Université de Montréal

Meta-heuristic Solution Methods for Rich Vehicle Routing Problems

par

Phuong Khanh Nguyen

Département d'informatique et de recherche opérationnelle

Faculté des arts et des sciences

Thèse présentée en vue de l'obtention du grade de

Philosophiae Doctor (Ph.D.)

en informatique

Juin, 2014

Université de Montréal

Faculté des arts et des sciences

Cette thèse intitulée :

Meta-heuristic Solution Methods
for Rich Vehicle Routing Problems

présentée par

Phuong Khanh Nguyen

a été évaluée par un jury composé des personnes suivantes:

Jean-Yves Potvin

(président-rapporteur)

Teodor Gabriel Crainic

(directeur de recherche)

Michel Toulouse

(co-directeur)

Michel Gendreau

(membre du jury)

Barrett Thomas

(examinateur externe)

Thèse acceptée le 10 Juin 2014

RÉSUMÉ

Le problème de tournées de véhicules (VRP), introduit par Dantzig and Ramser en 1959, est devenu l'un des problèmes les plus étudiés en recherche opérationnelle, et ce, en raison de son intérêt méthodologique et de ses retombées pratiques dans de nombreux domaines tels que le transport, la logistique, les télécommunications et la production. L'objectif général du VRP est d'optimiser l'utilisation des ressources de transport afin de répondre aux besoins des clients tout en respectant les contraintes découlant des exigences du contexte d'application.

Les applications réelles du VRP doivent tenir compte d'une grande variété de contraintes et plus ces contraintes sont nombreuse, plus le problème est difficile à résoudre. Les VRPs qui tiennent compte de l'ensemble de ces contraintes rencontrées en pratique et qui se rapprochent des applications réelles forment la classe des problèmes 'riches' de tournées de véhicules. Résoudre ces problèmes de manière efficiente pose des défis considérables pour la communauté de chercheurs qui se penchent sur les VRPs. Cette thèse, composée de deux parties, explore certaines extensions du VRP vers ces problèmes.

La première partie de cette thèse porte sur le VRP périodique avec des contraintes de fenêtres de temps (PVRPTW). Celui-ci est une extension du VRP classique avec fenêtres de temps (VRPTW) puisqu'il considère un horizon de planification de plusieurs jours pendant lesquels les clients n'ont généralement pas besoin d'être desservi à tous les jours, mais plutôt peuvent être visités selon un certain nombre de combinaisons possibles de jours de livraison. Cette généralisation étend l'éventail d'applications de ce problème à diverses activités de distributions commerciales, telle la collecte des déchets, le balayage des rues, la distribution de produits alimentaires, la livraison du courrier, etc. La principale contribution scientifique de la première partie de cette thèse est le développement d'une méta-heuristique hybride dans la quelle un ensemble de procédures de recherche locales et de méta-heuristiques basées sur les principes de voisinages coopèrent avec un algorithme génétique afin d'améliorer la qualité des solutions et de promouvoir la diversité de la population. Les résultats obtenus montrent que la méthode proposée est très performante et donne de nouvelles meilleures solutions pour certains grands exemplaires du problème.

La deuxième partie de cette étude a pour but de présenter, modéliser et résoudre deux problèmes riches de tournées de véhicules, qui sont des extensions du VRPTW en ce sens qu'ils incluent des demandes dépendantes du temps de ramassage et de livraison avec des restrictions au niveau de la synchronization temporelle. Ces problèmes sont connus respectivement sous le nom de Time-dependent Multi-zone Multi-Trip Vehicle Routing Problem with Time Windows (TMZT-VRPTW) et de Multi-zone Mult-Trip Pickup and Delivery Problem with Time Windows and Synchronization (MZT-PDTWS). Ces deux problèmes proviennent de la planification des opérations de systèmes logistiques urbains à deux niveaux. La difficulté de ces problèmes réside dans la manipulation de deux ensembles entrelacés de décisions: la composante des tournées de véhicules qui vise à déterminer les séquences de clients visités par chaque véhicule, et la composante de planification qui vise à faciliter l'arrivée des véhicules selon des restrictions au niveau de la synchronisation temporelle. Auparavant, ces questions ont été abordées séparément. La combinaison de ces types de décisions dans une seule formulation mathématique et dans une même méthode de résolution devrait donc donner de meilleurs résultats que de considérer ces décisions séparément. Dans cette étude, nous proposons des solutions heuristiques qui tiennent compte de ces deux types de décisions simultanément, et ce, d'une manière complète et efficace. Les résultats de tests expérimentaux confirment la performance de la méthode proposée lorsqu'on la compare aux autres méthodes présentées dans la littérature. En effet, la méthode développée propose des solutions nécessitant moins de véhicules et engendrant de moindres frais de déplacement pour effectuer efficacement la même quantité de travail. Dans le contexte des systèmes logistiques urbains, nos résultats impliquent une réduction de la présence de véhicules dans les rues de la ville et, par conséquent, de leur impact négatif sur la congestion et sur l'environnement.

Mots-clés: Problèmes de tournées de véhicules, ramassage et livraison, demandes dépendantes du temps, synchronisation, méta-heuristique, algorithme génétiques hybrides générationnels, recherche tabou.

ABSTRACT

For more than half of century, since the paper of Dantzig and Ramser (1959) was introduced, the Vehicle Routing Problem (VRP) has been one of the most extensively studied problems in operations research due to its methodological interest and practical relevance in many fields such as transportation, logistics, telecommunications, and production. The general goal of the VRP is to optimize the use of transportation resources to service customers with respect to side-constraints deriving from real-world applications.

The practical applications of the VRP may have a variety of constraints, and obviously, the larger the set of constraints that need to be considered, i.e., corresponding to ‘richer’ VRPs, the more difficult the task of problem solving. The needs to study closer representations of actual applications and methodologies producing high-quality solutions quickly to larger-sized application problems have increased steadily, providing significant challenges for the VRP research community. This dissertation explores these extensional issues of the VRP.

The first part of the dissertation addresses the Periodic Vehicle Routing Problem with Time Windows (PVRPTW) which generalizes the classical Vehicle Routing Problem with Time Windows (VRPTW) by extending the planning horizon to several days where customers generally do not require delivery on every day, but rather according to one of a limited number of possible combinations of visit days. This generalization extends the scope of applications to many commercial distribution activities such as waste collection, street sweeping, grocery distribution, mail delivery, etc. The major contribution of this part is the development of a population-based hybrid meta-heuristic in which a set of local search procedures and neighborhood-based meta-heuristics cooperate with the genetic algorithm population evolution mechanism to enhance the solution quality as well as to promote diversity of the genetic algorithm population. The results show that the proposed methodology is highly competitive, providing new best solutions in some large instances.

The second part of the dissertation aims to present, model and solve two rich vehicle routing problems which further extend the VRPTW with time-dependent demands of pickup and delivery, and hard time synchronization restrictions. They are called Time-dependent Multi-zone Multi-Trip Vehicle Routing Problem with Time Windows (TMZT-VRPTW), and Multi-zone Mult-Trip Pickup and Delivery Problem with Time Windows and Synchronization (MZT-PDTWS), respectively. These two problems originate from planning the operations of two-tiered City Logistics systems. The difficulty of these problems lies in handling two intertwined sets of decisions: the routing component which aims to determine the sequences of customers visited by each vehicle, and the scheduling component which consists in planning arrivals of vehicles at facilities within hard time synchronization restrictions. Previously, these issues have been addressed separately. Combining these decisions into one formulation and solution method should yield better results. In this dissertation we propose meta-heuristics that address the two decisions simultaneously, in a comprehensive and efficient way. Experiments confirm the good performance of the proposed methodology compared to the literature, providing system managers with solution requiring less vehicles and travel costs to perform efficiently the same amount of work. In the context of City Logistics systems, our results indicate a reduction in the presence of vehicles on the streets of the city and, thus, in their negative impact on congestion and environment.

Keywords: Vehicle Routing problem, pickup and delivery, time-dependent demand, synchronization, meta-heuristics, hybrid generational genetic algorithm, tabu search.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

ACKNOWLEDGEMENT

First and foremost, I am deeply thankful to my supervisors, Professor Teodor Gabriel Crainic and Professor Michel Toulouse for giving me the opportunity to do my graduate study at the Université de Montréal, and for patiently supporting my research with their invaluable assistance and guidance through the years. This dissertation would not have been possible without their aid, inspiration, and plenty fruitful discussions with them.

I am grateful to my labmates and staffs at the Interuniversity Research Centre on Enterprise Networks, Logistics and Transportation (CIRRELT), especially Marie-Ève Rancourt and Nadia Lahrichi, for their assistance and friendship which made my stay in Montréal more enjoyable. I would also like to express my gratitude to the CIRRELT in general, for providing such a rich and fulfilling research environment.

I would like to thank the School of Information and Communication Technology, Hanoi University of Science and Technology, for allowing me to come to Montréal to improve my professional knowledge through doctoral study. I would also like to thank all my colleges in the Computer Science department, for sharing their experiments and giving me continuous support.

Finally, on the personal side, I would like to thank my parents and my brother for their constant encouragement and understanding, especially when I am far from home.

Chapter 1

INTRODUCTION

1.1 Motivations

Transportation is a necessity in human society, allowing people, production and consumption of products to occur at different locations locally, nationally, and worldwide. Transportation is, however, also a major user of energy, it burns a large of the world's petroleum (Fuglestvedt et al., 2008). This creates environmental pollution, contributes significantly to global warming through emission of carbon dioxide. Therefore, improving the efficiency of transportation services is one of the key drivers for the reduction of transportation cost, as well as of global warming. As a result, applying operations research to problems of planning and management of transport operations has become a very progressive trend of academic study and industrial research. Operations research offers methodologies that provide solutions with high efficiency and quality in terms of economy and transportation services.

One specific problem related to planning the distribution process which has received a lot of attention is the Vehicle Routing Problem (VRP). It was first considered in Dantzig and Ramser (1959) and defined as the problem of designing the optimal set of routes for a fleet of vehicles needed to service a set of customers. Due to the high industrial applicability, the significant economic benefit that can be achieved when solving the problem, the VRP has been the object of numerous studies. In recent years, thanks to the increasing efficiency of solution methods and the availability of more powerful computers, the interest has shifted toward extended versions arising in real life. These extensions are identified as Rich Vehicle Routing Problems (RVRPs). In contrast to the classical VRP that work on idealized models with assumptions not always relevant to real-life problems, the

RVRPs consider more general models encountered in many aspects of industrial problems. Since the VRP is NP-hard (Lenstra and Rinnooy Kan, 1981), it is not always possible to solve instances to optimality within limited computation time. So far, exact algorithms have been able to solve the classical VRP instances having a small number of customers in term of real-life applications. The complexity of VRP therefore calls for heuristic solution approaches when 'rich' variants that involve multiple constraints, or realistically-sized instances are contemplated.

This dissertation consists of three papers, concerning the development of mathematical models as well as methodological solution approaches addressing three rich vehicle routing problems. In these problems, customers may require several services at different period of times during the planning horizon. Depending on the practical issues involved in each problem, different additional attributes have been addressed.

We first consider the ***Periodic Vehicle Routing Problem with Time Windows*** (PVRPTW) that generalize the classical Vehicle Routing Problem with Time Windows by extending the planning horizon to several days in which vehicle routes must be constructed over this period. Customers generally do not require delivery on every day, but rather according to one of a limited number of possible combinations of visit days. During each day within the planning period, a fleet of capacitated vehicles travels along routes that begin and end at a single depot, and serves only customers of that day within their time windows. This generalization extends the scope of applications to many commercial distribution activities such as waste collection, street sweeping, grocery distribution, mail delivery. The periodicity in the problem imposes a strong interaction between decisions that have to be taken during different days. It is therefore not possible to solve the problem on a single day basis and then to replicate the solution over the planning horizon.

We propose a new population-based hybrid meta-heuristic to solve the PVRPTW. This meta-heuristic is a generational genetic algorithm that uses two neighborhood-based meta-heuristics to optimize the offspring generated by crossover operators during the evolution. Local search methods have previously been proposed to enhance the fitness of offspring. In the proposed method, neighborhood-based meta-heuristics are instead used for their capacity to escape local optima, and deliver optimized and diversified solutions to the population of the next generation. Furthermore, the search performed by the neighborhood-based meta-heuristics repairs most of the constraint violations that naturally occur after the application of the crossover operators. Addressing the PVRPTW requires the assignment of customer-to-visit-day-patterns, and the scheduling of the deliveries. We therefore introduce two new crossover operators, one aiming at exploring different visit-day assignment, while the other exploits route components present in the parents. Together, these two crossover operators balance exploration and exploitation, thus improving the search efficiency of the algorithm. Extensive numerical experiments and comparisons with all methods proposed in the literature show that the proposed methodology is highly competitive, providing new best solutions for a number of large instances.

The second problem we consider is the ***Time-dependent Multi-zone Multi-trip Vehicle Routing Problem with Time Windows*** (TMZT-VRPTW). The TMZT-VRPTW originates from planning the operations of two-tiered City Logistics systems (Crainic et al., 2009). In such systems, incoming loads are first sorted and consolidated at a first-tier facility, located on the outskirts of the city. Then they are moved to a second-tier facility, satellite or supply point, by a fleet of first-tier vehicles, where they are transferred to smaller-capacity vehicles for final delivery to customers located within the controlled city area. Activities take place over a time horizon several hours long, vehicles, second-tier ones in particular, performing multiple tours before returning to the depots. Most importantly, the locations of most supply points within or close to the city center and their limited capacity, together with the need for efficient operations and on-time delivery to customers, induce the need for transdock load-transfer activities and the synchronization of the operations of first and second-tier vehicles, very short waiting times being allowed for vehicles at supply points. Planning determines the customer demands to service out of each supply point at each time period. When the set is non empty, it is called thereafter the supply-point zone. A second-tier vehicle then arrives at a supply point at an appointed time, meets the first-tier vehicles bringing in the demands for the zone, and loads the planned freight. It then performs a tour servicing its designated customer demands within the zone. Once the last customer is serviced, the vehicle moves either directly to a supply point for its next tour, the preferred move, or to a waiting station (when available) to wait for its next appointment, or to the depot to end the current activity period. The TMZT-VRPTW corresponds to the planning of the activities of second-tier vehicles.

We introduce the first tabu search for the TMZT-VRPTW, integrating multiple neighborhoods grouped into two classes to address the two sets of decisions of the problem simultaneously rather than separately as was previously done. A first set of neighborhoods and moves work on the construction of multiple-trip vehicle routes by modifying the facilities and availability periods a given vehicle visits, while the second aims to improve the routing of vehicles between two such visits by working on the customer to route assignments. The neighborhood selection rule is dynamically modified during the search, and a diversification strategy guided by an elite set of solutions and a frequency-based memory is called upon when the search begins to stagnate. The neighborhood definitions follow from a new formulation we propose for the TMZT-VRPTW. Extensive computational experiments show that the proposed tabu search yields very high quality solutions. It outperforms the currently available method (Crainic et al., 2012b) with new best-known solutions on all instances and an improvement in the solution quality by 4.42% on average.

In the TMZT-VRPTW, freight flow is only considered in the direction from regions outside the city to the city center. In reality, freight is moved in, out, and through a city, however. Integrating these flows into a single City Logistics system, facilities and vehicles serving simultaneously several traffic types, would reduce the presence of vehicles on the streets of the city, and make freight transportation more efficient. Such integration might make the planning, management of scheduling and synchronizing processes more complex, even difficult to implement in practice, however. Taking it as a challenge, in the last part of this dissertation, we address an integration of *outbound* freight flow, shipping freight from the city center to destinations outside the city limits, into the TMZT-VRPTW. We introduce a problem, ***Multi-zone Multi-trip Pickup and Delivery Problem with Time Windows and Synchronization***(MZT-PDTWS), which has not been studied before. The MZT-PDTWS is more complex than the TMZT-VRPTW in the sense that a new type of customer demands, say *pickup*, is now considered together with delivery-customer demands. Furthermore, pickup-customer-demands supply-point assignments are not known in advance as for delivery-customer demands, but rather each pickup-customer demands has a list of available supply points that can service it, thus requiring the assignment of each pickup-customer demand to one supply point selected from its list before routing them. We propose a model for the MZT-PDTWS, and generalize the tabu search proposed for the TMZT-VRPTW so that it can tackle the new problem efficiently. Different experiments on both algorithmic design and synchronization policies have been conducted. New benchmark instances with up to 72 supply points and 7200 customer demands have been built to show the performance of the proposed method.

In summary, we propose efficient meta-heuristics for three rich vehicle routing problems which are either studied in very few papers, or first considered in the literature. In the first study, we introduce the use of neighborhood-based meta-heuristics within a generational genetic algorithm. This hybridization provides the means to repair and enhance individuals produced during the evolution process, as well as to promote diversity of the genetic algorithm population. In the next studies, we study more complicated VRP variants. A decomposition approach has previously been proposed to solve the problems in which sets of decisions in the problems were addressed separately. We therefore propose a new neighborhood-based meta-heuristic that addresses these sets of decisions simultaneously, in a comprehensive and efficient way. Extensive numerical experiments and comparisons with the literature show that our proposed methods yield high quality solutions, providing many new best solutions or matching many of the existing ones.

1.2 Outline of the dissertation

The remainder of the dissertation is organized as follows. In Chapter 2, we give a short overview on characteristics and constraints encountered when solving vehicle routing in practice, as well as heuristics used to solve three VRP variants related to problems studied in this dissertation. Each of the next three chapters (Chapter 3, 4 and 5), corresponding to an associated paper, studies a specific problem (the PVRPTW, TMZT-VRPTW and MZT-PDTWS respectively), and follows the same structure. Each chapter gives a proper description of the problem studied, states and discusses published works, details our contributions, introduces available as well as newly generated benchmark when appropriate, reports the results compared with the literature, and finishes with conclusion and ideas for potential future works. Overall conclusions are drawn in Chapter 6. Appendix A, B, and C provide supplementary material of the PVRPTW, TMZT-VRPTW, and MZT-PDTWS, respectively.

Chapter 2

LITERATURE REVIEW

This chapter provides an overview of the work on the VRP variants related to the problems studied in this dissertation. The VRP and its variants have been the object of numerous studies, and a great number of papers have proposed solution methods for these problems. This chapter therefore begins with a classification of the VRP by describing the characteristics and constraints encountered when solving vehicle routing in practice (Section 2.1). Based on this classification, Section 2.2 refers to books and survey papers which provide more information on the VRP, its solution methods and recent works. Finally, three VRP variants related to the problems studied in this dissertation are reviewed: the Vehicle Routing problems with Time Windows (Section 2.3), the Pickup and Delivery problems (Section 2.4), and Vehicle Routing problems with Synchronization of vehicles (Section 2.5).

2.1 Classes of vehicle routing problems

The different VRP formulations that appear in the literature derive from the following four common attributes of real-life applications: depots, vehicles, customers, objective function. These attributes are described in the following:

- **Depots**. Either a single depot or multiple depots can be specified. If there are multiple depots, each depot will have either an upper bound on the number of vehicles of a particular type, or an upper bound on the total number of vehicles that can be housed there.

- **Vehicles.**

  – Characteristics

    ∗ Number of vehicles: limited or unlimited.

    - Capacity: the maximum weight that a vehicle can load.
    - Types: homogeneous or heterogeneous fleet.
    - Compartments: one or multiple (possibly subdivided in loading parts, each characterized by its capacity and by the types of freight that can be carried).
  - Operational rules
    - Route length (duration): each vehicle has a route-length constraint (for example, 10 hours).
    - Depot: vehicles must return to their starting depot (or not).
    - Pick up and delivery: vehicles available for the loading and unloading operations.
    - Multiple tours: a vehicle can do more than one route.

- **Customers.**
  - Amount of demand: freight quantity to be collected or delivered.
  - Splitting: allowing deliveries to be split, a customer may be serviced by more than one vehicle.
  - Time window: the period during which the customer can be served; Time window is either hard or soft. If the time window for a customer is $[a,b]$ and the window is hard, then servicing this customer by a vehicle has to be carried out between a and b. If this time window is soft, the vehicle need not service this customer during $[a,b]$; if the time window is missed, the algorithm usually assesses a penalty for the delayed/early service.
  - Numbers of visits: one or multiple in a planning period of several days.
  - A number of commodity: one or multiple (each customer may order one or several products).

- **Objective function.** Usually the objective function is to minimize a weighted combination of capital and operating costs for the fleet. It may also include a formula that represents penalties for not meeting all the time-window constraints and/or for violating other constraints. Sometimes these objectives are hierarchical; in other cases, they are considered concurrently.

Based on the objective function to be optimized and the types of constraints to be satisfied, different variants are introduced in the literature. The problems grouped under the 'rich' VRP variants have in common the characteristics of including additional realistic constraints, aiming a closer representation of real-life problems. As the number of constraints grow in a variant, the 'richer' VRP variant might more closely model the real-world applications, but it is likely more difficult to solve. In the following, we just introduce the most familiar basic VRP variants from which 'rich' VRPs are often built by considering the combination of these variants.

- Capacitated VRP (CVRP): each customer has its own demand and the total demand on a route can not exceed the vehicle capacity. In some time- or distance-constrained variants, additional constraints state that the length of a route should not exceed a given limit (DVRP) (see Golden et al. (1998), Altinel and Oncan (2005), Toth and Tramontani (2008)).

- VRP with time windows (VRPTW): a vehicle must arrive at each customer within a time interval (see Solomon (1987), Potvin and Rousseau (1993, 1995), Le Bouthillier and Crainic (2005)).

- Periodic VRP (PVRP): it is a variant of the VRP where the planning horizon extends over a number of periods. Routes are constructed for each period, and each customer is visited once or more over the horizon, depending on the customer requirements (see Christofides and Beasley (1984), Tan and Beasley (1984), Russell and Gribbin (1991), Cordeau et al. (1997), Drummond et al. (2001), Pirkwieser and Raidl (2008), Vidal et al. (2012)).

- VRP with heterogeneous fleet (VRPHF): the vehicles do not share the same characteristics (e.g. different capacities, different fixed costs) (see Gendreau et al. (1999), Lima et al. (2004), Dondo and Cerda (2007)).

- Multi-depot VRP (MDVRP): there are many depots and each vehicle can start or end its route from any of these depots (see Hadjiconstantinou and Baldacci (1998), Polacek et al. (2004), Ho et al. (2008), Xu et al. (2012)).

- Multiple tour VRP: one can assign more than one route to a vehicle over the planning horizon. Different names are given to problems with this setting such as "Multiple use of vehicles" (Taillard et al. (1995)), "multi-trip" (Brandão and Mercer (1997)), "multiple trips" (Petch and Salhi (2003)), "multiple vehicle trips" (Olivera and Viera (2007), Battarra et al. (2009)), "inter-depot" route (Crevier et al. (2007)), "multiple routes" (Azi et al. (2014)).

- Split delivery VRP (SDVRP): each customer may be served by different vehicles (the restriction that each customer is visited once is removed and the demand of each customer can be greater than the capacity of the vehicles) (see Dror et al. (1994), Frizzell and Giffin (1995), Belenguer et al. (2000), Ho and Haugland (2002)).

- The multi-compartment VRP (MC-VRP): products are incompatible and must be transported in independent vehicle compartments (see Fallahi et al. (2008), Mendoza et al. (2010)).

- General pick up and delivery problem. Basically, two problem classes can be distinguished:

  - The first class refers to situations where all goods delivered have to be loaded at one or several depots and all goods picked up have to be transported to one or several depots. Problems of this class are usually referred to as either Vehicle Routing Problems with Backhauls (VRPB) or VRP with Simultaneous Delivery and Pick-up (VRPSDP). In the VRPB, each route is a mix of linehaul customers (require deliveries) and backhaul customers (require pickups), where the backhauls are typically visited after the linehauls (see Goetschalckx and Jacobs-Blecha (1989); Jacobs-Blecha and Goetschalckx (1993), Thangiah et al. (1996), Brandão (2006), Zachariadis and Kiranoudis (2012), Salhi et al.

(2013)). In the VRPSDP, the same customer can ask both for freight to be delivered from the depot and for other freight to be picked up and brought to the depot. Each customer has to be visited either exactly once or twice (one for delivery and one for pickup) (see Montané and Galvao (2006), Ai and Kachitvichyanukul (2009), Zachariadis and Kiranoudis (2011), Tasan and Gen (2012)).

  – The second class, namely Pickup and Delivery VRP (VRPPD), considers all problems where freight are transported between pickup and delivery locations. It can be further divided into two subclasses which refer to situations where pickup and delivery locations are unpaired and paired, respectively. In the former problem class, every load can be picked up and transported anywhere (see Hernández-Pérez and Salazar-González (2003), Hernández-Pérez et al. (2009), Zhao et al. (2009)). The latter problem class considers transportation requests, each associated with an origin and a destination, resulting in paired pickup and delivery points with precedence constraint where the pick up point must be visited before the delivery point (see Bent and Hentenryck (2006)).

- VRP with synchronization considers the situations where vehicle routes must be synchronized in time and/or space (see Crainic et al. (2010, 2011), Dohn et al. (2011), Meisel and Kopfer (2012)). The spatial dimension of synchronization defines the locations where vehicle synchronization can take place, while the temporal dimension of synchronization defines the order in which vehicles must visit a synchronization point.

2.2 Survey Papers

A number of survey papers have appeared in various journals and books dealing with VRP. These papers provide the reader with a broad introduction, a synopsis of modeling and solution methods, and outline the status and prospects for future research to not only the general VRP but also to some of the main VRP variants.

A good overview of exact and heuristic methods, together with descriptions of some application areas, can be found in the book “The Vehicle Routing Problem”, by Toth and Vigo (2002).

Laporte (2007) gives a literature survey on the classical VRP in which only vehicle capacity constraints are present. It presents three main solution approaches used in the literature (i.e. exact algorithms, classical heuristics, and meta-heuristics) and the comparison of their results when available. Using the same approach, Cordeau et al. (2007) provide a general survey of the most important VRP variants (CVRP, VRPTW, inventory routing problems, stochastic vehicle routing problems), consisting of the mathematical formulations followed by the description of some of the most important available exact and heuristic algorithms.

The chapters in the book “The Vehicle Routing Problem: Latest Advances and New Challenges” by Golden et al. (2008) summarize the most significant results, methodological advances, new approaches for solving existing VRP since 2000, and highlights new challenges for the field.

Drexl (2012b) gives a comparison of the state of the art of scientific research and commercial software for modeling and solving VRP.

For more surveys and bibliographies on meta-heuristics for VRP, the reader is referred to Gendreau et al. (2002), Cordeau et al. (2005), Cordeau and Laporte (2005), Potvin (2009), Gendreau et al. (2008), Vidal et al. (2013b). Crainic (2008) provides a survey of exact and heuristic parallel algorithms for the VRP, while Crainic and Hail (2005) describe parallel meta-heuristics for VRP.

Surveys of some main VRP variants have also been published, such as the VRPTW (see Bräysy and Gendreau (2005a,b), Tripathi and Minocha (2006)), the VRP with pick up and delivery (see Savelsbergh and Solomon (1995), Mitrović-Minić (1998), Parragh et al. (2008a,b)), Periodic VRP (see Francis et al. (2008), Campbell and Wilson (2014)), Multi-trip VRP (see Şen and Bülbül (2008)), and VRP with synchronization (see Drexl (2012a)).

2.3 The Vehicle Routing problems with Time windows

The VRPTW is the problem of designing least cost vehicle routes, originating and ending at a central depot, servicing a number of geographically situated customers with known demands. The total demands of all customers on each vehicle route must not exceed the capacity of the vehicle. Each customer is serviced only once by exactly one vehicle within a given time interval, called a time window. The time window is hard when a vehicle route is not feasible if the service of a customer either starts before the earliest time or ends after the latest time of the time window bound, i.e., if a vehicle arrives too early at a customer, it must wait until the time window opens; and it is not allowed to arrive late (see Desrochers et al. (1992), Savelsbergh (1992), Garcia et al. (1994), Badeau et al. (1997), Schulze and Fahle (1999), Ioannou et al. (2001), Le Bouthillier and Crainic (2005)). In other cases, both lower and upper bounds of the time windows can be violated with a penalty. These are VRP with soft time windows (see Taillard et al. (1997), Ioannou et al. (2003), Calvete et al. (2007), Nai-Wen and Chang-Shi (2013)). The VRP with time deadlines is a special case of the VRPTW, where the time windows are replaced by time deadlines (there is only an upper bound) (see Thangiah et al. (1993)).

The VRPTW is an important problem occurring in many distribution systems, e.g., deliveries to super markets, bank and postal deliveries, school bus routing, industrial refuse collection, etc. As the VRPTW generalizes the CVRP by including time window at each customer, it can be seen as the core problem of most of VRP variants. Consequently, the VRPTW has been the object of intensive research efforts both for the design of heuristics and the development of optimal approaches. Here, we focus on heuristic and meta-heuristic approaches. Details about optimal methods can be found in the recent survey paper of Desaulniers et al. (2010).

Classical heuristic approaches can be roughly separated into two categories: constructive and improvement heuristics (Laporte et al. (2000)). Constructive heuristics work on building a feasible solution without a separate improvement scheme, while improvement heuristics work on improving an incumbent solution by using some types of edge exchange heuristics within and between routes.

Constructive heuristics were often proposed for the CVRP, but one could also apply them to the VRPTW without considering the feasibility of time windows at customers. Constructive heuristics fall into one of three classes:

- **Savings heuristics** initially build a solution where each customer is served on its own route. Routes are then merged one by one according to some criteria. Savings algorithms vary by the criterion used for merging routes (what saving is obtained by merging two routes) and by how routes are merged (see Clarke and Wright (1964), Gaskell (1967), Yellow (1970), Altinel and Oncan (2005)).

- **Insertion heuristics** build a solution by inserting one customer at a time. Insertion heuristics can build one route at a time (sequential insertion heuristics) or build many or all routes in parallel (parallel insertion heuristics). The choice of which customer to insert and where to insert the customer is what differentiates the insertion heuristics (see Solomon (1987), Potvin and Rousseau (1993), Ioannou et al. (2001), Figliozzi (2009)).

- **Clustering heuristics** are two-phase algorithms. The first phase consists of grouping customers into subsets (clusters). The second phase then creates routes for each subset. A third phase may be employed to repair the solution if it turns out that some of the clusters could not be served by a single vehicle (see Gillett and Miller (1974)).

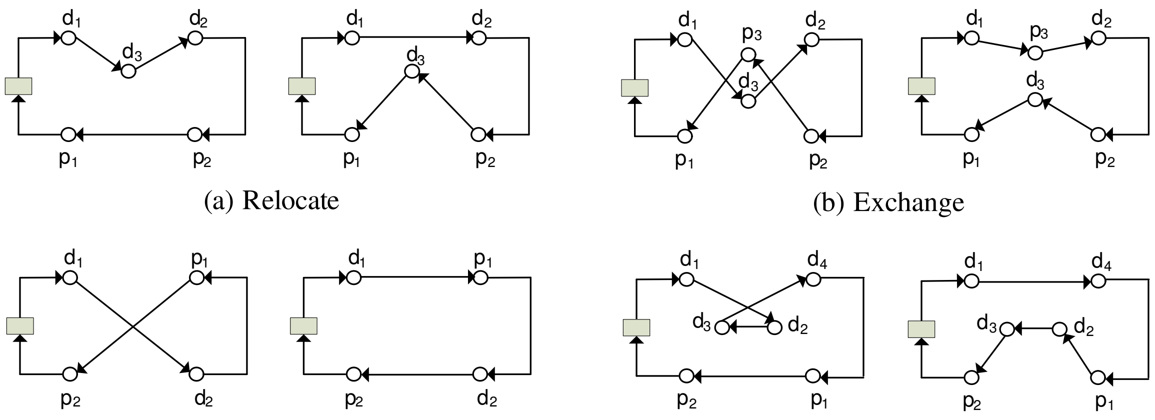
The common base of improvement heuristics for the VRP is that they are all local search methods. Local search is a commonly used technique in combinatorial optimization. It starts with an initial solution s, and repeatedly replaces the current solution s with a better solution s' in its neighborhood $N(s)$ until no better solution exits in $N(s)$, where neighborhood $N(s)$ is a set of solutions obtainable by slightly perturbing the current solution s.

Improvement heuristics for the VRP are based on the movements of either vertices or edges in one route (intra-route) (see Figure 2.1) or several routes (inter-route) (see Figure 2.2) at a time. These movements can be classified as follows:

- Vertex move: a vertex is removed from its position and inserted into another position (of the same route or a different route).

- Vertex swap: two vertices exchange their position in the solution.

- Edge exchange: replace λ edges in the solution by λ other edges.



(a) Relocate

(b) Exchange

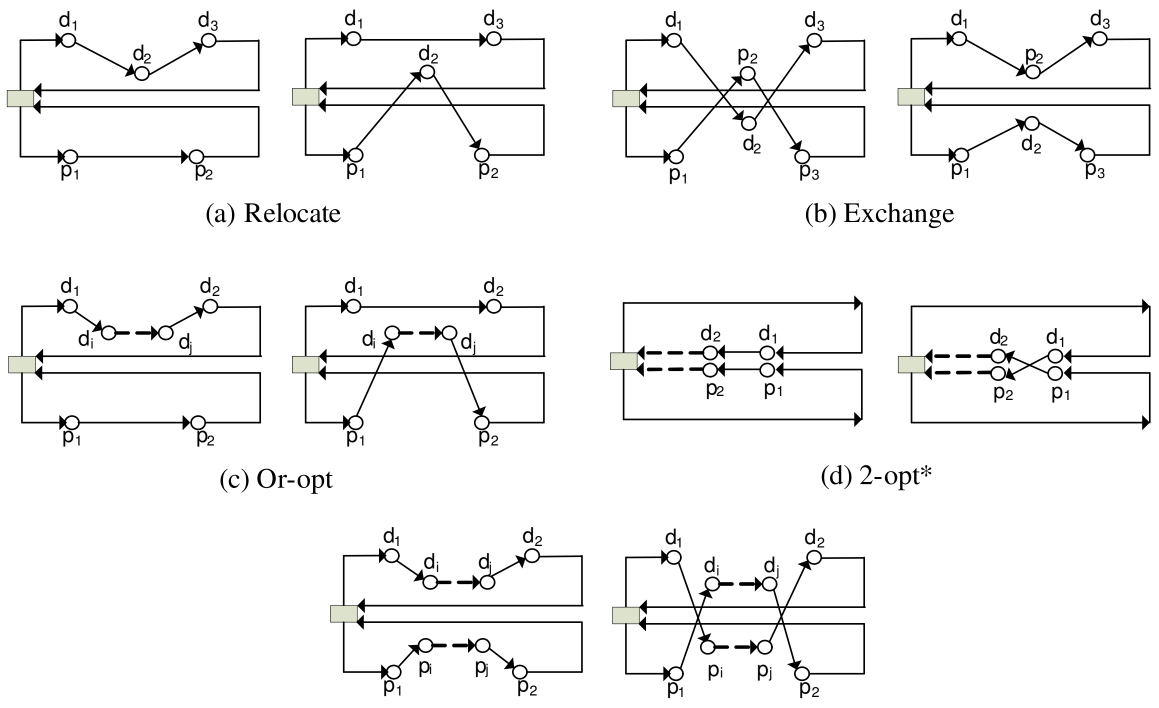
(c) 2-opt

(d) Or-opt

Figure 2.1: Intra-route operators

Numerous combinations, extensions and adaptations of these basic movements for different types of VRP are reported in the literature. For instance, it is possible to move or swap sequences of consecutive vertices (rather than a single vertex), e.g. CROSS exchange (Taillard et al. (1997)) with Or-opt (Or (1976)) and 2-opt* (Potvin and Rousseau (1995)) as special cases, GENI-exchange (Gendreau et al. (1992)), ejection chains (Xu and Kelly (1996), Rego and Roucairol (1996)).

One challenge of basic local search methods is that the search often converges to local optima. Thus, researchers have developed methods that allow the search to 'escape' local optima to find improved solutions. The improvement procedures can be embedded in a meta-heuristic such as tabu search (Garcia et al. (1994), Potvin et al. (1996b), Taillard et al. (1997), Chiang and Russell (1997), Tan et al. (2000), Cordeau et al. (2001), Homberger and Gehring (2005)), variable neighborhood search (Rousseau et al. (2002), Bräysy (2003)), large neighborhood search (Pisinger and Ropke (2007)), simulated annealing (Chiang and Russell (1996), Tan et al. (2000), Czech and Czarnas (2002)), path relinking (Hashimoto and Yagiura (2008)), or population-based algorithm (e.g., genetic algorithm (Thangiah et al. (1991), Potvin and Bengio (1996), Tan et al. (2000), Vidal et al.



(a) Relocate (b) Exchange

(c) Or-opt (d) 2-opt*

(e) CROSS-exchange

Figure 2.2: Inter-route operators

(2013a)), ant colony optimization (Gambardella et al. (1999)), or hybridizations between meta-heuristics (e.g., hybridization of genetic algorithm with tabu search (Wee-Kit et al. (2001)) and ant colony systems (Berger et al. (2003)), hybridization of simulated annealing with tabu search (Tan et al. (2001), Li and Lim (2003)) and large neighborhood search (Bent and Hentenryck (2004), etc.)

Parallel and cooperative search methods form another effective approach. Gehring and Homberger (2002) developed a parallel two-phased evolutionary algorithm combined with tabu search. In the first phase, an evolutionary algorithm is performed to minimize the number of vehicles, while in the second phase, tabu search is applied for the traveling distance minimization. The parallelization is based on the concept of cooperative autonomy, for which several autonomous two-phase meta-heuristics cooperate through the exchange of solutions. Each independent thread is performed with different configuration settings.

Le Bouthillier and Crainic (2005) presented a cooperative search method in which several threads communicate through asynchronous exchanges of information using a pool of feasible solutions called warehouse. Each thread implements either a genetic algorithm or a tabu search. Communications are initiated only by individual threads with access to the warehouse, no broadcasting takes place. Tabu search algorithms require a single solution from the warehouse either for restarting the search or for diversification, while the improved solutions are sent back to the warehouse. GAs use the warehouse as population, and each offspring is sent back to the warehouse. Later, Le Bouthillier et al. (2005) extended the approach of Le Bouthillier and Crainic (2005) to a guided parallel cooperative search method. It is based on a central memory structure, equipped with a mechanism to extract knowledge from the information exchanged among search threads in order to guide the search toward promising and unexplored regions of the solution space. The threads share information about their respective good solutions identified so far. When a thread improves the solution, it sends this solution to the post-optimization algorithms present in the central memory. The central memory sends both solutions and pattern information to each cooperative thread when needed. A pattern-identification based on the inclusion of arcs within solutions is used to guide the search. For this purpose, the central memory is divided into three subsets according to the quality of solutions (i.e., elite, average and worst). The frequency of an arc is calculated as the number of times it appears in solutions belonging to a given subset of solutions. Each pattern contains a set of arcs, with similar frequency of inclusion or not. Guidance is obtained by transmitting arcs patterns to individual threads indicating whether the arcs in the pattern should be fixed to intensify the search or, on the contrary, they should be prohibited to diversify the search.

The currently best-meta-heuristic for the VRPTW is reported in Vidal et al. (2013a) who developed a hybrid genetic algorithm with adaptive diversity management. Their method combines the capacity of exploration of genetic algorithm and the intensification of local search procedures. Each individual is represented as a giant tour without trip delimiters which allows the use of simple permutation-based crossover operators. A *Split* procedure is then needed to partition a given tour into several vehicle routes to obtain the associated VRPTW solution. Furthermore, the evaluation of individuals is based on both penalized costs and contribution to diversity metrics. This method not only solves successfully the VRPTW, but also many VRPTW variants with multiple depots (MDVRPTW), multiple periods (PVRPTW), and vehicle site dependencies (SDVRPTW).

2.4 The Pickup and Delivery problems

There has been extensive research on variants of the Pickup and Delivery problems that consider different types of constraints that occur in real-world applications; see a number of surveys (Savelsbergh and Solomon (1995), Parragh et al. (2008a,b), Berbeglia et al. (2007, 2010)) and book (Toth and Vigo (2002)). Figure 2.3 displays a classification scheme. According to Berbeglia et al. (2007), the Pickup and Delivery problems are categorized into one-to-one (1-1), one-to-many-to-one (1-M-1), and many-to-many (M-M) schemes. The main difference among these three schemes is the transportation endpoint. One-to-many-to-one scheme deliver goods from the depot to delivery (linehaul) customers and from pickup (backhaul) customers to the depot, while one-to-one and many-to-many schemes deal with transportation between customers.

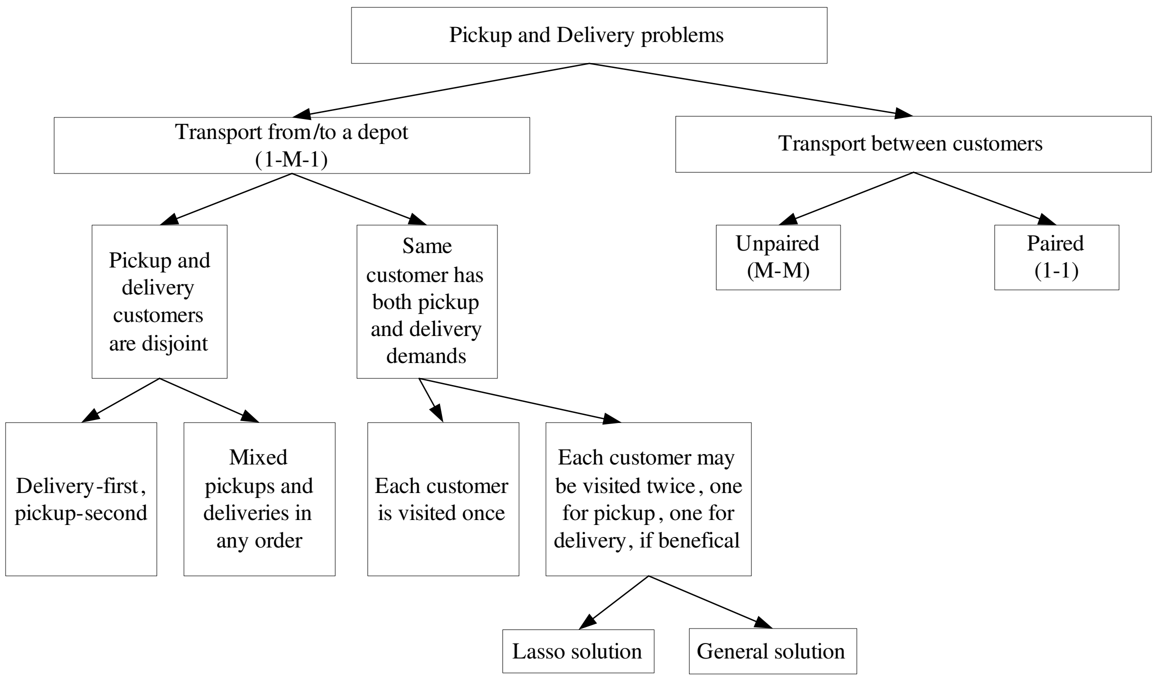

Pickup and Delivery problems
Transport from/to a depot (1-M-1)
Transport between customers
Pickup and delivery customers are disjoint
Same customer has both pickup and delivery demands
Unpaired (M-M)
Paired (1-1)
Delivery-first, pickup-second
Mixed pickups and deliveries in any order
Each customer is visited once
Each customer may be visited twice, one for pickup, one for delivery, if benefical
Lasso solution
General solution

Figure 2.3: A classification scheme for Pickup and Delivery Problems

2.4.1 One-to-many-to-one scheme

The class of one-to-many-to-one pickup and delivery problems is an extension of the VRP involving both delivery and pickup points. Linehaul (delivery) points are sites that are to receive goods from a main depot. Backhaul (pickup) points are sites that send goods to the main depot. The development of these problems was motivated by the fact that significant savings can be obtained by servicing both linehauls and backhauls in the same route as this results in less empty trips. In the literature there are many descriptions of applications of these problems, for example, in reverse logistics where full containers must be brought to customers, and empty containers must be returned from the customers to depot. There may be the case where the pickup and delivery service will be done either by one vehicle or a fleet of vehicles. This class of problems can be classified into three main categories:

- VRP with Backhauls (VRPB): In a route, first are served the linehaul customers and then the backhauls (delivery-first, pickup-second). No routes are allowed containing only backhauls, but a route can contain linehaul customers only.

- VRP with Mixed linehauls and Backhauls (VRPMB): any sequence of linehauls and backhauls in a route is permitted.

- VRP with Simultaneous Delivery and Pickup (VRPSDP): the same customer may have both a pickup and a delivery demand. Such customer may either be visited exactly once or twice, one for delivery and one for pickup. It is clear that the latter case which allows each customer to be visited once or twice is more flexible, thus can yield better solutions than the former. Parragh et al. (2008a) suggested a new name for the latter case - VRP with Divisible Deliveries and Pickups (VRPDDP). The literature further divides the VRPDDP into two categories according to distribution strategy: (1) ***lasso solution strategy***: each vehicle first follows a path performing deliveries; when sufficient space is created, it then starts visiting the remaining customers assigned to this vehicle along a loop by performing a simultaneous pickup and delivery at each visit, and finally a path is followed to perform the remaining pickups at the first visited customers in reverse order; (2) ***general solution strategy***:

customer may be visited twice either in two different routes or at different times on the same route.

We note that the VRPMB can be considered as a special case of the VRPSDP because when the delivery or pickup demand of each customer is set to zero, the VRPSDP reduces to VRPMB. Hence, a solution approach developed to solve the VRPSDP can be directly used to solve the VRPMB. In the same way, the VRPMB may be modeled as a VRPSDP by adding a pickup of zero to each linehaul and a delivery of zero to each backhaul. On the other hand, the customers of VRPDDP can be divided into pickup and delivery entities to give a mixed formulation. Table 2.1 presents a summary of the existing literature on these three variants.

VRP with Backhauls (VRPB) Among heuristic methods, early studies focused only on constructive methods. Deif and Bodin (1984) were among the first to develop classical constructive heuristics. They proposed two approaches based on the savings methods of Clarke and Wright (1964). In the first approach, a constraint is introduced to ensure that all deliveries are made before any pickup. In the second approach, pickup customers are delayed from early inclusion in routes by introducing a penalty factor in the basic savings function. The drawback of these two approaches is that the number of routes could not be controlled in the final solution. Therefore, the solution found may require more vehicles than the maximum available to service all customers.

The improvement heuristics were initiated by Goetschalckx and Jacobs-Blecha (1989). They presented two methods based on the idea of space-filling curves, where separate routes are developed for the pickup and delivery customers. These routes are then merged according to the space-filling mapping to obtain a final set of routes. The solution is then improved by using the 2-opt and 3-opt. Later, in Jacobs-Blecha and Goetschalckx (1993), they developed a cluster-first route-second algorithm based on the generalized assignment approach of Fisher and Jaikumar (1981). This method produced better results than their previous one.

Osman and Wassan (2002) were the first to develop a tabu search meta-heuristic for the VRPB. They proposed two route construction methods which are based on saving-insertion and saving-assignment procedures. The solution is then improved by a reactive

Table 2.1: A comparison of the different papers on the 1-M-1 Pickup and Delivery problems

Paper	Problem type					Solution method
	VRPB	VRPMB	VRPSDP			
			Each customer is visited			
			Once	Either once or twice (VRPDDP)		
				Lasso	General	
Deif and Bodin (1984)	x					Constructive heuristic
Goetschalckx and Jacobs-Blecha (1989)	x					Improvement heuristic
Jacobs-Blecha and Goetschalckx (1993)	x					Improvement heuristic
Osman and Wassan (2002)	x					Reactive tabu search
Brandão (2006)	x					Tabu search
Wassan (2007)	x					Reactive tabu search + Adaptive memory
Gajpal and Abad (2009b)	x					Multi-ant colony system
Golden et al. (1984)		x				Constructive heuristic
Salhi and Nagy (1999)		x			x	Constructive heuristic
Nagy and Salhi (2005)		x			x	Local search
Reimann and Ulrich (2006)		x				Ant colony optimization
Ropke and Pisinger (2006)	x	x	x			Adaptive large neighborhood search
Halse (1992)			x			Constructive heuristic
Crispim and Brandão (2005)		x	x			Tabu search + Variable neighborhood search (VNS)
Montané and Galvao (2006)			x			Tabu search
Chen and Wu (2006)			x			Tabu search
Wassan et al. (2008)			x			Reactive tabu search
Zachariadis et al. (2009)			x			Tabu search + Guided local search
Gajpal and Abad (2009a)			x			Ant colony system
Subramanian et al. (2010)			x			VNS + Iterated local search
Goksal et al. (2013)		x	x			Particle swarm optimization + VNS
Nagy et al. (2013)					x	Reactive tabu search
Hoff et al. (2009)				x		Tabu search

tabu search which considers single-node and two-node exchange neighborhood structures. Later, Brandão (2006) proposed a new tabu search algorithm where the initial solution is obtained from a pseudo lower bound based upon the K-tree approach. Their tabu search examines three neighborhood structures that involve relocating a customer to another route, exchanging two customers belonging to two different routes, and exchanging the positions of pickup and delivery customer within the same route. An intra-route repair operator is applied if the precedence constraint is violated. Their algorithm is better than Osman and Wassan (2002) algorithm in terms of average cost, number of optimal solutions found and also computing time on the benchmark instances of Goetschalckx and Jacobs-Blecha

(1989) and Toth and Vigo (1996). Ropke and Pisinger (2006) proposed a unified heuristic based on a large neighborhood search which effectively deals with three routing variants that consider backhaul customers: VRPB, VRPMB, and VRPSDP with and without time windows. Their method uses different removal and insertion algorithms. At every iteration a certain number of customers is removed from the routes by means of either random, worst customer, cluster or history based removal. The free customers are then inserted using a basic or a regret insertion heuristic. Later, Wassan (2007) combined adaptive memory programming with tabu search and proposed reactive tabu adaptive memory programming search (RTS-AMP). Their tabu search uses λ-interchange of customers as proposed by Osman (1993). RTS-AMP maintains a set of solutions called elitist solutions and uses these solutions to guide the search towards unexplored regions of the solution space. Gajpal and Abad (2009b) presented a multi-ant colony system: the first colony is used to assign customer to vehicles, and the second is used to construct a route for a vehicle given the assigned customers, i.e. to solve a Traveling Salesman Problem. After routes are constructed, three local search procedures are applied.

VRP with Mixed linehauls and Backhauls (VRPMB) Golden et al. (1984) presented an insertion-based procedure where routes are initially developed for delivery customers by using some VRP approaches, and then pickup customers are inserted into delivery routes according to an insertion criterion. Salhi and Nagy (1999) extended this scheme by allowing the insertion of a cluster of pickup customers, and addressed both the VRPMB and VRPDDP with general solution strategy.

Reimann and Ulrich (2006) proposed an insertion based ant colony optimization method. Solutions are generated iteratively based on pheromone information, using a sequential insertion based construction heuristic. The local search improvement phase consists of swap and shift operators.

Goksal et al. (2013) later presented a heuristic solution approach based on particle swarm optimization (PSO) in which a local search is performed by Variable Neighborhood Descent. Moreover, it implemented an annealing-like strategy to preserve the swarm diversity. The computational results indicated that their method improved 104 out of 141 best-known solutions on benchmark of VRPMB instances (the improvement is around 2%

on average).

VRP with Simultaneous Delivery and Pickup (VRPSDP) The VRPSDP was first proposed by Min (1989). The author presented a heuristic to solve a real-life problem concerning the distribution and collection of books of a public library. Customers are initially clustered into groups with respect to the capacity of a vehicle, and then for each group a Traveling Salesman Problem is solved. As mentioned before, the VRPSDP is divided into two categories according to patterns of goods movements:

- Each customer is visited once: Halse (1992) later proposed a two-phase heuristic based on the cluster-first-route-second concept. Crispim and Brandão (2005) were the first to present a meta-heuristic approach for the VRPSDP. Their method is a hybrid of tabu search and variable neighborhood search. Initial solutions are generated using a sweep method. If any route within this solution is infeasible due to intermediate arcs being overloaded, the order of customers on the route is exchanged until feasibility is established. The improvement phase is built on the moves insert and swap. Later, more works for the VRPSDP have been published. Montané and Galvao (2006) used a tabu search framework. The neighborhood is built using the moves insert, exchange, crossover (splitting and splicing two routes) and 2-opt. The balance between intensification and diversification of the search is controlled by a frequency penalization scheme. Chen and Wu (2006) also used the tabu search. The initial feasible solution is built by an insertion method, which relies on both distance- and load-based criteria. The neighborhood for the improvement phase is built on the moves 2-exchange, swap, shift, 2-opt and Or-opt. Wassan et al. (2008) presented a reactive tabu search framework which uses the general shift, swap operators together with a problem-specific move which reverses complete routes. The proposed dynamic control of the tabu list size achieves an effective balance between the intensification and diversification of the search. Zachariadis et al. (2009) proposed a hybrid meta-heuristic approach based on tabu search controlled by a guiding mechanism for diversifying the conducted search. Gajpal and Abad (2009a) proposed an Ant Colony System methodology which employs a construction rule as well as two multi-route local search schemes. Subramanian et al. (2010) presented a parallel algorithm which is embedded with a multi-start heuristic consisting of the Variable Neighborhood Descent integrated in an iterated local search framework. The main features of the proposed approach are the automatic calibration of some parameters and the ability of exploring the high-level of parallelism inherent to recent multi-core clusters.

- Each customer may be visited either once or twice, one for delivery, one for pickup, if beneficial (*VRP with Divisible Deliveries and Pickups* (VRPDDP)): As mentioned earlier, the VRPDDP is further classified along two distribution strategies: lasso solution strategy and general solution strategy.
  - Lasso solution strategy: Hoff et al. (2009) proposed a tabu search creating lasso solutions based on shift and swap neighborhoods. After each move, the path and loop parts of the current solution are re-optimized by 2.5-opt of Bentley (1992) and 2-opt of Lin (1965), respectively.
  - General solution strategy: Salhi and Nagy (1999) proposed an insertion-based heuristic in which linehaul customers are routed first. It then inserts a cluster of pickup customers into these linehaul routes rather than only one pickup customer at a time as was previously done by Golden et al. (1984). Nagy and Salhi (2005) proposed a local search heuristic that considers solutions with a certain degree of infeasibility. Based on the degree of infeasibility of the current solution, sequences of different improvement operators (2-opt, 3-opt, shift, exchange, etc.) are applied. Nagy et al. (2013) proposed a reactive tabu search which uses shift, swap operators together with a problem-specific move which splits customers that are currently served in a single visit into a delivery and pickup entity, and inserts whichever gives a better solution to the best possible position in another route.

2.4.2 *One-to-one and many-to-many schemes*

These schemes refer to problems of designing a set of least cost vehicle routes starting and ending at a common depot in order to transport goods from pickup to delivery points.

The one-to-one scheme considers transportation requests, each request originates at one location and is destined for one other pre-defined location. These requests apply to the transportation of goods (VRP with Pickups and Deliveries) or people (the Dial-a-Ride problem). The many-to-many scheme refers to situations where pickup and delivery points are unpaired, i.e., every good can be picked up and transported anywhere. The many-to-many scheme did not receive as much attention in the literature as the other schemes. Moreover, most of the literature is restricted to the Capacitated Pickup and Delivery Traveling Salesman Problem (CPDTSP) that consists of n pickup points and n delivery points, each of which provides or requests one unit of commodity, and one vehicle of limited capacity. The objective of the problem is to determine a minimum length feasible tour that picks up and delivers all loads and does not violate the vehicle capacity.

In this dissertation, we only deal with the 1-M-1 scheme. We therefore do not review the work published on 1-1 and M-M schemes. For more information on these two schemes, readers may see the surveys of Cordeau and Laporte (2007), Cordeau et al. (2008), Parragh et al. (2008b), Berbeglia et al. (2010).

2.5 *The Vehicle Routing problems with synchronization of vehicles*

In vehicle routing, synchronization of vehicles means to couple the route of two or more vehicles in time and space. The spatial dimension of synchronization defines the locations where vehicle synchronization can take place, while the temporal dimension defines the order in which vehicles must visit a synchronization point. Two of the most studied problems are multi-echelon VRP and VRP with cross-docking. Table 2.2 gives an overview of research on these problems for both time and space dimensions. The vehicles can be synchronized at a single location or multiple locations, while synchronization locations may be visited by vehicles either simultaneously or with a precedence order.

In multi-echelon VRP, delivery from one or several depots to customers is managed by routing and consolidating the freight through intermediate facilities which are called satellites. The most common version of multi-elechon VRP studied in the literature is the two-echelon VRP (2E-VRP). A general time-dependent formulation with fleet synchronization and customer time windows was introduced by Crainic et al. (2009) in the context

Table 2.2: Synchronization of vehicles

Problem type	Paper	Spatial synchronization		Temporal synchronization		Solution method
		Single	Multiple	Simultaneously	Precedence	
Multi-echelon VRP	Crainic et al. (2010)		x	x		Two-phase local search
	Crainic et al. (2011)		x	x		Multi-start local search
	Hemmelmayr et al. (2012)		x	x		GRASP, path relinking
	Crainic et al. (2013)		x	x		ALNS
VRP with cross-docking	Lee et al. (2006)	x		x		Tabu search
	Liao et al. (2010)	x		x		Tabu search
	Wen et al. (2008)	x			x	Tabu search, adaptive memory
	Tarantilis (2012)	x			x	Multi-start tabu search

of two-echelon City Logistics systems. In the 2E-VRP, there is only one depot considered, and a fixed number of capacitated satellites. All freight from the depot must transit through satellites, and then be delivered to the customers, i.e., direct shipping from the depot to customers is not allowed. There are therefore two levels of vehicles: 1st-level vehicles deliver freight from the depot to satellites, while 2nd-level vehicles start from satellites to load freight and deliver them to customers. All customer demands are fixed and known in advance and must be satisfied within the scheduling horizon. Each customer demand has to be satisfied by only one satellite, and customer-satellite assignments are not known in advance. To solve the 2E-VRP, the transportation is usually decomposed into two levels, with the upper one addressing the depot-to-satellites delivery, while the lower level building satellites-to-customers delivery routes. Satellites are thus called synchronization locations where freight is transferred from 1st-level vehicles to 2nd-level ones. The goal of the 2E-VRP is to service customers at minimum the total transportation cost at both levels, and satisfying the capacity constraints of the vehicles and satellites.

Crainic et al. (2010) proposed a two-phase heuristic based on a clustering first routing second procedure plus a classical local search procedure. They applied a separation strategy that splits the 2E-VRP problem into two major routing subproblems, one at each level. The second-level subproblem is further decomposed into as many VRPs as the number of satellites, assuming that the set of customers assigned to each satellite is known. The customer-to-satellite assignment problem is solved through a clustering-based heuristic procedure allocating customers to closer satellites. In the same way, the VRP at the first level involves a single depot and a set of satellites with each one featuring a demand equal to the sum of the demands of customers assigned to it. This heuristic was used to make a satellite location analysis in order to build instances up with to 250 customers, providing also a first sub-optimal solution as a reference for further methods for the 2E-VRP. Those results were improved by the same authors using multi-start strategy (Crainic et al. (2011)), and greedy randomized adaptive search procedure (GRASP) with path relinking (Crainic et al. (2013)). Crainic et al. (2011) is the extension of Crainic et al. (2010) by solving the resulting VRPs at the two levels iteratively, while adjusting satellite demands through customer-to-satellite reassignments. Using the same approach, Crainic et al. (2013) solved VRP subproblems by applying a GRASP and a local search procedure in sequence. Then, the resulting solution is linked to an elite solution by means of a path relinking procedure for further improvement. Crainic et al. (2013) produced an improvement of 8.7% with respect to Crainic et al. (2011). Hemmelmayr et al. (2012) developed an adaptive large neighborhood search (ALNS) heuristic. In their ALNS method, several different neighborhoods are applied and ranked according to their success in improving solutions. The highest ranked neighborhoods have a larger probability of being chosen. The proposed neighborhoods work on both levels, satellites and customers, by opening, closing, or swapping satellites; removing and reinserting customers. Their algorithm is capable of improving the best known solutions on several instances.

A closely related problem is the so-called VRP with cross-docking (VRPCD). The VRPCD involves transporting products from a set of suppliers (pickup nodes) to their customers (delivery nodes) via a single cross-dock. Products from the suppliers are picked up by a fleet of vehicles, consolidated at the cross-dock, and immediately delivered to customers by the same set of vehicles, without intermediate storage. Lee et al. (2006) were the first authors to take both VRP and cross-docking into consideration, assuming that all vehicles coming from suppliers arrive at the cross-dock simultaneously. Time windows at pickup and delivery nodes are not specified, but all demands must be satisfied within a given planning horizon. The objective is to determine the number of vehicles and the optimal vehicle routing schedule at the cross-dock to minimize the sum of transportation cost and fixed cost of vehicles. A mixed integer programming formulation and a tabu search were proposed. Their tabu search corresponds to solving two vehicle routing problems (one for pickup and one for delivery). The second routing problem can only start when the first one is finished. Liao et al. (2010) proposed another tabu search algorithm to solve the same problem. There are two major differences between the tabu search algorithms of Lee et al. (2006) and Liao et al. (2010). First, Liao et al. (2010) moved a customer from one route to another one at a time, whereas Lee et al. (2006) tried to exchange customers between two routes. Second, it was allowed to remove an empty vehicle in Liao et al. (2010), while it was not allowed in Lee et al. (2006). The tabu search of Liao et al. (2010) improved the solution quality by 20.5% when compared to Lee et al. (2006) with much less computation time.

A similar problem was studied by Wen et al. (2008), but in this case, each pickup and delivery have predetermined time windows, and there is no constraint on simultaneous arrival at the cross dock for all the vehicles at the cross-dock. Instead, the dependency among the vehicles is determined by the consolidation decisions. To solve the problem, a tabu search embedded within an adaptive memory procedure (AMP) was developed. In the AMP, a set of vehicles tours is stored in an adaptive memory. The initial solution is constructed by combining vehicle tours selected in the adaptive memory. In the neighborhood search phase, they used small neighborhood search and large neighborhood search. In the former, the algorithm tries to improve the solution by moving a small subset of nodes to a small subset of vehicles, while the whole solution space is explored in the latter (i.e., every node is moved to every position of every other vehicle). The algorithms starts with the small neighborhood, and switches to the large neighborhood if there is no improvement in the best solution after a number of iterations. When exploring the large neighborhood, if the best solution is updated within a number of iterations, the search process switches back to the small neighborhood, otherwise the algorithm stops. Tarantilis (2012) later proposed an adaptive multi-restart tabu search to improve the solution quality by up to 1.45% on average. Their proposed tabu search uses intra- and inter-route 2-Opt, 1-0 Relocate and 1-1 Exchange. At each iteration, all neighboring solutions involving pickup vehicle routes and delivery vehicle routes are evaluated. Only feasible moves are considered, and the oscillations among the neighborhood structures is random with equal selection probability.

A reference set consisting of good solutions found during the search process is used to provide a new working solution for the execution of the re-starting mechanism.

In summary, we may note that vehicles in the 2E-VRP are synchronized at multiple locations called satellites. Most algorithms addressing the problem in the literature did not consider the time windows for customers and satellite operations. The vehicles in the VRPCD are synchronized at a single location called cross-dock, time windows are defined for customers, but not defined for the cross-dock. In the last two problems we study in this dissertation, i.e., the TMZT-VRPTW and MZT-PDTWS, the operations of vehicles at one level are considered. The vehicles are synchronized at multiple locations called supply points to unload and/or load freight, time windows are defined for both customers and supply points. Each supply point has a no-wait, hard opening time window, specifying the earliest and latest times the vehicle must be available at it. *Waiting stations* (e.g., parking lots) are available for vehicles to wait in order to get to its next supply point just before the time window. Each vehicle can do multiple trips linking several supply points, where each trip serves a subset of customers while satisfying the time window at each customer.

2.6 Conclusion

In reviewing the literature, we see that VRPs have been the object of numerous studies and a very large number of papers proposed solution methods in which more and more powerful algorithms find increasingly better solutions to both real-world VRP instances and well-studied benchmark instances. However, research on rich VRP is still comparatively meager in relation to the body of literature accumulated for the basic variants of VRP.

This dissertation first studies a basic VRP variant with "rich" setting: the Periodic Vehicle Routing Problem with Time Windows (PVRPTW). Next, it tackles two new VRP variants which are encountered when planning the operations of two-tiered City Logistics systems, namely Time-dependent Multi-zone Multi-Trip Vehicle Routing Problem with Time Windows (TMZT-VRPTW), and Multi-zone Multi-Trip Pickup and Delivery Problem with Time Windows and Synchronization (MZT-PDTWS) respectively.

Chapter 3

A HYBRID GENERATIONAL GENETIC ALGORITHM FOR THE PERIODIC VEHICLE ROUTING PROBLEM WITH TIME WINDOWS

The article in this chapter has been accepted for publication in Journal of Heuristics: P. K. Nguyen, T. G. Crainic, and M. Toulouse. A hybrid generational genetic algorithm for the periodic vehicle routing problem with time windows. *Journal of Heuristics*, March 2014 (DOI 10.1007/s10732-014-9244-3).

Résumé: Nous proposons un algorithme génétique hybride à base de populations pour le problème de tournées de véhicules périodique avec fenêtres de temps. Cet algorithme hybride est une méta-heuristique qui combine un algorithme génétique avec deux méta-heuristiques de recherche à base de voisinages: la recherche tabou et la recherche à voisinage variable. Des méthodes de recherche locales ont été utilisées auparavant pour améliorer la qualité des solutions générées par les opérateurs de croisement des algorithmes génétiques. Dans notre travail, nous utilisons plutôt des méta-heuristiques de recherche à base de voisinages pour leur capacité à poursuivre leur recherche au-delà des optimum locaux, à générer des solutions de meilleures qualités et plus diversifiées, qui sont alors utilisées for construire la population de la génération suivante. De plus, la recherche exécutée par les méta-heuristiques de recherche à base de voisinages répare la plupart des violations de contraintes qui occurent naturellement après l'application des opérateurs de croisement. L'algorithme génétique que nous proposons introduit deux nouveaux opérateurs de croisement conçus pour le VRP périodique avec des contraintes de fenêtres de temps. Ces deux nouveaux opérateurs de croisement cherchent à utiliser le croisement des solutions pour améliorer la diversification de la recherche dans l'espace de solutions tout en préservant l'information à propos des routes qui ont déjà été calculées (le calcul des routes optimales est NP-difficile, nous cherchons donc à minimiser la destruction de cette information lors des croisements de solutions). Un grand nombre d'expérimentations ont été réalisées et de nombreuses comparaisons ont été conduites avec toutes les méthodes proposées dans la littérature, montrant que notre méthode est très compétitive, obtenant de meilleures solutions pour un nombre importants de problèmes de grande dimensionalité.

Abstract: We propose a new population-based hybrid meta-heuristic for the periodic vehicle routing problem with time windows. This meta-heuristic is a generational genetic algorithm that uses two neighborhood-based meta-heuristics to optimize offspring. Local search methods have previously been proposed to enhance the fitness of offspring generated by crossover operators. In the proposed method, neighborhood-based meta-heuristics are used for their capacity to escape local optima, and deliver optimized and diversified solutions to the population of the next generation. Furthermore, the search performed by the neighborhood-based meta-heuristics repairs most of the constraint violations that naturally occur after the application of the crossover operators. The genetic algorithm we propose introduces two new crossover operators addressing the periodic vehicle routing problem with time windows. The two crossover operators are seeking the diversification of the exploration in the solution space from solution recombination, while simultaneously preserving information about routes in the population as computing routes is NP-hard. Extensive numerical experiments and comparisons with all methods proposed in the literature show that the proposed methodology is highly competitive, providing new best solutions for a number of large instances.

Keywords: Periodic Vehicle Routing problem, time windows, hybrid generational genetic algorithm, meta-heuristics, tabu search, variable neighborhood search

3.1 Introduction

The Vehicle Routing Problem (VRP) is one of the most extensively studied problems in operations research due to its methodological interest and practical relevance to many fields, including transportation, logistics, telecommunications, and production; see, e.g., a number of recent surveys (Bräysy and Gendreau, 2005a,b; Cordeau et al., 2002a,b, 2007; El-Mihoub et al., 2006; Gendreau et al., 2002; Golden et al., 2002; Laporte and Semet, 2002; Laporte et al., 2000) and books (Golden et al., 2008; Toth and Vigo, 2002). Many of these contributions targeted basic problem settings such as the capacitated VRP and the Vehicle Routing Problem with Time Windows (VRPTW). More recent and significantly less studied are richer problem settings (Hartl et al., 2006) aiming at more refined representations of actual applications and combining several "complicating" requirements and restrictions, such as customers that require multiple visits, heterogeneous vehicle fleets, limits on route duration or length, etc.

We focus on such a rich, relatively little studied VRP setting, namely the *Periodic Vehicle Routing Problem with Time Windows* (*PVRPTW*). Addressing the PVRPTW requires the generation of a limited number of routes for each day of a given planning horizon, to minimize the total travel cost while satisfying the constraints on vehicle capacity, route duration, customer service time windows, and customer visit requirements. The PVRPTW generalizes the VRPTW by extending the planning horizon to several days where customers generally do not require delivery on every day in this period, but rather according to one of a limited number of possible combinations of visit days (the so-called *patterns*). This generalization extends the scope of applications to many commercial distribution activities such as waste collection, street sweeping, grocery distribution, mail delivery, etc. It also raises new resolution challenges due to the requirement of balancing aggregate daily workloads in order to achieve efficient feasible solutions. The PVRPTW is actually NP-hard as it includes the Periodic Vehicle Routing Problem (PVRP), known to be NP-hard, while the single-period case corresponds to the NP-hard VRPTW (Lenstra and Rinnooy Kan, 1981).

In this paper, we introduce a generational genetic algorithm (*GA*) for the PVRPTW. It is a population-based hybrid meta-heuristic in which a set of neighborhood-based meta-heuristics cooperate with the GA population evolution mechanism to enhance the solution quality. Therefore, a first contribution of this work is the hybridization, which uses local search procedures and neighborhood-based meta-heuristics as education strategies to repair individuals and enhance their fitness as well as to promote diversity of the GA population. Studies have been published where GAs are hybridized with local-search methods in order to improve their exploitation capability (e.g., El-Mihoub et al., 2006; Knowles and Corne, 2000; Ishibuchi and Narukawa, 2004; Vidal et al., 2013a). Local search may reduce the diversity of the population (Merz and Katayama, 2004), however, and therefore limit the exploration capability of the GA. A few studies have indicated that a more aggressive search with a neighborhood-based meta-heuristic starting from the crossover-generated offspring may improve the diversity makeup of educated offspring and the global performance of the GA in terms of solution quality (Crainic and Gendreau, 1999; Lü et al., 2010). Our concept of GA hybridization builds on these insights. In the hybrid algorithm we propose, offspring are thus first educated using neighborhood-based meta-heuristics, which extend the search along routes and patterns beyond the offspring's immediate local optimum. Then, more intensification-oriented local search procedures improve the educated offspring relative to its patterns and routes. This more extensive exploration of the offspring neighborhoods by meta-heuristics yields two additional benefits for the search: the first is the restoration of the feasibility of a very large percentage of the infeasible offspring produced by GA crossover operators; the second is the discovery of solutions with new customer-day assignments that once added to the current population increase substantially the capability of the algorithm to uncover new combinations of customer-day assignments through crossover operations.

A second contribution of this work is the design of a recombination operation for the PVRPTW. Each day in the planning horizon of PVRPTW requires the computation of a very good approximate solution to the NP-hard VRPTW routing problem using the above neighborhood-based heuristics. It would be too costly and wasteful to compute such good approximations for every offspring at each generation. Rather, we let the selection operator and individual fitness decide which routes are re-optimized through the application of the neighborhood-based heuristics on routes or route components that survive several generations. To achieve this objective, we need a recombination operation preserving as much as possible routes across generations while maintaining the exploration capabilities of generational genetic algorithms. We have implemented the genetic recombination operation using two crossover operators. The first crossover operator obtains new offspring by combining visit days belonging to two parents. This operator aims at exploring the domain of pattern assignments in the problem instance, thus favoring the exploration capabilities of the GA. However, after the application of this operator, many customers in the offspring are assigned to different days, leaving few significant routes or route components that can be further optimized in the next generations. To address this issue, a second offspring is generated by randomly selecting a parent for each day of the planning horizon and copying all the routes of that day from the selected parent. This second crossover operator preserves to a greater extent the existing routes of the parents which are further optimized by the above neighborhood-based heuristics in next generations.

This work is the first generational GA on PVRPTW (see Pirkwieser and Raidl, 2010; Vidal et al., 2013a, proposing steady state GAs). Typically, in a steady state GA (also called incremental GA), a single individual in the population is replaced by an offspring at every iteration of the method while in the generational GA the whole population is replaced. While generational GA is favored in general, this model certainly raises challenges than the steady state model if refined structures, such as highly optimized routes in PVRRPTW, are to be discovered and maintained across generations. We also note that PVRPTW has many constraints, offspring produced by crossover are likely to violate some of these constraints, which is an important and well known issue for generational GA. We believe the approaches proposed in this paper will be useful to address other problem settings raising similar issues.

We tested the algorithm we propose on all previously published benchmark instances and compare our results to all currently published results. The proposed *Hybrid Generational Genetic Algorithm* (*HGGA*) produces 9 new best-known solutions and finds 5 best-known solutions on the set of 20 PVRPTW instances of Cordeau et al. (2004), improving the solution quality by 0.05% on average in terms of best solution cost. We also improve all the 45 instances of Pirkwieser and Raidl (2009a), improving the average solution cost by 0.27%. We hope these encouraging results will contribute to stimulate more investigations into developing hybrid meta-heuristics for solving heavily constrained optimization problems.

The remainder of the paper is organized as follows. We define the problem and give a brief literature review in Section 3.2. The new HGGA and its components are introduced and discussed in Section 3.3. Section 3.4 is dedicated to the experimental results. Finally, Section 3.5 concludes the paper.

3.2 Problem definition and literature review

The PVRPTW is defined on a complete undirected graph $\mathscr{G} = (\mathscr{V}, \mathscr{E})$, where $\mathscr{V} = \{0, 1, \dots, n\}$ is the vertex set and $\mathscr{E} = \{(i, j) : i, j \in \mathscr{V}, i \neq j\}$ is the edge set. A distance (or travel time) c_{ij} is associated with every edge $(i, j) \in \mathscr{E}$. The depot vertex is indexed by 0. $\mathscr{V}_c = \mathscr{V} \backslash \{0\}$ is the set of customer vertices. Each vertex (customer) $i \in \mathscr{V}_c$ has a demand $q_i \geq 0$ on each day of its visit days over the planning horizon of $\mathscr{T}$ days, a service time $s_i \geq 0$, a time window $[e_i, l_i]$, where e_i is the earliest time service may begin and l_i is the latest time. Each customer requires a fixed number of visits f_i during the planning horizon. These f_i visits must be performed according to an allowable visit-day pattern, which is a subset of f_i days, for example days 1, 3 and 5 of a 7-day planning horizon. A customer may have several allowable visit-day patterns, they are part of the definition of a problem instance and stored in a list $\mathscr{R}_i$. The time window specifying the interval vehicles leave and return to the depot is given by $[e_0, l_0]$. A fleet of m vehicles, each with capacity Q_k, is based at the depot. Vehicles are grouped into set $\mathscr{K}$. Vehicle routes are restricted to a maximum duration of D_k, $k = 1, \ldots, m$.

In this paper, we address the case with a homogeneous vehicle fleet, $Q_k = Q$, and a common duration restriction $D_k = D$, $\forall k = 1, \ldots, m$. The PVRPTW can then be seen as the problem of generating (at most) m vehicle routes for each day of the planning horizon, to minimize the total cost over the entire planning horizon, such that 1) each vertex i is visited the required number of times, f_i, corresponding to a single pattern of visit days chosen from R_i, and is serviced within its time window; these are hard, i.e., a vehicle may arrive before e_i and wait to begin service; 2) each route starts from the depot, visits the vertices selected for that day, with a total demand not exceeding Q, and returns to the depot after a duration (travel time) not exceeding D.

The complexity class of PVRPTW calls for heuristic solution approaches when realistically-sized instances are contemplated. Cordeau et al. (2001, 2004) introduced the problem setting and pioneered the application of heuristics by proposing a tabu search algorithm, which allows infeasible solutions together with associated penalty terms in the objective function for violations of time windows, route duration, and vehicle capacity constraints. Moves either relocate a customer to a different route in the same day or change its pattern, which provides two ways to improve solutions, by modifying the routing and the visit pattern assignments to customers. The authors also introduced a set of 20 benchmark PVRPTW instances.

Pirkwieser and Raidl (2008) proposed a Variable Neighborhood Search (VNS) heuristic for the PVRPTW, with the particularity that it accepts worsening solutions based on a Metropolis criterion. Pirkwieser and Raidl (2009a) later introduced a hybrid scheme between this VNS heuristic and an ILP-based column generation procedure addressing a set-covering formulation. In this hybrid, the VNS is the sole provider of columns for the set-covering, which is solved via a generic ILP solver. If the latter improves on the current best solution, this new solution is transferred to the VNS for further enhancement. Since ILP solvers cannot tackle large instances, validation relied on a new set of smaller instances derived from the basic Solomon VRPTW 100-customer instances. The authors then proposed a hybrid between an evolutionary algorithm and the column generation approach (Pirkwieser and Raidl, 2010), as well as a rigid synchronous cooperative multi-search approach, named multiple VNS (mVNS) (Pirkwieser and Raidl, 2009b). In the latter setting, several VNS meta-heuristics run independently, synchronize after a given number of iterations to determine the best solution, the worst VNS thread being then restarted from this best solution, while the others continue their own search. The mVNS was also hybridized with the column generation approach as per Pirkwieser and Raidl (2009a). At a synchronization point, the best mVNS solution is passed to the ILP solver. If the resulting solution improves the mVNS best solution, the worst VNS search is initialized with it and the mVNS is restarted. This mVNS-ILP combination is repeated a fixed number of times. The mVNS-ILP hybrid generally produced better results than mVNS, without dominating over the entire instance set.

Cordeau and Maischberger (2012) proposed a parallel iterated tabu search heuristic which belongs to the pC/KS/MPDS category introduced by Crainic et al. (2005). This heuristic embeds a tabu search within the framework of iterated local search as the improvement phase, yielding an 'iterated tabu search'. Each 'iterated tabu search' thread starts from different solutions using different parameters, and these threads communicate through a central memory for exchanging their working solutions. The authors used up to 64 threads in the parallel variant.

Vidal et al. (2013a) proposed a hybrid genetic search which combines the exploration capabilities of genetic algorithms, the search intensification of local search-based improvement procedures, and diversity management mechanisms based on a generalized fitness function combining solution quality and diversity. To further improve solution quality, the authors also applied a decomposition phase in which pattern assigned to each customer was fixed, the resulting VRPTW subproblem for each period being then solved separately by their hybrid genetic algorithm.

Overall, the current best published results on the 20 instances of Cordeau et al. (2004) are reported by Vidal et al. (2013a). Only the latter authors and Pirkwieser and Raidl published results for the set of smaller instances introduced in Pirkwieser and Raidl (2009b). Pirkwieser and Raidl (2009a,b, 2010) reported only average solution costs and no best solution costs.

3.3 The Proposed Hybrid Meta-heuristic

This section is dedicated to introducing the *Hybrid Generational Genetic Algorithm* (*HGGA*) we propose. We start by describing the individual representation and evaluation procedure (Sections 3.3.1 and 3.3.2, respectively). The main phases of the HGGA, illustrated in Figure 3.1, are typical of generational genetic algorithms and are introduced next. The algorithm uses one population only, which may contain both feasible and infeasible individuals. The initial population is created using three greedy heuristics (Section 3.3.3). A new population is generated from the current one through the selection, crossover, mutation, education, and replacement phases of the algorithm (Sections 3.3.4 to 3.3.7, respectively). The population is always ordered in increasing order of fitness. Our contributions are both in adapting GA operators to the particular requirements of the PVRPTW and in designing the overall organization of the hybrid algorithm to answer the challenges of this problem setting.

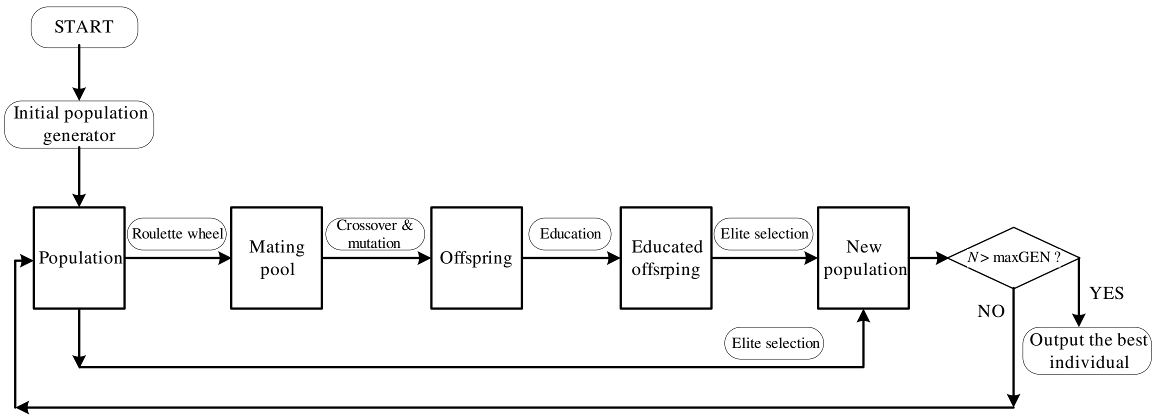


Figure 3.1: The Hybrid Generational Genetic Algorithm Structure

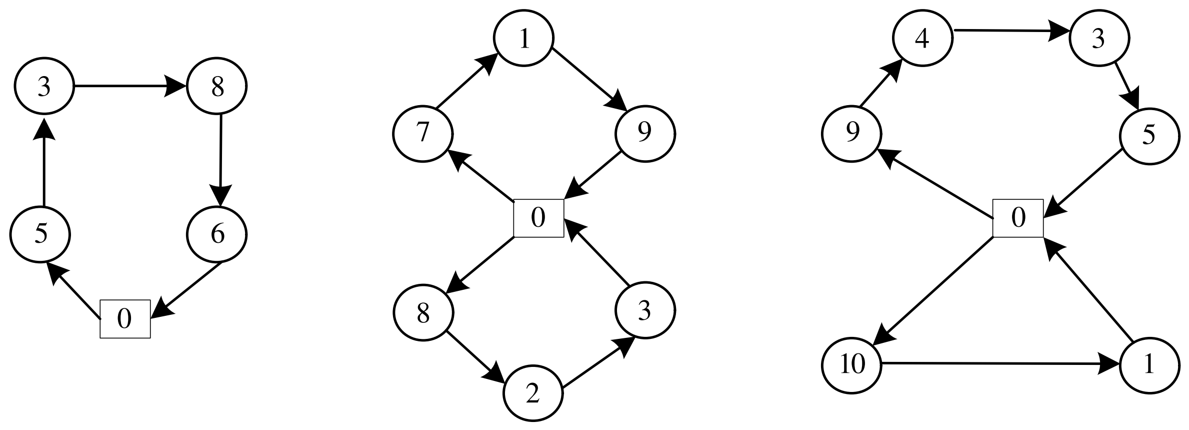
3.3.1 Individual representation

An individual for HGGA corresponds to a feasible or infeasible solution to the PVRPTW specifying the pattern assigned to each customer, the number of routes (that is, vehicles), and the delivery order within each route. The relevant characteristics of each individual are encoded into two chromosomes, the **Pattern** chromosome representing pattern-to-customer assignments and the **Route** chromosome which encodes the routes for each day of the planning horizon.

The Pattern chromosome is a binary vector of $n \times \mathscr{T}$ bits. Starting from the left, the i^{th} sequence of $\mathscr{T}$ bits encodes the pattern assigned to the corresponding customer i. Figure 3.2a illustrates the Pattern chromosome of a solution with 10 customers and 3 days. In this illustration, the first customer is serviced according to pattern [110], i.e., on days 2 and 3 (days are numbered from right to left).

110	010	111	100	101	001	010	011	110	100

(a) Pattern chromosome



Day 1:

Route 1: 0, 5, 3, 8, 6, 0

Day 2:

Route 1: 0, 7, 1, 9, 0
Route 2: 0, 8, 2, 3, 0

Day 3:

Route 1: 0, 10, 1, 0
Route 2: 0, 9, 4, 3, 5, 0

(b) Sequences of customers that are served on 3 days

5	3	8	6	11	7	1	9	12	8	2	3	12	10	1	13	9	4	3	5	13

(c) Route chromosome

Figure 3.2: Representation of an individual.

For each day in the planning horizon, a group of routes services customers on that day. A route is an ordered sequence of customers on one day. The Route chromosome is the concatenation of all the current sequences in the planning horizon. A number larger than the number of customers is added at the end of each route in the Route chromosome for the decoding of the routes and days. This number is different for each day. Figure 3.2b is a set of routes for the visit-day patterns in the Pattern chromosome of Figure 3.2a, while Figure 3.2c displays the corresponding Route chromosome.

3.3.2 Search space and individual evaluation

Allowing meta-heuristics to consider infeasible solutions often yields a better search able to reach higher-quality solutions more efficiently (e.g., Cordeau et al., 2004). We follow this trend and explicitly allow infeasible solutions during the search process by relaxing constraints on the maximum vehicle load, route duration, and customer service time windows.

Given a solution s, let $c(s)$ denote the total travel cost of its routes, and let $q(s)$, $d(s)$, and $w(s)$ denote the total violation of capacity, duration, and time window restrictions, respectively. The values of $q(s)$ and $d(s)$ are computed on a route basis with respect to the Q and D values, whereas $w(s) = \sum_{i=1}^{n} max\{(a_i - l_i), 0\}$, where a_i is the arrival time at customer i. Solutions are then evaluated according to the weighted *fitness function* $f(s) = c(s) + \alpha q(s) + \beta d(s) + \gamma w(s)$, where α, β, and γ are penalty parameters adjusted dynamically during the search.

Several techniques are available to adjust the penalty parameters. For example, Cordeau et al. (2004) based their update on the current solution. We prefer to follow Barbosa and Lemonge (2002), which makes use of information related to the complete population. Let $\overline{q}$, $\overline{d}$, and $\overline{w}$ stand for the violation of vehicle capacity, route duration, and customer service time window constraints, respectively, averaged over the current population.

Let

$$h = \begin{cases} c(s_{worst}) & \text{if there is no feasible solution in the population;} \\ c(s_{bestfeasible}) & \text{otherwise.} \end{cases}$$

The penalty parameters are then computed by the following rules:

$$\alpha = h\frac{\overline{q}}{\overline{q}^2 + \overline{d}^2 + \overline{w}^2}\ , \quad \beta = h\frac{\overline{d}}{\overline{q}^2 + \overline{d}^2 + \overline{w}^2}\ , \quad \gamma = h\frac{\overline{w}}{\overline{q}^2 + \overline{d}^2 + \overline{w}^2}\ .$$

Every time the current best feasible solution is improved, h is redefined, all fitness values are recomputed using the updated penalty coefficients, and the population is sorted accordingly. This adaptive scheme automatically determines the penalty parameter corresponding to each group of constraints during the evolutionary process so that the constraints that are more difficult to satisfy receive a relatively higher penalty coefficient.

3.3.3 Initial population

Solutions in the initial population are generated by, first, assigning randomly an allowable pattern of visit days to each customer and, second, by solving a VRPTW for each day using three greedy heuristics: 1) the Time-Oriented, Nearest-Neighbor heuristic of Solomon (1987); 2) the parallel route building heuristic of Potvin and Rousseau (1993); 3) our own route construction method. We use the methods of Solomon (1987) and of Potvin and Rousseau (1993) because these heuristics are very fast, and they appear to be complementary. Indeed, comparing the two, the former seems to perform better for clustered problem instances, while the opposite is true for the other problem settings (Bräysy and Gendreau, 2005a). Moreover, applying three different heuristics helps create diversity within the initial population.

Our own route construction method is quite flexible with regard to problems with a fixed number of vehicles. It follows the cluster first - route second scheme. During clustering, customers are first sorted in increasing order of the angle they make with the depot. Next, a customer j is chosen randomly and the sequence of n customers $j, j+1, \ldots, n, 1, \ldots, j-1$ is divided into m clusters of size $\lceil n/m \rceil$ (the last one may be smaller), one for each available vehicle. Clustering is performed by a sweep starting from j and proceeding counterclockwise through the customers.

Routing is performed iteratively for each cluster using Solomon's insertion heuristic of type I1. For added flexibility in routing, the procedure considers not only the customers in the cluster, but also a small number of customers not yet serviced by a route, selected from the two immediate neighboring clusters. Each route is initialized with the customer not yet assigned to a route displaying the lowest starting time for service. The remaining not-yet-assigned customers are then added sequentially to the route until it is full with respect to vehicle-capacity and route-duration constraints. The customers not yet assigned once the m routes are created, if any, are then inserted into the existing routes to minimize the increase in the total travel distance. The algorithm stops when all customers are serviced.

3.3.4 Mating selection

The selection operator chooses individuals within the population for mating purposes. At each generation of HGGA, a *mating pool* of *nPop* individuals is formed using a rank-based Roulette-wheel selection operator. This operator first sorts individuals in the population according to their fitness and then computes selection probabilities according to their ranks rather than fitness value. The rank for an individual is scaled linearly using the following formula:

$$Rank(Pos) = 2 - SP + 2(SP - 1)\frac{N - Pos}{N - 1}$$

where N is the number of individuals in the population, Pos is the position of an individual in this population (the best individual has $Pos = 1$, the worst individual $Pos = N$) and $SP \in [1.0, 2.0]$ is the selective pressure.

An individual may be selected more than once. Then, each time offspring are required during the course of the HGGA, two individuals in the mating pool are selected randomly and passed to the crossover operators. These two individuals are then deleted from the mating pool.

3.3.5 Crossover and mutation operators

A good solution to the PVRPTW involves a successful assignment of visit-day patterns to customers as well as strongly optimized solutions to the $\mathscr{T}$ NP-hard routing problems in a $\mathscr{T}$-day planning horizon. To get an efficient GA, diversified visit-day patterns must be sampled at every generation, requiring that new routing problems be solved, usually yielding under-optimized routes when time to solve these problems is limited. In turn, poorly solved routing problems increase the cost of otherwise potentially successful visit-day assignments, preventing the search from identifying truly good solutions. It is against this backdrop that we have designed the recombination phase of the genetic algorithm we propose. We introduce two crossover operators, one aiming at exploring different visit-day assignments, *the exploration crossover operator*, while the other, *the exploitation crossover operator* combines existing routes from different days. Applied to the same pair of parents and yielding two offspring, the former operator creates new visit-day assignments, sometime destroying useful information contained in the parents, while the latter helps intensify the search by preserving routes or route components present in the parents, which are iteratively optimized in the next generations. Together, these two crossover operators balance exploration and exploitation, thus improving the search efficiency of the algorithm.

In the following presentation, we use P_1 = {Day 1: (Route 1: 5, 3, 8, 6); Day 2: (Route 1: 7, 1, 9) and (Route 2: 8, 2, 3); Day 3: (Route 1: 10, 1) and (Route 2: 9, 4, 3, 5)}, and P_2 = {Day 1: (Route 1: 9, 5, 3, 4, 6) and (Route 2: 7, 1, 2, 10); Day 2: (Route 1: 8, 5, 3); Day 3: (Route 1: 1, 9, 3, 8)} as examples of parents involved in a crossover operation. The Pattern and Route chromosomes of P_1 and P_2 are shown in Figure 3.3.

110	010	111	100	101	001	010	011	110	100

(a) Pattern chromosome of parent P_1

5	3	8	6	11	7	1	9	12	8	2	3	12	10	1	13	9	4	3	5	13

(b) Route chromosome of parent P_1

101	001	111	001	011	001	001	110	101	001

(c) Pattern chromosome of parent P_2

9	5	3	4	6	11	7	1	2	10	11	8	5	3	12	1	9	3	8	13

(d) Route chromosome of parent P_2

Figure 3.3: Pattern and Route chromosomes for parents P_1 and P_2.

The first crossover operator creates offspring by transferring a partial set of routes from one parent, along with pattern assignments from both parents. The exploration crossover operator proceeds in two steps:

STEP 1. Assign a pattern to each customer

(a) *Inherit pattern assignments from parent P_1*. Randomly select two cutting points in the sequence of customers of the Route chromosome of parent P_1. The visit days of customers between these cutting points are copied into the Pattern chromosome of offspring C_1. For example, assume entries 8 and 14 of the Route chromosome of Figure 3.3b are the cutting points. Customers 9, 8, 2, 3 and 10 appear between the cutting points. Customers 9, 8, 2 and 3 are visited on the second day in P_1, therefore the second position of the visit-day pattern for these customers in the Pattern chromosome of offspring C_1 will inherit this visit day from P_1 as shown in Figure 3.4a. Similarly, as customer 10 found between the cutting points of P_1 is visited on the third day, the third position of the visit-day pattern of customer 10 in the Pattern chromosome of offspring C_1 is set to 1 as well. All the other positions of all the visit-day patterns in the Pattern chromosome of offspring C_1 are set to 0.

000	010	010	000	000	000	000	010	010	100

(a) After Step 1a

101	010	111	001	011	001	001	110	010	100

(b) After Step 1b

101	010	111	001	011	001	001	110	011	100

(c) After Step 1c

Figure 3.4: Example of pattern chromosome of offspring C_1.

(b) *Inherit pattern assignments from parent P_2*. Let $day(i)$ denote the visit-day pattern of customer i in offspring C_1 after step a). Scan the Pattern chromosome of parent P_2 from left to right. For each customer i such that the number of visit days in $day(i)$ is smaller than f_i, if customer i is visited on day t in P_2, customer i inherits day t in the Pattern chromosome of offspring C_1 if $day(i) \cup \{t\}$ is a sub-pattern of the visit-day patterns in R_i. For example, assume R_9 contains visit-day patterns [110] (visit day 2 and day 3), [101] (visit day 1 and day 3) and [011] (visit day 1 and 2). Customer 9 has $day(9) = [010]$ (assigned a visit at day 2) after step a) above. If in parent P_2, customer 9 has pattern [101], then the visit-day pattern $day(9)$ cannot be completed in step b). Otherwise, customer 9 will inherit day 3 (or day 1) from the visit-day pattern [110] (or [011]) of parent P_2, which will make $day(9)$ equal to one of the visit-day patterns in R_9. Figure 3.4b illustrates the state of the Pattern chromosome of offspring C_1 once step b) is completed.

(c) *Complete pattern assignments*. Assign a random pattern $r \in R_i$, such that r includes $day(i)$, to each customer i whose frequency is not satisfied. For example, after step b) above, there is only customer 9 whose frequency is not yet satisfied. Since $day(9)$ =[010] (assigned a visit at day 2) after step b), there are two patterns of R_9: [011] and [110] which have $day(9)$ as sub-pattern. Thus, one of these two patterns will be selected randomly, and assigned as pattern of customer 9. Then the Pattern chromosome of offspring C_1 is completed as shown in Figure 3.4c.

STEP 2. Assign customers to routes

(a) *Copy routes from parent P_1*. The customers between the two cutting points determined in Step 1a are routed as in parent P_1, i.e. the corresponding sequences are copied into the Route chromosome of C_1. If a route is cut by a cutting point, such as route 1 in day 2 of Figure 3.2b, the route that is copied in C_1 contains only the customers inside the cutting points, only customer 9 in this example.

(b) *Assign remaining customers to routes*. The customers not yet routed, customers before the first cutting point and after the second cutting point in P_1, are assigned to routes in C_1 using the cost insertion procedure of Potvin and Rousseau (1993). These customers may be inserted in routes copied from parent P_1 if the procedure of Potvin and Rousseau (1993) determines that this is the best insertion.

The exploration crossover operator is effective in changing visit-day assignments to customers. However, it disassembles potentially optimized routes contained in the two parents, and create new routes in the offspring which are likely not highly optimized.

The exploitation crossover operator processes each day t individually in the planning horizon, where it selects randomly one parent among $\{P_1, P_2\}$. All routes of the selected parent in day t are copied into the Route chromosome of offspring C_2. Then, customers are removed/inserted from/into days such that the pattern of visit days in C_2 are satisfied for all customers. If there is more than one possibility to delete a customer, the one that minimizes cost is deleted. Customers are inserted using the cost insertion procedure of Potvin and Rousseau (1993).

For example, assume day 2 is selected from parent P_1, while days 1 and 3 are selected from parent P_2. Then, all routes of parent P_1 from day 2, and all routes of parent P_2 from days 1 and 3 are copied into the offspring C_2, making C_2 = {Day 1: (Route 1: 9, 5, 3, 4, 6) and (Route 2: 7, 1, 2, 10); Day 2: (Route 1: 0, 7, 1, 9, 0) and (Route 2: 0, 8, 2, 3, 0); Day 3: (Route 1: 1, 9, 3, 8)}. Each customer from 1 to 10 is then sequentially examined for satisfying its visit-day pattern. For example, customer 1 is currently serviced in day 1, 2 and 3 in offspring C_2. Assume R_1 contains visit-day patterns [110] (visit day 2 and day 3), and [101] (visit day 1 and day 3). Therefore, customer 1 should be removed either from day 1 to satisfy the pattern [110] or from day 2 to satisfy the pattern [101]. If the former case gives a higher gain compared to the latter case, customer 1 is removed from day 1. Otherwise, customer 1 is removed from day 2.

The exploitation operator maintains some of the pre-existing routes in parents P_1 and P_2 that might have been disassembled by the exploration operator. Pre-existing routes can be furthermore optimized during the education phase of HGGA in future generations. In turn, well optimized routes yield more accurate approximations of the cost of solutions produced by different pattern assignments.

The mutation operator is applied to each offspring yielded by the crossover operators. Mutation consists in changing the pattern assignment of a few customers, which are selected through a low probability P_m. A new pattern is then assigned to each selected customer i: Either an eligible pattern (i.e., in R_i) not assigned to i in any of the individuals of the current population, if such a pattern exists, or one not assigned to i in the current offspring.

3.3.6 Education

Crossover and mutation operators yield offspring, which may be feasible or infeasible, but is "sent to school" in all cases. The goal of the *Education* procedure is to improve the quality of the offspring, as well as to restore the feasibility as much as possible (when needed).

Two issues must be addressed in this context. First, while GAs proved their worth in exploring broad and complex search spaces, they also appear less well suited for finetuning solutions that are near local optima (Gendreau and Potvin, 2005; García-Martínez and Lozano, 2008). Hybridization with local-search methods has been proposed to address this issue, but this strategy often degrades the diversity of the population (Merz and Katayama, 2004), and therefore limits the capability of the GA to find successions of improvements through its generational process. The second issue concerns the fact that crossover and mutation operators often yield offspring that violates some of the problem requirements and a repair phase is required. Local search has also been proposed to address this issue, but it might not be sufficient for heavily constrained optimization problems like PVRPTW, as our initial experiments have shown.

We therefore propose an education procedure based on a different principle, one which embeds neighborhood-based meta-heuristics into the GA to both maintain its exploring ability and restore the feasibility of offspring before its insertion in the population. This principle follows the insight of previous work on cooperative search methods (Crainic and Gendreau, 1999) and has been recently reinforced by the results of Lü et al. (2010). Compared with local search procedures, a higher proportion of offspring have their feasibility restored by meta-heuristics and the educated offspring have a higher average fitness.

The proposed education procedure integrates two meta-heuristics, the *Unified Tabu Search* (*UTS*) of Cordeau et al. (2004) and the *Random Variable Neighborhood Search* (*RVNS*) of Pirkwieser and Raidl (2008). We selected these meta-heuristics for several reasons. On one hand, they were applied to the PVRPTW and, thus, one obtains not only a basis for comparisons, but also known behavior and performance for the problem of interest. On the other hand, while both can contribute to routing and pattern-assignment improvements by changing the patterns assigned to customers and the routes of each day, the way moves are selected, evaluated, and performed is particular to each meta-heuristic. UTS selects between a routing or a pattern move based on the maximization of the cost improvement, while RVNS selects randomly at each iteration whether to execute a pattern or a route move. Combining the two yields HGGA, which produces more diversified individuals across generations.

The pseudo code of Algorithm 1, where CNG stands for the current number of generations, gives the general structure of the education procedure: Offspring are first educated through the neighborhood-based meta-heuristics and, then, intensification-oriented local search further enhances the educated offspring in their pattern assignment and routing dimensions. To save computation time and improve the diversity of the GA, UTS and RVNS are applied alternately every generation. A rather small number of iterations is allowed to each meta-heuristic. This may impair the performance of RVNS by "clashing" with its random move-selection characteristic, which helps to explore new points in the search space but yields poor solutions if the meta-heuristic is interrupted too soon. Consequently, a local search *Pattern-Improvement* procedure is applied to further improve the solutions. Finally, a *Route-Improvement* procedure locally re-optimizes the routes for each day separately.

Algorithm 1 EducationProcedure(solution s, CNG)

```
1: if CNG is even then
2:     UTS($s$)
3: else
4:     RVNS($s$)
5:     Pattern_improvement($s$)
6: end if
7: Route_improvement($s$)
8: return $s$
```

The *Pattern-Improvement* procedure proceeds by assigning a new pattern to each customer, keeping those that actually improve the solution. Customers are handled in random order. Then, for each customer i and each of its unassigned patterns $r' \in R_i$ (if any), i is removed from its current routes and the cheapest fitness insertion is performed to insert i into routes of corresponding days of the new pattern r'. A 2-opt heuristic is then applied to each route changed by this reassignment. If the new solution improves over the current one, a 2-opt* heuristic is applied for further improvement of the new current solution. One then proceeds to the next pattern or, if all have been tried out, to the next customer.

Route improvement is the last education activity and is performed by applying a number of well-known local search route improvement techniques. Two are intra-route operators, the 2-opt of Lin (1965) and the Or-opt of Or (1976). The others are inter-route operators, the λ-interchange of Osman (1993), the 2-opt* of Potvin and Rousseau (1995), and the CROSS-exchange of Taillard et al. (1997). For the λ-interchange, we only consider the cases where $\lambda = 1$ and $\lambda = 2$ corresponding to the (1,0), (1,1), (2,0), (2,1), and (2,2)-interchange operators.

The procedure starts by applying in random order the five λ-interchange and the 2-opt* and CROSS-exchange inter-route operators. Each neighborhood is searched on all possible pairs of routes (in random order) of the same day and stopped on the first improvement. The solution is then modified and the process is repeated until no further improvement can be found. The search is then continued by locally improving each route of the current day in turn. The intra-route 2-opt and Or-opt neighborhoods are sequentially and repeatedly applied until no more improvement is found.

3.3.7 Generation replacement and HGGA general structure

Once the mating pool is emptied, the generational change takes place. The goal is to produce a new generation that conserves the best characteristics of the individuals encountered so far and that displays a good variety of genetic material. An elitist approach is thus used, keeping some of the parents and all children which have different pattern assignments, while also welcoming individuals with unused pattern assignments generated by the education procedure.

HGGA evolves a population in which each individual has a different pattern assignment. Recall (Section 3.3.4) that the mating pool is populated with *nPop* individuals selected from the current population (the same individual may appear more than once according to its fitness) and that *nPop* offspring are created through crossover, mutation and education. The next generation is then composed of these *nPop* new individuals, the best *nKeep* individuals in the current population ($nKeep < nPop$), plus feasible solutions found by the education procedure that have a pattern assignment not yet seen in the population. Finally, for those individuals that share the same pattern assignment, only the best one is kept in the next generation. Algorithm 2 summarizes the hybrid meta-heuristic we propose for the PVRPTW.

As in the mating pool individuals may appear more than once, the diversity of the

Algorithm 2 Hybrid Generational Genetic Algorithm

```
1:  Randomly generate an initial population of $nPop$ individuals
2:  repeat
3:      Evaluate the fitness for each individual in the population
4:      Create the mating pool of $nPop$ size using rank-based Roulette wheel
5:      while the mating pool is not empty do
6:          Select two parents at random from the mating pool
7:          Apply crossover operators to produce two offspring
8:          Apply mutation operator to each offspring
9:          Apply education procedure to each offspring
10:         Delete both selected parents from the mating pool
11:     end while
12:     Generation replacement
13: until maximum number of generations $> maxGEN$
14: Print the current best solution
```

pattern assignments in the mating pool is smaller than in the initial population. This phenomena will be accentuated if selection schemes with higher selection pressure were used. Therefore, among the *nPop* offspring obtained from the crossover and education phases, it is likely that some will be rejected because they have same pattern assignments. In HGGA, rejected offspring can easily be replaced by individuals generated by the execution of the UTS and RVNS procedures during the education phase. These procedures execute between 50 and 800 local transformations on solutions generated by crossover, many of which are pattern assignment transformations. Each new feasible solution following a transformation is stored, yielding plenty of solutions to fill the population. Nonetheless, in the eventuality that local transformation procedures are not used, some approaches should be considered to obtain the needed individuals such as mutation operations on offspring with repeated pattern assignments or by generating random individuals which do not repeat existing pattern assignments.

3.4 Computational Analysis

The objective of the numerical experiments is twofold. First, to study a number of variants of the main algorithmic components and strategies and thus develop insights into addressing the challenges of designing hybrid meta-heuristics for tightly constrained combinatorial optimization problems (Section 3.4.1). The second objective consists in evaluating the performance of the proposed HGGA through comparisons with published results (Section 3.4.2).

HGGA is implemented in C++. Experiments were run on an Intel Xeon 2.8 GHz, 16 GB RAM. Two sets of instances were used throughout the experiments. The first, introduced by Cordeau et al. (2004), is made up of two sets (identified as *a* and *b*) of ten Euclidean instances each, ranging from 48 to 288 customers, 3 to 20 homogeneous vehicles, and with a planning horizon of 4 or 6 days. Instances *a* and *b* have narrow and large time windows, respectively. The depot has a [0,1000] time window in all instances. The second set of instances was generated by Pirkwieser and Raidl (2009a) and is made up of three sets of fifteen instances each. These Euclidean instances were created based on the Solomon VRPTW 100-customer instance set with a planning horizon of four, six, and eight days, denoted *p4*, *p6*, and *p8*, respectively.

3.4.1 Analysis of design decisions

A hybrid meta-heuristic like HGGA is always the result of a number of decisions on the structure, components, and parameter values of the method. Three main design components are studied through their impact on the behavior of the proposed HGGA: the education procedure (Sections 3.4.1.1 and 3.4.1.2), the selection and replacement strategies (Section 3.4.1.3), and the crossover operators (Section 3.4.1.4). The calibration of the search parameters is presented in Section 3.4.1.5.

3.4.1.1 Variants of the education procedure

The first experiment examines the contribution of the education procedure on search performance. We consider the execution of the GA with and without the inclusion of the education procedure phase. We also evaluate the impact of each neighborhood-based meta-heuristic on the GA performance in order to determine an appropriate hybridization scheme.

To determine the hybridization scheme, we have implemented the UTS and RVNS meta-heuristics following the descriptions given in Cordeau et al. (2004) and Pirkwieser and Raidl (2008), respectively (Annex A.1 details the algorithms), and set up four hybrid algorithms following the general structure of Algorithm 2. The first two, *HGGA-UTS* and *HGGA-VNS*, embed one method only, UTS or RVNS being called at each iteration, respectively. The third, *HGGA-UTS/VNS*, combines the two, UTS and RVNS being used alternately at every generation. The fourth, *HGGA*, enhances the HGGA-UTS/VNS structure by performing the pattern improvement procedure following the RVNS execution.

To level the comparison field, the number of iterations of UTS (200) and RVNS (1200) was set to yield comparable computing times. Similarly, the number of iterations of RVNS was reduced for HGGA such that the total computing effort for RVNS and pattern improvement be approximately the same as that of RVNS in HGGA-UTS/VNS. Experiments were conducted on the Cordeau et al. (2004) instances.

Table 3.1: Aggregated performance comparison between education schemes on Cordeau et al. (2004) instances

	HGGA-notEdu	*HGGA-UTS*	*HGGA-VNS*	*HGGA-UTS/VNS*	*HGGA*
Time (hours)	0.35	5.22	5.77	5.20	5.18
GAP	+8.84%	+1.07%	+1.48%	+0.34%	-0.05%

Table 3.1 displays the aggregated results of this experiment in terms of average CPU time and gap to the best known solution (BKS). The last four columns display results for GA with the education phase under the different hybridization schemes while the second column (*HGGA-notEdu*) displays the performance obtained without an education phase. As expected, we can observe from the second column that the education procedure contributes significantly to improve the performance of the algorithm. The computational time is significantly less compared to the hybridized versions, but the average gap to the BKS of +8.84% is the highest compared to all the variants using the education phase.

In Table 3.1, we also observe that hybridization based on UTS or RVNS impacts the method differently, the aggregate performances being too close to discriminate (slight advantage to HGGA-UTS). Mainly, this is explained by how the move types are selected by each meta-heuristic, UTS selects between routing or pattern modification moves based on cost improvement while RVNS makes a random selection satisfying the Metropolis selection criteria. The results also showed that both hybridization schemes improved in solution quality upon UTS but not relative to RVNS, and the situation did not change with increased education effort (e.g., doubling the number of UTS and RVNS iterations in the hybrids). Combining the two neighborhood-based meta-heuristics into the same hybrid algorithm produced the desired result, HGGA-UTS/VNS outperforming HGGA-UTS and HGGA-VNS (even when the education effort was reduced, e.g., running 100 and 1000 iterations of UTS and VNS, respectively). The hybrid also improved the BKS for small instances. The best performance, both in solution quality and computation time, is offered by the HGGA hybrid, however, which adds the pattern-improvement post-optimization procedure to RVNS.

3.4.1.2 Capacity of UTS and RVNS to repair offspring

Based on the constraint violations penalty system described in Section 3.3.2, infeasible individuals with low constraint violation may get good objective function values and be competitive in terms of fitness selection. This is desirable as the solution of these individuals, beside few violated constraints, may have components that facilitate finding improved feasible individuals. However, keeping too many infeasible solutions may lead to a prolonged processing time to synthesize the population. In the next experiment, we show that UTS and RVNS are very efficient in repairing the offspring generated by the crossover operators. This experiment measures the percentage of feasible individuals among the new offspring and in the overall population during the evolutionary process.

Table 3.2 displays the results for all small instances (less than 100 customers) and large instances (100 or more customers). The percentage values in each row of the Range columns represent the smallest and the largest percentages of feasible individuals obtained among all the generations. The first row is before education and the second row is after

Table 3.2: Percentage of presence of feasible individuals during the evolution

	Small instances		Large instances	
	Range	Mean	Range	Mean
Offspring before education	29% - 82%	66%	26% - 77%	62%
Offspring after education	70% - 100%	91%	60% - 100%	90%
Individuals in population	61% - 100%	96%	57% - 100%	93%

education, both rows apply to offspring only. The third row is after education considering the percentages of feasible individuals in the overall population. The mean number of feasible individuals over all the generations are reported in the Mean columns. From Table 3.2, it is clear that the education procedure is able to restore the feasibility of offspring, i.e., the percentage of feasible individuals in *nPop* offspring is increased from 66% (62%) to 91% (90%) after the education for small (large) instances on average. Together, for small and large instances, we get percentages of feasible individuals in the population of 96% and 93% respectively.

3.4.1.3 Variants of selection and replacement schemes

In the next set of experiments, we analyzed the impact of the selection operator and replacement policies on the solution quality. Two types of selection operators, rank-based roulette wheel and binary tournament, were investigated in conjunction with two replacement strategies:

- Strategy 1. The population of generation $i+1$ is made up of the *nKeep* best solutions of generation *i* plus all *nPop* offspring.
- Strategy 2. The population of generation $i+1$ is made up of the *nKeep* best solutions of generation *i*, all *nPop* offspring, plus feasible solutions discovered during the education procedure displaying pattern assignments not yet used by *nKeep* best solutions and *nPop* offspring.

Note that in both strategies, each individual in the population has a different pattern assignment as only the best one among those using the same pattern assignment is kept.

Table 3.3: Proportion of individuals in the mating pool from *nKeep* best solutions of previous generation

Strategy	*nKeep*	Binary Tournament		Rank-based roulette wheel	
		%*nKeep*	GAP	%*nKeep*	GAP
1	40	6% - 8%	2.54%	26% - 37%	0.38%
	70	13 % - 16%	2.09%	48% - 53%	0.05%
2	40	4% - 7%	2.2%	24% - 34%	0.07%
	70	10% - 15%	2.06%	37% - 50%	0%

The experiment was run for 2000 generations with $nPop = 100$ and different values of *nKeep*. The selective pressure for the rank-based roulette wheel selection operator was set to 1.2. The experiment was performed using a small set of instances {1a,11a,3a, 3b, 9a, 9b} from Cordeau et al. (2004) with diverse characteristics, such as small and large time windows, number of customers, and number of periods, which impact the ease of addressing instances. Thus, larger time windows imply more feasible places to insert customers into routes, thereby increasing the number of feasible solutions with respect to this constraint. Similarly, more customers or longer planning horizons imply more feasible pattern combinations. We report the average results over 10 runs in Table 3.3. The %*nKeep* column reports the proportion of solutions in the mating pool that comes from the *nKeep* best solutions of the previous generation, the percentage values in each row of this column representing the lower and upper bounds on the number of solutions from *nKeep*, respectively. Over all, rank-based roulette wheel selection and the second replacement strategy yield better solutions than the others. Thus, we report in the GAP column the gaps for the average values obtained by each selection and replacement strategy with respect to that of the rank-based roulette wheel selection and the second replacement strategy.

We observe a significant difference in Table 3.3 between the two selection operators in terms of the impact they have on the performance of the HGGA. Further, for both selection operators, Strategy 2 yields better solutions than Strategy 1. Consequently, rank-based roulette wheel selection and the second replacement strategy were used in all subsequent experiments.

3.4.1.4 Complementarity of the crossover operators

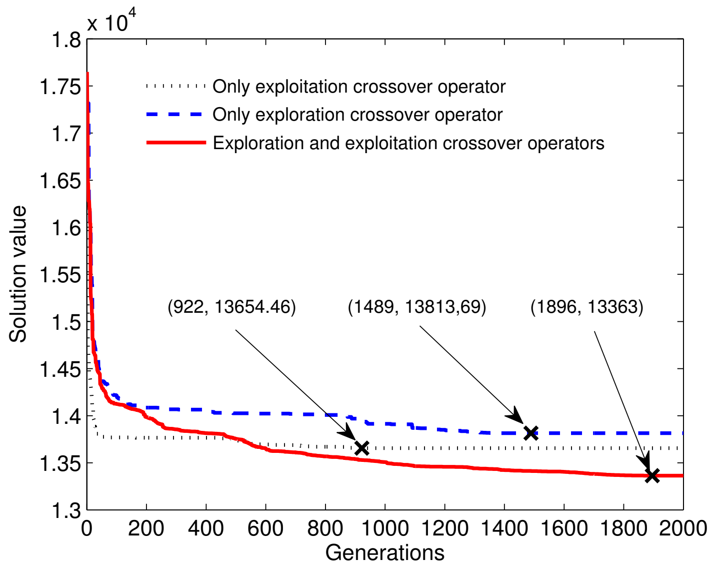


Figure 3.5: Performance comparison between variants of crossover operators.

To show the complementarity of the two crossover operators in HGGA, we have run tests where HGGA uses only one of the crossover operators. Figure 3.5 displays the value of the best solutions found by HGGA over 2000 generations on instance 10b of Cordeau et al. (2004). The value of the best solution when HGGA uses only the exploitation operator is 13813.69 obtained at generation 922, with only the exploration operator the best solution is 13654.46 obtained at generation 1489, while using both crossover operators the best solution is 13363 obtained at generation 1896. From Figure 3.5, we can see that using only the exploitation crossover, HGGA converges rapidly but the search yields a sub-optimal solution. With a slower convergence, HGGA using only the exploration crossover produces the worst solution, 1.17% higher than when only the exploitation operator is used. Employing both crossover operators improves the solution by -2.18% and -3.37% compared to using only either one of crossover operators, exploitation and exploration, respectively.

3.4.1.5 Calibration of search parameters

The main parameters of the proposed HGGA requiring calibration are the population size, the cardinality of the elite group, and the number of iterations for UTS and RVNS. The parameter values were selected considering a number of criteria: solution quality, improvement of the best solution through generations, and the computational cost for HGGA. The calibration was performed using the same small set of instances used in Section 3.4.1.3.

Experiments showed that a somewhat larger population size and longer education explorations (number of iterations of UTS and RVNS meta-heuristics) yield a more extensive sampling of the solution space and favor identifying good solutions based on correct selections of patterns for customers together with good routings. The experiments also showed that appropriate parameter values change with the size of the instance. The interval examined and the final parameter settings appear in Table 3.4, where small and large instances consist of less and more than or equal to 100 customers, respectively. Additionally, for all instances, the UTS procedure uses a tabu length of $\theta = 1.5log_{10}(n)$ (smaller than the $7.5log_{10}(n)$ in Cordeau et al., 2004), and the RVNS procedure lowers the temperature every $\tau = 10$ iterations (smaller than the 100 in Pirkwieser and Raidl, 2008).

Table 3.4: Calibration of main HGGA parameters

Parameters	Interval explored	Final values	
		Small instances	Large instances
Population size (*nPop*)	[50, 400]	70	100
Elite set cardinality (*nKeep*)	[25, 300]	40	70
Number of UTS iterations	[50, 150]	50	100
Number of RVNS iterations	[100, 800]	400	800
Maximum number of generations (*maxGEN*)	[250, 5000]	500	1500, 2000

The last calibration step examined the stopping criterion, which is based on the total number of HGGA iterations. The best solution and corresponding computation time were measured for each instance from generation 250 to generation 5000, by steps of 250. The results indicate that, for small instances, the best solutions can not be improved after 500 generations, while the average improvement of the best solution between generations 500 and 750 is less than 0.001%. Consequently, the number of generations was set to 500 for small instances. For large instances, the number of generations was set to 1500 for instances with less than 200 customers, and to 2000 for instances with more than or equal to 200 customers.

3.4.2 *Numerical Results*

The performance of the proposed HGGA is evaluated through comparisons with published results on the instances provided by Cordeau et al. (2004) and Pirkwieser and Raidl (2009a). For brevity, only aggregated results are provided in this section. Details may be found in Annex A.2.

In order to compare computational times of meta-heuristic algorithms that were run on different machines, we have scaled all CPU times into their equivalent Intel Xeon 2.8 GHz run times. This conversion is based on the assumption that CPU time is approximately linearly proportional to the number of floating point operations per second (flop/s) performed by the processor. Various computer systems have been tested using the linear equation solver software LINPACK, their (flop/s) measures are reported in Dongarra (2013). Table 3.5 provides these values for each of the algorithms we compared with, along with the resulting scaling factors used for the time conversion. MFlop/s values were obtained from the Net Library at the website with URL: http://www.netlib.org/benchmark/linpackjava/timings_list.html.

Table 3.5: Scaling factors for computation times

Authors	Processors	MFlop/s	Factor
Cordeau et al. (2004)	Pentium 4 2 GHz	198	0.47
Pirkwieser and Raidl (2008)	2.2 GHz Dual-Core AMD Opteron	300	0.7
Cordeau and Maischberger (2012); Vidal et al. (2013a)	2.93 GHz Intel Xeon CPU	448	1.05
Pirkwieser and Raidl (2009b, 2010)	Core2 Quad 2.83 GHz Q9550	687	1.6
This paper	2.8 GHz Intel Xeon CPU	428	1.00

Table 3.6 compares the gaps for the HGGA relative to previous BKS and those of the other algorithms on the Cordeau et al. (2004) instances:

- CLM04: UTS of Cordeau et al. (2004)

- PR08: RVNS of Pirkwieser and Raidl (2008)
- CM12: Parallel iterated tabu search of Cordeau and Maischberger (2012)
- VCGP13: Hybrid genetic algorithm of Vidal et al. (2013a)

Table 3.6: Best performance comparison among PVRPTW algorithms; Cordeau et al. (2004) instances

Instances				Prev BKS	CLM04	PR08	CM12	VCGP13	HGGA			
No	n	m	T	Cost	Best 10	Best X	Best 30	Best 10	Avg 10	Best 10 Cost	GAP	Time (min)
1a	48	3	4	2909.02	0.07	0	0	0	0	2909.02	0	2
2a	96	6	4	5026.57	0.57	0.19	0	0	0.04	5026.57	0	16
3a	144	9	4	7023.9	2.93	1.63	0.54	0.38	0.48	7024.96	0.02	129
4a	192	12	4	7755.77	2.54	1.63	0.66	0.47	0.25	**7738.25**	-0.23	219
5a	240	15	4	8311.17	3.39	2.18	0.58	0.37	0.56	8319.89	0.11	327
6a	288	18	4	10473.24	4.34	2.3	0.66	0.04	0.76	10478.80	0.05	468
7a	72	5	6	6782.68	0.62	0.07	0	0.01	0.1	6782.68	0	10
8a	144	10	6	9574.8	1.81	1.53	0.3	0.19	0.32	9574.86	0.0006	252
9a	216	15	6	13201.06	3.13	1.99	0.75	0.35	0.64	**13188.40**	-0.1	523
10a	288	20	6	16920.96	4.81	4.31	2.01	0.47	0.11	**16906.80**	-0.08	903
1b	48	3	4	2277.44	0.73	0	0	0	0	2277.44	0	2
2b	96	6	4	4121.5	3.3	0.39	0.08	0.01	0.39	4122.03	0.01	26
3b	144	9	4	5489.33	2.9	1.57	0.01	0.59	0.41	5489.77	0.01	140
4b	192	12	4	6347.77	3.89	2.03	0.56	0.08	0.73	**6345.65**	-0.03	248
5b	240	15	4	6777.54	4.09	2.84	0.34	0.19	0.6	**6775.39**	-0.03	446
6b	288	18	4	8582.72	4.03	2.76	0.89	0.14	0.19	**8580.06**	-0.03	504
7b	72	5	6	5481.61	0.43	0.42	0	0	0.26	5481.61	0	16
8b	144	10	6	7599.01	3.64	1.71	0.75	0.28	0.31	**7595.75**	-0.04	287
9b	216	15	6	10532.51	3.39	3.36	0.45	0.54	0.42	**10508.80**	-0.23	610
10b	288	20	6	13406.89	4.28	4	0.63	0.27	-0.1	**13363.00**	-0.33	1080
Avg Gap to BKS (%)					2.74	1.75	0.46	0.22	0.32		-0.05	
Avg Origin run time (min)					160	N/A	11.32	32.74				310.46
Avg Scaled time (min)					75.2	N/A	11.88	34.37				310.46
Processor					P4-2G	Opt-2.2G	Xe-2.93G	Xe-2.93G				Xe-2.8G

The first four columns indicate the instance name, the number of customers n, the number of vehicles m, and the number of days T, respectively. The fifth column displays the previous BKS reported in Vidal et al. (2013a). All gaps are reported as % value w.r.t. the previous BKS. The next four columns provide, respectively, the gaps for the values reported by Cordeau et al. (2004), Pirkwieser and Raidl (2008), Cordeau and Maischberger (2012), Vidal et al. (2013a). The last four columns report the gaps for the average values obtained by our algorithm over 10 runs, the cost of the best solution found during these runs, the corresponding gap and the average computation time over the 10 runs. The last three rows provide average measures over all instances: the average gap to the previous BKS, the original computation time, and the scaled computation time which use our machine (Xe-2.8G) as the baseline in minutes, and the type of processor used by each algorithm. Boldface indicates instances for which HGGA improves previous BKS.

HGGA produces high quality solutions, with an average error gap of -0.05% to the previous BKS, compared to more than 0.22% for the other algorithms. HGGA produces 9 new best-known solutions and finds 5 best-known solutions over 20 instances. Experiments also showed that the best solutions in the initial population were 27.59% greater than the best solutions obtained by HGGA on average, illustrating the significant solution-improvement effect of the hybrid strategy.

The next round of experiments focused on the instances proposed by Pirkwieser and Raidl, and used in Pirkwieser and Raidl (2009a,b, 2010) and Vidal et al. (2013a). It is noteworthy that the authors truncated the calculated travel costs to one digit, which reduced the total cost. More importantly, for instances with tight time windows, truncation can produce many more feasible solutions compared to the no-truncation case, hence resulting in a quite huge reduction of total cost. We therefore give the performance results of HGGA with and without truncation (the detailed results are provided in Annex A.2). One also notes that best solutions were reported by Vidal et al. (2013a) only, and just for the truncation case. Pirkwieser and Raidl (2009a,b, 2010) reported only average costs and standard deviations computed over 30 runs for each instance. Average costs over 10 runs for each instance set p4, p6, p8 are available in Vidal et al. (2013a). We therefore selected the best average cost of each instance set over all previous algorithms, identified it as *Previous Best Average*, and used it to measure the performance of the proposed HGGA meta-heuristic (notice that all *Previous Best Average* were produced by Vidal et al. (2013a)).

Table 3.7 sums up this comparison, based on averages. It displays the average results of HGGA over 10 runs (with truncation), the reported results of the algorithms in Pirkwieser and Raidl (2009a,b, 2010) – VNS and VNS-ILP (Pirkwieser and Raidl, 2009a); mVNS and mVNS-ILP (Pirkwieser and Raidl, 2009b); Evolutionary Algorithm (EA), combination of column generation and evolutionary algorithm (CG-EA), and CG-ILP (Pirkwieser and Raidl, 2010) – as well as the average results of the Hybrid Genetic Algorithm (VCGP13) of (Vidal et al., 2013a). We report for each algorithm the averages of the percentages of deviation from the Previous Best Average (Column *GAP*), the published computation time (Column *Origin*) and the scaled computation time based on our machine (Xe-2.8G) (Column *Scale*). Several settings were given for VNS-ILP, mVNS, and mVNS-ILP, without a clear dominating one. We therefore selected the best solution over all settings (5 or 7, each with 30 repetitions) for each instance set, together with the corresponding computation time. Pirkwieser and Raidl (2009a) used an Opteron 2.2 GHz, Pirkwieser and Raidl (2009b, 2010) used a Core2 Quad 2.83 GHz, and Vidal et al. (2013a) used an Intel Xeon 2.93 GHz. One observes that HGGA performs again very well, obtaining the smallest average gap from the Previous Best Average for all 45 instances, and improving the average costs by 0.09%, 0.25%, and 0.48% for p4, p6, and p8 instances, respectively. A detailed comparison, instance by instance, further supports this conclusion (see Table A.2 in the Annex). Relative to the best known solutions of Vidal et al. (2013a), the proposed metaheuristic obtained better results on 9/45 instances, the same results on 13/45, and worse on 23/45 instances. In all cases, improvements and deficits are negligible, the gaps for the three instance sets being 0.002%, 0.022% and -0.004% for p4, p6, and p8, respectively.

Table 3.7: Comparative performances on Pirkwieser and Raidl (2009a) instances

Algorithms	Processor	p4				p6				p8			
		Runs	GAP	Time (min)		Runs	GAP	Time (min)		Runs	GAP	Time (min)	
				Origin	Scale			Origin	Scale			Origin	Scale
VNS	Opt-2.2G	90	1.32%	0.81	1.15	90	1.21%	0.93	1.33	30	2.18%	0.51	0.73
VNS-ILP		150	1.12%	0.57	0.81	150	1.05%	0.81	1.16	30	2.05%	0.58	0.83
mVNS		150	0.95%	0.43	0.69	210	0.96%	0.80	1.28	-	-	-	-
mVNS-ILP		150	0.41%	0.70	1.12	210	0.81%	0.82	1.31	-	-	-	-
EA	Qd-2.83G	30	3.55%	0.48	0.77	30	4.25%	0.61	0.98	30	6.33%	0.73	1.17
CG-EA		30	2.47%	0.60	0.96	30	3.41%	0.80	1.28	30	5.19%	1.00	1.6
CG-ILP		30	3.52%	0.56	0.89	30	9.61%	0.80	1.28	30	14.88%	1.00	1.6
VCGP13	Xe-2.93G	10	0%	3.21	3.37	10	0%	4.44	4.66	10	0%	5.06	5.31
HGGA	Xe-2.8G	10	-0.09%	33.88	33.88	10	-0.25%	43.20	43.20	10	-0.48%	48.75	48.75

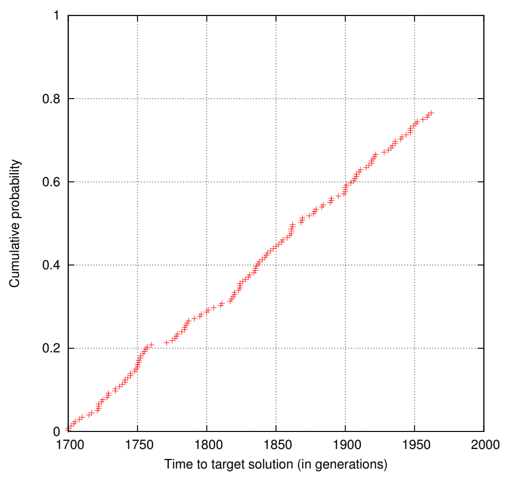
Table 3.8 displays the performance of HGGA for the two cases, with and without travel-cost truncation. As expected, HGGA displays enhanced performance in the former case compared to the latter. Experiments also showed that the best solutions in the initial population were, on average, 23.17% and 21.86% greater than the best solution obtained in the cases without and with truncation, respectively, illustrating again the significant effect of the hybrid meta-heuristic.

Table 3.8: HGGA and the travel-cost truncation issue; Pirkwieser and Raidl (2009a) instances

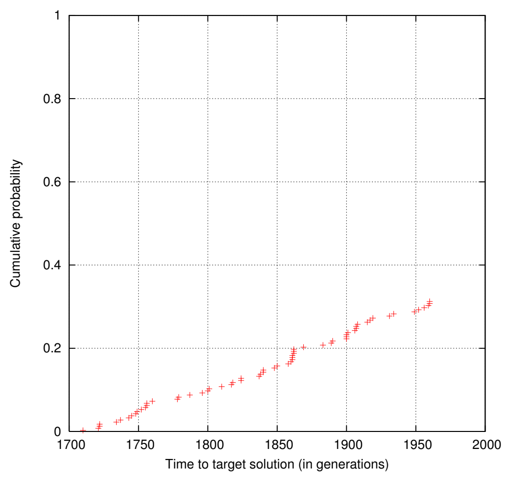
Result		p4	p6	p8
Previous Best Average	Avg cost	3280.30	4381.19	5387.09
HGGA with truncation	Avg cost	3277.23	4370.43	5361.22
	GAP	-0.09%	-0.25%	-0.48%
HGGA without truncation	Avg cost	3297.31	4393.97	5394.08
	GAP	0.55%	0.29%	0.13%

To further validate the performance of HGGA, we sample the success rate of HGGA for reaching the previous BKS and the new BKS by constructing a time-to-target (TTT) plot (Aiex et al., 2007) for two problem instances. A TTT plot is generated from the execution of an algorithm κ times during which one measures the time required to reach a solution at least as good as a target solution. The running times are sorted in increasing order. The i-sorted running time t_i is associated with a probability $\rho_i = (i-1/2)/\kappa$, the points $z_i = [t_i, \rho_i]$, $i = 1, \ldots, \kappa$, are plotted. Each plotted point indicates the probability (vertical axis) for the algorithm to achieve the target solution in the indicated time (horizontal axis). For this experiment we use the two largest instances of Cordeau et al. (2004), i.e., instances 10a and 10b, and replace, in the TTT plots, the running times by the number of generations. The plots in Figure 3.6a and 3.6b were generated from the execution of HGGA 200 times on instances 10a, using the previous BKS and the new BKS as target values, i.e., values 16920.96 and 16906.80, respectively. Similarly, the plots in Figure 3.6c and 3.6d were generated for instance 10b. We observe that HGGA reaches the previous

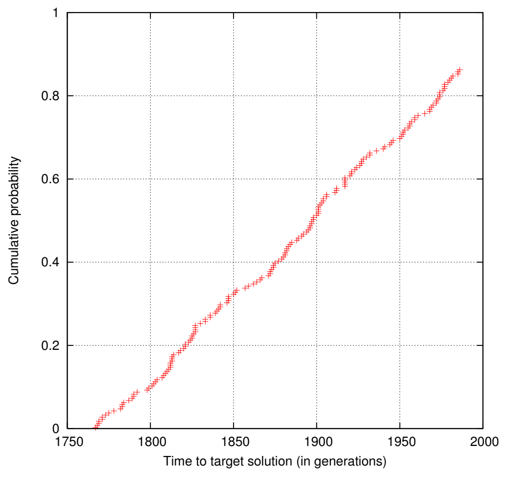
BKS with a maximum probability of 0.75 and 0.86 on instances 10a and 10b respectively while reaching the new BKS with a maximum probability of 0.32 and 0.37 respectively for instances 10a and 10b.



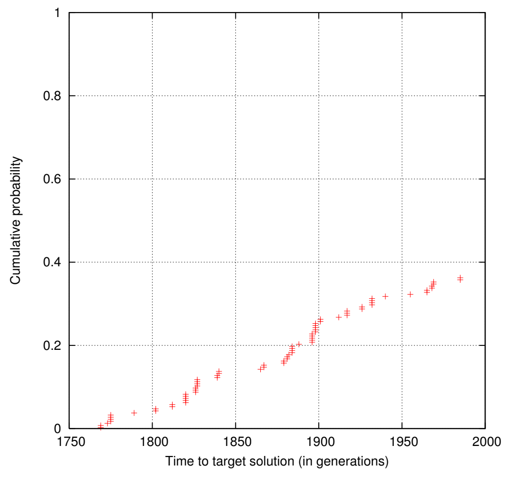
(a) Target solution is previous BKS of 10a



(b) Target solution is new BKS of 10a



(c) Target solution is previous BKS of 10b



(d) The target solution is new BKS of 10b

Figure 3.6: Time-to-target plot for the instance 10a and 10b of Cordeau et al. (2004)

We conclude this section with a few remarks on the computational effort of HGGA. In its current implementation, this effort appears indeed high compared to the methods in the literature, the meta-heuristic education procedure compounding the issue. Yet, the issue is not really significant. On the one hand, the proposed methodology may be greatly accelerated by using the classical strategy of performing the crossover and education procedures in parallel (Crainic and Toulouse, 2010). This strategy is particularly adapted to the present case for two main reasons. First, in the generational GA paradigm, all mating and offspring generation and education is performed before a new generation is considered. Decomposing this component of the work clearly then yields independent tasks. Second, each such task is still computation intensive, involving at least a sequence of mating and offspring generation and education (this case corresponds to the finer-grained decomposition assigning each pair of parents to a particular processor). Quasi-linear speedup factors may consequently be expected.

On the other hand, the proposed hybrid GA meta-heuristic is actually using the computational effort to provide higher-quality solutions. To support this claim, we let the two neighborhood-based meta-heuristics search address each instance for a computing time equivalent to that of the HGGA for the same instance. All algorithms were run 10 times and the best results were taken for comparison. The results are summed up in Table 3.9, where columns CLM04 and PR08 indicate the gap to the cost of solutions obtained by HGGA from the cost of solutions obtained by the meta-heuristics for the same computing time as HGGA (for example, row 3a, $-2.69 = \frac{HGGA(3a)-CLM04(3a)}{HGGA(3a)} \times 100$). The negative values observed for all instances and procedures indicate the computing effort of HGGA is well spent, the proposed hybrid meta-heuristic is outperforming the two meta-heuristics, UTS of Cordeau et al. (2004) and RVNS of Pirkwieser and Raidl (2008), which are used in the education procedure.

3.5 Conclusion

We introduced HGGA, a population-based hybrid meta-heuristic for the periodic vehicle routing problem with time windows. For this hybrid, a persistent algorithm, the genetic algorithm, calls at each generation for the execution of one among two neighborhood-based meta-heuristics to “educate” each individual of the current population. Extensive numerical experiments and comparisons with all methods proposed in the literature show that the proposed methodology is highly competitive, providing new best solutions in some large instances.

In the present algorithm, the education of each individual is completely independent of the others for a given generation, education can be easily parallelized with an expected speed-up closed to linear. Future work will exploit this parallelism by expanding the current hybrid algorithm into a cooperative search algorithm. To fully exploit parallelism, the neighborhood-based meta-heuristics of the current implementation will be used in a persistent manner and executed in parallel while GA operators will provide cooperation

Table 3.9: Performance comparison for fixed computing effort; Cordeau et al. (2004) instances

Instances	Time (hours)	CLM04	PR08	HGGA
1a	0.03	-0.04	0	2909.02
2a	0.27	-0.51	-0.15	5026.57
3a	2.15	-2.69	-1.71	7024.96
4a	3.66	-2.11	-1.79	7738.25
5a	5.46	-2.91	-1.32	8321.15
6a	7.81	-3.88	-2.11	10478.80
7a	0.17	-0.28	-0.1	6782.68
8a	4.21	-1.7	-2.05	9574.86
9a	8.71	-3.04	-1.65	13188.40
10a	15.05	-4.52	-3.12	16906.80
1b	0.03	-0.51	0	2277.44
2b	0.43	-2.67	-0.8	4122.03
3b	2.34	-2.49	-1.69	5489.77
4b	4.14	-3.53	-2.05	6345.65
5b	7.43	-2.08	-3.1	6775.39
6b	8.41	-2.67	-3.09	8580.06
7b	0.26	-0.36	-0.33	5481.61
8b	4.78	-3.54	-1.31	7595.75
9b	10.17	-3.33	-3.12	10508.80
10b	18	-2.75	-4.14	13363.00
Average	5.18	-2.28	-1.68	7924.49

among neighborhood-based meta-heuristics. Current and new crossover operators will be called on solutions that belong to a pair of neighborhood-based meta-heuristics, generating offspring that will guide the two cooperating meta-heuristics. A distributed selection procedure will be designed to select these solutions. Besides speeding-up computation through parallelism, this cooperative approach to PVRPTW will study the distributed selection, crossover and mutation operators and their capability to provide, through diffusion, a consistent global search strategy, which we expect to be comparable to the evolution of a steady state GA.

Chapter 4

A TABU SEARCH FOR TIME-DEPENDENT MULTI-ZONE MULTI-TRIP VEHICLE ROUTING PROBLEM WITH TIME WINDOWS

This chapter has been published as journal article: P. K. Nguyen, T. G. Crainic, and M. Toulouse. A tabu search for Time-dependent Multi-zone Multi-trip Vehicle Routing Problem with Time Windows. *European Journal of Operational Research*, 231(1):43–56, 2013.

Résumé: Nous proposons une méta-heuristique basée sur la recherche tabou pour le Time-dependent Multi-zone Multi-trip Vehicle Routing Problem with Time Windows. Cette méta-heuristique utilise deux types de voisinages correspondant aux deux types de décisions du problème. Une stratégie de sélection du type de voisinage est définie en fonction des phases de la recherche dans l'espace de solutions, cette stratégie permettant de combiner les capacités d'exploration et d'intensification de la recherche. Une stratégie de diversification guidée par un ensemble de solutions élites et une mémoire à base de fréquences est aussi utilisée pour conduire la recherche dans des régions non explorées de l'espace de solutions dans le but d'améliorer la qualité de la recherche. Un grand nombre d'expérimentations ont été conduites, de même que de nombreuses comparaisons avec les résultats dans la littérature. Ces expérimentations et comparaisons montrent que la méthode de recherche tabou proposée donne des solutions de très hautes qualités, améliorant les solutions déjà publiées.

Abstract: We propose a tabu search meta-heuristic for the Time-dependent Multi-zone Multi-trip Vehicle Routing Problem with Time Windows. Two types of neighborhoods, corresponding to the two sets of decisions of the problem, together with a strategy controlling the selection of the neighborhood type for particular phases of the search, provide the means to set up and combine exploration and exploitation capabilities for the search. A diversification strategy, guided by an elite solution set and a frequency-based memory, is also used to drive the search to potentially unexplored good regions and, hopefully, enhance the solution quality. Extensive numerical experiments and comparisons with the literature show that the proposed tabu search yields very high quality solutions, improving those currently published.

Keywords: Multi-trip Vehicle Routing with Time windows; Synchronization; Time-dependent demand; Tabu search

4.1 Introduction

The basic Vehicle Routing Problem with Time Windows (VRPTW) aims to design least cost routes from a single depot to a set of geographically distributed customers, while satisfying time window constraints at customers and the capacity of vehicles. In this paper, we consider the *Time-dependent Multi-zone Multi-trip Vehicle Routing Problem with Time Windows* (*TMZT-VRPTW*), which is an extension of the VRPTW involving both designing and assigning routes to vehicles within time synchronization restrictions.

In the TMZT-VRPTW setting, a homogeneous fleet of vehicles operates out of a single garage to deliver customer-specific loads, available at particular facilities during particular operating time intervals. Deliveries at customers must be performed according to hard time windows. Vehicles must synchronize their arrivals at facilities with the respective operating time periods, that is, time windows at facilities are hard and vehicles are not permitted to arrive in advance and wait. Particular waiting stations may be used by the vehicles to wait for the next appointment. A vehicle route thus leaves the garage to visit a first facility within its operating time periods and load freight, proceeds to deliver it to customers within their time windows, and then moves to its next appointment at a facility, possibly stopping to wait for the appropriate time at a waiting station. The route continues until either there are no more loads to deliver or its cost becomes noncompetitive compared to other routes. The vehicle returns to the garage in both cases. The goal of the TMZT-VRPTW is to determine the set of routes, and assign them to particular vehicles, providing timely customer service and synchronized arrival at facilities for loading freight, minimizing the total cost made up of the (variable) costs of operating vehicles and the (fixed) costs of using them. The "time dependency" characterizing the problem setting follows from the time stamps of the origin-to-destination demand, indicating the time interval when the load is available at the origin facility and the delivery time window at the customer. This is different from most time-dependent vehicle routing problem contributions in the literature where travel costs or travel times are considered to vary with time.

The TMZT-VRPTW is encountered in several settings, in particular in the context of planning the operations of two-tiered City Logistics systems (Crainic et al., 2009). In such systems, the first tier involves large-capacity vehicles delivering freight from the city distribution centers (CDCs) located on the outskirts of the city to intermediary facilities, called satellites, where it is transferred to smaller-capacity vehicles performing the satellite-to-customer delivery routes. Given the concerns regarding the impact of freight transport on the city living conditions (e.g., congestion and environment), as well as the locations of most satellites within or close to the city center, very short waiting times are allowed, most transfer operations being performed according to trans-dock practices, without intermediate storage. The arrival of first-tier vehicles at a given time period define the set of customers to be serviced, and the time required to unload and transfer the freight thus defining the availability period during which second-tier vehicles must arrive at the satellite and load. Second-tier vehicles must therefore synchronize their arrivals at satellites with these availability periods. After loading the planned freight, each second-tier vehicle undertakes a trip servicing the customers assigned to it. Once the last customer is serviced, the vehicle moves empty either directly to a satellite for its next trip, to a waiting station (when available) to wait for its next appointment at a satellite, or to the garage to end the current work assignment. The TMZT-VRPTW corresponds to the planning of the activities of second-tier vehicles.

To our knowledge, Crainic et al. (2009) were the first (and only) to propose a method to address the TMZT-VRPTW. The authors proposed a decomposition approach that first addressed the VRPTW subproblems representing the delivery to the customers associated with each combination of satellite and availability period. The vehicle trips resulting from the solution of the subproblems were then put together into multi-trip routes by solving a minimum cost network flow problem. These two sets of decisions, 1) how to service customers associated to given facility-availability period combinations, and 2) how to combine the resulting trips into vehicle-specific multi-trip routes abiding by the synchronization requirements at facilities, are not independent, however. Combining them into one formulation and solution method should yield better results. The objective of this paper is to take up on this challenge and present a meta-heuristic that addresses the two decisions simultaneously, in a comprehensive and efficient way.

We thus introduce the first tabu search for the TMZT-VRPTW, integrating multiple neighborhoods grouped into two classes to address the two sets of decisions identified above. A first set of neighborhoods and moves work on the construction of the multiple-trip vehicle routes by modifying the facilities and availability periods a given vehicle visits. A second set aims to improve the routing of vehicles between two such visits by working on the customer to route/vehicle assignments. The former perturbs significantly the solution and thus favors exploration of the search space, while the latter applied to each vehicle trip exploits good assignments. Hence, dynamically adjusting their utilization during the search provides the proposed algorithm with desired exploration and exploitation capabilities.

The proposed algorithm starts by freely exploring the search space made up of feasible and infeasible solutions. As the search advances, one lowers the probability of selecting neighborhoods modifying the facility-to-vehicle route assignments, thus limiting the size of the search region and giving routing moves more time to optimize routes. Of course, customer time windows and synchronization requirements constrain these decisions and moves. A diversification strategy guided by an elite set of solutions and a frequency-based memory is called upon when the search begins to stagnate. Creating new working solutions from the elite set helps to capitalize on the best solution attributes obtained so far. On the other hand, employing a frequency-based memory to perturb new working solutions provides a certain level of diversity to the search.

The main contributions of the paper are the following: 1) a new formulation for the TMZT-VRPTW, which is the source to define neighborhoods in the proposed tabu search; 2) the neighborhood structure and the dynamic strategy used to control the selection of neighborhoods; 3) a new tabu search meta-heuristic outperforming the available method (Crainic et al., 2012b) with new best-known solutions on all instances and an improvement in the solution quality by 4.42% on average.

The remainder of the paper is organized as follows. Section 4.2 contains a detailed problem description. The problem formulation is then provided in Section 4.3. Section 4.4 reviews the literature. The details of the proposed methodology are described in Section 4.5. Computational results are then reported and analyzed in Section 4.6, while conclusions and future works are considered in Section 4.7.

4.2 Problem Description

The time-dependency characterizing demand in the TMZT-VRPTW setting translates into two phenomena. The first concerns facilities, which become available for work at particular time periods only with a set of loads destined to specific customers. A given facility may be available at several periods during the planning period considered, with a different set of loads at each occurrence. To model this time dependency, we define ***supply points*** as particular combinations of facilities and availability time periods. A supply point is then characterized by a set of loads to be delivered to particular customers, and by a no-wait hard time window, meaning that vehicles cannot arrive before the beginning of the time window and wait for the opening of the facility, nor after the end of the time window by paying a penalty. The second phenomenon concerns customers, which may receive several loads, from different facilities and time periods. We model this time dependency by identifying each particular load as a ***customer demand***, characterized by the supply point where it is available for delivery, the customer it must be delivered to, and the particular time window for the delivery at the customer.

Synchronization at supply points requires that vehicles arrive at supply points at appointed times. Consequently, a direct move that gets the vehicle to a supply point sooner than the appointed time is forbidden. In this case, the vehicle may go to a location, which we call *waiting station* (e.g., a parking lot), and wait there in order to get to its next supply point just before the appointed time. Otherwise, if there is no waiting station available, the vehicle goes to the garage to finish its route.

The TMZT-VRPTW can then be described as follows. There is a garage, or main depot, g, a set of waiting stations $w \in \mathscr{W}$, a set of supply points $s \in \mathscr{S}$, and a set of customer-demand nodes $d \in \mathscr{D}$ (i.e., one node for each customer demand). We assume that there is a limited allowable waiting time, defined by η, at each supply point. Each supply point $s \in \mathscr{S}$ has a no-wait, hard opening time window $[t(s) - \eta, t(s)]$, specifying the earliest and latest times the vehicle may be at s, respectively, a vehicle loading time $\delta(s)$, and a set of customer-demand nodes $D_s \in \mathscr{D}$ making up its *service zone*. Each customer-demand node $d \in D_s$ has a volume q_d to be delivered, a service time $\delta(d)$ to unload freight from the vehicle, and a time window $[e_d, l_d]$, where e_d is the earliest time service may begin and l_d is the latest time. It is assumed all customer-demand volumes are less than the capacity of the vehicle (otherwise, the classical technique of duplicating customer demands such that each conforms to this requirement is applied in a processing phase).

The TMZT-VRPTW can be seen as the problem of determining a set of routes made up of a sequence of supply-point visits, each followed by a trip servicing customer loads in the zone of the respective supply point, and of assigning each route to one vehicle. The objective is to minimize the total cost, which is made up of the fixed cost of using the vehicles and the routing costs of servicing customer demands and moving between supply points, while the following conditions are satisfied:

1. Every vehicle starts and ends its route at the main depot g;

2. Every vehicle servicing customer-demand nodes in D_s must reach the supply point $s \in \mathscr{S}$ within its time window, i.e., it must not arrive sooner than $(t(s) - \eta)$ and no later than $t(s)$; When needed, the vehicle may wait at a waiting station $w \in \mathscr{W}$ before moving to s; Once at s, the vehicle starts loading at time $t(s)$ and continues loading for a time $\delta(s)$, after which it leaves s to service the assigned customer-demand nodes in D_s. After performing a route within zone D_s, the vehicle may move to another supply point for the next trip or go to the main depot g to complete its route;

3. Every customer-demand node $d \in \cup_{s \in \mathscr{S}} D_s$ is visited by exactly one vehicle within its time window (these are hard).

4.3 Model Formulation

The TMZT-VRPTW is defined on a directed graph $\mathscr{G} = (\mathscr{V}, \mathscr{A})$, with vertex set $\mathscr{V} = g \cup \mathscr{S} \cup \mathscr{D} \cup \mathscr{W}$, where g is the main depot, $\mathscr{S}$ is the set of supply points, $\mathscr{D} = \{\cup D_s : s \in \mathscr{S}\}$ is the customer-demand node set, $\mathscr{W}$ is the set of waiting stations, and the arc set $\mathscr{A} = \{(g,s) : s \in \mathscr{S}\} \cup \{(s,d) : s \in \mathscr{S}, d \in D_s\} \cup \{(d,j) : d \in \mathscr{D}, j \in g \cup \mathscr{W}\} \cup \{(i,j) : i,j \in D_s, s \in \mathscr{S}\} \cup \{(d,s') : d \in D_s, s, s' \in \mathscr{S}, t(s) < t(s')\} \cup \{(w,s) : w \in \mathscr{W}, s \in \mathscr{S}\}$. Hence, the set $\mathscr{A}$ does not include arcs representing direct travel

- From the main depot g to any customer-demand node or waiting station;

- From any customer-demand node to its supply point or to supply points with opening times earlier than that of its supply point;

- From a supply point to any waiting station or to the main depot g.

A routing cost (or travel time) c_{ij} is associated with each arc $(i, j) \in \mathscr{A}$. A fleet of m identical vehicles with capacity Q is based at the main depot g. Vehicles are grouped into set $\mathscr{K}$.

Let a ***route leg*** be a trip that links a pair of supply points, or starts and ends at a supply point and the main depot g, respectively. Thus, there are two types of route legs and their feasibility is defined as follows:

- A ***single-supply point route leg*** l, starting at supply point s and ending at the main depot, is feasible if it starts loading a total of goods not exceeding Q at supply point s at time $t(s)$, then leaves s at time $t(s) + \delta(s)$ to deliver to a subset of customer-demand nodes in D_s within their time windows.

- An ***inter-supply point route leg*** l that starts and ends at a pair of supply points s and s', respectively, is feasible if it starts loading a total of goods not exceeding Q at supply point s at time $t(s)$, then leaves s at time $t(s) + \delta(s)$ to deliver to a subset of customer-demand nodes in D_s within their time windows, and arrives to s' within the opening time window $[t(s') - \eta, t(s')]$ (the vehicle can wait at a waiting station $w \in \mathscr{W}$ before moving to s' in case the direct move from the last serviced customer in leg l to s' gets the vehicle to s' before $(t(s') - \eta)$).

A sequence of route legs, starting and ending at the main depot, assigned to a vehicle is called a route or ***work assignment***. For the sake of simplicity, from now on, the terms vehicle route and work assignment are used interchangeably. Figure 4.1 illustrates a three-leg work assignment, where s_1, s_2, s_3 are supply points, g and w_1 are the main depot and a waiting station, respectively, $D_{s_1} = \{d_1, d_2, d_3, d_4, d_5\}$, $D_{s_2} = \{d_6, d_7, d_8, d_9\}$, and $D_{s_3} = \{d_{10}, d_{11}, d_{12}, d_{13}, d_{14}, d_{15}\}$. The dashed lines stand for the empty arrival from the depot g or from a waiting station, the empty movement from the last customer in the previous leg to the supply point of the next leg or to a waiting station, and the empty movement to the depot g once the work assignment is finished. This work assignment consists of a sequence of three legs $\{l_1, l_2, l_3\}$, where $l_1 = \{s_1, d_1, d_4, d_3, s_2\}$ and $l_2 = \{s_2, d_7, d_9, w_1, s_3\}$ are two inter-supply point route legs, while $l_3 = \{s_3, d_{11}, d_{14}, d_{12}, d_{15}, g\}$ is a single-supply point route leg.

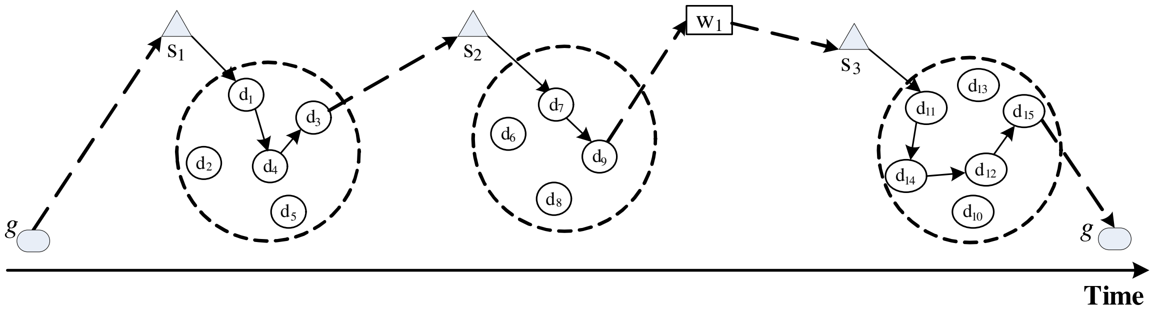


Figure 4.1: A three-leg work assignment illustration

Let $\mathscr{L}$ denote the set of all feasible legs satisfying the total load of vehicles, and the time windows at customers and supply points. Define the e_{dl} and f_{ls} coefficients:

$$e_{dl} = \begin{cases} 1 & \text{if customer demand } d \in D \text{ is on leg } l \in \mathscr{L}; \\ 0 & \text{otherwise.} \end{cases}$$

$$f_{ls} = \begin{cases} 1 & \text{if leg } l \text{ starts at supply point } s \in \mathscr{S}; \\ -1 & \text{if leg } l \text{ ends at supply point } s \in \mathscr{S}; \\ 0 & \text{otherwise.} \end{cases}$$

Binary decision variables are used in the formulation:

- $x_l^k = \begin{cases} 1 & \text{if leg } l \in \mathscr{L} \text{ is assigned to work assignment } k \in \mathscr{K}; \\ 0 & \text{otherwise.} \end{cases}$

- $y_s^k = \begin{cases} 1 & \text{if work assignment } k \text{ has the first leg starting at supply point } s; \\ 0 & \text{otherwise.} \end{cases}$

- $z_s^k = \begin{cases} 1 & \text{if work assignment } k \text{ has the last leg starting at supply point } s; \\ 0 & \text{otherwise.} \end{cases}$

Let π_l be the total cost of route leg l, and F be the fixed cost of using a vehicle. The TMZT-VRPTW can then be formulated as

$$\text{Minimize} \sum_{k \in \mathscr{K}} \sum_{l \in \mathscr{L}} \pi_l x_l^k + \sum_{k \in \mathscr{K}} F \sum_{s \in \mathscr{S}} y_s^k \tag{4.1}$$

$$\text{S.t.} \sum_{k\in\mathscr{K}} \sum_{l\in\mathscr{L}} e_{dl} x_l^k = 1 \;\; \forall d \in \mathscr{D}, \tag{4.2}$$

$$\sum_{s\in\mathscr{S}} y_s^k \le 1 \;\; \forall k \in \mathscr{K}, \tag{4.3}$$

$$\sum_{s\in\mathscr{S}} y_s^k = \sum_{s\in\mathscr{S}} z_s^k \;\; \forall k \in \mathscr{K}, \tag{4.4}$$

$$\sum_{l\in\mathscr{L}} f_{ls} x_l^k = y_s^k \;\; \forall s \in \mathscr{S}, k \in \mathscr{K}, \tag{4.5}$$

$$x_l^k \in \{0,1\} \;\; \forall l \in \mathscr{L}, k \in \mathscr{K}, \tag{4.6}$$

$$y_s^k \in \{0,1\} \;\; \forall s \in \mathscr{S}, k \in \mathscr{K}, \tag{4.7}$$

$$z_s^k \in \{0,1\} \;\; \forall s \in \mathscr{S}, k \in \mathscr{K} \tag{4.8}$$

The objective function (4.1) minimizes the total cost made up of the costs of operating and using vehicles. Constraints (4.2) guarantee that each customer demand is visited exactly once, while Constraints (4.3) state that at most one work assignment is assigned to each vehicle. Constraints (4.4) ensure that each work assignment starts and ends at the main depot. In fact, by summing over all supply points, the left-hand side counts the number of first legs assigned to each vehicle k, while the right-hand side counts the number of last legs assigned to each vehicle k. Then, from Constraints (4.3) and (4.4), either the numbers of first and last legs assigned to the vehicle k are both equal to zero, e.g., the vehicle k is not used ($\sum_{s\in\mathscr{S}} y_s^k = \sum_{s\in\mathscr{S}} z_s^k = 0$), or are both equal to 1, e.g., the vehicle k is used ($\sum_{s\in\mathscr{S}} y_s^k = \sum_{s\in\mathscr{S}} z_s^k = 1$).

Constraints (4.5) ensure that when a vehicle goes to a supply point, it also leaves it, except for the starting supply point of the first leg. In fact, for any given vehicle k and supply point s, the left-hand side of the equality sums the value of f_{ls} on all legs starting or ending at supply point s which are assigned to vehicle k. Consequently, when $y_s^k = 1$, the equality (4.5) becomes $\sum_{l\in\mathscr{L}} f_{ls} x_l^k = 1$, which means that there must be one leg starting at supply point s assigned to vehicle k as the first leg. Constraints (4.6), (4.7), and (4.8) define the sets of decision variables.

4.4 Literature Review

The literature on TMZT-VRPTW is limited. In the TMZT-VRPTW setting, customer demands are divided into zones associated to supply points, that is facilities and time periods. Taking advantage of this special structure, Crainic et al. (2009) proposed a decomposition-based heuristic approach for TMZT-VRPTW, but no implementation was reported. The general idea is to decompose the problem by (facility, period) zone, solve the resulting small VRPTW at each zone, and finally determine the flow of vehicles to operate the routes associated with these zones at minimum cost by solving a minimum cost network flow problem. Crainic et al. (2012b) later implemented this idea, and calculated a lower bound by relaxing vehicle capacity and time window constraints at supply points and customers.

A number of VRP variants share the multi-trip setting with the TMZT-VRPTW, e.g., the Multi-trip Vehicle Routing Problem (or the Vehicle Routing Problem with Multiple Use of Vehicles; Taillard et al., 1995; Brandão and Mercer, 1998; Petch and Salhi, 2003; Salhi and Petch, 2007), the VRP with Intermediate Facilities or with inter-depot routes (Tarantilis et al., 2008; Crevier et al., 2007), and the Waste Collection VRP (Kim et al., 2006; Ombuki-Berman et al., 2007; Benjamin and Beasley, 2010). In the first variant, only one depot is used to replenish vehicles between their trips, while in the latter variants, vehicles may be replenished at intermediate depots along their trips. In addition, unlike the first two variants, each driver is assumed to take a lunch break in a period of time in the Waste Collection VRP. The challenging setting in our problem compared to these three variants is the time synchronization restrictions at supply points and the waiting stations.

The School Bus Routing problem (SBRP) resembles our problem setting quite closely. In general, the SBRP involves transporting students from predefined locations to their schools using a fleet of buses with varying capacities, while satisfying all school timing requirements (see the surveys of Desrosiers et al., 1981; Braca et al., 1997; Park and Kim, 2010). The SBRP consists of three components: determine the bus stop locations, assign students to bus stops, route and schedule the buses. However, most of the problems described in the literature just consider some parts of the SBRP. In the multi-school setting, the SBRP shares some constraint settings with the TMZT-VRPTW, e.g., vehicle capacity, school time window, multi-trip. Yet, there are also differences in conditions and settings between the SBRP and the TMZT-VRPTW. Mixed-load settings may occur in the SBRP, where students from different schools can be put on the same bus at the same time; Maximum riding times for students on buses may also be specified. These settings are not imposed in the TMZT-VRPTW. In the SBRP, only the exact earliest pick-up time for all students is considered, while there is a time window for each customer in the TMZT-VRPTW.

4.5 Tabu Search Meta-heuristic

Among the meta-heuristics proposed for the vehicle routing problem, tabu search has been shown to be a very effective one, providing a good compromise between solution quality and computation time. Various techniques have also been proposed to further enhance the performance of tabu search for complex problems with multiple constraints and characteristics. Inspired in part by these developments, we propose a tabu search with multiple neighborhoods and appropriate memory mechanisms for the TMZT-VRPTW. This section describes our tabu search algorithm, from its general structure to detailing its main components. The search space and initial solution generator are presented in Sections 4.5.2 and 4.5.3, respectively. Sections 4.5.4 - 4.5.6 describe the neighborhood structures, our neighborhood-selection strategy, and the tabu status mechanism for each neighborhood. We detail in Section 4.5.7 the management of the elite set and a diversification mechanism. Finally, a post optimization procedure is described in Section 4.5.8.

4.5.1 General structure

The TMZT-VRPTW schedules vehicles from the main depot to supply points in order to load freight. The vehicles then deliver freight from supply points to customers. Solutions to this problem involve two types of decisions: the first ones assign vehicles to supply points while the second ones assign customer demands to vehicles. We define neighborhood structures for each of these two types of decisions. The *leg neighborhoods* aim to change the vehicle assignments to supply points, while *routing neighborhoods* move customer demands among vehicle routes.

Several studies of tabu search with multiple neighborhoods exist. In some studies (e.g., Dell'Amico and Trubian, 1993; Gaspero and Schaerf, 2007), all neighborhoods are evaluated simultaneously. In other ones (e.g., Xu et al., 2006; Hamiez et al., 2009), neighborhoods are explored in a serial way, one after another in either a fixed or randomized order. Our experiments show that these approaches do not work well for the TMZT-VRPTW (see details in Annex B.1). This is because, in our problem, each type of neighborhood is applied to a particular decision domain, whereas size is the main difference between neighborhoods in the literature on tabu search with multiple neighborhoods. Moreover, the two decision domains of our problem are related and impact each other. Therefore, we propose an approach where only one neighborhood type is used in a given iteration of the tabu search method. Our selection strategy of the neighborhood type at each iteration is probabilistic, the distribution of our probability function being biased by the structure of our problem and the state of the search in order to obtain the right balance of exploration within each neighborhood space. The bias of our function is implemented through a parameter r that specifies *the ratio of selecting routing neighborhoods to leg neighborhoods*. The value of this parameter is adjusted during the course of the algorithm to favor the best possible exploration of the problem solution space.

The general structure of the tabu search meta-heuristic (TS) we propose is introduced in Algorithm 3. First, an initial feasible solution p is generated using a greedy method seeking to fully utilize vehicles and minimize the total cost. At each iteration of the tabu search method, one neighborhood is selected probabilistically based on the current value of r, then the selected neighborhood is explored, and the best move is chosen (lines 7-8). This move must not be tabu, unless it improves the current best solution p_{best} (aspiration criterion). The algorithm adds the new solution to an elite set $\mathscr{E}$ if it improves on p_{best}. It also remembers the value of the parameter r when the new best solution was found (lines 9-13), and finally updates the elite set $\mathscr{E}$ by removing a solution based on its value and the difference between solutions (Section 4.5.7).

Initially, the search freely explores the solution space by selecting each neighborhood

Algorithm 3 Tabu search

```
1: Generate an initial feasible solution $p$
2: $p_{best} \leftarrow p$
3: Elite set $\mathcal{E} \leftarrow \emptyset$
4: Probability of selecting routing neighborhood with respect to leg neighborhood $r \leftarrow 1$
5: STOP $\leftarrow 0$
6: **repeat**
7:     A neighborhood is selected based on the value of $r$
8:     Find the best solution $p'$ in the selected neighborhood of $p$
9:     **if** $p'$ is better than $p_{best}$ **then**
10:        $p_{best} \leftarrow p'$
11:        $r_{best} \leftarrow r$
12:        Add $(p_{best}, r_{best})$ to the elite set $\mathcal{E}$; update $\mathcal{E}$
13:    **end if**
14:    $p \leftarrow p'$
15:    **if** $p_{best}$ not improved for $IT_{cNS}$ iterations **then**
16:        **if** $p_{best}$ not improved after $C_{cNS}$ consecutive executions of *Control* procedure **then**
17:            **if** $\mathcal{E} = \emptyset$ **then**
18:                STOP $\leftarrow 1$
19:            **else**
20:                Select randomly $(p, r_p)$ (and remove it) from the elite set $\mathcal{E}$
21:                Diversify the current solution $p$
22:                Set $r \leftarrow r_p$ and reset tabu lists
23:            **end if**
24:        **else**
25:            Apply *Control* procedure to update the value of $r$
26:            $p \leftarrow p_{best}$
27:        **end if**
28:    **end if**
29: **until** STOP
30: $p_{best} \leftarrow$ *Post-optimization*$(p_{best})$
31: return $p_{best}$
```

with equal probability. Whenever the best solution is not improved for IT_{cNS} TS iterations (line 25), the *Control* procedure is called to reduce the probability of selecting leg neighborhoods. Consequently, route neighborhoods are selected proportionally more often, which gives routing moves more opportunity to optimize trips. The search is re-initialized from the current best solution p_{best} after the execution of the *Control* procedure (line 26). Moreover, after C_{cNS} consecutive executions of this procedure without improvement of the current best solution p_{best}, a solution p is selected randomly and removed from the elite set $\mathscr{E}$ (line 20), and a *Diversification* mechanism is applied to perturb p (line 21). The value of r is reset to the value it had when the corresponding elite solution was found, and all tabu lists are reset to the empty state (line 22). The search then proceeds from the perturbed solution p. The search is stopped when the elite set $\mathscr{E}$ is empty. Finally, a post-optimization procedure is performed to potentially improve the current best solution p_{best} (line 30).

4.5.2 Search space

As described in Section 4.3, a solution is a set of work assignments, each work assignment consisting of a sequence of route legs linking supply points. The search space is thus made up of feasible and infeasible work assignments.

For a given solution p, let $c(p)$ denote the total travel cost of its work assignments, and let $q(p), w_c(p)$, and $w_s(p)$ denote the total violation of vehicle load, customer time windows, and supply-point time windows, respectively. The total vehicle-load violation is computed on a route leg basis with respect to the value Q, whereas the total violation of time windows of customers is equal to $\sum_{d \in p} max\{(a_d - l_d), 0\}$, and the total violation of time windows of supply points is equal to $\sum_{s \in p} max\{(t(s) - \eta - a_s), (a_s - t(s)), 0\}$, where a_d and a_s are the arrival time at customer demand d and supply point s, respectively.

Due to the time synchronization restrictions at supply points, the arrival time at the first customer d of each leg l starting at supply point s is always calculated as $a_d = t(s) + \delta(s) + c_{sd}$, no matter when the vehicle arrives at supply point s. This helps to prevent propagating time-window infeasibility among the legs of a work assignment.

Solutions are then evaluated according to the weighted fitness function $f(p) = c(p) + \alpha_1 q(p) + \alpha_2 w_c(p) + \alpha_3 w_s(p) + F * m$, where α_1, α_2, α_3 are penalty parameters adjusted dynamically during the search. The updating scheme is based on the idea of Cordeau et al. (2001). At each iteration, the value of α_1, α_2, and α_3 are modified by a factor $1+\beta > 1$. If the current solution is feasible with respect to load constraints, the value of α_1 is divided by $1+\beta$; otherwise it is multiplied by $1+\beta$. The same rule applies to α_2 and α_3 with respect to time window constraints of customers and supply points, respectively. In our algorithm, we set $\alpha_1 = \alpha_2 = \alpha_3 = 1$ and $\beta = 0.5$.

4.5.3 Initial solution

We sort the supply points (customer zones) and index them in increasing order of their opening time. Thus, if $t(s_1) \leq t(s_2)$, then $s_1 < s_2$ and vice-versa. We then construct an initial solution by building each work assignment sequentially. Each work-assignment construction consists of two phases: the first phase determines the first supply point for the current work assignment; the second phase creates sequentially each leg using a greedy algorithm.

In the first phase, the supply point s with earliest opening time and unserviced customers is assigned as the initial supply point of the first leg of the current work assignment. During the second phase, the first leg l is created using a greedy algorithm. If the leg l ends at a supply point s', we continue applying the greedy algorithm to build the next leg of l in which s' is now used as the initial supply point. Otherwise, if the leg l ends at the main depot, it means the current work assignment cannot be used anymore, and we return to the first phase to build another work assignment. This process is repeated until all customers are serviced (assigned to a vehicle route).

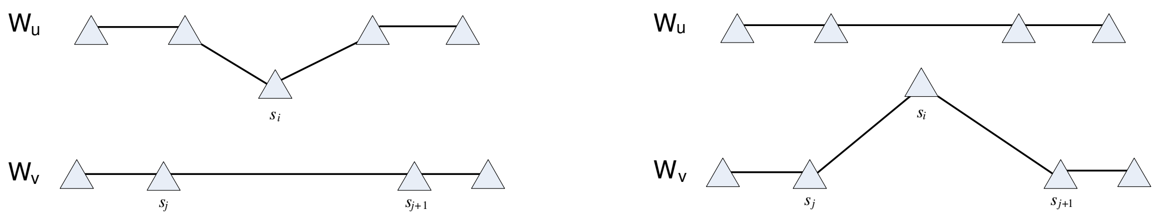
The greedy algorithm constructs each leg by attempting to minimize the cost and keep the vehicle working at full capacity as much as possible. Thus, for a given initial supply point s assigned to the leg, it finds a set of supply points $S' = \{s' \in \mathscr{S} | s'$ with unserviced customers and $t(s') > t(s)\}$. If $S' \neq \emptyset$, for each pair (s,s'), it creates an empty leg with s and s' as the initial and end supply point, respectively. It then assigns unserviced customers of customer zone s to this leg sequentially by applying the heuristic I1 of Solomon (1987) until the vehicle is full. When feasible legs exist, the one with minimum cost is selected. In the case there are no feasible legs or $S' = \emptyset$, it builds the last leg (s,g) by applying the heuristic I1 of Solomon (1987).

4.5.4 Neighborhoods

Leg neighborhoods focus on repositioning legs at supply points within the time restrictions. Let W_u be the work assignment assigned to vehicle u. Let s_{i-1} and s_{i+1} denote the predecessor and successor supply points, respectively, of s_i within a work assignment. The leg-move operators are:

- *Relocate supply point*. Consider two work assignments W_u and W_v as illustrated in Figure 4.2. For supply point $s_i \in W_u$, such that $s_i \notin W_v$, and for each two successive supply points s_j, $s_{j+1} \in W_v$, if $s_j < s_i < s_{j+1}$ then move supply point s_i from work assignment W_u to W_v locating it between s_j and s_{j+1}. All customers serviced by s_i on W_u are also moved to W_v.

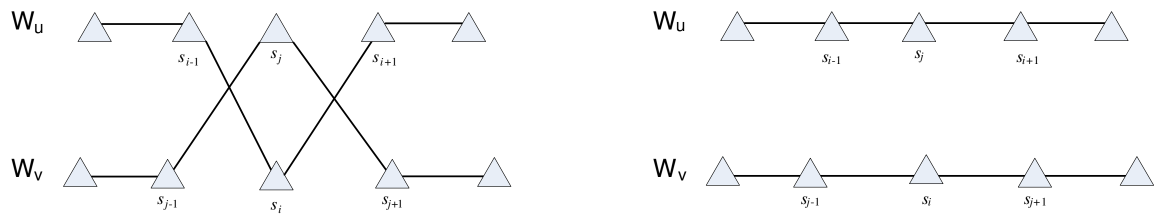
- *Exchange supply points*. Consider two work assignments W_u and W_v as illustrated in Figure 4.3. For supply points $s_i \in W_u$ and $s_j \in W_v$ such that $s_{i-1} < s_j < s_{i+1}$ and $s_{j-1} < s_i < s_{j+1}$, swap s_i and s_j together with their customers.



(a) Work assignments before *Relocate* (b) Work assignments after *Relocate*

Figure 4.2: Relocate supply point

When moving a supply point, all customers serviced by it are also moved. Therefore, the violations of load and time windows for customers are not changed by the move. The *move value* is thus defined as $\Delta f = \Delta c + F * \Delta m + \Delta w_s$. The three components of the summation are the difference in travel cost, the fixed cost of using vehicles, and the difference in violation of time windows at supply points between the value of the neighboring solution and the value of the current solution.



(a) Work assignments before *Exchange* (b) Work assignments after *Exchange*

Figure 4.3: Exchange supply points

Routing neighborhoods try to improve trip routing by using different intra- and inter-route neighborhoods commonly used in the VRPTW literature: Relocation, Exchange and 2-opt. For each move in each neighborhood, two customers are considered.

- *Relocation move*: one of the two customers is taken from its current position and inserted after the other one.

- *Exchange move*: two customers are swapped.

- *2-opt move*: for two customers in the same leg, the edges emanating from them are removed, two edges are added, one of which connects these two customers, and the other connects their successor customers. For two customers in different legs, the work assignment segments following them are swapped preserving the order of customers succeeding them in each segment.

Moving customers could change the travel cost and the number of vehicles, as well as the level of constraint violations of load, time windows of customers, and time windows of supply points. Consequently, the value of a routing move is defined as $\Delta f = \Delta c + F * \Delta m + \Delta q + \Delta w_c + \Delta w_s$. Note that for Δc, the change in the routing cost, may involve, beside a change in the routing cost between customers, a change in the routing cost from the last customer to the supply point at the end of the modified leg(s). For example, a routing move may impact on whether a vehicle has to go to a waiting station or not, therefore impacting the traveling cost from the last customer to its supply point.

4.5.5 Neighborhood selection strategy

The algorithm explores one neighborhood at each iteration. The neighborhood to explore is randomly selected among the five previously defined neighborhoods. The main issue in this case is to define a probability distribution for the neighborhoods as well as whether the probability distribution should evolve over time.

Using a fixed a priori distribution has drawbacks which limit the exploration capability of the algorithm. For example, this would mean that the algorithm will display the same behavior during the entire search. Moreover, the calibration of the probabilities would be extremely challenging and instance dependent. Indeed, too low routing-neighborhood probabilities (high leg-neighborhood probabilities) would result in an insufficient number of routing moves to adequately optimize the customer routes after the leg moves. The search may easily get stuck in the opposite case, as it needs to move to less-explored regions of the search space once a succession of routing moves have "optimized" trips.

Considering the drawbacks of having a fixed probability distribution for the selection of neighborhoods, we have elected to vary these probabilities as the search progresses. At the beginning of the search, both leg and routing neighborhoods are given the same probability of being selected, which allows the TS algorithm to freely explore the solution space. Given that the number of supply points is much smaller than the number of customers in most TMZT-VRPTW instances, the algorithm should perform more routing than leg moves to ensure adequate optimization of routes. Consequently, after the initial phase, the probability of selecting leg neighborhoods becomes lower than the probability of selecting routing neighborhoods. We control this probability distribution using the neighborhood-selection parameter r, assigning to a routing neighborhood the probability $r/(2+3r)$ of being selected, and to a leg neighborhood the probability $1/(2+3r)$ of being selected. The equal initial probabilities are then obtained by setting $r = 1$. The *Control* procedure in our algorithm varies the value of r during execution to monotonically reduce (increase) the probability of selecting leg (routing) neighborhoods after each IT_{cNS} iterations without improvement of the best solution. A linear scheme $r_{k+1} = r_k + \Delta_r$ is used, where Δ_r is a user defined parameter.

4.5.6 Tabu lists and tabu duration

We keep a separate tabu list for each type of move. Elements of a solution generated by a move are given a tabu status as follows:

- Leg moves:
  - Relocate supply point: the position of supply point s_i just inserted into work assignment W_v cannot be changed by another relocate supply point move while it is tabu.
  - Exchange supply points: supply points s_i and s_j just swapped cannot be swapped again while they are tabu.
- Routing moves:
  - Relocation move: the position of customer i just inserted after customer j, cannot be changed by the same type of move while it is tabu.
  - Exchange move: customers i and j just swapped cannot be swapped again while they are tabu.
  - 2-opt move: a 2-opt move applied to customers i and j cannot be applied again to the same customers while tabu.

A tabu status is assigned to each tabu list element for θ iterations, where θ is randomly selected from a uniform interval. Generally, the tabu status of a move stays so for a number of iterations proportional to the number of possible moves. Consequently, we use different intervals of tabu list size for leg and routing moves. Since there are $O(m' * |\mathscr{S}|)$ possible leg moves, we set the interval of tabu list size for leg moves to $[m'*|\mathscr{S}|/a_1, m'*|\mathscr{S}|/a_2]$, where m' is the number of vehicles used in the initial solution, and a_1 and a_2 are user-defined parameters.

In TMZT-VRPTW, each supply point has its own customer demands. Therefore, the number of iterations during which a routing move within the zone of a supply point s remains tabu is only counted each time the algorithm deals with customer demands in that zone. The interval of tabu list size for routing moves for each supply point s with

$|D_s|$ associated customer demands is therefore calculated as $[a_3 log_{10}(|D_s|), a_4 log_{10}(|D_s|)]$, where a_3 and a_4 are user defined parameters.

In our tabu search, a move declared tabu is accepted if it improves the current best solution.

4.5.7 Diversification strategy

A diversification strategy, based on an elite set and a frequency-based memory, moves the search to potentially unexplored promising regions when it begins to stagnate. In a nutshell, diversification aims to capitalize on the best attributes obtained so far by selecting a new working solution from the elite set and perturbing it based on long-term trends.

In more details, we use the elite set as a diversified pool of high-quality solutions found during the tabu search. The elite set starts empty and is limited in size. The quality and diversity of the elite set is controlled by the insertion of new best solutions produced by the tabu search and the elimination of the existing solutions in the elite set. The elimination is based on the Hamming distance $\Delta(p_1, p_2)$ measuring the number of customer positions that differ between solutions p_1 and p_2. (see, e.g., Ehmke et al., 2012; Vidal et al., 2012, for the utilization of the Hamming distance in other VRP settings).

The elimination of a solution from the elite set is considered each time a new best solution p_{best} is inserted. There are two cases. While the elite set is not yet full, we delete only when there exists a solution very similar to the new p_{best}, i.e., we delete the solution p with the smallest $\Delta(p, p_{best}) \leq 0.05(n + |\mathscr{S}|)$. This aims to balance the impact on pool quality and diversity. When the elite set is full, p_{best} replaces the solution p that is the most similar to it, i.e., the one with the smallest $\Delta(p, p_{best})$.

The long-term frequency memory keeps a history of the arcs most frequently added to the current solution. Let t_{ij} be the number of times arc (i, j) has been added to the solution during the search process. The frequency of arc (i, j) is then defined as $\rho_{ij} = t_{ij}/T$, where T is the total number of iterations executed so far.

Diversification then proceeds to perturb the search that starts from the solution taken from the elite set by removing arcs with high frequency and inserting arcs with low frequency. Thus, the evaluation of neighbor solutions is biased so as to penalize the arcs most frequently added to the current solution. More precisely, a penalty $g(\bar{p}) = \bar{C}(\sum_{(i,j)\in A_a} \rho_{ij} + \sum_{(i',j')\in A_r}(1-\rho_{i'j'}))$ is added to the evaluation of the fitness $f(\bar{p})$ (Section 4.5.2) of a neighbor $\bar{p}$ of the current solution p, where $\bar{C}$ is the average cost of all arcs in the problem, and A_a and A_r are the sets of arcs that are added to and removed from the solution p in the move to $\bar{p}$, respectively. The diversification mechanism is executed IT_{div} iterations.

4.5.8 *Post-optimization*

The best solution obtained through the tabu search is enhanced by applying a number of well-known local search route improvement techniques. Two are intra-route operators, the 2-opt of Lin (1965) and the Or-opt of Or (1976). The others are inter-route operators, the λ-interchange of Osman (1993), and the CROSS-exchange of Taillard et al. (1997). For the λ-interchange, we only consider the cases where $\lambda = 1$ and $\lambda = 2$ corresponding to the (1,0), (1,1), (2,0), (2,1), and (2,2)-interchange operators. A customer is re-allocated only to legs with the same initial supply point. The post-optimization procedure is executed for each customer zone separately.

The post-optimization procedure starts by applying in random order the five λ-interchange and CROSS-exchange inter-route operators. Each neighborhood is searched on all possible pairs of legs (in random order) of the same starting supply point and stopped on the first improvement. The solution is then modified and the process is repeated until no further improvement can be found. The search is then continued by locally improving each leg of the current starting supply point in turn. The intra-route 2-opt and Or-opt neighborhoods are sequentially and repeatedly applied until no more improvement is found.

4.6 Computational Results

The objective of the numerical experimentation is threefold. First, to study the impact of a number of major parameters and search strategies on the performance of the proposed algorithm in order to identify the most efficient ones. The second objective consists in evaluating the performance of the method through comparisons with currently published results. We finally analyze the impact of synchronization and vehicle fixed cost on solution quality.

Our tabu search algorithm is implemented in C++. Experiments were run on a 2.8 GHz Intel Xeon 4-core processor with 16GB of RAM. Six sets of ten instances each, generated by Crainic et al. (2012b), were used throughout the experiments. These Euclidean instances are identified as A1, A2, B1, B2, C1, and C2. The numbers of customer zones for these sets are 4, 8, 16, 32, 36, and 72, respectively. The numbers of customer demands are 400, 1600, and 3600 for set of type A, B, and C, respectively. Supply points, waiting stations, and customers are uniformly distributed in a square, with the X and Y coordinates in the interval [0, 100], [0, 200], and [0, 300] for set of type A, B, and C, respectively. The opening times of supply points are generated randomly in the [0, 14400] range, while the limited allowable waiting time at supply points $\eta = 100$. The vehicle-loading times at supply points are set to 30, for all supply points. The time window of each customer demand is then generated based on the opening time of the supply point to which it is assigned. The ready time and time window duration of each customer demand are generated randomly in the interval [0, 300] and [150, 450], respectively. The due time for delivery is thus set by adding the time window duration to the ready time. To ensure feasibility of movements from a supply point to its customer demands, a number of values are added to the ready times and due times: the opening time of the supply point to which the customer demand is assigned, the loading time at the supply point, and the smallest integer higher than the value of the distance between the customer demand and the supply point. One waiting station is set for every 100 customers in all instances. The fixed cost and the capacity of each vehicle are set to 500 and 100, respectively, for all instance sets.

4.6.1 Algorithm design and calibration

We aim for a general algorithmic structure avoiding instance-related parameter settings. We therefore defined settings as functions of problem size for the main parameters of the proposed algorithm, tabu tenure, neighborhood selection-control, diversification triggering, and size of the elite set.

4.6.1.1 Tabu tenure calibration

The intervals for the tabu list tenures for leg and routing moves were defined in Section 4.5.6 as [m'*$|\mathscr{S}|$/a_1, m'*$|\mathscr{S}|$/a_2] and [$a_3 log_{10}(|D_s|)$, $a_4 log_{10}(|D_s|)$], respectively. Using a large interval for routing moves, [10, 20], we tested different values for a_1 in the interval [6, 8] and a_4 in the interval [10, 12]. We observed that too large an interval is not productive as low values cannot prevent cycling, while high ones overly restrict the search path. We therefore set a_3 and a_4 to 7 and 10, respectively.

A similar process has been used to explore different values of a_1 in the integer interval [7, 9] and a_2 in the integer interval [4, 6]. using the leg-move tabu tag interval as defined above. We found that the most appropriate values for a_1 and a_2 are 7 and 5, respectively.

4.6.1.2 Calibration of the neighborhood selection probabilities

Adjustments to the neighborhood selection probabilities depend on two parameters: IT_{cNS}, the number of consecutive iterations without improvement of the best solution (this number triggers the execution of the *Control* procedure that modifies probabilities), and Δ_r, the adjustment factor of the neighborhood-selection parameter r.

The value of IT_{cNS} is defined as a function of the problem size. This value should be large enough to give each customer demand and supply point in each leg the possibility to be moved. Thus, $IT_{cNS} = e_1 * (m' * |\mathscr{S}| + n)$, where m' is the number of vehicles used in the initial solution, $|\mathscr{S}|$ and n are the numbers of supply points and customer demands, respectively, and e_1 is a user defined parameter. Similarly, Δ_r, the amplitude of the modifications in the probabilities, is set to be proportional to the ratio of the number of customers with the number of supply points. Thus, $\Delta_r = e_2 \log_{10}(n/|S|)$, where e_2 is a user defined parameter.

Searching for a good combination of values for e_1 and e_2 concerns balancing between exploration and exploitation. On one hand, the higher the value of IT_{cNS}, the more chances customer demands and supply points are to be moved between routes, thus favoring exploration. On the other hand, a too high IT_{cNS} value may waste time in useless moves. We experimented with different values of e_1 in the integer interval [1, 5] and e_2 in the integer interval [1, 7]. Three runs were performed for each instance for 1 million iterations. Computational results for each combination of values (e_1, e_2) over all 60 instances are summed up in Table 4.1, which displays the average gaps to the previous best known solutions (BKS) of the solutions obtained by each combination.

Table 4.1: Performance comparison between (e_1, e_2) combinations

e_1	e_2						
	1	2	3	4	5	6	7
1	-2.21%	-2.42%	-2.98%	-3.09%	-3.11%	-3.18%	-3.15%
2	-2.27%	-2.45%	-3.24%	-3.21%	-3.17%	-3.14%	-3.12%
3	-2.34%	-2.73%	-3.37%	-3.39%	-3.44%	-3.36%	-3.28%
4	-2.46%	-2.74%	-3.31%	-3.36%	-3.41%	-3.28%	-3.24%
5	-2.41%	-2.78%	-3.32%	-3.37%	-3.39%	-3.29%	-3.19%

Table 4.1 indicates that (3, 5) is the most appropriate combination for (e_1, e_2), improving the solution quality by 3.44% on average. We also observed that executing the algorithm with r greater than 50 $\log_{10}(n/|S|)$ yields an average improvement of the best solution of less than 0.1%, while requiring about 37.3% more time. Based on these results, we used (e_1, e_2) = (3, 5) and $r_{max} = 50\log_{10}(n/|S|)$, the maximum value of r, in the remaining experiments.

4.6.1.3 Neighborhood search strategy

The neighborhood exploration strategy specifies how the different neighborhoods can be used to explore efficiently the solution space of the problem. Several such strategies can be envisioned and we actually experimented with quite a number of them before selecting the one introduced in Section 4.5.5. (For brevity, we placed in Annex B.1 the description of the alternate strategies we explored and the numerical results supporting our selection.)

The neighborhood-search strategy also specifies which move in the neighborhood is to be chosen at each iteration. We studied two strategies, first and best improvement, respectively. The former chooses the first neighbor solution that improves the objective function as the next starting solution, while the latter chooses the best neighbor thus requiring to evaluate all neighbors. The customers in each route are searched sequentially.

Table 4.2 reports comparison results between these two strategies. Corresponding average gaps to the previous BKS and average computation times are displayed in Columns *GAP to BKS* and *Time (min)*, respectively.

Table 4.2: Comparative performances between neighborhood search strategies

Problem set	Best improvement		First improvement			
	Elite set size = 5		Elite set size = 5		Elite set size = 10	
	GAP to BKS	Time (min)	GAP to BKS	Time (min)	GAP to BKS	Time (min)
A1	-3.01%	18	-1.95%	16	-2.06%	24
A2	-6.22%	10	-4.42%	8	-4.57%	14
B1	-4.47%	60	-2.82%	45	-2.98%	69
B2	-4.71%	39	-3.23%	27	-3.35%	46
C1	-3.78%	165	-1.14%	115	-1.24%	176
C2	-3.82%	104	-1.89%	80	-1.91%	108
Average	-4.33%	66	-2.57%	49	-2.68%	73

The computational results in Table 4.2 show that using the same elite set size (=5), best improvement gives better solutions for all sets, while first improvement has a lower computation time. One observes, however, that the difference in computation time is smaller than the difference in solution quality, indicating that the best improvement strategy yields a better algorithm. More importantly, even doubling size of the elite set (=10), which results in longer computation times, the solutions of the first improvement strategy are still significantly worse than the solutions of the best improvement strategy.

4.6.1.4 Elite set calibration, diversification, and run-time behavior

We now turn to the parameters characterizing the diversification procedure and the elite set utilization, and examine their impact on the performance of the algorithm. We complete this part of reporting on the computational experiments, by examining the run-time behavior of the proposed tabu search meta-heuristic.

Four variants of the algorithm were studied corresponding to the different ways to set an elite solution as the new working solution and the inclusion, or not, of the diversification phase. The first two variants simply select an elite solution p at random and re-start the algorithm from it. The *Diversification* mechanism described in Section 4.5.7 is applied in the last two variants to diversify from the elite solution p.

The initialization of the r parameter following the selection of p is a component common to the four variants. We studied two alternatives where r was set to either the full or half the value at which p was found, respectively (i.e., $r = r_p$ or $r = r_p/2$). The size of the elite set is relevant for the *Diversification* mechanism only. Three values were tested, 1, 5, and 10.

Similar to previous experiments, we used formulas dependent on the problem dimensions for IT_{div} and C_{cNS}, which determine for how long exploration can proceed. Thus, the number of diversification phases is set to $IT_{div} = m' * |\mathscr{S}| + n$, where m' is the number of vehicles used in the initial solution, and $|\mathscr{S}|$ and n are the numbers of supply points and customers, respectively. We also set the number of consecutive executions of the *Control* procedure without improvement of the best solution to $C_{cNS} = min(3\log_{10}(n/|S|), (r_{max} - r)/\Delta_r)$, which keeps the value of C_{cNS} sufficiently high during the course of the algorithm, even though *Control* procedure is started with different values of r (remember that $r_{max} = 50\log_{10}(n/|S|))$. Intuitively, in the beginning, r is small and C_{cNS} takes the value $3\log_{10}(n/|S|)$, while when r becomes large enough, C_{cNS} takes the value $(r_{max} - r)/\Delta_r$.

Table 4.3 displays the performance comparison between the four variants with the three different values for the elite set size. For each variant and size of the elite set, the table shows the average gaps to the previous BKS of the average cost of the best solutions of all instances, together with the corresponding average computation time in minutes over 10 runs.

As expected, results indicate that guidance using elite solutions contributes significantly to improve the performance of the algorithm. Without using the elite set, the algorithm requires the lowest computation effort but produces solutions with the lowest average (improvement) gap to the previous BKS of -2.96%, compared to all the variants using the elite

Table 4.3: Performance comparison between diversification settings

Elite set size	Without diversification				With diversification			
	1st variant		2nd variant		3rd variant		4th variant	
	$r = r_p$		$r = r_p/2$		$r = r_p$		$r = r_p/2$	
	GAP to BKS	Time	GAP to BKS	Time	GAP to BKS	Time	GAP to BKS	Time
0	-2.96%	24	-	-	-	-	-	-
1	-3.18%	29	-3.03%	38	-3.93%	35	-3.97%	47
5	-3.39%	39	-3.34%	47	-4.33%	66	-4.24%	71
10	-3.53%	50	-3.61%	59	-4.35%	92	-4.25%	98

set. Comparing the two variants corresponding to the two values at which r is reset, one observes that the solution quality is not very sensitive to this value, but computing effort is increasing when the value of r is lower ($r = r_p/2$).

One observes that the third and fourth variants are significantly better in terms of finding high quality solutions. This indicates that the long-term memory and the diversification mechanism added to the algorithm are important features for high performance. Moreover, setting the size of the elite set to 5 achieves a better balance between solution quality and computation time, compared to a larger size of 10. Indeed, doubling the size of the elite set improves only slightly the solution quality, 0.02%, but requires 39% more time. We therefore set the size of the elite set to 5 and reset $r = r_p$.

Figures 4.4a and 4.4b illustrate the impact of the diversification procedure and the elite set on search performance. These figures report the solution value and the value of r, respectively, along the iterations of the tabu search algorithm on instance C2-1. The number of customers n and supply points $|\mathscr{S}|$ for the instance C2-1 are 3600 and 72, respectively, therefore, the maximum value of the neighborhood-selection parameter $r_{max} = 80$. The value of r was reset six times (Figure 4.4b) corresponding to the re-initialization of the search using six different solutions extracted from the elite set. Prior to the first reset of r, the algorithm proceeded from the solution generated by the initialization procedure, the value of r increasing from 1 to 50 during that search sequence. Aligning vertically the

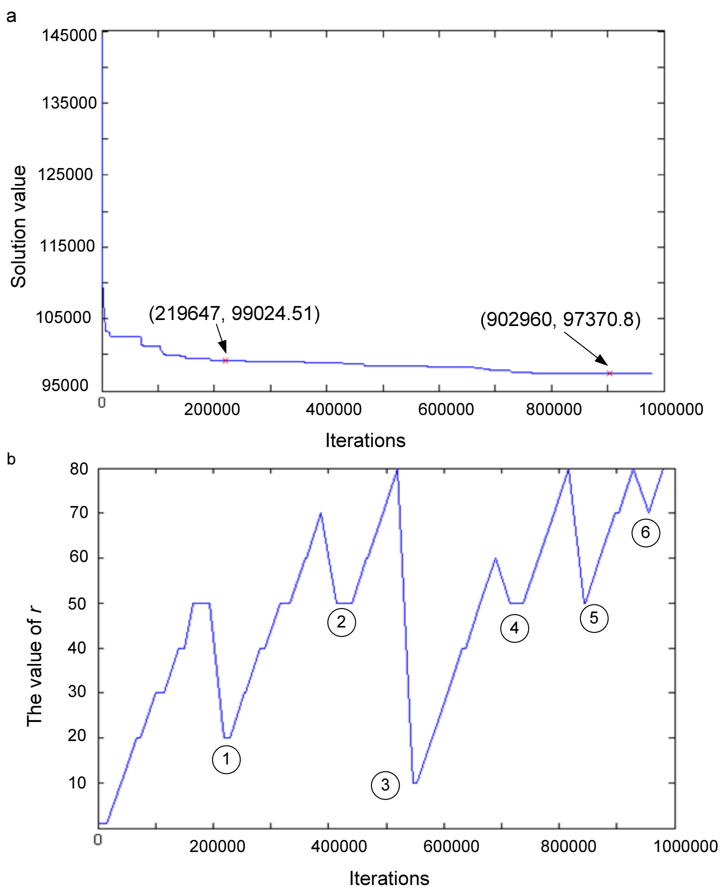


Figure 4.4: Impact of diversification and elite-set utilization on algorithm performance

iteration axis of Figures 4.4b and Figure 4.4a, one obtains the representation of how the solution quality varies with each search sequence between two consecutive resets. We can see that without using the elite set, the search stopped quite early, namely at iteration 219,647, with a best solution value of 99,024.51. On the other hand, using the elite set and the diversification procedure, new improved solutions with substantially better costs were found through the next five search sequences when a solution from the elite set was used to diversify the search. The best solution was found at iteration 902,960 (with $r = 70$) from the fifth solution extracted from the elite set. Its value is 97,370.8, improving the solution quality by 1.67% compared to the case without using the elite set. This improvement indicates the efficiency of the diversification procedure and the elite set applied in the algorithm. The search stopped at iteration 980,723.

To further emphasize the importance of the diversification feature of the proposed meta-

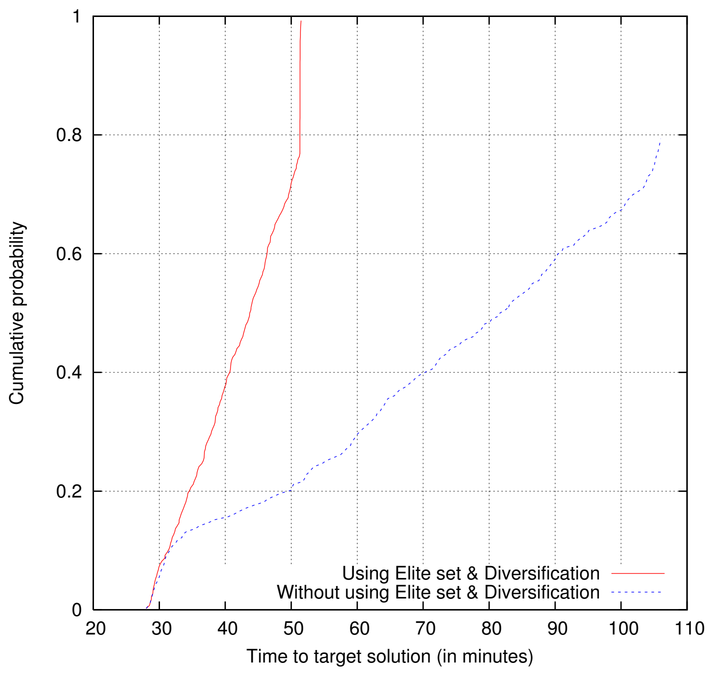


Figure 4.5: Time-to-target plot for the instance C2-1

heuristic on performance, we focused on the run-time behavior of the proposed algorithm and constructed a time-to-target (TTT) plot (Aiex et al., 2007). A TTT plot is generated by executing an algorithm κ times and measuring the time required to reach a solution at least as good as a target solution. The running times are sorted in increasing order. The i-sorted running time t_i is associated with a probability $\rho_i = (i-1/2)/\kappa$ and the points $z_i = [t_i, \rho_i]$, $i = 1, ..., \kappa$, are plotted. Each plotted point indicates the probability (vertical axis) for the algorithm to achieve the target solution in the indicated time (horizontal axis). Once again, we used the C2-1 instance, with a best known solution value of 97,350.2, for this experiment. The plot in Figure 4.5 compares two algorithms: the first does not use the elite set and diversification mechanisms while the second applies them. This plot is produced by the execution of both algorithms 200 times, using a target value of 1.5% above the best known solution, i.e., value of 98,810.45, together with a maximum computational time of 120 minutes. We observe in Figure 4.5 that both algorithms have similar behaviors until about 31 minutes. During that period, the second algorithm has not yet applied the elite set and diversification mechanisms. Starting from that point, the probability for the second algorithm to find the target value starts to be greater than for the first algorithm. Moreover, within 120 minutes, the first algorithm reaches the target with a maximum probability of 0.8. While the second algorithm always reaches the target and requires less than 55 minutes. This analysis indeed indicates that better performances are obtained when the elite set and diversification procedure are integrated to the search algorithm.

4.6.2 *Comparing with results in the literature*

This section compares the performance of the proposed tabu search algorithm with results available in the literature. We initiate this analysis examining the effectiveness of the routing component of the search, as well as the contribution of each set of decisions to the solution quality.

4.6.2.1 *Effectiveness of routing neighborhoods*

We have run the proposed tabu search algorithm using only our routing neighborhoods on the Solomon and Desrosiers (1988) 56 VRPTW 100-customer instances. All parameters related to the supply points were discarded. Therefore, $IT_{cNS} = 3*n$, and $IT_{div} = n$. Moreover, when the elite set is not yet full, the solution which is most similar to p_{best} is deleted, i.e., the solution p with the smallest $\Delta(p, p_{best}) \leq 0.05*n$. We executed three runs for each instance, and report the best one.

Table 4.4 compares the performances obtained by this striped-down version of the tabu search algorithm with the results of other tabu search algorithms reported in the survey on the VRPTW of Bräysy and Gendreau (2005b). For completion sake, we also included the results of the currently best-metaheuristic, the hybrid genetic algorithm with adaptive diversity management of Vidal et al. (2013a) (best in five repetitions). The first column gives the name of the authors of the study. Columns R1, R2, C1, C2, RC1, and RC2 present the average number of vehicles and average total distance with respect to the six groups of problem instances, respectively. Finally, the rightmost column indicates the cumulative number of vehicles (CNV) and cumulative total distance (CTD) over all 56 instances. The performance of the routing neighborhood search of our algorithm appears satisfactory, considering that all algorithms in the literature were tailored for the VRPTW, most of them (except Backer and Furnon, 1997; Tan et al., 2000) actually aiming first to reduce the number of vehicles. We do not, as our algorithm treats vehicles through supply-point neighborhoods not considered in this experiments, and we do not compete with the other meta-heuristics on this count. We do compete with respect to the total distance, though, outperforming six out of the eleven meta-heuristics.

Using routing neighborhoods only is not sufficient for the problem setting at hand, however. We have actually also run our algorithm by fixing the supply point-to-vehicle assignment of the initial solution and applying only routing neighborhoods. We obtained worse solutions than the proposed tabu search handling both neighborhood types, with an average error gap of 5.91% (see details in Annex B.1). Therefore, although we have created efficient routing neighborhoods, we cannot get good solutions by considering routing only. Both sets of decisions are important. Our algorithm handles both neighborhood types efficiently, and is able to enhance the solution quality significantly by using long-term memory and diversification mechanisms. Performance comparisons of the proposed algorithm with the literature are presented in the next subsection.

4.6.2.2 The performance of the proposed algorithm

The performance of the proposed tabu search meta-heuristic is evaluated by comparing its output with published results on the instances provided by Crainic et al. (2012b). For brevity, only aggregated results are provided in this section. (Details can be found in Annex B.2.)

Table 4.5 displays the comparison between the (best) results reported by Crainic et al. (2012b) and those obtained by the proposed tabu search meta-heuristic run 10 times for each instance. For comparison sake, we report the best results we obtained for the 10 runs, but provide both best and average results in the detailed tables of Annex B.2. Table 4.5 gives the best results (*Best* column), the number of vehicles (*#Vehicles* column), the number of times vehicles move directly from one customer zone to another customer zone without using waiting stations (*DM* column), and the number of times waiting stations are used for moving between customer zones (*MWS* column). Average computation times in minutes are displayed in the *Time* column, while the corresponding gaps to the previous

Table 4.4: Performance comparison with meta-heuristics for the VRPTW

Authors	R1	R2	C1	C2	RC1	RC2	CNV/CTD
Garcia et al. (1994)	12.92	3.09	10.00	3.00	12.88	3.75	436
	1317.70	1222.60	877.10	602.30	1473.50	1527.00	65977.00
Rochat and Taillard (1995)	12.25	2.91	10.00	3.00	11.88	3.38	415
	1208.50	961.72	828.38	589.86	1377.39	1119.59	57231.00
Potvin et al. (1996b)	12.50	3.09	10.00	3.00	12.63	3.38	426
	1294.50	1154.40	850.20	594.60	1456.30	1404.80	63530.00
Taillard et al. (1997)	12.17	2.82	10.00	3.00	11.50	3.38	410
	1209.35	980.27	828.38	589.86	1389.22	1117.44	57523.00
Chiang and Russell (1997)	12.17	2.73	10.00	3.00	11.88	3.25	411
	1204.19	986.32	828.38	591.42	1397.44	1229.54	58502.00
Backer and Furnon (1997)	14.17	5.27	10.00	3.00	14.25	6.25	508
	1214.86	930.18	829.77	604.84	1385.12	1099.96	56998.00
Schulze and Fahle (1999)	12.25	2.82	10.00	3.00	11.75	3.38	414
	1239.15	1066.68	828.94	589.93	1409.26	1286.05	60346.00
Tan et al. (2000)	13.83	3.82	10.00	3.25	13.63	4.25	467
	1266.37	1080.24	870.87	634.85	1458.16	1293.38	62008.00
Cordeau et al. (2001)	12.08	2.73	10.00	3.00	11.50	3.25	407.00
	1210.14	969.57	828.38	589.86	1389.78	1134.52	57556.00
Lau et al. (2003)	12.17	3.00	10.00	3.00	12.25	3.38	418
	1211.55	1001.12	832.13	589.86	1418.77	1170.93	58477.00
Vidal et al. (2013a)	11.92	2.73	10.00	3.00	11.50	3.25	405
	1210.69	951.51	828.38	589.86	1384.17	1119.24	57196
TS	12.66	3.72	10.00	3.00	12.50	4.25	442
	1231.40	986.70	850.34	609.11	1396.84	1136.72	58425.00

Table 4.5: Comparative performances on Crainic et al. (2012b) instances

Problem set	Crainic et al. (2012b)					TS						GAP to BKS
	Best	#Vehicles	DM	MWS	Time	Avg 10	Best 10	#Vehicles	DM	MWS	Time	%
A1	18575	24	1	36	5	18043.84	18010.52	23	2	36	18	-3.04
A2	15411	19	2	42	3	14495.23	14440.05	17	5	42	10	-6.29
B1	55653	51	14	180	22	53124.00	53036.13	45	31	168	60	-4.62
B2	47396	39	20	193	10	45272.96	45074.16	34	40	177	39	-4.79
C1	117426	87	39	423	50	112975.99	112810.90	79	73	394	165	-3.92
C2	101570	64	68	434	23	97747.18	97585.83	60	108	391	104	-3.91
Average	59338.50	47.33	24	218	18.83	56943.20	56826.27	43	43.17	201.50	66	-4.42

BKS are given in the last column.

TS produces high-quality solutions, with an average improvement gap of -4.42% compared to the previous BKS, yielding better solutions than Crainic et al. (2012b) on all instances. Moreover, the proposed tabu search meta-heuristic produces consistently good solutions, the average solution quality being very close to that of the best one. Noteworthy, as shown in Table 4.3, without post-optimization and using the same size of the elite set (=5), TS obtains an average gap to the previous BKS of -4.33%. Thus, the post-optimization process helped to improve solution quality by 0.09% on average, requiring only a few extra seconds.

TS produces solutions that not only require fewer vehicles (8.87% on average), but also require less usage of waiting stations. More precisely, the TS we propose used waiting stations 12050 times compared to 13071 times in Crainic et al. (2012b), vehicles move directly from one customer zone to another customer zone on 2590 occasions compared to 1449 occasions in Crainic et al. (2012b). Thus, in our TS, 17.69% of the times vehicles move directly to another customer zone without using waiting stations, compared to 9.98% in Crainic et al. (2012b). Moreover, the proposed TS provides better customer routing (i.e., traveling cost), with an average gap of -1.30% to the previous BKS. This advantage goes beyond the simple numerical performance in terms of cost to propose a distribution system that is structurally better with less vehicles traveling for idling purposes.

4.6.3 Synchronization at supply points

Waiting stations are introduced as locations where vehicles can wait when the direct move would get them at supply points sooner than the opening times. In this section, we analyze the impact of waiting stations on solution quality.

In all previous experiments, traveling cost and time were equivalent. In order to analyze the impact of waiting without modifying the travel costs, we explicitly introduced into the model a waiting cost measure related to the customer-to-waiting station movement generating the need to wait. Thus, the waiting cost for a (customer demand, supply point) pair is computed as a percentage of the total cost from the customer to the waiting station and from the latter to the supply point. We performed 6 runs corresponding to a percentage equal to 10%, 20%, 30%, 40%, 50%, and 100%.

The experiment was run on the C2 set, which includes the largest instances in terms of the numbers of customers, supply points, and waiting stations. Table 4.6 sums up the solution-quality variations for the six cases compared to the case without waiting costs. The table displays the solution-quality variations in terms of the total cost, traveling cost, number of vehicles, and synchronization requirements at supply points. One observes that higher waiting cost results in vehicles performing longer routes (the routing cost increases) to avoid going to waiting stations. Consequently, the number of direct moves increases, and accordingly, the number of moves using waiting stations is reduced. Moreover, it also requires more vehicles, resulting in a higher total cost.

4.6.4 Impact of vehicle fixed cost

The objective of the TMZT-VRPTW is to minimize the total cost of the system, comprised on the total "fixed" cost of using vehicles and the "variable" cost of delivering the loads to customers by the vehicles. In this section, we analyze the impact of vehicle fixed cost on solution quality.

In all previous experiments, a value of 500 was set to the fixed cost F of using a vehicle. We performed three additional runs setting the fixed cost F equal to 0, 250, and 1000. Table 4.7 sums up the solution-quality variations in terms of total cost, traveling cost, number of vehicles, synchronization requirements at supply points, and number of legs. One observes

Table 4.6: Impact of waiting cost on solution quality

Impact on solution quality	Increase the cost of waiting by					
	10%	20%	30%	40%	50%	100%
Total cost	2.87%	5.15%	7.46%	9.75%	11.77%	20.23%
Routing cost	3.78%	6.78%	10.04%	13.13%	15.39%	26.59%
#Vehicles	0.82%	1.47%	1.64%	2.13%	3.61%	5.91%
DM	3.91%	15.14%	23.24%	31.30%	34.19%	58.41%
MWS	-5.45%	-8.68%	-10.90%	-12.97%	-14.14%	-21.38%

that vehicle fixed cost plays an important role in the problem solution. Setting the fixed cost to zero represents the case where the objective of the problem aims only to minimize the traveling cost. The experimental results in Table 4.7 illustrate clearly that the traveling cost is lower for this case compared to the other cases where vehicle fixed costs are considered. On the other hand, this case puts on the road the largest number of vehicles, each vehicle performing exactly one leg. While this case eliminates the need for waiting stations, it would correspond to the highest cost in terms of manpower, as well as impact on congestion and emissions. Introducing a vehicle fixed cost dramatically reduces the number of vehicles and generates solutions where each vehicle performs a sequence of several of legs and, sometimes stops at waiting stations. Increasing the vehicle fixed cost continues to decrease the number of vehicles, as well as the utilization of the waiting stations, while increasing the traveling cost. These modifications are not dramatic, however. Thus, for example, doubling the value of F (i.e., from 250 to 500, or from 500 to 1000 in this experiment) increases the traveling cost by less than 1% and reduces the number of vehicles by less than 2.5%. Generally speaking, once vehicle fixed costs are introduced, the pattern of solutions is not very sensitive to its value.

Table 4.7: Impact of vehicle fixed cost on solution quality

Fixed cost F	Total cost	Traveling cost	#Vehicles	DM	MWS	#Legs
0	17584.74	17584.74	290	0	0	290
250	46105.71	35105.71	44	41	202	287
500	56943.06	35443.06	43	42	202	287
1000	77740.31	35740.31	42	44	201	287

4.7 Conclusion

We proposed a tabu search meta-heuristic for the Time-dependent Multi-zone Multi-trip Vehicle Routing Problem with Time Windows. The proposed model formulation provided the means to identify clearly the main components of the decision set of the problem. We could thus propose a tabu search method that works on multiple neighborhoods, which are used to improve both the routing and the assignment of routes to vehicles. The selection of neighborhoods is dynamically adjusted along the search to keep the balance between exploration and exploitation. Moreover, a diversification strategy guided by an elite set and a frequency-based memory is introduced to not only provide a certain level of diversity to the search, but also help incorporate good attributes into newly created solutions.

Experimental results illustrated clearly the superior performance of the proposed methodology compared to the literature. It yields higher quality solutions in terms of both required number of vehicles and traveling cost. In addition, the utilization of waiting stations, resulting from the synchronization restriction at supply points, was significantly reduced. The quality of these results in terms of number of required vehicles, costs, times, and frequency of direct movements are not only extremely satisfying when evaluating the proposed meta-heuristic, but are also interesting from a managerial point of view. Indeed, they indicate that our tabu search will prove efficient in actual applications, contributing to provide system managers with solution requiring less vehicles to perform efficiently the same amount of work. This is particularly interesting when City Logistics systems are contemplated as our results indicate a reduction in the presence of vehicles on the streets of the city and, thus, in their negative impact on congestion and environment.

Chapter 5

MULTI-ZONE MULTI-TRIP PICKUP AND DELIVERY PROBLEM WITH TIME WINDOWS AND SYNCHRONIZATION

This chapter has been published as technical report: P. K. Nguyen, T. G. Crainic, and M. Toulouse. Multi-zone Multi-trip Pickup and Delivery Problem with Time Windows and Synchronization. Technical report, Publication CIRRELT-2014-18, Centre interuniversitaire de recherche sur les réseaux d'entreprise, la logistique et le transport, Université de Montréal, Montréal, QC, Canada, 2014.

Résumé: Dans ce travail nous considérons les systèmes de logistique urbaine à deux niveaux dans lesquels nous prenons en considération à la fois le trafic entrant et sortant. Ce problème avec trafic entrant et sortant n'a pas encore été pris en considération dans les modèles et algorithmes sur les tournées de véhicules. Le problème étudié, appelé Multi-zone Multi-trip Pickup and Delivery Problem with Time Windows and Synchronisation, comporte deux ensembles de décisions entrelacés: des décisions de routage qui déterminent la séquence de clients visités par chaque itinéraire d'un véhicule, des décisions de planification qui prévoient les mouvements de véhicules entre les installations à l'intérieur de restrictions liées à la synchronisation temporelle. Nous proposons un algorithme de recherche tabou intégrant plusieurs voisinages adaptés aux types de décisions du problème. Pour évaluer l'algorithme proposé, des tests ont été effectués sur des problèmes de référence que nous avons conçus pour ce problème, et qui ont jusqu'à 72 installations et 7200 demandes de clients. De plus, comme il n'y a pas de résultats disponibles dans la littérature pour ce problème, nous avons aussi évalué la performance de notre méthode par comparaison avec des résultats publiés sur le problème de tournées de véhicules avec voyages de retour. L'algorithme proposé est compétitif avec d'autres méta-heuristiques avec et sans fenêtres de temps.

Abstract: In this paper, we consider two-tier City Logistics systems accounting for both the inbound and outbound traffic, that have not been taken into account in models and algorithms for vehicle routing research. The problem under study, called the Multi-zone Multi-trip Pickup and Delivery Problem with Time Windows and Synchronization, has two sets of intertwined decisions: the routing decisions which determine the sequence of customers visited by each vehicle route, the scheduling decisions which plan movements of vehicles between facilities within time synchronization restrictions. We propose a tabu search algorithm integrating multiple neighborhoods targeted to the decision sets of the problem. To assess the proposed algorithm, tests have been conducted on the first benchmark instances of the problem which have up to 72 facilities and 7200 customer demands. As no previous results are available in the literature for the problem, we also evaluate the performance of the method through comparisons with currently published results on the vehicle routing problem with backhauls. The proposed algorithm is competitive with other

meta-heuristics both with and without time windows.

Keywords: Multi-trip Pickup and Delivery problem with Time windows; Synchronization; Tabu search; Multiple neighborhoods

5.1 Introduction

Research for city logistics is receiving more and more attention, with most researchers focusing on ***inbound*** freight flow. Hence, either the single-tier case or the multiple tiers case, freight flow is often considered in the direction from regions outside the city, say ***external zones***, to the city center, and the objective is to minimize the total cost of the associated system. In reality, freight is moved in, out, and through a city. When dedicated fleet is used for each type of freight flow, this might be simple to implement and manage, but it results in the presence of more vehicles and empty trips on the streets of the city, that eventually increases the operating costs and environmental pollution. In this paper, we therefore make an effort on integrating the ***outbound*** freight flow, shipping freight from the city center to external zones, with the 'classical' inbound freight flow into a single City Logistics system. We study a new problem, the so-called *Multi-zone Multi-trip Pickup and Delivery Problem with Time Windows and Synchronization*(MZT-PDTWS), which addresses an integrated system servicing both traffic types while sharing the same fleet of vehicles.

The MZT-PDTWS originates from planning the operations of two-tiered City Logistics systems (Crainic et al., 2009). In such systems, inbound loads are sorted and consolidated at first-tier facilities located on the outskirts of the city, moved to second-tier facilities, called satellites, located close to or within the city-center area, by a fleet of medium-sized vehicles. In the second tier, a smaller-capacity fleet performs tours to pick up outbound demands within the city center and transport them to satellites. Once at satellites, planned appropriate pairs of first-tier and second-tier vehicles transfer inbound and outbound demands to each other according to cross-docking, without intermediate storage. The first-tier vehicles then move the outbound demands to external zones, while the second-tier vehicles deliver the inbound demands to designated customers situated within the city center. This integration of inbound and outbound operations helps to reduce the number of empty vehicle movements in both tiers, as well as freight traffic in the city center. Since satellites are used as intermediate transshipment points for the freight distribution, the synchronization of the operations of first-tier and second-tier vehicles at satellites becomes one of the most constraining aspects of the problem.

In the MZT-PDTWS setting, a homogeneous fleet of vehicles operates out of a single garage to pickup- or delivery-customer demands associated to a given set of satellites. Each ***customer demand*** is defined by its specific location, commodity volume, as well as a particular service time requirement. Customer demands are divided into two categories, pickup (*backhaul*) demands representing outbound demands, and delivery (*linehaul*) demands representing inbound demands. The arrival of first-tier vehicles at a given time period define the set of delivery-customer demands to be serviced, and the time required to unload and transfer the freight thus define the availability period during which second-tier vehicles must arrive at the satellite and load. Second-tier vehicles must therefore synchronize their arrivals at satellites with these availability periods for loading the planned freight. The integration of outbound freight flow concerns pickup demands, which should be collected by second-tier vehicles, and brought to assigned satellites at their availability periods. Consequently, at satellites, second-tier vehicles may either unload pickup demands (if any), or load delivery demands (when available), or do both. In case the vehicle loads the planned freight, it undertakes a route delivering freight to the assigned customers. Otherwise, when the vehicle leaves a satellite empty (i.e., only unloads at this satellite), it either undergoes a route collecting new pickup demands, or moves empty to another satellite for loading delivery demands, or to the garage to end its activity. The waiting stations may be used by vehicles to wait for its next appointment at a satellite. The MZT-PDTWS corresponds to the planning of the activities of second-tier vehicles.

The MZT-PDTWS is a generalization of the Time-dependent Multi-zone Multitrip Vehicle Routing Problem with Time Windows (TMZT-VRPTW) which has been studied in Crainic et al. (2012b) and Nguyen et al. (2013). The TMZT-VRPTW considers only the inbound traffic flow, while the MZT-PDTWS generalizes the TMZT-VRPTW by considering an additional traffic flow, say outbound, shipping freight in the opposite direction, i.e., from the city center to destinations outside the city limits. Both traffic types share a same fleet of vehicles. Furthermore, satellites would also be shared. This version has not been investigated in the literature. Such services sharing services increases the synchronization challenges at satellites, the complexity to manage the traffic of vehicles into and out of satellites in particular, as well as to route vehicles doing both pickup and delivery operations through satellites. We make three contributions: 1) we present the first formulation for the MZT-PDTWS, 2) we propose a tabu search meta-heuristic to solve the problem, 3) we introduce a new benchmark with instances up to 72 supply points and 7200 customer demands to show the performance of the proposed method.

The remainder of the paper is organized as follows. Section 5.2 contains a detailed problem description. Section 5.3 reviews the literature. The problem formulation is then provided in Section 5.4. Details of the proposed methodology are described in Section 5.5. Computational results are then reported and analyzed in Section 5.6, while conclusions and future works are considered in Section 5.7.

5.2 Problem Description

In this paper, we address the integration of outbound traffic into city logistics. The physical flow from customers to satellites is called the pickup phase. The process from satellites to customers is called the delivery phase. Vehicles operate according to the ***Pseudo-Backhaul*** strategy described in Crainic et al. (2012a), in which any delivery or pickup phase must be completed before another one may be started. Each satellite may operate at several periods during the planning period considered. For the sake of simplicity, we define ***supply points*** as particular combinations of satellites and availability time periods in our model.

The MZT-PDTWS can be described as follows. There is a garage, or main depot, g, a set of pickup-customer demands $p \in \mathscr{C}^P$, a set of delivery-customer demands $d \in \mathscr{C}^D$, a set of waiting stations $w \in \mathscr{W}$, and a set of supply points $s \in \mathscr{S}$. A traveling cost (or travel time) $c_{i,j}$ is associated with each pair of i and j where $i, j \in g \cup \mathscr{C}^D \cup \mathscr{C}^P \cup \mathscr{S} \cup \mathscr{W}$. We use the terms cost, travel time, and distance interchangeably. Each supply point $s \in \mathscr{S}$ has a no-wait, hard opening time window $[t(s) - \eta, t(s)]$, specifying the earliest and latest times the vehicle may be at s, respectively. Hence, the vehicle must not arrive at s sooner than $(t(s) - \eta)$ and no later than $t(s)$; in the former case, the vehicle has to wait at a waiting station $w \in \mathscr{W}$ before moving to s. Each customer demand is serviced by exactly one supply point. The supply point that services each delivery-customer demand is fixed and known in advance. For each pickup-customer demand $p \in \mathscr{C}^P$, it is given a set of supply points $\mathscr{S}_p \in \mathscr{S}$ that can service p. Therefore, it is required to assign each pickup-customer demand p to exactly one supply point $s \in \mathscr{S}_p$. Consequently, each supply point s may service a group of either pickup-customer demands $\mathscr{C}_s^P \subseteq \mathscr{C}^P$, or delivery-customer demands $\mathscr{C}_s^D \subseteq \mathscr{C}^D$ or both. Thus, it refers to movements where all freight collected from pickup-customer demands in $\mathscr{C}_s^P$ have to be transported to s and all freight delivered to delivery-customer demands in $\mathscr{C}_s^D$ have to be loaded at s. Accordingly, each supply point s requires an unloading time $\varphi'(s)$, which is the time required to unload freight picked up at a set of customer demands in $\mathscr{C}_s^P$, and a loading time $\varphi(s)$, which is the time required to load freight to service a set of customer demands in $\mathscr{C}_s^D$. For each customer demand $i \in \mathscr{C}^P \cup \mathscr{C}^D$, we use $(i, q_i, \delta(i), [e_i, l_i])$ to mean that customer demand i requires a quantity q_i of demand in the hard time window $[e_i, l_i]$ with a service time $\delta(i)$.

In general, once arriving at a supply point, the vehicle in the MZT-PDTWS may either unload pickup demands or load delivery demands or do both. Figure 5.1 represents these activities of a vehicle at a supply point s. The dashed lines stand for the empty move.

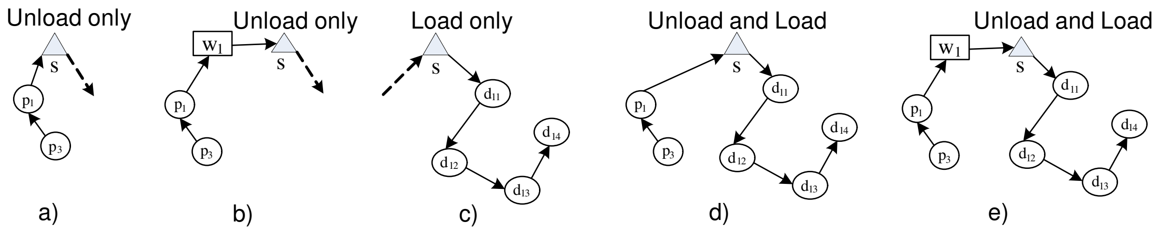


Figure 5.1: Activities at supply points

Figures 5.1a and 5.1b depict instances of ‘unload only’ operation in which after arriving at the supply point with the collected freight from pickup-customer demands, the vehicle unloads all freight, then it leaves the supply point empty for its next trip or goes to the depot to end its activity. From the last serviced pickup-customer demand, the vehicle may go directly to the supply point s as shown in Figure 5.1a if it can arrive at s within the time window $[t(s) - \eta, t(s)]$. Otherwise, in case the direct move gets the vehicle to s sooner than $t(s) - \eta$, as shown in Figure 5.1b, the vehicle goes to a waiting station, and wait there in order to get to s within the time window of s. Figure 5.1c represents the case of 'load only' operation in which the vehicle goes empty to supply point s and loads freight. Figures 5.1d and 5.1e depict instances of 'unload and load' operation in which at a supply point s, after unloading all freight collected from pickup-customer demands, the vehicle starts to load freight, it then leaves s to deliver freight to designated delivery-customer demands.

Let a pickup (delivery) leg be a route that links one or several pickup- (delivery-) customer demands of the same type with a supply point. Thus, we define two types for each pickup and delivery legs as well as their feasibility rules as follows:

- ***Direct-pickup leg*** is a route run by a vehicle that goes to one or several pickup-customer demands to collect freight and goes to the supply point directly to unload all freight (see Figure 5.1a).

- ***Indirect-pickup leg*** is similar to the case of direct-pickup leg, except that after servicing the last pickup-customer demand, the vehicle has to go to a waiting station and wait there due to the synchronization requirement at the supply point, then ends at the supply point to unload all freight (see Figure 5.1b).

  A pickup leg assigned to the supply point s is feasible if the vehicle with a total load not exceeding Q can arrive at s within its time window $[t(s)-\eta, t(s)]$ after servicing a subset of pickup-customer demands in $\mathscr{C}_s^P$ within their time windows.

- ***Single-delivery leg*** is a route run by a vehicle that arrives empty at a supply point s to load freight and delivers all freight to one or several delivery-customer demands in $\mathscr{C}_s^D$ (see Figure 5.1c). A single-delivery leg assigned to the supply point s is feasible if the vehicle arrives empty at s at time $t' \in [t(s)-\eta, t(s)]$ to load freight with a total load not exceeding Q, then leaves s at time $t'+\varphi(s)$ to perform the delivery for serving a subset of customer demands in $\mathscr{C}_s^D$ within their time windows.

- ***Coordinate-delivery leg*** is a route run by a vehicle that starts empty at a supply point s for loading freight, then delivers all freight to designated delivery-customer demands in $\mathscr{C}_s^D$, given that this vehicle does both unload and load at supply point s, i.e, it does a direct-pickup leg (see Figure 5.1d) or an indirect-pickup leg (see

Figure 5.1e) at the same supply point s right before this coordinate-delivery leg. A coordinate-delivery leg assigned to the supply point s is feasible if the vehicle arrives at s at time $t' \in [t(s)-\eta, t(s)]$ to unload all collected freight, it then starts to load delivery demands at time $t'+\varphi'(s)$ with a total load not exceeding Q, leaves s at time $t(s)+\varphi'(s)+\varphi(s)$ to perform the delivery for serving a subset of customer demands in $\mathscr{C}_s^D$ within their time windows.

A sequence of legs, starting and ending at the main depot, assigned to a vehicle is called a ***work assignment***. For the sake of simplicity, from now on, the terms *vehicle* and *work assignment* are used interchangeably. Figure 5.2 illustrates a four-leg work assignment, where s_1, s_2, s_3 are supply points, g and w_1 are respectively the main depot and waiting station, $\mathscr{C}_{s_1}^P = \{p_1, p_2, p_3, p_4, p_5\}$, $\mathscr{C}_{s_1}^D = \{d_1, d_2, d_3, d_4, d_5\}$, $\mathscr{C}_{s_2}^P = \{p_6, p_7, p_8, p_9, p_{10}\}$, $\mathscr{C}_{s_2}^D = \{d_6, d_7, d_8, d_9, d_{10}, d_{11}\}$, $\mathscr{C}_{s_3}^P = \{p_{11}, p_{12}, p_{13}, p_{14}, p_{15}\}$, and $\mathscr{C}_{s_3}^D = \{d_{12}, d_{13}, d_{14}, d_{15}\}$. The dashed lines stand for the empty arrival. This vehicle performs a sequence of four legs $\{r_1, r_2, r_3, r_4\}$ where $r_1 = \{s_1, d_1, d_3, d_4\}$ is a single-delivery leg, $r_2 = \{p_6, p_8, p_9, w_1, s_2\}$ is an indirect-pickup leg, $r_3 = \{s_2, d_6, d_9, d_8, d_7\}$ is a coordinate-delivery leg, and $r_4 = \{p_{11}, p_{13}, p_{12}, s_3\}$ is a direct-pickup leg. This vehicle first moves empty from the depot g to supply point s_1. Once at s_1, this vehicle start loading delivery demands. After loading for a time $\varphi(s_1)$, it leaves s_1 to service customer demands (d_1, d_3, d_4) in $\mathscr{C}_{s_1}^D$, then moves empty to pickup customer zone $\mathscr{C}_{s_2}^P$ for collecting freight at pickup-customer demands (p_6, p_8, p_9). In order to arrive at s_2 within its opening time window, after collecting freight from customer demand p_9, this vehicle has to go to waiting station w_1 and wait there. Once at s_2 (assuming arrival time is t), it does both unloading and loading operations: (1) at time t, it starts unloading and keeps doing for a time $\varphi'(s_2)$, and (2) then from time $t+\varphi'(s_2)$, it loads delivery demands and continues loading for a time $\varphi(s_2)$, after which it leaves s_2 to service customer demands (d_6, d_9, d_8, d_7) in $\mathscr{C}_{s_2}^D$. After servicing the last customer demand d_7, it moves empty to pickup customer zone $\mathscr{C}_{s_3}^P$. There, after loading freight at pickup-customer demands (p_{11}, p_{13}, p_{12}), this vehicle moves to supply point s_3 within the opening time window of s_3. Once at s_3, this vehicle starts unloading freight for a duration of $\varphi'(s_3)$. At the end, this vehicle moves empty back to the depot g to complete its task.

The MZT-PDTWS can be seen as the problem of (1) assigning pickup-customer de-

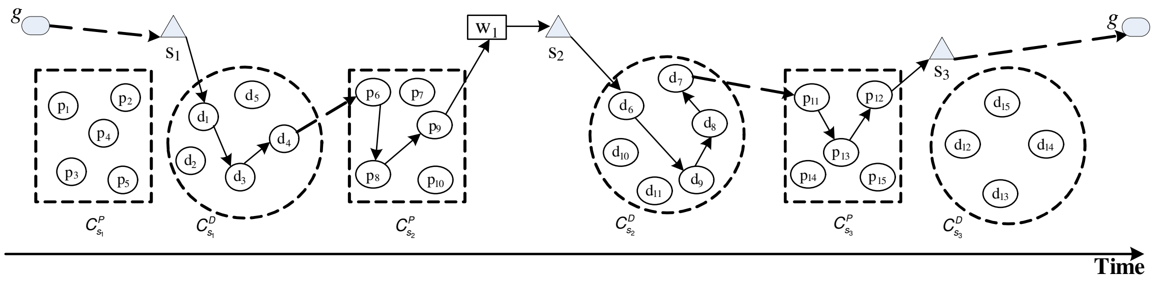


Figure 5.2: A four-leg work assignment illustration

mands to supply points, and (2) determining a set of pickup and delivery legs and an assignment of each leg to one vehicle, such that each vehicle can perform several legs sequentially. The objective is to minimize the total cost, which is comprised of routing cost and fixed cost on the use of vehicles, while the following conditions are satisfied:

1. Every vehicle starts and ends its leg sequence at the main depot g;
2. Each pickup-customer demand p is assigned to exactly one supply point $s \in \mathscr{S}_p$;
3. Every vehicle required to service customer demands in $\mathscr{C}_s^P \cup \mathscr{C}_s^D$ must reach its supply point $s \in \mathscr{S}$ within its no-wait, hard opening time window. Assume the arrival time at s is t; Once at s:
    - if the vehicle is not empty, i.e., it is containing goods picked up from customer demands in $\mathscr{C}_s^P$, it has to unload them first. The vehicle starts unloading goods at time t, and continues unloading for a duration of $\varphi'(s)$, after which it may either:
        - (1) load goods shipped from external zones for a duration of $\varphi(s)$ and then leave s to deliver goods to customer demands in $\mathscr{C}_s^D$, or
        - (2) move empty either to another pickup customer zone to collect goods, or directly to another supply point for loading goods, or
        - (3) go to the main depot g to complete its task;

- otherwise, the vehicle starts to load goods to service customer demands in $\mathscr{C}_s^D$. It starts loading goods at time t and continues loading for a duration of $\varphi(s)$, after which it leaves s to deliver goods to customers in $\mathscr{C}_s^D$. After performing a trip within the delivery customer zone $\mathscr{C}_s^D$, the vehicle may continue its movement as either the situation (2) or (3) described above;

4. Every customer demand is visited by exactly one leg with a total load not exceeding Q, and each customer demand $i \in \mathscr{C}^D \cup \mathscr{C}^P$ is serviced within its hard time window $[e_i, l_i]$, i.e., a vehicle may arrive before e_i and wait to begin service, but must not arrive later than l_i.

5.3 Literature Review

In this paper, we present a new variant of the VRP. It extends the Time-dependent Multi-zone Multi-trip Vehicle Routing problem with Time Windows (TMZT-VRPTW) by considering an additional type of customer demands. The TMZT-VRPTW just addresses inbound demands, resulting in only one type of customers, i.e., delivery-customer demands, while the MZT-PDTWS considers delivery and pickup-customer demands as it addresses both inbound and outbound demands. Crainic et al. (2009) pioneered the introduction of the TMZT-VRPTW and proposed a decomposition-based heuristic approach to solve it. The general idea is to solve each customer-zone routing out of each supply point subproblem independently, and then put the created vehicle tours together into multi-tour routes by solving a minimum cost network flow problem. Yet, as routing decisions affect the supply point assignment decisions and vice versa, these two decision levels are intertwined and should not be solved separately. Nguyen et al. (2013) later investigated an alternative approach that addresses these two decisions simultaneously, in a tabu search framework. Thus, in comparing to the previous approach, it yields solutions with higher quality up to 4.42% in term of total cost, requiring not only less vehicles, but also less usage of waiting stations.

In the context of Pickup and Delivery problems, there has been extensive research on variants of the problem with respect to additional and different types of constraints which occur in real-world applications; see, e.g., a number of surveys (Savelsbergh and Solomon, 1995; Parragh et al., 2008a,b; Berbeglia et al., 2007, 2010) and book (Toth and Vigo, 2002). Based on the difference in transportation endpoints, Parragh et al. (2008a,b) divided them into two subclasses: the first refers to transportation of goods from the depot to delivery (*linehaul*) customers and from pickup (*backhaul*) customers to the depot; the second refers to those problems where goods are transported between pickup and delivery locations. As we follow the Pseudo-Backhaul strategy in which any delivery or pickup phase must be completed before another one may be started, our problem belongs to the first subclass.

In reviewing the literature, the first subclass of Pickup and Delivery problems includes the single demand case where linehaul and backhaul customers are disjoint, and the combined case where the same customer has both a pickup and a delivery demand. In the former case, there are either problems in which linehaul customers of a given trip have to be serviced before backhaul customers of the same trip (Osman and Wassan, 2002; Brandão, 2006), that are denoted as Vehicle Routing problem with Backhauls (VRPB); or problems in which linehaul and backhaul customers may be visited in any order (Dethloff, 2002; Ropke and Pisinger, 2006), that are denoted as Vehicle Routing problem with Mixed linehauls and Backhauls (VRPMB). In the combined case, each customer may be either visited exactly once (Nagy and Salhi, 2005; Dell'Amico et al., 2006) or visited twice, one for delivery and one for pickup (Salhi and Nagy, 1999; Gribkovskaia et al., 2001). Problems in this case are denoted as Vehicle Routing problem with Simultaneous Delivery and Pickup.

In our problem, a customer, which is identified by a location, may have both types of demands: pickup and delivery. These demands of a customer might be available at different periods with different commodity volumes, thus we define them as customer demands. However, pickup and delivery phases are completed separately. The VRPB can be considered as a subproblem of our problem. More precisely, VRPB can be seen as the problem of routing delivery-customer demands of supply point s and pickup-customer demands assigned to supply point s' where $t(s) < t(s')$ in our problem, while time synchronization restrictions at supply points and waiting stations are not considered. Starting from the supply point s, the vehicles first deliver freight to delivery-customer demands. Then, they collect new freight at pickup-customer demands and bring to supply point s'. Two variants with and without time windows at customers are considered in the VRPB literature. However, the number of studies dealing with the time window variant is relatively smaller than those without time window.

Vehicle Routing problem with Cross-Docking (VRPCD) is a VRP variant sharing synchronization of vehicles' operations requirement with our problem. In general, the VRPCD involves transporting products from a set of suppliers to their corresponding customers via a cross-dock. More precisely, products from the suppliers are picked up by a fleet of vehicles, consolidated at the cross-dock (i.e., classified into a certain group according to their destination), and immediately delivered to customers by the same set of vehicles, without delay or storage. A supplier and its corresponding customers are not necessarily served by the same vehicle. At the cross-dock, for each vehicle the unloading must be completed before reloading starts. There may exist constraint on simultaneous arrival at cross-dock for all the vehicles (Lee et al., 2006; Liao et al., 2010) or the arrival dependency among the vehicles is determined by the consolidation decisions (Wen et al., 2008). As our problem, each vehicle in the VRPCD operates pickup and delivery phases separately. However, there are also differences between the VRPCD and the MZT-PDTWS.

In the VRPCD, each vehicle performs a sequence of two trips, i.e., pickup and then delivery, using cross-dock as intermediate storage. While in our problem, there are neither limitation on the number of trips nor the requirement for the ordering between pickup and delivery trips performed by each vehicle. Vehicles in our problem are synchronized at multiple locations (supply points) rather than a single location (the cross-dock) in the VRPCD. The synchronized arrival time of all or several vehicles at the cross-dock is considered as a variable in the VRPCD, which is determined by the consolidation decisions, while it is given in advance in our problem.

5.4 Model Formulation

5.4.1 Notation

We define for each supply point $s \in \mathscr{S}$:

- $A_s^{DS} = \{(d,s) | s' \in \mathscr{S}, d \in \mathscr{C}_{s'}^D, t(s') < t(s), e_d + \delta(d) + c_{d,s} \leq t(s)\ \}$ contains all the arcs (d,s) from delivery-customer demands d to the supply point s such that $e_d + \delta(d) + c_{d,s} \leq t(s)$, i.e., the vehicle could arrive at s before the opening time $t(s)$. By including this constraint, we eliminate inadmissible arcs, thus reduce the set;

- $A_s^{PS} = \{(p,s) | p \in \mathscr{C}_s^P\}$ contains all the arcs from pickup-customer demands p to the supply point s such that $s \in S_p$.

- $A_s^{S-} = \{(s',s) | t(s) - \eta \leq t(s') - \eta + \varphi'(s') + c_{s',s} \leq t(s)\}$ contains all the arcs (s',s) from any supply points s' to the supply point s such that $t(s) - \eta \leq t(s') - \eta + \varphi'(s') + c_{s',s} \leq t(s)$, i.e., the vehicle could travel directly from s' to s when it only unloads at s' (leaves s' empty);

- $A_s^{S+} = \{(s,s') | t(s') - \eta \leq t(s) - \eta + \varphi'(s) + c_{s,s'} \leq t(s')\}$ contains all the arcs (s,s') from the supply point s to any supply points s' such that $t(s') - \eta \leq t(s) - \eta + \varphi'(s) + c_{s,s'} \leq t(s')$, i.e., the vehicle could travel directly from s to s' when it only unloads at s;

- $A_s^{SP} = \{(s,p) | t(s) - \eta + \varphi'(s) + c_{s,p} \leq l_p, p \in \mathscr{C}^P\}$ contains all the arcs (s,p) from the supply point s to any pickup-customer demands $p \in \mathscr{C}^P$ such that $t(s) - \eta + \varphi'(s) + c_{s,p} \leq l_p$, i.e., the vehicle could arrive at p from s before the due time l_p when it only unloads at s (there exists arcs (s,p), i.e., the vehicle goes from s to p, only if the vehicle leaves s empty, i.e., only unloads at s);

- $A_s^{SD} = \{(s,d) | d \in \mathscr{C}_s^D\}$ contains all the arcs from the supply point s to any delivery-customer demands $d \in \mathscr{C}_s^D$;

We define for each delivery-customer demand $d \in \mathscr{C}_s^D, s \in \mathscr{S}$:

- $A_d^{DS} = \{(d,s') | s' \in \mathscr{S}, e_d + \delta(d) + c_{d,s'} \leq t(s')\}$ contains all the arcs (d,s') from the delivery-customer demand d to any supply points $s' \in \mathscr{S}$ such that $e_d + \delta(d) + c_{d,s'} \leq t(s')$, i.e., the vehicle could arrive at s' from d before the opening time $t(s')$;

- $A_d^{DP} = \{(d,p) | p \in \mathscr{C}^P, e_d + \delta(d) + c_{d,p} \leq l_p\}$ contains all the arcs (d,p) from the delivery-customer demand d to any pickup-customer demands $p \in \mathscr{C}^P$ such that $e_d + \delta(d) + c_{d,p} \leq l_p$, i.e., the vehicle could arrive at p before the due time l_p;

- $A_d^{D^+} = \{(d,d') | d' \in \mathscr{C}_s^D, e_d + \delta(d) + c_{d,d'} \leq l_{d'}\}$ contains all the arcs (d,d') from the delivery-customer demand d to any other delivery-customer demands d' of the same zone $\mathscr{C}_s^D$ such that $e_d + \delta(d) + c_{d,d'} \leq l_{d'}$;

- $A_d^{D^-} = \{(d',d) | d' \in \mathscr{C}_s^D, e_{d'} + \delta(d') + c_{d',d} \leq l_d\}$ contains all the arcs (d',d) from any delivery-customer demands d' of the same zone $\mathscr{C}_s^D$ to the delivery-customer demand d such that $e_{d'} + \delta(d') + c_{d',d} \leq l_d$;

We define for each pickup-customer demand $p \in \mathscr{C}^P$:

- $A_p^{PS} = \{(p,s) | s \in \mathscr{S}_p\}$ contains all the arcs from the pickup-customer demand p to its available supply point $s \in \mathscr{S}_p$;

- $A_p^{PD} = \{(p,d) | d \in \mathscr{C}^D, e_p + \delta(p) + c_{p,d} \leq l_d\}$ contains all the arcs (p,d) from the pickup-customer demand p to any delivery-customer demands d such that $e_p + \delta(p) + c_{p,d} \leq l_d$;

- $A_p^{P^+} = \{(p,p') | p' \in \mathscr{C}^P, e_p + \delta(p) + c_{p,p'} \leq l_{p'}, S_p \cap S_{p'} \neq \emptyset\}$ contains all the arcs (p,p') from the pickup-customer demand p to any other pickup-customer demands p' such that (1) $S_p \cap S_{p'} \neq \emptyset$, i.e., p' has at least one available supply point in common with p, and (2) $e_p + \delta(p) + c_{p,p'} \leq l_{p'}$;

- $A_p^{P^-} = \{(p',p) | p' \in \mathscr{C}^P, e_{p'} + \delta(p') + c_{p',p} \leq l_p, S_p \cap S_{p'} \neq \emptyset\}$ contains all the arcs (p',p) from any pickup-customer demands p' to the pickup-customer demand p such that (1) $S_p \cap S_{p'} \neq \emptyset$, i.e., p' has at least one available supply point in common with p, and (2) $e_{p'} + \delta(p') + c_{p',p} \leq l_p$;

- $A_p^{DP} = \{(d,p) | d \in \mathscr{C}^D, e_d + \delta(d) + c_{d,p} \leq l_p\}$ contains all the arcs (d,p) from any delivery-customer demands d to the pickup-customer demand p such that $e_d + \delta(d) + c_{d,p} \leq l_p$;

- $A_p^{SP} = \{(s,p)|t(s) - \eta + \varphi'(s) + c_{s,p} \leq l_p\}$ contains all the arcs (s,p) from any supply points s to the pickup-customer demand p such that $t(s) - \eta + \varphi'(s) + c_{s,p} \leq l_p$, i.e., the vehicle could arrive at p from s before the due time l_p when it only unload at s;

We define for each waiting station $w \in \mathscr{W}$:

- $A^{W-} = \{(i,w)|i \in \{\mathscr{S} \cup \mathscr{C}^D \cup \mathscr{C}^P\}, w \in \mathscr{W}\}$ contains all the arcs from pickup-customer demands, delivery-customer demands and supply points to waiting stations;
- $A^{WS} = \{(w,s)|w \in \mathscr{W}, s \in \mathscr{S}\}$ contains all the arcs from waiting stations to supply points;

For the depot g, we define:

- $A^{DG} = \{(d,g)|d \in \mathscr{C}^D\}$ contains all the arcs from delivery-customer demands d to the main depot g;
- $A^{GS} = \{(g,s)|s \in \mathscr{S}\}$ contains all the arcs from the main depot g to supply points;
- $A^{GP} = \{(g,p)|p \in \mathscr{C}^P\}$ contains all the arcs from the main depot g to pickup-customer demands p;

Let F stand for fixed cost for operating a vehicle. The set of available vehicles is denoted by $\mathscr{K}$. The maximal number of arcs included in any vehicle is given by e and we define $\mathscr{R}$ as $\{1,...,e\}$. Also, M is a large positive constant.

5.4.2 Formulation

The MZT-PDTWS is defined on a space-time network $\mathscr{G} = (\mathscr{V}, \mathscr{A})$, where $\mathscr{V}$ is the set of nodes, and the arcs in $\mathscr{A}$ stand for the possible movements between these nodes. Set $\mathscr{V}$ is made up of the main depot g and the sets of customer demands, supply points and waiting stations, i.e., $\mathscr{V} = g \cup \mathscr{C}^P \cup \mathscr{C}^D \cup \mathscr{S} \cup \mathscr{W}$. Set $\mathscr{A} = \cup_{s \in \mathscr{S}} [A_s^{SD} \cup A_s^{SP} \cup A_s^{S^+}] \cup_{d \in \mathscr{C}^D} [A_d^{DS} \cup A_d^{DP} \cup A_d^{D^+}] \cup_{p \in \mathscr{C}^P} [A_s^{PS} \cup A_s^{PD} \cup A_s^{P^+}] \cup A^{W^-} \cup A^{WS} \cup A^{DG} \cup A^{GS} \cup A^{GP}$, which consists of admissible arcs.

We define the following decision variables:

- x^r_{ijk}, a binary variable that takes value 1 if arc $(i,j) \in \mathscr{A}$ is traversed by vehicle k and appears in the rth position of the work assignment of vehicle k, and value 0 otherwise;

- y_{ps}, a binary variable that takes value 1 if pickup-customer demand $p \in \mathscr{C}^P$ is assigned to supply point $s \in \mathscr{S}$, and value 0 otherwise;

- z_{sk}, a binary variable that takes value 1 if vehicle k unloads at supply point s, and value 0 otherwise.

Note that we preliminary set $y_{ps} = 0 \, \forall p \in \mathscr{C}^P, s \notin S_p$ given that such s does not service p. Demands at each supply point $s \in \mathscr{S}$, waiting station $w \in \mathscr{W}$ and the main depot g are equal to zero, i.e, $q_s = q_w = q_g = 0$. For convenience, we set demand at each delivery node $d \in \mathscr{C}^D$: $q_d = -q_d < 0$. In addition,

B_{ik}	is the starting time of service at customer demand $i \in \mathscr{C}^P \cup \mathscr{C}^D$ by vehicle k;
B_{sk}	is the arrival time of vehicle k at supply point $s \in \mathscr{S}$;
B_{iwk}	is the arrival time of vehicle k at waiting station $w \in \mathscr{W}$ from a supply point, a delivery-customer demand, or a pickup-customer demand $i \in \{\mathscr{S} \cup \mathscr{C}^D \cup \mathscr{C}^P\}$;
Q_{ik}	is the load of vehicle k when leaving $i \in \mathscr{V}$;
L_{sk}	is the load of vehicle k when arriving at supply point $s \in \mathscr{S}$.

We set $Q_{gk} = 0 \, \forall k \in \mathscr{K}$ as the vehicle leaves the depot empty.

The MZT-PDTWS can then be formulated as the following:

$$\text{Minimize} \sum_{k\in\mathscr{K}} \sum_{(i,j)\in\mathscr{A}} c_{ij} \sum_{r\in\mathscr{R}} x^r_{ijk} + F \sum_{k\in\mathscr{K}} \left(\sum_{s\in\mathscr{S}} x^1_{gsk} + \sum_{p\in\mathscr{C}^P} x^1_{gpk} \right) \tag{5.1}$$

$$\text{S.t.} \sum_{r\in\mathscr{R}} \left(x^r_{sgk} + \sum_{(s,d)\in A^{SD}_s} x^r_{sdk} + \sum_{w\in\mathscr{W}} x^r_{swk} + \sum_{(s,s')\in A^{S+}_s} x^r_{ss'k} + \sum_{(s,p)\in A^{SP}_s} x^r_{spk} \right) \leq 1 \qquad \forall s \in \mathscr{S}, k \in \mathscr{K} \tag{5.2}$$

$$x^1_{gsk} + \sum_{r\in\mathscr{R}} \left(\sum_{w\in\mathscr{W}} x^r_{wsk} + \sum_{(s',s)\in A^{S-}_s} x^r_{s'sk} + \sum_{(p,s)\in A^{PS}_s} x^r_{psk} + \sum_{(d,s)\in A^{DS}_s} x^r_{dsk} \right) = \sum_{r\in\mathscr{R}} \left(x^r_{sgk} + \sum_{(s,d)\in A^{SD}_s} x^r_{sdk} + \sum_{w\in\mathscr{W}} x^r_{swk} + \sum_{(s,s')\in A^{S+}_s} x^r_{ss'k} + \sum_{(s,p)\in A^{SP}_s} x^r_{spk} \right) \quad \forall s\in\mathscr{S}, k\in\mathscr{K} \tag{5.3}$$

$$\sum_{s\in\mathscr{S}} x^1_{gsk} + \sum_{p\in\mathscr{C}^P} x^1_{gpk} = \sum_{r\in\mathscr{R}} \left(\sum_{d\in\mathscr{C}^D} x^r_{dgk} + \sum_{s\in\mathscr{S}} x^r_{sgk} \right) \quad \forall k\in\mathscr{K} \tag{5.4}$$

$$\sum_{r\in\mathscr{R}} \sum_{s\in\mathscr{S}} x^r_{wsk} = \sum_{r\in\mathscr{R}} \left(\sum_{d\in\mathscr{C}^D} x^r_{dwk} + \sum_{p\in\mathscr{C}^P} x^r_{pwk} + \sum_{s\in\mathscr{S}} x^r_{swk} \right) \quad \forall w\in\mathscr{W}, k\in\mathscr{K} \tag{5.5}$$

$$\sum_{s\in\mathscr{S}_p} y_{ps} = 1 \quad \forall p\in\mathscr{C}^P \tag{5.6}$$

$$\sum_{k\in\mathscr{K}} \sum_{r\in\mathscr{R}} x^r_{psk} \le y_{ps} \quad \forall p\in\mathscr{C}^P, s\in\mathscr{S} \tag{5.7}$$

$$x^r_{pwk} + y_{ps} \le x^{r+1}_{wsk} + 1 \quad \forall p\in\mathscr{C}^P, s\in\mathscr{S}_p, w\in\mathscr{W}, r\in\mathscr{R}, k\in\mathscr{K} \tag{5.8}$$

$$\sum_{k\in\mathscr{K}} \sum_{r\in\mathscr{R}} x^r_{pp'k} + y_{ps} \le y_{p's} + 1 \quad \forall p,p'\in\mathscr{C}^P, p\ne p', s\in\mathscr{S}_p \tag{5.9}$$

$$\sum_{k\in\mathscr{K}} \sum_{r\in\mathscr{R}} \left(\sum_{(p,p')\in A^{P+}_p} x^r_{pp'k} + \sum_{w\in\mathscr{W}} x^r_{pwk} + \sum_{s\in\mathscr{S}_p} x^r_{psk} \right) = 1 \quad \forall p\in\mathscr{C}^P \tag{5.10}$$

$$\sum_{k\in\mathscr{K}} \sum_{r\in\mathscr{R}} \left(\sum_{(s,p)\in A^{SP}_p} x^r_{spk} + \sum_{(p',p)\in A^{P-}_p} x^r_{p'pk} + \sum_{(d,p)\in A^{DP}_p} x^r_{dpk} \right) + \sum_{k\in\mathscr{K}} x^1_{gpk} = 1 \quad \forall p\in\mathscr{C}^P \tag{5.11}$$

$$\sum_{k\in\mathscr{K}} \sum_{r\in\mathscr{R}} \left(\sum_{(d,d')\in A^{D+}_d} x^r_{dd'k} + \sum_{(d,p)\in A^{DP}_d} x^r_{dpk} + \sum_{(d,s')\in A^{DS}_d} x^r_{ds'k} + \sum_{w\in\mathscr{W}} x^r_{dwk} + x^r_{dgk} \right) = 1 \quad \forall d\in\mathscr{C}^D_s, s\in\mathscr{S} \tag{5.12}$$

$$\sum_{k\in\mathscr{K}}\sum_{r\in\mathscr{R}}\left(x^r_{sdk}+\sum_{(d',d)\in A^{D-}_d}x^r_{d'dk}\right)=1\ \ \forall d\in\mathscr{C}^D_s, s\in\mathscr{S} \tag{5.13}$$

$$Q_{jk}\geq(Q_{ik}+q_j)-Q(1-\sum_{r\in\mathscr{R}}x^r_{ijk})\ \ \forall(i,j)\in\mathscr{A}, j\notin\mathscr{S}, k\in\mathscr{K} \tag{5.14}$$

$$L_{sk}\geq Q_{ik}-Q(1-\sum_{r\in\mathscr{R}}x^r_{isk})\ \forall(i,s)\in\mathscr{A}, s\in\mathscr{S}, k\in\mathscr{K} \tag{5.15}$$

$$max\{0,q_i\}\leq Q_{ik}\leq min\{Q,Q+q_i\}\ \ \forall i\in\mathscr{V}, k\in\mathscr{K} \tag{5.16}$$

$$Q_{sk}\leq Q\sum_{d\in\mathscr{C}^D_s}\sum_{r\in\mathscr{R}}x^r_{sdk}\ \ \forall s\in\mathscr{S}, k\in\mathscr{K} \tag{5.17}$$

$$Q_{sk}\geq\min_{d\in\mathscr{C}^D_s}\{q_d\}\sum_{d\in\mathscr{C}^D_s}\sum_{r\in\mathscr{R}}x^r_{sdk}\ \ \forall s\in\mathscr{S}, k\in\mathscr{K} \tag{5.18}$$

$$L_{sk}\geq\min_{p\in\mathscr{C}^P}\{q_p\}\sum_{i\in\mathscr{V}\setminus\mathscr{C}^D}\sum_{r\in\mathscr{R}}x^r_{sik}\ \ \forall s\in\mathscr{S}, k\in\mathscr{K} \tag{5.19}$$

$$Q_{sk}\leq Q(1-\sum_{i\in\mathscr{V}\setminus\mathscr{C}^D}\sum_{r\in\mathscr{R}}x^r_{sik})\ \ \forall s\in\mathscr{S}, k\in\mathscr{K} \tag{5.20}$$

$$\sum_{k\in\mathscr{K}}L_{sk}=\sum_{p\in\mathscr{C}^P}q_p y_{ps}\ \ \forall s\in\mathscr{S} \tag{5.21}$$

$$\begin{gathered}B_{jk}\geq B_{ik}+\delta(i)+c_{i,j}-M(1-\sum_{r\in\mathscr{R}}x^r_{ijk})\\ \forall(i,j)\in\mathscr{A}, i\in\{\mathscr{C}^P\cup\mathscr{C}^D\}, j\in\{\mathscr{C}^P\cup\mathscr{C}^D\cup\mathscr{S}\}, i\neq j, k\in\mathscr{K}\end{gathered} \tag{5.22}$$

$$B_{ik}\geq B_{sk}+\varphi'(s)+c_{s,i}-M(1-\sum_{r\in\mathscr{R}}x^r_{sik})\ \ \forall s\in\mathscr{S}, i\in\{\mathscr{C}^P\cup\mathscr{S}\setminus s\}, k\in\mathscr{K} \tag{5.23}$$

$$B_{dk}\geq B_{sk}+\varphi(s)+z_{sk}\varphi'(s)+c_{s,d}-M(1-\sum_{r\in\mathscr{R}}x^r_{sdk})\ \ \forall s\in\mathscr{S}, d\in\mathscr{C}^D_s, k\in\mathscr{K} \tag{5.24}$$

$$B_{iwk} \geq B_{iw} + \delta(i) + c_{i,w} - M(1 - \sum_{r\in\mathscr{R}} x^r_{iwk}) \;\; \forall w \in \mathscr{W}, i \in \{\mathscr{C}^P \cup \mathscr{C}^D\}, k \in \mathscr{K} \tag{5.25}$$

$$B_{swk} \geq B_{sk} + \varphi'(s) + c_{s,w} - M(1 - \sum_{r\in\mathscr{R}} x^r_{swk}) \;\; \forall w \in \mathscr{W}, s \in \mathscr{S}, k \in \mathscr{K} \tag{5.26}$$

$$\begin{gathered} \text{If } \sum_{r\in\mathscr{R}\setminus e} \left(x^r_{iwk}\, x^{r+1}_{wsk}\right) = 1 \text{ then } B_{sk} \geq B_{iwk} + c_{ws} \\ \forall w \in \mathscr{W}, s \in \mathscr{S}, i \in \{\mathscr{C}^P \cup \mathscr{C}^D \cup \mathscr{S} \setminus s\}, k \in \mathscr{K} \end{gathered} \tag{5.27}$$

$$\begin{aligned} z_{sk} = 1 \text{ if and only if } \sum_{p\in\mathscr{C}^P} \left(\sum_{r\in\mathscr{R}} x^r_{psk} + \sum_{r\in\mathscr{R}\setminus e} \sum_{w\in\mathscr{W}} x^r_{pwk}\, x^{r+1}_{wsk} \right) = 1 \\ \forall s \in \mathscr{S}, k \in \mathscr{K} \end{aligned} \tag{5.28}$$

$$\begin{aligned} &(t(s) - \eta) \sum_{r\in\mathscr{R}} \left(x^r_{gsk} + \sum_{w\in\mathscr{W}} x^r_{wsk} + \sum_{(s',s)\in A^{S-}_s} x^r_{s'sk} + \sum_{(p,s)\in A^{PS}_s} x^r_{psk} + \sum_{(d,s)\in A^{DS}_s} x^r_{dsk} \right) \leq B_{sk} \\ &\leq t(s) \sum_{r\in\mathscr{R}} \left(x^r_{sgk} + \sum_{(s,d)\in A^{SD}_s} x^r_{sdk} + \sum_{w\in\mathscr{W}} x^r_{swk} + \sum_{(s,s')\in A^{S+}_s} x^r_{ss'k} + \sum_{(s,p)\in A^{SP}_s} x^r_{spk} \right) \\ &\;\forall s \in \mathscr{S}, k \in \mathscr{K} \end{aligned} \tag{5.29}$$

$$\begin{aligned} &e_p \sum_{r\in\mathscr{R}} \left(\sum_{(p,p')\in A^{P+}_p} x^r_{pp'k} + \sum_{w\in\mathscr{W}} x^r_{pwk} + \sum_{s\in\mathscr{S}_p} x^r_{psk} \right) \leq B_{pk} \\ &\leq l_p \sum_{r\in\mathscr{R}} \left(\sum_{(s,p)\in A^{SP}_p} x^r_{spk} + \sum_{(p',p)\in A^{P-}_p} x^r_{p'pk} + \sum_{(d,p)\in A^{DP}_p} x^r_{dpk} + x^r_{gpk} \right) \\ &\;\forall p \in \mathscr{C}^P, k \in \mathscr{K} \end{aligned} \tag{5.30}$$

$$\begin{aligned} &e_d \sum_{r\in\mathscr{R}} \left(\sum_{(d,d')\in A^{D+}_d} x^r_{dd'k} + \sum_{(d,p)\in A^{DP}_d} x^r_{dpk} + \sum_{(d,s')\in A^{DS}_d} x^r_{ds'k} + \sum_{w\in\mathscr{W}} x^r_{dwk} + x^r_{dgk} \right) \\ &\leq B_{dk} \leq l_d \sum_{r\in\mathscr{R}} \left(x^r_{sdk} + \sum_{(d',d)\in A^{D-}_d} x^r_{d'dk} \right) \;\; \forall s \in \mathscr{S}, d \in \mathscr{C}^D_s, k \in \mathscr{K} \end{aligned} \tag{5.31}$$

$$0 \leq L_{sk} \leq Q \ \ \forall s \in \mathscr{S}, k \in \mathscr{K} \tag{5.32}$$

$$x_{ijk}^{r} \in \{0,1\} \ \ \forall (i,j) \in \mathscr{A}, r \in \mathscr{R}, k \in \mathscr{K} \tag{5.33}$$

$$y_{ps} \in \{0,1\} \ \ \forall p \in \mathscr{C}^{P}, s \in \mathscr{S} \tag{5.34}$$

$$z_{sk} \in \{0,1\} \ \ \forall s \in \mathscr{S}, k \in \mathscr{K} \tag{5.35}$$

The objective function (5.1) minimizes the total transportation cost, including the fixed costs incurred for using vehicles. Constraints (5.2) ensure that a vehicle leaving a supply point visits either a customer demand, a waiting station, another supply point, or the main depot g. The conservation of flow at supply point is completed by constraints (5.3). Constraints (5.4) and (5.5) represent the conservation of flow at main depot and waiting stations respectively. Constraints (5.6) ensure that each pickup-customer demand must be assigned to only one supply point. Constraints (5.7) - (5.9) forbid the illegal pickup legs which bring either pickup demands to the supply point to which they are not assigned or pickup demands not assigned to the same supply point.

Constraints (5.10) ensure that when a vehicle leaves a pickup-customer demand p, it goes either to another pickup-customer demand, a waiting station $w \in \mathscr{W}$, or a supply point $s \in \mathscr{W}$. These constraints together with (5.11) enforce the flow conservation at pickup-customer demands, and the single assignment of pickup-customer demands to legs. Similarly, constraints (5.12) ensure that when a vehicle leaves a delivery-customer demand $d \in \mathscr{C}_{s}^{D}$, it goes either to another delivery-customer demand of the same set $\mathscr{C}_{s}^{D}$, a pickup-customer demand p, a supply point s', a waiting station w, or the main depot g. Constraints (5.12) and (5.13) also enforce the flow conservation at delivery-customer demands, and the single assignment of delivery-customer demands to legs.

Consistency of load variables is ensured through constraints (5.14) and (5.15), while constraints (5.16) enforce the restrictions on the vehicle capacity. Constraints (5.17) and (5.18) ensure that the vehicle k brings load from a supply point s to a delivery-customer demand d of the customer zone $\mathscr{C}_s^D$ if and only if it loads freight at supply point s, i.e., $Q_{ks} > 0$. Constraints (5.19) and (5.20) ensure that the vehicle k goes directly from the supply point s to either a pickup-customer demand p, any other supply point, a waiting station, or the main depot g if it only unloads at s and then leaves s empty. Constraints (5.21) guarantee that the total pickup load entering each supply point equals to the total pickup demands of customers, which are assigned to the corresponding supply point.

Consistency of the time variables is ensured through constraints (5.22) - (5.27). Note that when a waiting station w is reached in the rth position of the work assignment of vehicle k, the outgoing arc (w,s) should be in the $(r+1)$th position of the same work assignment. Constraints (5.27) can be linearized by introducing new variables $v_{iwsk}^r \in \{0,1\}$ such that $v_{iwsk}^r = x_{iwk}^r x_{wsk}^{r+1}$ $\forall w \in \mathscr{W}, s \in \mathscr{S}, i \in \{\mathscr{C}^P \cup \mathscr{C}^D \cup \mathscr{S} \setminus s\}, k \in \mathscr{K}$. Constraints (5.27) can be made explicit by means of the following linear constraints:

$$x_{iwk}^r \geq v_{iwsk}^r \quad \forall r \in \mathscr{R}, w \in \mathscr{W}, s \in \mathscr{S}, i \in \{\mathscr{C}^P \cup \mathscr{C}^D \cup \mathscr{S} \setminus s\}, k \in \mathscr{K} \tag{5.36}$$

$$x_{wsk}^{r+1} \geq v_{iwsk}^r \quad \forall r \in \mathscr{R}, w \in \mathscr{W}, s \in \mathscr{S}, i \in \{\mathscr{C}^P \cup \mathscr{C}^D \cup \mathscr{S} \setminus s\}, k \in \mathscr{K} \tag{5.37}$$

$$x_{iwk}^r + x_{wsk}^{r+1} \leq 1 + v_{iwsk}^r \quad \forall r \in \mathscr{R}, w \in \mathscr{W}, s \in \mathscr{S}, i \in \{\mathscr{C}^P \cup \mathscr{C}^D \cup \mathscr{S} \setminus s\}, k \in \mathscr{K} \tag{5.38}$$

$$B_{sk} \geq B_{iwk} + c_{ws} - M\Big(1 - \sum_{r \in \mathscr{R}} v_{iwsk}^r\Big) \quad \forall w \in \mathscr{W}, s \in \mathscr{S}, i \in \{\mathscr{C}^P \cup \mathscr{C}^D \cup \mathscr{S} \setminus s\}, k \in \mathscr{K} \tag{5.39}$$

Constraints (5.28) ensure that the vehicle k unloads at a supply point s if and only if it brings pick-up demands to s. Because each pickup-customer demand p is serviced only once, these constraints can be linearized and rewritten as follows:

$$z_{sk} = \sum_{p \in \mathscr{C}^P} \left(\sum_{r \in \mathscr{R}} x_{psk}^r + \sum_{r \in \mathscr{R}} v_{pwsk}^r \right) \qquad \forall s \in \mathscr{S}, k \in \mathscr{K} \tag{5.40}$$

The respect of time windows at supply points and customer demands is enforced through constraints (5.29) - (5.31). Constraints (5.32) are bounding constraints for variables L_s^k. Finally, constraints (5.33) - (5.35) define the decision variables.

5.5 Solution method

In this section, we present the extension of our proposed tabu search for the TMZT-VRPTW in Nguyen et al. (2013) so that it can tackle the MZT-PDTWS. The general structure and search space are presented in Section 5.5.1 and 5.5.2, respectively. Section 5.5.3 describes the initial solution construction. All components of TS are then given: the neighborhood structures (Section 5.5.4), the neighborhood-selection *Control* procedure (Section 5.5.5), the tabu status mechanism (Section 5.5.6), the diversification mechanism (Section 5.5.7), and *Post-optimization* procedure (Section 5.5.8).

5.5.1 General structure

The structure of TS is presented in Algorithm 4. Let $\bar{r}$ as the ratio of selecting routing neighborhoods to leg neighborhoods. First, an initial feasible solution z is generated using a greedy method seeking to fully utilize vehicles and minimize the total cost. At each iteration of the TS method, one neighborhood is selected probabilistically based on the current value of $\bar{r}$, then the selected neighborhood is explored, and the best move is chosen (lines 7-8). This move must not be tabu, unless it improves the current best TS solution z_{best} (aspiration criterion). The algorithm adds the new solution to an elite set $\mathscr{E}$ if it improves on z_{best}. It also remembers the value of the parameter $\bar{r}$ when the new best solution was found (lines 9-13), and finally updates the elite set $\mathscr{E}$ by removing a solution based on its value and the difference between solutions (Section 5.5.7).

Initially, the search freely explores the solution space by assigning each neighborhood with the same probability of being selected. Whenever the best TS solution z_{best} is not improved for IT_{cNS} TS iterations (line 15), the *Control* procedure is called to reduce the probability of selecting leg neighborhoods (line 25). Consequently, routing neighborhoods are selected proportionally more often, which gives customer moves more opportunity to fully optimize routes. The search is re-initialized from the current best TS solution z_{best}

Algorithm 4 Tabu search

```
 1: Generate an initial feasible solution $z$
 2: $z_{best} \leftarrow z$
 3: Elite set $\mathscr{E} \leftarrow \oslash$
 4: Probability of selecting routing neighborhood with respect to leg neighborhood $\bar{r} \leftarrow 1$
 5: STOP $\leftarrow 0$
 6: **repeat**
 7:     A neighborhood is selected based on the value of $\bar{r}$
 8:     Find the best solution $z'$ in the selected neighborhood of $z$
 9:     **if** $z'$ is better than $z_{best}$ **then**
10:         $z_{best} \leftarrow z'$
11:         $\bar{r}_{best} \leftarrow \bar{r}$
12:         Add $(z_{best}, \bar{r}_{best})$ to the elite set $\mathscr{E}$; update $\mathscr{E}$
13:     **end if**
14:     $z \leftarrow z'$
15:     **if** $z_{best}$ not improved for $IT_{cNS}$ iterations **then**
16:         **if** $z_{best}$ not improved after $C_{cNS}$ consecutive executions of *Control* procedure **then**
17:             **if** $\mathscr{E} = \oslash$ **then**
18:                 STOP $\leftarrow 1$
19:             **else**
20:                 Select randomly $(z, \bar{r}_z)$ (and remove it) from the elite set $\mathscr{E}$
21:                 Diversify the current solution $z$
22:                 Set $\bar{r} \leftarrow \bar{r}_z$ and reset tabu lists
23:             **end if**
24:         **else**
25:             Apply *Control* procedure to update the value of $\bar{r}$
26:             $z \leftarrow z_{best}$
27:         **end if**
28:     **end if**
29: **until** STOP
30: $z_{best} \leftarrow$ *Post-optimization*$(z_{best})$
31: return $z_{best}$
```

after the execution of the *Control* procedure (line 26). Moreover, after C_{cNS} consecutive executions of this procedure without improvement of the current best TS solution z_{best}, a solution z is selected randomly and removed from the elite set (line 20), and a *Diversification* mechanism is applied to perturb z (line 21). The value of $\bar{r}$ is reset to the value it had when the corresponding elite solution was found, and all tabu lists are reset to the empty state (line 22). The search then proceeds from the perturbed solution z. The search is stopped when the elite set $\mathscr{E}$ is empty. Finally, a post-optimization procedure is performed to potentially improve the current best solution z_{best} (line 30).

5.5.2 Search space

We allow infeasible solutions in our algorithm. Infeasible solutions are penalized in proportion to the violations of the constraints on vehicle capacity, the time windows of customer demands and supply points. More precisely, for a solution z, let $c(z)$ denote the total traveling cost, and let $q(z), w_c(z)$ and $w_s(z)$ denote the total violation of vehicle load, customer demands time windows, supply points time windows, respectively. The total vehicle-load violation is computed on a leg basis with respect to the value Q, whereas the total violation of time windows of customer demands is equal to $\sum_{i\in z} max\{(a_i - l_i), 0\}$, and the total violation of time windows of supply points is equal to $\sum_{s\in z} max\{(t(s) - \eta - a_s), (a_s - t(s)), 0\}$, where a_i and a_s are the arrival time at customer demand i and supply point s, respectively.

Solutions are then evaluated according to the weighted fitness function $f(z) = c(z) + \alpha^Q q(z) + \alpha^C w_c(z) + \alpha^S w_s(z) + F * m$, where α^Q, α^C, α^S are penalty parameters adjusted dynamically during the search. The updating scheme is based on the idea of Cordeau et al. (2001). At each iteration, the value of α^Q, α^C and α^S are modified by a factor $1+\beta > 1$. If the current solution is feasible with respect to load constraints, the value of α^Q is divided by $1+\beta$; otherwise it is multiplied by $1+\beta$. The same rule applies to α^C and α^S with respect to time window constraints of customers and supply points, respectively. In our algorithm, we set $\alpha^Q = \alpha^C = \alpha^S = 1$ and $\beta = 0.3$.

5.5.3 Initial solution

We sort the supply points and index them in increasing order of their opening times. Thus, if $t(s_1) \leq t(s_2)$, then $s_1 < s_2$ and vice versa. We then construct an initial solution by assigning each pickup-customer demand to one supply point, and building each feasible work assignment sequentially.

There are several ways to assign pickup-customer demands to supply points. A simple way is to assign each pickup-customer demand to its closest supply point. Another way is that each supply point s services a predefined number of its pickup-customer demands closest to it. However, these strategies fail to take the significant variation of delivery loads at each supply point into account. The imbalance of pickup and delivery demands might happen at some supply points, which reduces the possibility of 'unload and load' operation at a supply point, and thus increases the number of empty movements.

In order to overcome this issue, we first estimate what the maximum total pickup demands at each supply point should be. As delivery-customer demands are already assigned to supply points, we calculate the total delivery demands assigned to each supply point. Let K_s denote this number for each supply point $s \in \mathscr{S}$. We then use K_s as the maximum capacity of collected freight which vehicles can unload at supply point s in hope of balancing the unloading and loading operations at each supply point. Each pickup-customer demand p is then assigned to the nearest supply point in $\mathscr{S}_p$. When the assignment violates the maximum capacity of the nearest supply point in $\mathscr{S}_p$, it is randomly allocated to the supply point in S_p whose residual capacity is large enough for the assignment. Pickup-customer demands are handled in random order. By considering both the distance from pickup-customer demands to supply points and the capacity of delivery demands at each supply point when allocating pickup-customer demands, we aim to generate a solution which satisfy the following two conditions. The first condition is a small total traveling cost. The other condition is to avoid the imbalance of the number of generated pickup legs and delivery legs at each supply point, so vehicles can do more 'unload and load' activities, which then helps to reduce empty movements.

Once the assignment of pickup-customer demands to supply points is completed, each work assignment of the initial solution is built sequentially. Each work-assignment construction consists of two phases: the first phase determines the first supply point for the current work assignment; the second phase creates sequentially each leg using a greedy algorithm.

In the first phase, the supply point s with earliest opening time and unserviced customer demands is assigned as the initial supply point of the first leg of the current work assignment. During the second phase, one or a sequence of first legs between supply point s and either other supply point s' or the depot g is created using a greedy algorithm. If the first created leg(s) ends at a supply point s', we continue applying the greedy algorithm to build next leg(s) in which s' is now used as the initial supply point. Otherwise, if it ends at the main depot, it means the current work assignment cannot be used anymore, and we return to the first phase to build another work assignment. This process is repeated until all customer demands are serviced (assigned to a vehicle route).

The greedy algorithm is implemented as follows: for a given initial supply point s assigned to the leg, it finds a set of supply points $S' = \{s' \in \mathscr{S} | s'$ with unserviced customer demands and $t(s') > t(s)\}$. If $S' \neq \varnothing$, for each pair (s, s'), all unrouted customer demands of s and unrouted pickup-customer demands of s' are candidates for insertion to the vehicle according to a priority order which considers unrouted pickup-customer demands of s first, then unrouted delivery-customer demands of s, and unrouted pickup-customer demands of s' as the latest. Each unserviced customer demand is assigned to the vehicle sequentially by applying the heuristic I1 of Solomon (1987) until the vehicle is full.

Figure 5.3 illustrates different possibilities when routing customer demands between two supply points s and s'. If there exists unrouted pickup-customer demands of s, the greedy algorithm assigns them to the current vehicle first, for instance, generating a pickup leg $\{p_2, p_5, p_3, s\}$. Between supply point s and s', the algorithm then may generate either a leg or a sequence of legs:

- (1) A sequence of two legs: a delivery leg $\{s, d_1, d_2, d_4\}$ and a pickup leg $\{p_7, p_{10}, s'\}$
- (2) A pickup leg: $\{p_9, p_{11}, p_{12}, s'\}$
- (3) A delivery leg: $\{s, d_3, d_5, d_7\}$

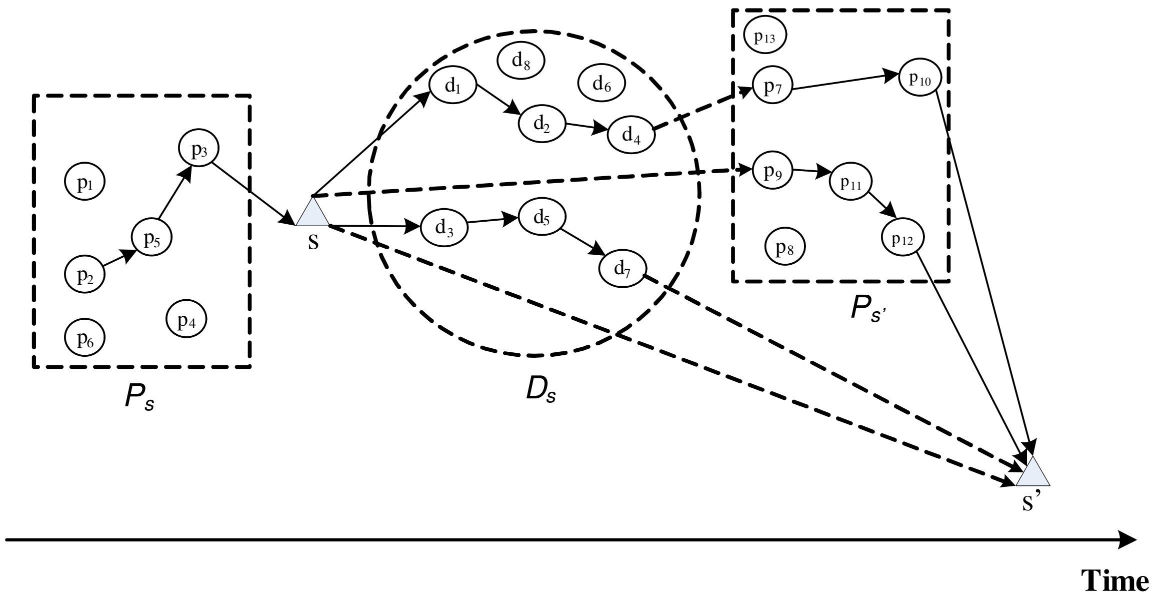


Figure 5.3: A generation of a sequence of legs between two supply points.

- (4) An empty leg connecting s and s'

Which one is generated depends on the departure time at supply point s, time windows and distance between unserviced delivery- and pickup-customer demands of supply point s and s', respectively.

Among feasible legs generated between all pairs of s and s', the one with the smallest average cost per unit demand is selected and assigned to the current work assignment. The average cost per unit demand is expressed as the ratio of the total traveling time over the total demand carried by the vehicle between s and s'. We set total demand to one for empty legs. In the case there are no feasible legs or $S' = \varnothing$, the greedy algorithm builds the last leg (s, g) by applying the heuristic I1 of Solomon (1987).

5.5.4 Neighborhoods

The MZT-PDTWS has the same problem structure as the TMZT-VRPTW, where each work assignment consists of a sequence of legs and where each leg is made of a sequence of customer demands. Therefore, the MZT-PDTWS is also solved by applying neighborhoods at leg and customer levels. However, the MZT-PDTWS is more complex than the

TMZT-VRPTW in the sense that a new type of customer demands, say pickup, is now considered together with delivery-customer demands. Furthermore, pickup-customer-demand-to-supply point assignments are not known in advance as for delivery-customer demands, but rather each pickup-customer demand has a list of available supply points that can service it. Consequently, it is asking for expanding neighborhoods so that the MZT-PDTWS can be addressed more efficiently. In this section, we describe in detail two types of neighborhoods, i.e., routing neighborhoods and leg neighborhoods, used in our tabu search.

5.5.4.1 Routing neighborhoods

In the MZT-PDTWS, each vehicle performs a sequence of legs, each leg services either pickup- or delivery-customer demands but cannot do both. As a result, routing neighborhoods work on two sets of pickup legs and delivery legs separately. These neighborhoods try to improve routing by using different intra and inter route neighborhoods commonly used in the VRP literature: Relocation, Exchange and 2-opt.

For delivery-customer demands whose serviced supply points are pre-assigned as those in the TMZT-VRPTW, moving them between supply points, i.e., causing reassignments to other supply points, is forbidden. Routing neighborhoods working on delivery-customer demands are therefore kept unchanged as in the TMZT-VRPTW. More precisely, for each move in each neighborhood, two delivery-customer demands which belong to a same supply point are considered:

- *Relocation move*: one of the two customer demands is taken from its current position and inserted after the other one.

- *Exchange move*: two customer demands are swapped.

- *2-opt move*: for two customer demands in the same leg, the edges emanating from them are removed, two edges are added, one of which connects these two customer demands, and the other connects their successor customer demands. For two customer demands in different legs, the remaining segments of these legs are swapped preserving the order of customer demands.

For pickup-customer demands whose lists of available serviced supply points are only given, the assignment of each pickup-customer demand to a supply point selected from the given list is required before routing it. Consequently, we extend routing neighborhoods working on pickup-customer demands so that they could address the routing but also the supply-point assignment. It eventually helps to improve the routing through pickup-customer-demand-to-supply-point assignment and vice-versa. Since pickup-customer demands can be reassigned to other supply points, we do not restrict routing neighborhoods to work on each zone of pickup-customer demands separately as for delivery-customer demands. However, when a move reassigns a pickup-customer demand p, assuming from s_i to s_j, it is constrained to reassign the demand to a supply point that belongs to the list of available serviced supply points for p, i.e., $s_j \in \mathscr{S}_p$. Three types of routing neighborhoods are thus considered for all pairs of pickup-customer demands satisfying the above condition for reassignment:

- *Relocation move*: one pickup-customer demand is shifted from its current position to another position, in the same leg or in a different leg which may be assigned to the same supply point or not, provided the condition for supply-point reassignment is respected.

- *Exchange move*: two pickup-customer demands are exchanged. They may belong to the same leg or, if the condition for supply-point reassignment allows them, to two distinct legs sharing one common supply point or not.

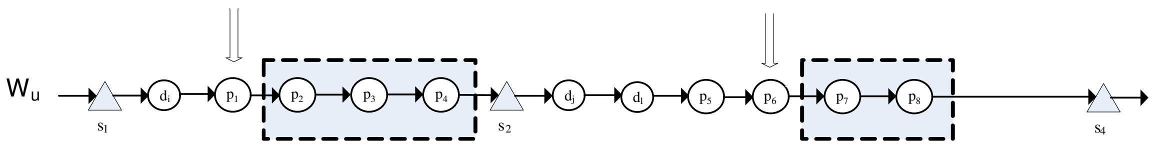
- *2-opt move*: for two pickup-customer demands in the same leg, the edges emanating from them are removed, two edges are added, one of which connects these two pickup-customer demands, and the other connects their successor pickup-customer demands. For two pickup-customer demands in different legs sharing one common supply point, thus in different vehicles, the remaining segments of these legs are swapped preserving the order of customer demands. Finally, for two pickup-customer demands in different legs sharing two distinct supply points, i.e., in a same vehicle or different vehicles, the remaining segments of these legs are swapped preserving the order of customer demands, when the condition for supply-point reassignment is respected.

Let us take a simple example to illustrate the condition for supply-point reassignment. Consider Table 5.2 where the lists $\mathscr{S}_p$ for pickup-customer demands $p \in \mathscr{P}$ are given for the work assignment W_u shown in Figure 5.4a. Consider two pickup-customer demands p_1 and p_6 in W_u which belong to different supply points, s_2 and s_4, respectively. The 2-opt move of p_1 and p_6 applied on W_u requires the supply-point reassignments of $\{p_2, p_3, p_4\}$ to s_4 and of $\{p_7, p_8\}$ to s_2. Customer demands p_7 and p_8 can be reassigned to supply point s_2 as $s_2 \in \mathscr{S}_{p_7} \cap \mathscr{S}_{p_8}$. Similarity, p_2, p_3, p_4 can be reassigned to s_4 as $s_4 \in \mathscr{S}_{p_2} \cap \mathscr{S}_{p_3} \cap \mathscr{S}_{p_4}$. The condition for supply-point reassignment is satisfied, therefore this 2-opt move is accepted. Figure 5.4b illustrates W_u after the move.

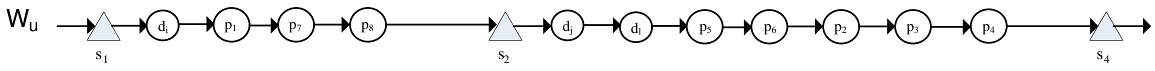
Table 5.2: The lists of available serviced supply points $\mathscr{S}_p$

Pickup-customer demand p	List of available serviced supply points $\mathscr{S}_p$
p_1	$\{s_2, s_4\}$
p_2	$\{s_2, s_3, s_4\}$
p_3	$\{s_1, s_2, s_4\}$
p_4	$\{s_2, s_4\}$
p_5	$\{s_4\}$
p_6	$\{s_4, s_5\}$
p_7	$\{s_2, s_4\}$
p_8	$\{s_1, s_2, s_4\}$

On the other hand, the 2-opt move of p_1 and p_5 applied on W_u requires the supply-point reassignments of $\{p_2, p_3, p_4\}$ to s_4 and of $\{p_6, p_7, p_8\}$ to s_2. However, $s_2 \notin \mathscr{S}_{p_6}$, so p_6 can not be reassigned to supply point s_2. Due to the infeasibility of the supply-point reassignment, this 2-opt move is not accepted.



(a) Work assignment W u before *2-opt*



(b) Work assignment W u after *2-opt*

Figure 5.4: An example of 2-opt routing neighborhood on pickup-customer demands

5.5.4.2 Leg neighborhoods

In the MZT-PDTWS, each leg is assigned to the supply point where the vehicle returns the collected freight and/or loads new freight. Let W_u be the work assignment performed by vehicle u. Let s_{i-1} and s_{i+1} denote the predecessor and successor supply points, respectively, of s_i within a work assignment. When moving a supply point, all customer demands serviced by it in the vehicle are also moved. Leg neighborhoods focus on repositioning legs within the time restrictions. They are described in the following:

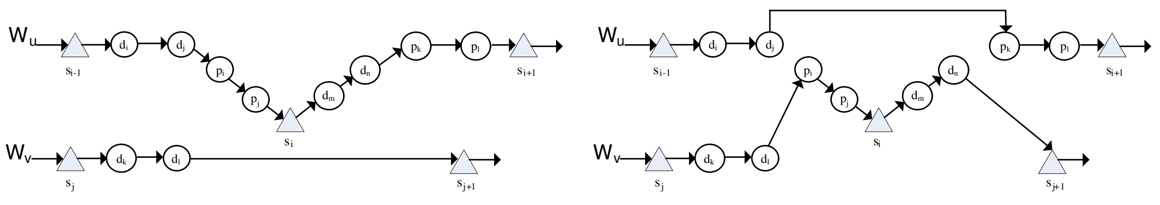
Relocate supply point This neighborhood removes a supply point together with customer demands serviced by it in a work assignment and inserts them into another work assignment. Consider two work assignments W_u and W_v. For each supply point $s_i \in W_u$:

- if $s_i \notin W_v$: for each two successive supply points s_j, $s_{j+1} \in W_v$, such that $s_j < s_i < s_{j+1}$, then move s_i from work assignment W_u to W_v locating it between s_j and s_{j+1} (see Figure 5.5 for example). As we promote the 'unload and load' operation at a supply point to reduce empty movements, the reassignment of pickup-customer demands to other supply points may be required. More precisely, whenever a pickup (or a single-delivery) leg assigned to s_i is relocated between s_j and s_{j+1}, and the leg assigned to s_{j+1} is a single-delivery leg (or the leg assigned to s_j is pickup leg), the reassignment of all pickup-customer demands in the leg of s_i (or s_j) to supply point s_{j+1} (or s_i) is checked. If the reassignment is feasible, it is executed, customer demands in the leg of s_i is then relocated between s_j and s_{j+1} to create a new 'unload and load' operation at supply point s_{j+1} (or s_i) on the work assignment W_v. Otherwise, the leg assigned to s_i is just simply relocated between s_j and s_{j+1}. Figure 5.6 illustrates the relocation of a pickup leg assigned to s_i on W_u between s_j and s_{j+1} on W_v. As there are two pickup-customer demands p_i and p_j in this leg, and the leg assigned to s_{j+1} of W_v is a single delivery leg, we check whether we can reassign p_i and p_j to supply point s_{j+1} so that vehicle v can do 'unload and load' at s_{j+1}. If $s_{j+1} \in \mathscr{S}_{p_i} \cap \mathscr{S}_{p_j}$ then the reassignment is applied, and the movement is executed as shown in Figure 5.6b. Otherwise, see Figure 5.6c.

- if $s_i \in W_v$:
  - Case 1: if vehicle u only unloads at s_i: this is a relocate move of the pickup leg r_i assigned to s_i in vehicle u. Three cases of vehicle's operation at supply point s_i in vehicle v are considered:
    * Case 1.1: if vehicle v only unloads at s_i: denote r_j the pickup leg assigned to s_i in W_v, then move s_i from work assignment W_u to W_v by concatenating two pickup legs r_i and r_j. Appending r_i to r_j and r_j to r_i are both considered (see Figure 5.7).
    * Case 1.2: if vehicle v only loads at s_i: denote r_j the single-delivery leg assigned to s_i in W_v, then move s_i from work assignment W_u to W_v by locating pickup leg r_i right before single-delivery leg r_j so that vehicle v does 'unload and load' operation at s_i (see Figure 5.8).
    * Case 1.3: if vehicle v both 'unloads and loads' at s_i: denote r_j the pickup leg and r'_j the coordinate leg assigned to s_i in W_v, then move s_i from work assignment W_u to W_v by concatenating two pickup legs r_i and r_j. Both cases of appending r_i to r_j and r_j to r_i are also considered as in the case 1.1.
  - Case 2: if vehicle u only loads at s_i: this is a relocate move of the single-delivery leg r_i assigned to s_i of vehicle u. Three cases of vehicle's operation at supply point s_i in vehicle v are considered as in the Case 1. Similarly, if

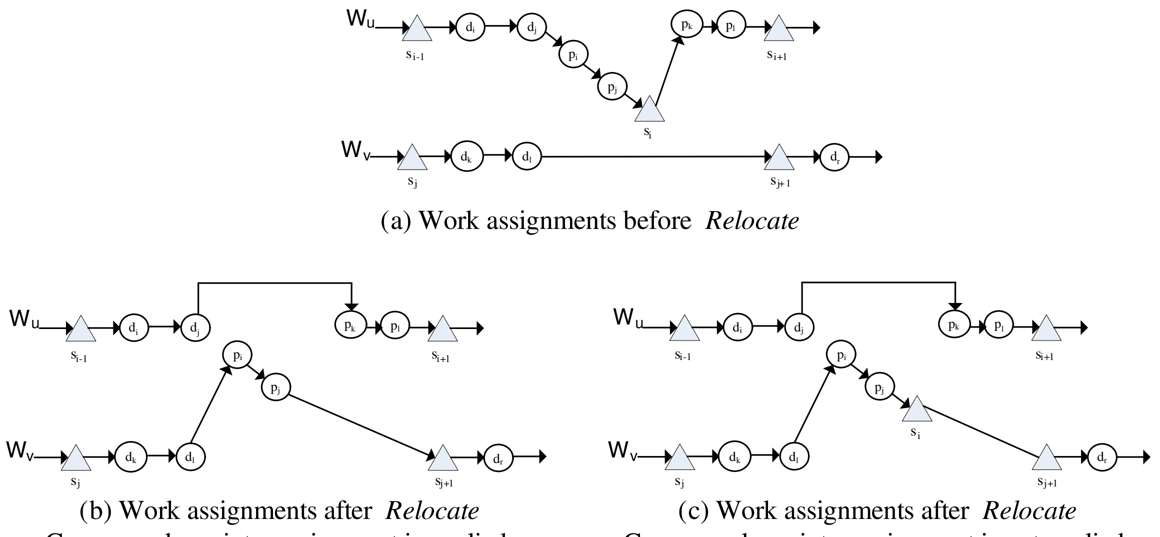
vehicle v loads at s_i, i.e., there exists delivery leg r_j assigned to s_i in vehicle v, the concatenation of delivery leg r_i and delivery leg r_j is also examined in two cases: one appending r_i to r_j and the other appending r_j to r_i.

- Case 3: if vehicle u both 'unloads and loads' at s_i: this is a relocate move of both the pickup leg r_i and the coordinate delivery leg r'_i assigned to the same supply point s_i in vehicle u. Three cases of vehicle's operation at supply point s_i in vehicle v are considered as in previous cases. All possibilities of concatenation delivery (pickup) legs assigned to the same supply point s_i in both vehicle u and v are also examined.



(a) Work assignments before *Relocate* (b) Work assignments after *Relocate*

Figure 5.5: Relocate a supply point: relocate both pickup leg and coordinate-delivery leg assigned to a same supply point

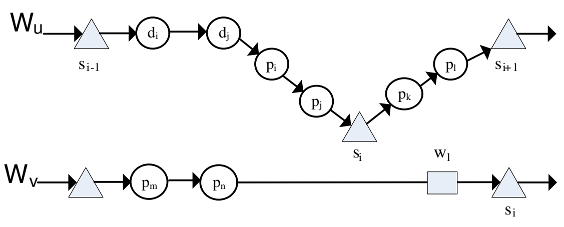


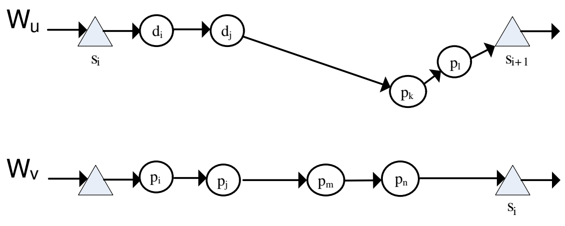
(a) Work assignments before *Relocate*

(b) Work assignments after *Relocate*
Case: supply-point reassignment is applied

(c) Work assignments after *Relocate*
Case: supply-point reassignment is not applied

Figure 5.6: Relocate a supply point: relocate a pickup leg


(a) Work assignments before *Relocate*


(b) Work assignments after *Relocate*
Case: append (r_j,s_i) to (r_i,s_i)

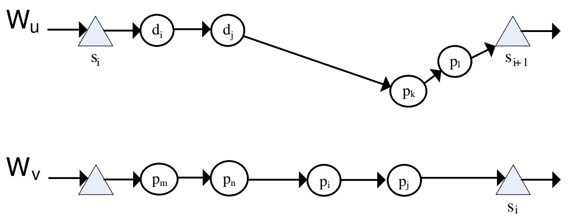
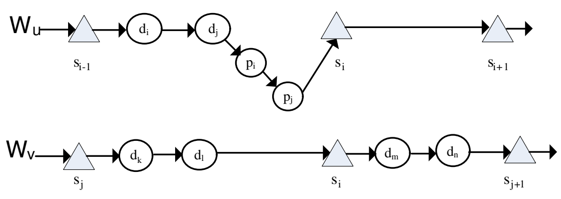

(c) Work assignments after *Relocate*
Case: append (r_i,s_i) to (r_j,s_i)

Figure 5.7: Relocate a supply point: concatenation of two pickup legs


(a) Work assignments before *Relocate*

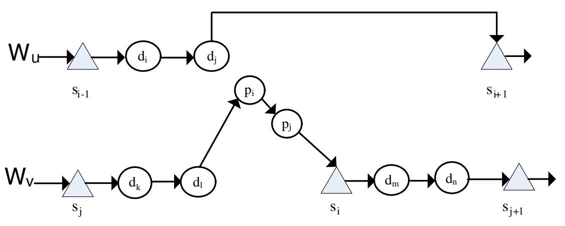

(b) Work assignments after *Relocate*

Figure 5.8: Relocate a supply point: creation of an 'unload and load' operation

Exchange supply point This neighborhood exchanges legs between work assignments. Consider two work assignments W_u and W_v. For supply points $s_i \in W_u$ and $s_j \in W_v$:

- if $s_{i-1} < s_j < s_{i+1}$:
  - if $s_{j-1} < s_i < s_{j+1}$: simply swap s_i and s_j together with customer demands serviced by them (both pickup- and delivery-customer demands if any);
  - if $s_{j-1} = s_i < s_{j+1}$: first swap s_i and s_j together with customer demands serviced by them; next if there were pickup-customer demands assigned to s_i in both vehicle u and vehicle v, then in vehicle v we concatenate pickup-customer demands from both vehicles as described in the case 1.1; and also concatenate

delivery-customer demands in both vehicles if applicable;

  – if $s_{j-1} < s_i <= s_{j+1}$: same as previous case.

- otherwise if $s_{i-1} = s_j$ or if $s_j = s_{i+1}$: as supply points s_i and s_j play a symmetric role in this exchange leg move, these two cases are considered the same as the previous one.

The reassignment of pickup-customer demands to new supply points is considered the same as those in the relocate supply point neighborhood whenever it could create 'unload and load' operation. And only the feasible reassignments are accepted.

Moving legs or customer demands could change the traveling cost and the number of vehicles, as well as the level of constraint violations of load, time windows of customer demands, and time windows of supply points. As a result, the move value is defined as a sum of five terms $\Delta f = \Delta c + F * \Delta m + \Delta q + \Delta w_c + \Delta w_s$. The five components of the summation are the difference in traveling cost, the fixed cost of using vehicles, and the difference in violation of load, time windows at customer demands and supply points between the value of the neighboring solution and the value of the current solution.

5.5.5 *Neighborhood selection strategy*

The algorithm explores one neighborhood at each iteration. The neighborhood to explore is randomly selected among the five previously defined neighborhoods. As in Nguyen et al. (2013), the probability for the selection of neighborhoods is controlled by a neighborhood-selection parameter $\bar{r}$. At the beginning of the search, both leg and routing neighborhoods are given the same probability of being selected, which allows the TS algorithm to freely explore the solution space. Given that the number of supply points is much smaller than the number of customer demands in most MZT-PDTWS instances, the algorithm should perform more customer than leg moves to ensure adequate optimization of routes. Consequently, after the initial phase, the probability of selecting leg neighborhoods becomes lower than the probability of selecting routing neighborhoods. We assign to a routing neighborhood the probability $\bar{r}/(2+6\bar{r})$ of being selected, and to a leg neighborhood the probability $1/(2+6\bar{r})$ of being selected. The equal initial probabilities are then obtained by setting $\overline{r} = 1$. The *Control* procedure in our algorithm varies the value of $\overline{r}$ during execution to monotonically reduce (increase) the probability of selecting leg (routing) neighborhoods after each IT_{cNS} iterations without improvement of the best solution. A linear scheme $\overline{r}_{k+1} = \overline{r}_k + \Delta\overline{r}$ is used, where $\Delta\overline{r}$ is a user defined parameter, $\overline{r}_{k+1}$ and $\overline{r}_k$ are values of $\overline{r}$ at iteration $k+1$ and k, respectively.

5.5.6 Tabu lists and tabu duration

We keep a separate tabu list for each type of move. Elements of a solution generated by a move are given a tabu status as follows:

- Leg moves:
  - Relocation move: the position of supply point s_i just inserted into work assignment W_v cannot be changed by another relocate supply point move while it is tabu.
  - Exchange move: supply points s_i and s_j just swapped cannot be swapped again while they are tabu.
- Customer moves:
  - Relocation move: the position of customer demand i just inserted after customer demand j, cannot be changed by the same type of move while it is tabu.
  - Exchange move: customer demands i and j just swapped cannot be swapped again while they are tabu.
  - 2-opt move: a 2-opt move applied to customer demands i and j cannot be applied again to the same customer demands while tabu.

A tabu status is assigned to each tabu list element for θ iterations, where θ is randomly selected from a uniform interval. Any move declared tabu cannot be performed unless it yields a solution which improves the current best solution. Generally, the tabu status of a move stays so for a number of iterations proportional to the number of possible moves. Consequently, we use different intervals of the tabu list size for leg and routing moves.

Since there are $O(m' * |\mathscr{S}|)$ possible leg moves, we set the interval of tabu list size for leg moves to [m'*$|\mathscr{S}|/a_1$, m'*$|\mathscr{S}|/a_2$], where m' is the number of vehicles used in the initial solution, and a_1 and a_2 are user-defined parameters.

In MZT-PDTWS, each delivery-customer demand is serviced by a fixed supply point which is known in advance. Therefore, the number of iterations during which a customer demand moves within the delivery customer zone of a supply point s remains tabu is only counted each time the algorithm deals with customer demands in that zone. The interval of tabu list size for delivery-customer demand moves for each supply point s with $|\mathscr{C}_s^D|$ associated customers is therefore calculated as [$a_3 log_{10}(|\mathscr{C}_s^D|)$, $a_4 log_{10}(|\mathscr{C}_s^D|)$], where a_3 and a_4 are user defined parameters. As pickup-customer demands are not fixed to any supply points yet, the interval of tabu list size for pickup-customer demand moves is [$a_5 log_{10}(|\mathscr{C}^P|)$, $a_6 log_{10}(|\mathscr{C}^P|)$], where a_5 and a_6 are user defined parameters.

5.5.7 Diversification strategy

A diversification strategy, based on an elite set and a frequency-based memory, directs the search to potentially unexplored promising regions when the search begins to stagnate. In a nutshell, diversification aims to capitalize on the best attributes obtained so far by selecting a new working solution from the elite set and perturbing it based on long-term trends.

In more details, we use the elite set as a diversified pool of high-quality solutions found during the tabu search. The elite set starts empty and is limited in size. The quality and diversity of the elite set is controlled by the insertion of new best solutions produced by the tabu search and the elimination of the existing solutions in the elite set. The elimination is based on the Hamming distance $\Delta(z_1, z_2)$ measuring not only the number of customer demand positions that differ between solutions z_1 and z_2 as in the TMZT-VRPTW, but also the differences between supply-point assignments of pickup-customer demands. More precisely, this distance is computed according to Equation (5.41), where $\mathbf{T}(cond)$ is a valuation function that returns 1 if the condition *cond* is true, 0, otherwise; $N_z[i]$ is the next place (which is either a customer demand, the depot, or a supply point) visited by the vehicle after servicing customer demand i in solution z; and $S_z[i]$ is the supply point assigned to pickup-customer demand i in the solution z.

$$\Delta(z_1,z_2) = \sum_{i\in\mathscr{C}^P\cup\mathscr{C}^D} \mathbf{T}(N_{z_1}[i] \neq N_{z_2}[i]) + \sum_{i\in\mathscr{C}^P} \mathbf{T}(S_{z_1}[i] \neq S_{z_2}[i]) \tag{5.41}$$

The elimination of a solution from the elite set is considered each time a new best solution z_{best} is inserted. There are two cases. If the elite set is not yet full, we delete only when there exists a solution very similar to the new z_{best}, i.e., we delete the solution z with the smallest $\Delta(z,z_{best}) \leq 0.05(|\mathscr{C}^D|+2|\mathscr{C}^P|+|\mathscr{S}|)$. When the elite set is full, z_{best} replaces the solution z that is the most similar to it, i.e., the one with the smallest $\Delta(z,z_{best})$.

The long-term frequency memory keeps a history of the arcs most frequently added to the current solution as well as the supply-point assignments of pickup-customer demands most frequently used. Let t_{ij} be the number of times arc (i, j) has been added to the solution during the search process. The frequency of arc (i, j) is then defined as $\rho_{ij} = t_{ij}/T$, where T is the total number of iterations executed so far. Similarly, let t'_{ps} be the number of times pickup-customer demand p has been assigned to supply point s during the search. The frequency of the supply-point assignment of customer demand p to s is defined as $\chi_{ps} = t'_{ps}/T$.

Diversification then proceeds to perturb the search that starts from the solution taken from the elite set by removing arcs with high frequency and inserting arcs with low frequency and promoting never-seen supply-point assignments. Thus, the evaluation of neighbor solutions is biased so as to penalize the arcs most frequently added to the current solution and the supply-point assignment most frequently used.

More precisely, the corresponding two penalties $g_1(\bar{z})$ and $g_2(\bar{z})$ are added to the evaluation of the fitness $f(\bar{z})$ (Section 5.5.2) of a neighbor $\bar{z}$ of the current solution z:

$$g_1(\bar{z}) = \bar{C}\Big(\sum_{(i,j)\in A_a} \rho_{ij} + \sum_{(i',j')\in A_r} (1-\rho_{i'j'})\Big) \tag{5.42}$$

$$g_2(\bar{z}) = \bar{C} \sum_{p\in\mathscr{C}^P} \left[\sum_{\substack{s\in\mathscr{S}_p \\ S_z(p)=S_{\bar{z}}(p)=s}} \chi_{ps} + \sum_{\substack{s\in\mathscr{S}_p \\ S_z(p)\neq s \\ S_{\bar{z}}(p)=s}} \chi_{ps} + \sum_{\substack{s\in\mathscr{S}_p \\ S_z(p)=s \\ S_{\bar{z}}(p)\neq s}} (1-\chi_{ps}) \right] \tag{5.43}$$

where $\bar{C}$ is the average cost of all arcs in the problem, and A_a and A_r are the sets of arcs that are added to and removed from the solution z in the move to $\bar{z}$, respectively.

In this way, we introduce into the solution new arcs and supply-point assignments which helps to direct the search into unexplored regions. The diversification mechanism is executed IT_{div} iterations.

5.5.8 Post optimization

The best solution obtained through the tabu search is enhanced by applying a local-search *Supply-point-improvement* procedure and a *Leg-improvement* procedure sequentially. The purpose of implementing two procedures is to improve the routing and the supply-point assignment.

The *Supply-point-improvement* proceeds by assigning a new supply point to each pickup-customer demand, keeping those that actually improve the solution. Pickup-customer demands are handled in random order. Then, for each pickup-customer demand p and each of its unassigned supply point $s \in \mathscr{S}_p$ (if any), p is removed from its current leg (i.e., current assigned supply point) and the cheapest fitness insertion is performed to insert p into each pickup leg assigned to s. The best feasible improvement one is executed (if any). One then proceeds to the next unassigned supply point or, if all have been tried out, to the next pickup-customer demand.

The *Leg-improvement* is then performed by applying a number of well-known local-search route improvement techniques. Two are intra-route operators, the 2-opt of Lin (1965) and the Or-opt of Or (1976). The others are inter-route operators, the λ-interchange of Osman (1993), and the CROSS-exchange of Taillard et al. (1997). For the λ-interchange, we only consider the cases where $\lambda = 1$ and $\lambda = 2$ corresponding to the (1,0), (1,1), (2,0), (2,1), and (2,2)-interchange operators. A delivery-customer demand is re-allocated only to legs with the same initial supply point. This procedure is therefore executed for each delivery customer zone separately. For pickup-customer demands which could change their supply points, the procedure is executed for all pairs of pickup-customer demands satisfying the supply-point assignment.

The procedure starts by applying in random order the five λ-interchange and CROSS-exchange inter-route operators. Each neighborhood is searched on all possible pairs of legs (in random order) and stopped on the first feasible improvement. The solution is then modified and the process is repeated until no further improvement can be found. The search is then continued by locally improving each leg of each vehicle in turn. The intra-route 2-opt and Or-opt neighborhoods are sequentially and repeatedly applied until no more improvement is found.

5.6 Experiments

Because our problem is new, no benchmark instances are available for it. We have first created MZT-PDTWS test instances from known TMZT-VRPTW benchmark problems. Next, we have studied the impact of a number of major parameters and search strategies on the performance of the proposed algorithm in order to identify the best design. We then have analyzed the impact of sharing the same fleet of vehicles and synchronization schemes on solution quality. Finally, in order to evaluate the performance of the method, we made comparisons with published results of the VRPB with and without time windows.

Our tabu search algorithm is implemented in C++. Experiments were run on a 2.8 GHz Intel Xeon 4-core processor with 16GB of RAM.

5.6.1 Test data generation

We have generated new data sets for our problem based on the TMZT-VRPTW instances proposed in Crainic et al. (2009) and two given parameters, the ratio $BH = |\sum_{p\in\mathscr{C}^P} q_p / \sum_{i\in\mathscr{C}^P\cup\mathscr{C}^D} q_i|$ - the total demand of backhauls over the total demand of backhauls and linehauls, and the value $M_{SP} = max_{p\in\mathscr{C}^P} \left\|S_p\right\|$.

Based on the hypothesis that the volume of goods moving out of the city is relatively lower than the volume of goods moving in, we have set the values of BH at {0.1, 0.3, 0.5}. For the sake of simplification, we have also used BH as the ratio of the number of backhauls over the total number of backhauls and linehauls.

We have generated six sets of 15 instances each, for a total of 90 problem instances. The six sets are called A1, A2, B1, B2, C1, and C2. Each set is further divided into three groups of 5 instances, each group is defined by one of the three different values of $BH = \{0.1, 0.3, 0.5\}$. Table 5.3 summarizes the parameters of all the MZT-PDTWS instances. The last column lists the names of the TMZT-VRPTW instances used to create our new MZT-PDTWS benchmark.

Each MZT-PDTWS instance is constructed from a TMZT-VRPTW to which a set of new pickup-customer demands has been added, while all delivery-customer demands, supply points, waiting stations and their attributes in the TMZT-VRPTW are kept unchanged. The added pickup-customer demands are generated based on a value of BH. The attributes of each pickup-customer demand p are generated as follows:

- The coordinates $[X_p, Y_p]$ are uniformly distributed in the same interval used to generate coordinates of the delivery-customer demands.
- The volume of demand q_p is randomly generated in the same interval as for delivery-customer demands, i.e., [5, 25], with respect to the value of BH.
- The service time $\delta(p)$ is set to 20 as in TMZT-VRPTW.
- The number of available supply points assigned to pickup-customer demand p is selected randomly in the range [1, M_{SP}]. Let x denote this number. Then, the list of available supply points assigned to p is determined by randomly selecting x supply points in the problem.
- Time window $[e_p, l_p]$: Assuming $s_1, s_2, ..., s_x$ are x supply points available to service pickup-customer demand p in increasing order of opening times. To ensure feasibility, the values of e_p and l_p are then chosen randomly in the interval [E_p - 300, E_p] and [L_p - 300, L_p], respectively, where $E_p = t(s_1) - \delta(p) - \lceil c_{p,s_1} \rceil$ and $L_p = t(s_x) - \delta(p) - \lceil c_{p,s_x} \rceil$.

All other attributes are kept the same as in the TMZT-VRPTW instances. The numbers of supply points (waiting stations) for the six sets are 4(4), 8(4), 16(16), 32(16), 36(36), and 72(36), respectively. Supply points, waiting stations, and customers are uniformly distributed in a square, with the X and Y coordinates in the interval [0, 100], [0, 200], and [0,300] for set of type A, B, and C, respectively. The opening times of supply points are generated randomly in the [1000, 15,400] range, while the limited allowable waiting time at supply points $\eta = 100$. The vehicle-loading and vehicle-unloading times at supply points are set to 30, for all supply points. The fixed cost and the capacity of each vehicle are set to 500 and 100, respectively, for all instance sets.

Table 5.3: Summary of the benchmark test

Problem set	Instances name	Number of supply points	Number of waiting stations	BH	Number of customers		[X,Y] customer coordinates	M_{SP}	Original instances by Crainic et al. (2009)
					Linehauls	Backhauls			
A1	A1-1 ... A1-5	4	4	0.1	400	44	[0,100]	2	A1-1 ... A1-5
	A1-6 ... A1-10			0.3		171			
	A1-11 ... A1-15			0.5		400			
A2	A2-1 ... A2-5	8	4	0.1	400	44	[0,100]	2	A2-1 ... A2-5
	A2-6 ... A2-10			0.3		171			
	A2-11 ... A2-15			0.5		400			
B1	B1-1 ... B1-5	16	16	0.1	1600	177	[0,200]	3	B1-1 ... B1-5
	B1-6 ... B1-10			0.3		685			
	B1-11 ... B1-15			0.5		1600			
B2	B2-1 ... B2-5	32	16	0.1	1600	177	[0,200]	3	B2-1 ... B2-5
	B2-6 ... B2-10			0.3		685			
	B2-11 ... B2-15			0.5		1600			
C1	C1-1 ... C1-5	36	36	0.1	3600	400	[0,300]	4	C1-1 ... C1-5
	C1-6 ... C1-10			0.3		1542			
	C1-11 ... C1-15			0.5		3600			
C2	C2-1 ... C2-5	72	36	0.1	3600	400	[0,300]	4	C2-1 ... C2-5
	C2-6 ... C2-10			0.3		1542			
	C2-11 ... C2-15			0.5		3600			

5.6.2 *Algorithm design and calibration*

We have aimed for a general algorithmic structure avoiding instance-related parameter settings. We have therefore defined settings as functions of problem size for the main parameters of the proposed algorithm, tabu tenure, neighborhood selection-control.

5.6.2.1 Tabu tenure calibration

The intervals for the tabu list tenures for leg, delivery, and pickup routing moves were defined in Section 5.5.6 as $[m'*|\mathscr{S}|/a_1,\ m'*|\mathscr{S}|/a_2]$, $[a_3 log_{10}(|\mathscr{C}_s^D|),\ a_4 log_{10}(|\mathscr{C}_s^D|)]$, and $[a_5 log_{10}(|\mathscr{C}^P|),\ a_6 log_{10}(|\mathscr{C}^P|)]$, respectively. Using a large interval for routing moves, [10, 20], we tested different values for a_1 in the integer interval [7, 10] and for a_2 in the integer interval [4, 6]. We have observed that too large an interval is not productive as low values cannot prevent cycling, while high ones overly restrict the search path. We have therefore set a_1 and a_2 to 7 and 5, respectively.

A similar process has been used to explore different values of a_3, a_4, a_5, a_6 in the integer interval [4, 6], [7, 9], [6, 8] and [10, 12], respectively, using delivery and pickup routing tabu as defined above. We have used a larger value of tabu tenure for routing moves on pickup-customer demands as they are not restricted to one customer zone as those on delivery-customer demands. We found that the most appropriate values for a_3, a_4, a_5 and a_6 are 6, 8, 7 and 10, respectively.

5.6.2.2 Calibration of the neighborhood selection probabilities

Adjustments to the neighborhood selection probabilities depend on two parameters: IT_{cNS}, the number of consecutive iterations without improvement of the best solution (this number triggers the execution of the *Control* procedure that modifies probabilities), and $\Delta\bar{r}$, the adjustment factor of the neighborhood-selection parameter $\bar{r}$.

The value of IT_{cNS} is defined as a function of the problem size. This value should be large enough to give each customer and supply point in each leg the possibility to be moved. Thus, $IT_{cNS} = e_1 * (m' * |\mathscr{S}| + n)$, where m' is the number of vehicles used in the initial solution, $|\mathscr{S}|$ and n are the numbers of supply points and customer demands, respectively, and e_1 is a user defined parameter. Similarly, $\Delta\bar{r}$, the amplitude of the modifications in the probabilities, is set to be proportional to the ratio of the number of customer demands with the number of supply points. Thus, $\Delta\bar{r} = e_2 \log_{10}(n/|S|)$, where e_2 is a user defined parameter.

Searching for a good combination of values for e_1 and e_2 concerns balancing between exploration and exploitation. On one hand, the higher the value of IT_{cNS}, the more chances customers and supply points are to be moved between routes, thus favoring exploration. On the other hand, a too high IT_{cNS} value may waste time in useless moves. We have experimented with different values of e_1 in the integer interval [1,5] and e_2 in the integer interval [1, 7]. Three runs were performed for each instance for one million iterations. Computational results for each combination of values (e_1, e_2) over all instances are summed up in Table 5.4, which displays the average gaps between the best solutions obtained by each combination and the best combination.

Table 5.4: Performance comparison between (e_1, e_2) combinations

e_1	e_2						
	1	2	3	4	5	6	7
1	1.25%	1.04%	0.43%	0.34%	0.32%	0.28%	0.28%
2	1.14%	0.98%	0.21%	0.23%	0.26%	0.31%	0.31%
3	1.12%	0.73%	0.09%	0.06%	0%	0.08%	0.17%
4	0.97%	0.71%	0.14%	0.08%	0.04%	0.18%	0.21%
5	1.05%	0.68%	0.12%	0.07%	0.05%	0.17%	0.28%

Table 5.4 indicates that (3,5) is the most appropriate combination for (e_1, e_2), giving best solutions on average. We have also observed that executing the algorithm with $\bar{r}$ greater than 60 $\log_{10}(n/|S|)$ yields an average improvement of the best solution of less than 0.1%, while requiring about 41% more time. Based on these results, we used (e_1, e_2) = (3, 5) and $\bar{r}_{max} = 60\log_{10}(n/|S|)$, the maximum value of $\bar{r}$, in the remaining experiments.

5.6.2.3 Elite set calibration, diversification

We now turn to the parameters characterizing the diversification procedure and the elite set utilization, and examine their impact on the performance of the algorithm. Four variants of the algorithm were studied corresponding to the different ways to set an elite solution as the new working solution and the inclusion, or not, of the diversification phase. The first two variants simply select an elite solution z at random and re-start the algorithm from it. The *Diversification* mechanism described in Section 5.5.7 is applied in the last two variants to diversify from the elite solution z.

The initialization of the $\overline{r}$ parameter following the selection of z is a component common to the four variants. We have studied two alternatives where $\overline{r}$ was set to either the full or half the value at which z was found, respectively (i.e., $\overline{r} = \overline{r}_z$ or $\overline{r} = \overline{r}_z/2$). The size of the elite set is relevant for the *Diversification* mechanism only. Three values were tested, 1, 5, and 10.

Similar to previous experiments, we have used formulas dependent on the problem dimensions for IT_{div} and C_{cNS}, which determine for how long exploration can proceed. Thus, the number of diversification phases is set to $IT_{div} = m' * |\mathscr{S}| + n$, where m' is the number of vehicles used in the initial solution, and $|\mathscr{S}|$ and n are the numbers of supply points and customer demands, respectively.

We have also set the number of consecutive executions of the *Control* procedure without improvement of the best solution to $C_{cNS} = min(3\log_{10}(n/|S|), (\overline{r}_{max} - \overline{r})/\Delta_{\overline{r}})$, which keeps the value of C_{cNS} sufficiently high during the course of the algorithm, even though *Control* procedure is started with different values of $\overline{r}$ (remember that $\overline{r}_{max} = 60\log_{10}(n/|S|)$). Intuitively, in the beginning, $\overline{r}$ is small and C_{cNS} takes the value $3\log_{10}(n/|S|)$, while when $\overline{r}$ becomes large enough, C_{cNS} takes the value $(\overline{r}_{max} - \overline{r})/\Delta_{\overline{r}}$.

Table 5.5 displays the performance comparison between the four variants with the three different values for the elite set size. For each variant and size of the elite set, the table shows the average gaps to the cost of the best solutions obtained by it from those obtained without using the elite set and diversification, together with the corresponding average computation time in minutes over 10 runs.

As expected, results indicate that guidance using elite solutions contributes significantly to improve the performance of the algorithm. Without using the elite set, the algorithm requires the lowest computation effort but produces worst solutions compared to all the variants using the elite set. Comparing the two variants corresponding to the two values at which $\overline{r}$ is reset, one observes that the solution quality is not very sensitive to this value, but computing effort is increasing when the value of $\overline{r}$ is lower ($\overline{r} = \overline{r}_z/2$).

Table 5.5: Performance comparison between diversification settings

Elite set size	Without diversification				With diversification			
	1st variant		2nd variant		3rd variant		4th variant	
	$r = r_z$		$r = r_z/2$		$r = r_z$		$r = r_z/2$	
	GAP (%)	Time	GAP (%)	Time	GAP (%)	Time	GAP (%)	Time
0	0	50	-	-	-	-	-	-
1	-0.37	66	-0.36	92	-1.02	88	-1.05	103
5	-0.64	95	-0.69	117	-1.54	157	-1.48	194
10	-0.78	121	-0.74	139	-1.55	223	-1.50	260

One observes that the third and fourth variants are significantly better in terms of finding high quality solutions. This indicates that the long-term memory and the diversification mechanism added to the algorithm are important features for high performance. Moreover, setting the size of the elite set to 5 achieves a better balance between solution quality and computation time, compared to a larger size of 10. Indeed, doubling the size of the elite set improves only slightly the solution quality, 0.01%, but requires 42% more time. We therefore set the size of the elite set to 5 and reset $\bar{r} = \bar{r}_z$.

5.6.3 Numerical results

Table 5.6 displays the results obtained by the proposed tabu search meta-heuristic over 10 runs for each group of instances. It gives the average results (*Avg 10* column), the best results *(Best 10* column), the number of vehicles (#*Vehicles* column), the percentage of times vehicles move directly to supply points without using waiting stations (*DM(%)* column), and the percentage of times vehicles do both unload and load once they arrive at supply points (*PD(%)* column). Average computation times in minutes are displayed in the *Time* column.

Experiment results show that, in total, 4874 vehicles are used in the 90 problem instances, servicing a total of 39790 legs. Hence, on average, each vehicle services 8 legs. Table 5.6 shows that the percentage of times vehicles do both unload and load increases

Table 5.6: Performance of TS on all instances

Problem set	BH	Avg 10	Best 10	#Vehicles	DM (%)	PD (%)	Time(min)
	0.1	19873.29	19758.67	21.8	10.45	21.89	20
A1	0.3	21007.60	20854.25	22	27.44	60.93	34
	0.5	23455.87	23245.62	22.2	51.29	87.1	58
	0.1	16884.05	16756.85	16.4	14.77	21.52	12
A2	0.3	18462.56	18295.76	16.4	31.75	56.75	19
	0.5	21150.77	20981.06	17.2	45.28	88.05	33
	0.1	66979.79	66763.80	46.8	19.33	15.01	66
B1	0.3	75587.05	75398.22	47.8	31.3	46.73	139
	0.5	99155.77	99025.96	54.8	38.31	80.39	231
	0.1	59828.68	59717.48	36.4	19.06	16.53	42
B2	0.3	72098.73	71945.56	40	23.64	46.76	97
	0.5	94024.35	93838.52	46	32.63	78.41	198
	0.1	153335.20	153106.40	90.4	17.65	13.84	172
C1	0.3	200072.40	199848.80	99.4	21.78	46.01	310
	0.5	292032.84	291836.60	119.8	30.91	82.58	705
	0.1	141018.12	140803.04	76.2	18.26	15.65	112
C2	0.3	195573.18	195206.00	94.4	24.59	41.92	213
	0.5	278354.82	278058.20	106.8	26.45	77.77	348
Average		102716.39	102524.49	54.16	26.94	49.88	156.06

proportionally to the percentage of pickup-customer demands (i.e., the value of BH). On average, 49.88% of times the vehicles do both unload and load once they arrive at supply points. This factor of 'unload and load' operations at supply points not only reduces the number of empty moves but also the traveling cost. Moreover, experiments show that the traveling cost and the number of vehicles of initial solutions are 32.45% and 20.76% greater than those of best solutions on average, respectively, illustrating the significant solution-improvement effect of the proposed algorithm.

5.6.4 The benefits of combining linehauls and backhauls

The combining of linehaul- and backhaul-customer demands on each vehicle is expected to reduce the total number of vehicles and total traveling cost with respect to the case when linehauls and backhauls are serviced on separate vehicles. Table 5.7 compares the best solutions of both alternatives for all instances over 10 runs. In this table, LH-BH refers to the solutions with linehaul- and backhaul-customer demands on the same vehicles, while LH+BH refers to the solutions with linehaul- and backhaul-customer demands on different vehicles. The LH+BH solutions can be seen as the summation of solutions to the problem in two cases, one dealing with only linehaul-customer demands and the other dealing with only backhaul-customer demands. The average number of vehicles, traveling cost, and total cost of LH-BH solutions on each set of problems are given in column *#Vehicles*, *Traveling cost*, and *Total cost*, respectively. For the column 'LH+BH', each entry consists of three numbers indicating the gaps between attributes of solutions obtained by LH+BH and those obtained by LH-BH. The first number displays the gap to the average number of vehicles obtained by the LH+BH strategy from the average number of vehicles obtained by the LH-BH strategy, while the second and third numbers indicate the gaps of average traveling cost and total cost, respectively.

As expected, results indicate that assigning linehauls and backhauls to separate fleets of vehicles leads to an increase in both the average number of vehicles and traveling cost. This increase becomes significant when more backhauls are serviced, i.e., higher value of BH. In all cases, an increase in total cost is thus also observed, with an average gap of 27.63% and a maximal gap of 62.48%.

Table 5.7: Comparison of separate and combined linehaul and backhaul solutions in number of vehicles, traveling cost, and total cost

Problem set	BH	LH-BH			LH+BH		
		#Vehicles	Traveling cost	Total cost	GAP (%)		
A1	0.1	21.8	8858.67	19758.67	12.84	9.08	11.16
	0.3	22	9854.25	20854.25	45.45	26.25	36.38
	0.5	22.2	12145.62	23245.62	89.19	38.06	62.48
A2	0.1	16.4	8556.85	16756.85	13.41	10.14	11.74
	0.3	16.4	10095.76	18295.76	35.37	21.91	27.94
	0.5	17.2	12381.06	20981.06	69.77	34.57	49.00
B1	0.1	46.8	43363.80	66763.80	8.55	11.52	10.48
	0.3	47.8	51498.22	75398.22	35.98	29.50	31.55
	0.5	54.8	71625.96	99025.96	48.18	33.71	37.71
B2	0.1	36.4	41517.48	59717.48	10.99	10.65	10.76
	0.3	40	51945.56	71945.56	38.00	28.73	31.31
	0.5	46	70838.52	93838.52	57.39	29.76	36.53
C1	0.1	90.4	107906.40	153106.40	0.22	14.68	10.41
	0.3	99.4	150148.80	199848.80	24.55	28.87	27.79
	0.5	119.8	231936.60	291836.60	39.07	26.11	28.77
C2	0.1	76.2	102703.04	140803.04	0.79	22.56	16.67
	0.3	94.4	148006.00	195206.00	10.59	31.48	26.43
	0.5	106.8	224658.20	278058.20	35.39	29.00	30.23
Average		54.16	75446.71	102524.49	31.98	24.25	27.63

5.6.5 *Synchronization at supply points*

Each supply point is defined as a combination of a satellite and an availability time period in our problem. Thus the vehicles must arrive at supply points during these predefined periods to unload and/or load freight. In this section, we analyze the impact of synchronization of vehicles' operations requirement at supply points on solution quality.

In all previous experiments, the requirement for availability of vehicle at each supply point s is characterized by only one time window $[e_s, l_s]$ of s which is used for both unload and load operations. In order to analyze the impact of available requirements without modifying time windows at customer demands, we introduce into the model two time windows for unloading and loading respectively at each supply point, but keep the availability time periods of supply points unchanged. More precisely, we use $[e_s^u, l_s^u]$ and $[e_s^l, l_s^l]$, specifying the earliest and latest times at which the vehicle has to be available at s for unloading collected demands and loading delivery demands, respectively, where $l_s^u + \varphi'(s) \leq l_s^l$, $e_s^u = e_s$ and $l_s^l = l_s$. Activities of a vehicle at s are then described as follows:

- Only unload at s: the vehicle arrives at s with pickup demands at time t within its unload time window $[e_s^u, l_s^u]$, i.e., the vehicle must not arrive at s sooner than e_s^u and no later than l_s^u; it takes $\varphi'(s)$ for unloading all demands, the vehicle thus leaves s empty at time $t + \varphi'(s)$;

- Only load at s: the vehicle arrives at s empty at time t within its load time window $[e_s^l, l_s^l]$, i.e., the vehicle must not arrive at s sooner than e_s^l and no later than l_s^l; it takes $\varphi(s)$ for loading delivery demands, with a total load not exceeding Q; the vehicle then leaves s at time $t + \varphi(s)$ to perform the delivery for serving a subset of delivery customers in $\mathscr{C}_s^D$;

- Unload and load at s: the vehicle arrives at s with pickup demands at time t within its unload time window $[e_s^u, l_s^u]$; it takes $\varphi'(s)$ for unloading all demands; in case $t + \varphi'(s) < e_s^l$, the vehicle has to wait at supply point till e_s^l to start loading freight; otherwise it starts to load freight at $t + \varphi'(s)$; it takes $\varphi(s)$ for loading, then the vehicle leaves s to deliver all loaded freight to a subset of customers in $\mathscr{C}_s^D$.

The unload and load time windows at each supply point are defined by two parameters: the length of each unload and load time window (denoted by len_u and len_l, respectively; we set $len_u = len_l$ in our experiment), and the difference between e_s^l and l_s^u (see Figure 5.9). We performed three runs with values of these two parameters equal to (20, 60), (30, 40) and (40,20) (remember that the length of time window at each supply point equals to 100 in all instances).

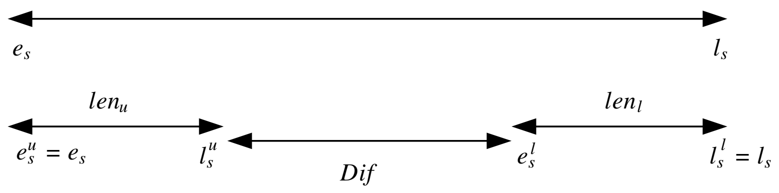


Figure 5.9: Illustration of two time windows at a supply point s

Table 5.8: Impact of synchronization at supply points on solution quality

	One time window	Two time windows		
	Len = 100, Dif = 0	Len = 20, Dif = 60	Len = 30, Dif = 40	Len = 40, Dif = 20
#Vehicles (%)	0	1.03	0.79	0.52
Traveling cost (%)	0	2.17	0.82	0.74
Total cost (%)	0	1.94	0.88	0.75
PD (%)	49.88	45.40	46.98	48.31
DM (%)	26.94	22.45	23.95	24.01

The experiment was run on all instances. Table 5.8 sums up the solution-quality variations for three cases of two time windows compared to the case of only one time window. The table displays the solution-quality variations in terms of the number of vehicles, traveling cost, and total cost. The percentage of times vehicles do both unload and load at supply points (*PD(%)* row) and the percentage of times vehicles move directly to a supply point without using waiting stations (*DM(%)* row) are also given.

Results indicate that solutions to the cases with two time windows are worse than those with one time window in terms of both number of vehicles and traveling cost. Moreover, longer waiting-time capabilities of supply points result in vehicles moving directly to supply points more frequently and doing more ‘unload and load’ operations (maximum of 26.94% and 49.88% respectively, both from the case of only one time window).

5.6.6 Comparing with the published results for the VRPB

As mentioned previously, the proposed algorithm can be directly applied to solve the VRPB which is a special case of the MZT-PDTWS. Hence, in this subsection, we compare our algorithm with the existing algorithms in the literature for the VRPB, in both cases with and without time windows.

5.6.6.1 Vehicle Routing problem with Backhauls and Time windows

In the Vehicle Routing problem with Backhauls and Time windows (VRPBTW), time windows at customers and duration of the route are considered. We have run our tabu search using only our routing neighborhoods on the Gélinas et al. (1995) 15 VRPBTW 100-customer instances. All parameters related to the supply points were discarded.

Different exact and meta-heuristic algorithms for the VRPBTW may be found in the literature. Gélinas et al. (1995) proposed a branching strategy for branch-and-bound approaches based on column generation. This algorithm found optimal solutions to 6 test problems. Potvin et al. (1996a) designed a genetic algorithm to identify an ordering of customers that produces good routes. This ordering was then used by a greedy route construction heuristic to insert the customers one by one into the routes. Simple construction and improvement algorithms (using λ-interchange and 2-opt*) were developed by Thangiah et al. (1996). The ant system approach was used by Reimann et al. (2002) where only global pheromone updating was applied. A two-phase heuristic was proposed by Zhong and Cole (2005) in which customers were clustered in the first phase, then in the second phase, three route improvement routines (2-opt, 1-move, 1-exchange) were applied within a guided local search framework. Ropke and Pisinger (2006) developed a large neighborhood search which applied several competing removal and insertion heuristics. The selection of a heuristic was based on statistics gathered during the search.

Table 5.9 compares the performances obtained by our algorithm with the results of these algorithms. The first column gives the name of the authors of the study. The next fifteen columns present the number of vehicles and total distance with respect to the 15 instances, respectively. These 15 instances are divided into five groups, i.e., R101, R102, R103, R104, R105, with three different percentages of backhaul customers (%BH) in each group. Finally, the rightmost column indicates the cumulative number of vehicles (CNV) and cumulative total distance (CTD) over all 15 instances. Most algorithms in the literature (except Gélinas et al. (1995)) actually aimed first to reduce the number of vehicles. We do not, as our algorithm treats vehicles through supply point neighborhoods not considered in these experiments, and we do not compete with the other meta-heuristics on this count. We do compete with respect to the total distance, though, outperforming four out of the five meta-heuristics (with an average gap of 1.08%, a maximal gap of 2.81% and a minimal gap of -0.34%). This comparison, however, is not totally fair given that the other methods optimize another objective by first minimizing the number of vehicles.

5.6.6.2 Vehicle Routing problem with Backhauls

The next round of experiments focused on the VRPB which is obtained by removing the constraints of time windows at customers and the route duration from the VRPBTW. The performance of the proposed tabu search is evaluated through comparison with results of other tabu search algorithms on two sets of instances in the VRPB literature. The first set of 62 instances was proposed in Goetschalckx and Jacobs-Blecha (1989). The instances range in size between 25 and 150 customers with backhauls ranging between 20 and 50%. The second set of 33 instances was proposed by Toth and Vigo (1997) where the number of customers range between 21 and 100, and backhauls percentages are either 20, 34 or 50%. In the VRPB literature, there are two different ways to compute the Euclidean distances between pairs of customers, namely real-valued and integer-valued cost matrix, respectively. The former matrix was used for all three tabu search algorithms with which we compare our method. Therefore, in this experiment, we only use real-valued cost matrix whose entries are the Euclidean distances.

Table 5.10 sums up the comparison of average of the best solutions for two sets of instances. The first column gives the name of the authors of the study. The average of the best solutions obtained by each study is given in the columns *Cost*. For completion

Table 5.9: Performance comparison with algorithms for the VRPBTW

Authors	R101			R102			R103			R104			R105			CNV/CTD
	%BH			%BH			%BH			%BH			%BH			
	10%	30%	50%	10%	30%	50%	10%	30%	50%	10%	30%	50%	10%	30%	50%	
Gélinas et al. (1995)	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
	1767.9	1877.6	1895.1	1600.5	1639.2	1721.3	-	-	-	-	-	-	-	-	-	-
Thangiah et al. (1996)	24	24	25	20	21	21	15	16	17	13	12	13	17	18	18	274
	1842.3	1928.6	1937.6	1654.1	1764.3	1745.7	1371.6	1477.6	1543.2	1220.3	1302.5	1346.6	1553.4	1706.7	1657.4	24051.9
Potvin et al. (1996a)	23	23	24	20	20	21	16	15	17	12	12	13	17	16	18	267
	1815	1896.6	1905.9	1622.9	1688.1	1735.7	1343.7	1381.6	1456.6	1117.7	1169.1	1203.7	1621	1652.8	1706.7	23317.1
Reimann et al. (2002)	22	23	24	19	22	22	16	16	17	11	12	12	16	16	17	265
	1831.68	1999.16	1945.29	1677.62	1754.43	1782.21	1348.41	1395.88	1467.66	1205.78	1128.3	1208.46	1544.81	1592.23	1633.01	23514.93
Zhong and Cole (2005)	24	24	25	-	-	-	-	-	-	-	-	-	17	17	19	-
	1848.04	2034.61	2057.05	-	-	-	-	-	-	-	-	-	1590.54	1667.92	1699.88	-
Reimann and Ulrich (2006)	22	23	24	19	22	22	15	15	17	11	11	11	16	16	17	261
	1853.45	1985.23	1964.04	1663.16	1759.02	1782.91	1454.25	1407.29	1478.48	1153.06	1228.62	1306.97	1570.11	1646.11	1689.74	23942.44
Ropke and Pisinger (2006)	22	23	24	19	22	22	15	15	17	11	11	11	15	16	16	259
	1818.86	1959.56	1939.1	1653.19	1750.7	1775.76	1387.57	1390.33	1456.58	1084.17	1154.84	1191.38	1561.28	1583.3	1710.19	23416.81
TS	22	23	24	19	22	22	15	15	17	11	11	12	16	16	17	263
	1823.64	1897.79	1917.87	1665.93	1758.31	1781.46	1402.63	1382.08	1471.43	1102.21	1181.17	1204.59	1571.42	1586.66	1648.32	23395.51

sake, we also included the GAP for these studies relative to the average of best known solutions in the *GAP to BKS (%)* columns. One observes that the proposed TS performs well, outperforming all three other tabu search algorithms on Goetschalckx and Jacobs-Blecha (1989) instances (with an average gap of 0.05% and a maximal gap of 0.1%), and only worse than Brandão (2006) on Toth and Vigo (1997) instances (with a gap of -0.47%).

Table 5.10: Performance comparison with tabu search algorithms for the VRPB

Authors	Goetschalckx and Jacobs-Blecha (1989)		Toth and Vigo (1997)	
	Cost	GAP to BKS (%)	Cost	GAP to BKS (%)
Osman and Wassan (2002)	291261.7	0.25	708.42	1.09
Brandão (2006)	291160.5	0.21	702.15	0.19
Wassan (2007)	290981.8	0.15	706.48	0.81
TS	290964.7	0.14	705.49	0.67

5.7 Conclusion

We studied the MZT-PDTWS, a new vehicle routing problem variant in which each vehicle performs multiple sequences of delivery and pickup through supply points within time synchronization restrictions. We proposed the first model formulation and a tabu search meta-heuristic integrating multiple neighborhoods for the problem. The computational study was performed on the first benchmark instances with up to 72 supply points and 7200 customer demands. Our experiments reveal that longer waiting-time capabilities of supply points tend to yield better results as it reduces the utilization of waiting stations, but also the number of empty trips. The MZT-PDTWS is a new problem and no previous results are available. We thus evaluated the performance of the proposed method through comparisons with published results on the VRPB as the MZT-PDTWS generalizes this problem. The experiments indicated that the proposed method is competitive with other meta-heuristics for both the cases with and without time windows.

Chapter 6

CONCLUSIONS

The Vehicle Routing Problem is a problem related to the distribution of goods, and to logistics in general. Solving the VRP efficiently is key to efficient transport planning and management. Today's exact algorithms can solve instances having a small number of customers in terms of real-life applications. Hence, solving VRP optimally in reasonable time is still a challenge, particularly for large-scale instances and real-life, rich VRP variants. Meta-heuristic approaches do not guarantee optimality, but they significantly reduce the computational time needed to find solutions, which are often the best known solutions.

The main objective of this dissertation was to introduce models and efficient algorithms for three rich vehicle routing problems arising in many real-life applications. They are the PVRPTW, the TMZT-VRPTW and the MZT-PDTWS. The first two problems are studied in very few papers, while the last problem is considered for the first time in the literature. Due to the high computational complexity and the need to solve realistically-sized instances, our focus has been on developing meta-heuristics.

Since the problems we have studied in this dissertation belong to the VRP domain, we have first summarized the most important concepts, solution methods and the progress made within this area in Chapter 2.

In Chapter 3, we have tackled the Periodic VRP with Time Windows (PVRPTW) which is a generalization of VRPTW as it extends the planning horizon to several days. We have proposed a new population-based hybrid meta-heuristic to address this problem in which two neighborhood based meta-heuristics are used to educate the offspring generated by new crossover operators to enhance the solution quality. This hybridization provides the means to combine the exploration capabilities of population-based methods and the systematic, sometimes aggressive search capabilities of neighborhood based methods, as well as their proficiency to explore the infeasible part of the search space to repair infeasible solutions. Numerical experiments show that the proposed methodology is highly competitive, providing new best solutions in some large instances.

Chapter 4 addressed the Time-dependent Multi-zone Multi-trip VRP with Time Windows (TMZT-VRPTW). A decomposition approach has previously been proposed to solve the problem (Crainic et al., 2009, 2012b) in which two key sets of decisions in the problem were addressed separately. These two sets of decisions, 1- how to service customers associated to given supply points, and 2- how to combine the resulting trips into vehicle-specific multi-trip routes abiding by the synchronization requirements at supply points, are not independent, however. We thus have proposed a tabu search meta-heuristic, integrating multiple neighborhoods into two classes to address these two sets of decisions of the problem simultaneously. The first set applied to each vehicle trip exploits good routing, while the second works on the construction of the multiple-trip vehicle routes, perturbing significantly the solution, and thus favor exploration of the search space. The selection of neighborhoods is dynamically adjusted along the search, providing the proposed algorithm with desired exploration and exploitation capabilities. Further, a diversification strategy guided by an elite set of solutions and a frequency-based memory is also used to provide a certain level of diversity to the search, but also helps incorporate good attributes into newly created solutions. Extensive computational experiments illustrate clearly the good performance of the proposed methodology compared to the literature. It yields higher quality solutions in terms of both required number of vehicles and traveling cost.

Chapter 5 studied the Multi-zone Multi trip Pickup and Delivery Problem with Time Windows and Synchronization (MZT-PDTWS). The MZT-PDTWS generalizes the TMZT-VRPTW as it accounts for both inbound and outbound traffic. Hence, the same fleet of vehicles and given set of supply points are shared for both types of traffic. These shared services increase the synchronization challenges at supply points, the complexity to manage the traffic of vehicles into and out of supply points in particular, as well as to route vehicles doing both pickup and delivery operations through supply points. This problem has not been investigated in the literature. We have introduced a model for the MZT-PDTWS, and generalized the tabu search proposed for the TMZT-VRPTW in order to tackle this new problem efficiently. Extensive computational experiments have been conducted to qualify the impact of a number of major problem characteristics, parameters and search strategies on the quality of the solutions and to underline the good performance of the proposed algorithm. The MZT-PDTWS also generalizes the VRP with Backhauls. As no previous results were available in the literature for the MZT-PDTWS, we have also evaluated the performance of the method through comparisons with currently published results on the VRPB for both cases with and without time windows.

In summary, we proposed efficient meta-heuristics for three rich vehicle routing problems. In the first study, we introduced the use of neighborhood-based meta-heuristics within a generational genetic algorithm. This hybridization provided the means to repair and enhance individuals produced during the evolution process, as well as to promote diversity of the genetic algorithm population. In the next studies, we studied more complicated VRP variants. A decomposition approach has previously been proposed to solve the problems in which sets of decisions in the problems were addressed separately. We therefore proposed a new neighborhood-based meta-heuristic that addresses these sets of decisions simultaneously, in a comprehensive and efficient way. Extensive numerical experiments and comparisons with the literature showed that our proposed methods yield high quality solutions, providing many new best solutions or matching many of the existing ones.

Through the three problems studied in this dissertation, we have found that real-world rich VRP variants pose challenging research issues arising from the multiple intertwined optimization subproblems. Some studies address these subproblems sequentially and separately, while others take a more holistic perspective in which subproblems cooperate and are intertwined. The latter yields better results, but also asks for more complex designs of methodology, intelligent cooperation between decisions that have to be taken during the solution process. The utilization of multiple neighborhoods targeting different levels of decisions is a good approach to handle such problems. However, the definition of neighborhood structures, neighborhood combinations, transitions and cooperation between neighborhoods are challenges when designing a meta-heuristic algorithm, and often dependent on the specifics of problem at hand. We think that future research should focus on more complex neighborhood structures which help to explore new neighbors, thus hopefully, improving the solution quality. In order to speed-up computation and enhance solutions, parallelization and cooperation schemes between different neighborhoods embedded in tabu search threads are also one research direction that we want to follow. Furthermore, one can run tabu searches and genetic methods in a cooperative search framework to combine the strength of both population-based and neighborhood-based methods to explore the solution space. The tabu searches aim to explore the search space, while the genetic methods contribute toward increasing the diversity of solutions exchanged among the cooperating methods.

Finally, the two last problems studied in this dissertation being new VRP variants, several research issues still need to be addressed, in particular the one concerning the synchronization of vehicles operating at different levels of a supply chain. For example, in the case of two-tier City Logistics systems, both urban vehicles in the first tier and city freighters in the second tier are integrated into a single system. The vehicles in each tier may bring different types of freight traffic, and do multiple trips to exchange freight through shared facilities between the two tiers. Routing customers and scheduling movements of vehicles in this setting impose a more general and interesting VRP variant which has not been investigated in the literature.

BIBLIOGRAPHY

T. J. Ai and V. Kachitvichyanukul. A particle swarm optimization for the Vehicle Routing Problem with simultaneous pickup and delivery. *Computers & Operations Research*, 36 (5):1693–1702, 2009.

R. M. Aiex, M. G. C. Resende, and C. C. Ribeiro. TTTPLOTS: A perl program to create time-to-target plots. *Optimization Letters*, 1:355–366, 2007.

I. K. Altinel and T. Oncan. A new enhancement of the Clarke and Wright savings heuristic for the capacitated vehicle routing problem. *Journal of the Operational Research Society*, 56:954–961, 2005.

N. Azi, M. Gendreau, and J.-Y. Potvin. An adaptive large neighborhood search for a vehicle routing problem with multiple routes. *Computers & Operations Research*, 41:167–173, 2014.

B. D. Backer and V. Furnon. Meta-heuristics in Constraint programming Experiments with Tabu Search on the vehicle routing problem. In *Proceeding Second International Conference Metaheuristics (MIC'97)*, pages 1–14, Sophia Antipolis, France, 1997.

P. Badeau, F. Guertin, M. Gendreau, J.-Y. Potvin, and E. D. Taillard. A parallel tabu search heuristic for the Vehicle Routing Problem with time windows. *Transporation Research Part C: Emerging Technologies*, 5(2):109–122, 1997.

H. Barbosa and A. Lemonge. An adaptive penalty scheme in genetic algorithms for constrained optimization problems. In *Proceedings of the Genetic and Evolutionary Computation Conference, San Francisco, CA, USA*, pages 287–294. Morgan Kaufmann Publishers Inc., 2002.

M. Battarra, M. Monaci, and D. Vigo. An adaptive guidance approach for the heuristic solution of a minimum multiple trip Vehicle Routing Problem. *Computers & Operations Research*, 36(11):3041–3050, 2009.

J. M. Belenguer, M. C. Martinez, and E. Mota. A lower bound for the split delivery vehicle problem. *Operations Research*, 48(5):801–810, 2000.

A. Benjamin and J. E. Beasley. Metaheuristics for the waste collection vehicle routing problem with time windows, driver rest period and multiple disposal facilities. *Computers & Operations Research*, 37(12):2270–2280, 2010.

R. Bent and P. V. Hentenryck. A Two-Stage Hybrid Local Search for the Vehicle Routing Problem with Time Windows. *Transportation Science*, 38(4):515–530, 2004.

R. Bent and P. V. Hentenryck. A two-stage hybrid algorithm for pickup and delivery vehicle routing problems with time windows. *Computers & Operations Research*, 33(4):875 – 893, 2006.

J. J. Bentley. Fast algorithms for geometric traveling salesman problems. *ORSA Journal on Computing*, 4(4):387–411, 1992.

G. Berbeglia, J.-F. Cordeau, I. Gribkovskaia, and G. Laporte. Static pickup and delivery problems: a classification scheme and survey. *TOP*, 15:1–31, 2007.

G. Berbeglia, J.-F. Cordeau, and G. Laporte. Dynamic pickup and delivery problems. *European Journal of Operational Research*, 202:8–15, 2010.

J. Berger, M. Barkaoui, and O. Bräysy. A route-directed hybrid genetic approach for the Vehicle Routing Problem with time windows. *Information Systems and Operational Research*, 41:179–194, 2003.

J. Braca, J. Bramel, and D. Simchi-levi. A Computerized Approach to the New York City School Bus Routing Problem. *IIE Transactions*, 29:693–702, 1997.

J. Brandão. A new tabu search algorithm for the Vehicle Routing Problem with backhauls. *European Journal of Operational Research*, 173(2):540–555, 2006.

J. Brandão and A. Mercer. A tabu search algorithm for the multi-trip vehicle routing and scheduling problem. *European Journal of Operational Research*, 100(1):180–191, 1997.

J. Brandão and A. Mercer. The Multi-Trip Vehicle Routing Problem. *Journal of the Operational Research Society*, 49(8):799–805, 1998.

O. Bräysy. A reactive variable neighborhood search for the vehicle routing problem with time windows. *INFORMS Journal on Computing*, 15:347–368, 2003.

O. Bräysy and M. Gendreau. Vehicle Routing Problem with time windows, Part I: Route Construction and Local Search Algorithms. *Transportation Science*, 39(1):104–118, 2005a.

O. Bräysy and M. Gendreau. Vehicle Routing Problem with time windows, Part II: Metaheuristics. *Transportation Science*, 39(1):119–139, 2005b.

H. I. Calvete, C. Galé, M.-J. Oliveros, and B. Sánchez-Valverde. A goal programming approach to vehicle routing problems with soft time windows. *European Journal of Operational Research*, 177(3):1720–1733, 2007.

A. M. Campbell and J. H. Wilson. Forty years of Periodic Vehicle Routing. *Networks*, 63 (1):2–15, 2014.

J. F. Chen and T. H. Wu. Vehicle Routing Problem with Simultaneous Deliveries and Pickups. *Journal of the Operational Research Society*, 57(5):579–587, 2006.

W.-C. Chiang and R. A. Russell. Simulated annealing metaheuristics for the Vehicle Routing Problem with time windows. *Annals of Operations Research*, 63:3–27, 1996.

W.-C. Chiang and R. A. Russell. A Reactive Tabu Search Metaheuristic for the Vehicle Routing Problem with Time Windows. *INFORMS Journal on Computing*, 9:417–43, 1997.

N. Christofides and J. E. Beasley. The period routing problem. *Networks*, 14(2):237–256, 1984.

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

J.-F. Cordeau and G. Laporte. Tabu Search Heuristics for the Vehicle Routing Problem. In R. Sharda, S. Voß, C. Rego, and B. Alidaee, editors, *Metaheuristic Optimization via Memory and Evolution*, volume 30 of *Operations Research/Computer Science Interfaces Series*, pages 145–163. Springer US, 2005.

J.-F. Cordeau and G. Laporte. The dial-a-ride problem: models and algorithms. *Annals of Operations Research*, 153(1):29–46, 2007.

J.-F. Cordeau and M. Maischberger. A parallel iterated tabu search heuristic for vehicle routing problems. *Computers & Operations Research*, 39(9):2033–2050, 2012.

J.-F. Cordeau, M. Gendreau, and G. Laporte. A tabu search heuristic for periodic and multi-depot vehicle routing problems. *Networks*, 30(2):105–119, 1997.

J.-F. Cordeau, G. Laporte, and A. Mercier. A unified Tabu Search heuristic for vehicle routing problems with time windows. *Journal of the Operational Research Society*, 52: 928–936, 2001.

J.-F. Cordeau, G. Desaulniers, J. Desrosiers, M. M. Solomon, and F. Soumis. The VRP with Time Windows. In P. Toth and D. Vigo, editors, *The Vehicle Routing Problem*, SIAM Monographs on Discrete Mathematics and Applications, pages 129–154. SIAM, Philadelphia, PA, 2002a.

J.-F. Cordeau, M. Gendreau, G. Laporte, J.-Y. Potvin, and F. Semet. A Guide to Vehicle Routing Heuristics. *Journal of the Operational Research Society*, 53:512–522, 2002b.

J.-F. Cordeau, G. Laporte, and A. Mercier. An improved tabu search algorithm for the handling of route duration constraints in vehicle routing problems with time windows. *Journal of the Operational Research Society*, 55:542–546, 2004.

J.-F. Cordeau, M. Gendreau, A. Hertz, G. Laporte, and J.-S. Sormany. New Heuristics for the Vehicle Routing Problem. In A. Langevin and D. Riopel, editors, *Logistics Systems: Design and Optimization*, pages 279–297. Springer US, 2005.

J.-F. Cordeau, G. Laporte, M. W. P. Savelsbergh, and D. Vigo. Vehicle Routing. In C. Barnhart and G. Laporte, editors, *Transportation*, volume 14 of *Handbooks in Operations Research and Management Science*, pages 367–428. North-Holland, Amsterdam, 2007.

J.-F. Cordeau, G. Laporte, and S. Ropke. Recent models and algorithms for one-to-one pickup and delivery problems. In B. Golden, S. Raghavan, and E. Wasil, editors, *The Vehicle Routing Problem: Latest Advances and New Challenges*, volume 43 of *Operations Research/Computer Science Interfaces*, pages 327–357. Springer US, 2008.

T. G. Crainic. Parallel Solution Methods for Vehicle Routing Problems. In B. Golden, S. Raghavan, and E. Wasil, editors, *The Vehicle Routing Problem: Latest Advances and New Challenges*, volume 43 of *Operations Research/Computer Science Interfaces*, pages 171–198. Springer US, 2008.

T. G. Crainic and M. Gendreau. Towards an Evolutionary Method - Cooperating Multi-Thread Parallel Tabu Search Hybrid. In S. Voß, S. Martello, C. Roucairol, and I.H. Osman, editors, *Meta-Heuristics 98: Theory & Applications*, pages 331–344. Kluwer Academic Publishers, Norwell, MA, 1999.

T. G. Crainic and N. Hail. Parallel Metaheuristics application. In E. Alba, editor, *Parallel Metaheuristics: A New Class of Algorithms*, pages 447–494. John Wiley & Sons, Inc., 2005.

T. G. Crainic, M. Gendreau, and J.-Y. Potvin. Parallel Tabu Search. In E. Alba, editor, *Parallel Metaheuristics: A New Class of Algorithms*, pages 289–313. John Wiley & Sons, Inc., 2005.

T. G. Crainic, N. Ricciardi, and G. Storchi. Models for Evaluating and Planning City Logistics Systems. *Transportation Science*, 43(4):432–454, 2009.

T. G. Crainic, G. Perboli, S. Mancini, and R. Tadei. Two-Echelon Vehicle Routing Problem: A satellite location analysis. *Procedia - Social and Behavioral Sciences*, 2(3):5944–5955, 2010.

T. G. Crainic, S. Mancini, G. Perboli, and R. Tadei. Multi-start Heuristics for the Two-Echelon Vehicle Routing Problem. In P. Merz and J.-K. Hao, editors, *Evolutionary Computation in Combinatorial Optimization*, volume 6622 of *Lecture Notes in Computer Science*, pages 179–190. Springer Berlin Heidelberg, 2011.

T. G. Crainic, F. Errico, W. Rei, and N. Ricciardi. Integrating c2e and c2c Traffic into City Logistics Planning. *Procedia - Social and Behavioral Sciences*, 39(0):47–60, 2012a.

T. G. Crainic, Y. Gajpal, and M. Gendreau. Multi-Zone Multi-Trip Vehicle Routing Problem with Time Windows. Technical report, Publication CIRRELT-2012-36, Centre interuniversitaire de recherche sur les réseaux d'entreprise, la logistique et le transport, Université de Montréal, Montréal, QC, Canada, 2012b.

T. G. Crainic, S. Mancini, G. Perboli, and R. Tadei. GRASP with Path Relinking for the Two-Echelon Vehicle Routing Problem. In L. Di Gaspero, A. Schaerf, and T. Stützle, editors, *Advances in Metaheuristics*, volume 53 of *Operations Research/Computer Science Interfaces Series*, pages 113–125. Springer New York, 2013.

T. G. Crainic and M. Toulouse. Parallel Meta-Heuristics. In M. Gendreau and J.-Y. Potvin, editors, *Handbook of Metaheuristics*, pages 497–541. Springer, 2010.

B. Crevier, J.-F. Cordeau, and G. Laporte. The Multi-depot Vehicle Routing Problem with inter-depot routes. *European Journal of Operational Research*, 176(2):756–773, 2007.

J. Crispim and J. Brandão. Metaheuristics Applied to Mixed and Simultaneous Extensions of Vehicle Routing Problems with Backhauls. *Journal of the Operational Research Society*, 56(11):1296–1302, 2005.

A. Şen and K. Bülbül. A Survey On Multi-Trip Vehicle Routing Problem. In *Proceedings of the VI. International Logistics & Supply Chain Congress*, 2008.

Z. J. Czech and P. Czarnas. Parallel simulated annealing for the Vehicle Routing Problem with time windows. In *10th Euromicro Workshop on Parallel, Distributed and Network-based Processing, Spain*, pages 376–383, 2002.

G. B. Dantzig and J. H. Ramser. The truck dispatching problem. *Management Science*, 6 (1):80–91, 1959.

I. Deif and L. Bodin. Extension of the Clarke and Wright algorithm for solving the vehicle routing problem with backhauling. In A. Kidder, editor, *Proceedings of the Babson Conference on Software Uses in Transportation and Logistic Management*, pages 75–96, 1984.

M. Dell'Amico and M. Trubian. Applying tabu search to the job-shop scheduling problem. *Annals of Operations Research*, 41:231–252, 1993.

M. Dell'Amico, G. Righini, and M. Salani. A Branch-and-Price Approach to the Vehicle Routing Problem with Simultaneous Distribution and Collection. *Transportation Science*, 40(2):235–247, 2006.

G. Desaulniers, J. Desrosiers, and S. Spoorendonk. The Vehicle Routing Problem with Time Windows: State-of-the-Art Exact Solution Methods. In *Wiley Encyclopedia of Operations Research and Management Science*. John Wiley & Sons, Inc., 2010.

M. Desrochers, J. Desrosiers, and M. M. Solomon. A new optimization algorithm for the Vehicle Routing Problem with time windows. *Operations Research*, 40:342–354, 1992.

J. Desrosiers, J. Ferland, J.-M. Rousseau, G. Lapalme, and L. Chapleau. An Overview of a School Busing System. In N. Jaiswal, editor, *Scientific Management of Transport Systems*, pages 235–243. North-Holland, 1981.

J. Dethloff. Relation between Vehicle Routing Problems: An Insertion Heuristic for the Vehicle Routing Problem with Simultaneous Delivery and Pick-Up Applied to the Vehicle Routing Problem with Backhauls. *Journal of the Operational Research Society*, 53 (1):115–118, 2002.

A. Dohn, M. S. Rasmussen, and J. Larsen. The vehicle routing problem with time windows and temporal dependencies. *Networks*, 58(4):273–289, 2011.

R. Dondo and J. Cerda. A cluster-based optimization approach for the multi-depot heterogeneous fleet Vehicle Routing Problem with time windows. *European Journal of Operational Research*, 176(3):1478–1507, 2007.

J. J. Dongarra. Performance of various computers using standard linear equations software. Technical report, University of Tennessee, 2013.

M. Drexl. Synchronization in Vehicle Routing - A Survey of VRPs with Multiple Synchronization Constraints. *Transportation Science*, 46(3):297–316, 2012a.

M. Drexl. Rich vehicle routing in theory and practice. *Logistics Research*, 5(1-2):47–63, 2012b.

M. Dror, G. Laporte, and P. Trudeau. Vehicle routing with split deliveries. *Discrete Applied Mathematics*, 50(3):239–254, 1994.

L. M. A. Drummond, L. S. Ochi, and D. S. Vianna. An asynchronous parallel metaheuristic for the Period Vehicle Routing Problem. *Future Generation Computer Systems*, 17(4): 379–386, 2001.

J. F. Ehmke, A. Steinert, and D. C. Mattfeld. Advanced routing for city logistics service providers based on time-dependent travel times. *Journal of Computational Science*, 3 (4):193–205, 2012.

T. A. El-Mihoub, A. A. Hopgood, L. Nolle, and A. Battersby. Hybrid Genetic Algorithms: A Review. *Engineering Letters*, 13(2):124–137, 2006.

A. E. Fallahi, C. Prins, and R. W. Calvo. A memetic algorithm and a tabu search for the Multi-compartment Vehicle Routing Problem. *Computers & Operations Research*, 35 (5):1725–1741, 2008. Part Special Issue: Algorithms and Computational Methods in Feasibility and Infeasibility.

M. A. Figliozzi. A route improvement algorithm for the vehicle routing problem with time dependent travel times. In *Proceeding of the 88th Transportation Research Board Annual Meeting CD ROM, Washington, DC*, 2009.

M. L. Fisher and R. Jaikumar. A generalized assignment heuristic for vehicle routing. *Networks*, 11:109–124, 1981.

P. Francis, K. Smilowitz, and M. Tzur. The Period Vehicle Routing Problem and its Extensions. In B. Golden, S. Raghavan, and E. Wasil, editors, *The Vehicle Routing Problem: Latest Advances and New Challenges*, volume 43 of *Operations Research/Computer Science Interfaces*, pages 73–102. Springer US, 2008.

P. W. Frizzell and J. W. Giffin. The split delivery vehicle scheduling problem with time windows and grid network distances. *Computers & Operations Research*, 22(6):655–667, 1995.

J. Fuglestvedt, T. Berntsen, G. Myhre, K. Rypdal, and R. B. Skeie. Climate forcing from the transport sectors. *Proceedings of the National Academy of Sciences*, 105(2):454–458, 2008.

Y. Gajpal and P. Abad. An ant colony system (ACS) for Vehicle Routing Problem with simultaneous delivery and pickup. *Computers & Operation Research*, 36(12):3215–3223, 2009a.

Y. Gajpal and P. Abad. Multi-ant colony system (MACS) for a vehicle routing problem with backhauls. *European Journal of Operational Research*, 196(1):102–117, 2009b.

L. M. Gambardella, E. D. Taillard, and G. Agazzi. MACS-VRPTW: A Multiple Colony System For Vehicle Routing Problems With Time Windows. In *New Ideas in Optimization*, pages 63–76. McGraw-Hill, 1999.

B.-L. Garcia, J.-Y. Potvin, and J.-M. Rousseau. A parallel implementation of the Tabu search heuristic for vehicle routing problems with time window constraints. *Computers & Operations Research*, 21(9):1025–1033, 1994.

C. García-Martínez and M. Lozano. Local search based on genetic algorithms. In P. Siarry and Z. Michalewicz, editors, *Advances in Metaheuristics for Hard Optimization*, Natural Computing Series, pages 199–221. Springer, Berlin Heidelberg, 2008.

T. Gaskell. Bases for vehicle fleet scheduling. *Operations Research Quarterly*, 18:281–295, 1967.

L. D. Gaspero and A. Schaerf. A composite-neighborhood tabu search approach to the traveling tournament problem. *Journal of Heuristics*, 13(2):189–207, 2007.

H. Gehring and J. Homberger. Parallelization of a Two-Phase Metaheuristic for Routing Problems with Time Windows. *Journal of Heuristics*, 8(3):251–276, 2002.

S. Gélinas, M. Desrochers, J. Desrosiers, and M. Solomon. A new branching strategy for time constrained routing problems with application to backhauling. *Annals of Operations Research*, 61(1):91–109, 1995.

M. Gendreau and J.-Y. Potvin. Metaheuristics in combinatorial optimization. *Annals of Operations Research*, 140(1):189–213, 2005.

M. Gendreau, A. Hertz, and G. Laporte. A new insertion and postoptimization procedures for the traveling salesman problem. *Operations Research*, 40:1086–1093, 1992.

M. Gendreau, G. Laporte, M. Christophe, and E. D. Taillard. A tabu search heuristic for the Heterogenous fleet Vehicle Routing Problem. *Computers & Operations Research*, 26(12):1153–1173, 1999.

M. Gendreau, G. Laporte, and J.-Y. Potvin. Metaheuristics for the Vehicle Routing Problem. In P. Toth and D. Vigo, editors, *The Vehicle Routing Problem*, SIAM Monographs on Discrete Mathematics and Applications, pages 129–154. SIAM, Philadelphia, PA, 2002.

M. Gendreau, J.-Y. Potvin, B. Olli, G. Hasle, and L. Arne. Metaheuristics for the Vehicle Routing Problem and Its Extensions: A Categorized Bibliography. In B. Golden, S. Raghavan, and E. Wasil, editors, *The Vehicle Routing Problem: Latest Advances and New Challenges*, volume 43 of *Operations Research/Computer Science Interfaces*, pages 143–169. Springer US, 2008.

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

M. Goetschalckx and C. Jacobs-Blecha. The Vehicle Routing Problem with backhauls. *European Journal of Operational Research*, 42(1):39–51, 1989.

F. P. Goksal, I. Karaoglan, and F. Altiparmak. A hybrid discrete particle swarm optimization for vehicle routing problem with simultaneous pickup and delivery. *Computers & Industrial Engineering*, 65(1):39–53, 2013.

B. Golden, E. Wasil, J. Kelly, and I.-M. Chao. The impact of metaheuristics on solving the vehicle routing problem: Algorithms, problem sets, and computational results. In T. G. Crainic and G. Laporte, editors, *Fleet Management and Logistics*, Centre for Research on Transportation, pages 33–56. Springer US, 1998.

B. L. Golden, E. Baker, J. Alfaro, and J. Schaffer. The vehicle routing problem with backhauling: Two approaches. In R. Hammesfahr, editor, *Proceedings of the Twenty-first Annual Meeting of S.E. TIMS*, pages 90–92, 1984.

B. L. Golden, A. A. Assad, and E. A. Wasil. Routing vehicles in the real world: Applications in the solid waste, beverage, food, dairy, and newspaper industries. In P. Toth and D. Vigo, editors, *The Vehicle Routing Problem*, pages 245–286. SIAM, Philadelphia, PA, 2002.

B. L. Golden, S. Raghavan, and E. A. Wasil. *The Vehicle Routing Problem: Latest Advances and New Challenges*. Springer, 2008.

I. Gribkovskaia, O. Halskau, and K. Myklebost. Models for Pick-up and Deliveries from Depots with Lasso Solutions. In *Proceedings of the 13th Annual Conference on Logistics Research*, pages 279–293. NOFOMA 2001, Collaboration in logistics: Connecting Islands using Information Technology, 2001.

E. Hadjiconstantinou and R. Baldacci. A Multi-depot Period Vehicle Routing Problem arising in the utilities sector. *Journal of the Operational Research Society*, 49(12):1239–1248, 1998.

K. Halse. *Modeling and solving complex vehicle routing problems*. PhD thesis, Institute of Mathematical Statistics and Operations Research, Technical University of Denmark, 1992.

J.-P. Hamiez, J. Robet, and J.-K. Hao. A Tabu Search Algorithm with Direct Representation for Strip Packing. In *Proceedings of the 9th European Conference on Evolutionary Computation in Combinatorial Optimization*, EvoCOP '09, pages 61–72. Springer-Verlag, 2009.

R. Hartl, G. Hasle, and G. E. Jansens. Special Issue on Rich Vehicle Routing Problems. *Central European Journal of Operations Research*, 13(2):103–104, 2006.

H. Hashimoto and M. Yagiura. A Path Relinking Approach with an Adaptive Mechanism to Control Parameters for the Vehicle Routing Problem with Time Windows. In J. Hemert and C. Cotta, editors, *Evolutionary Computation in Combinatorial Optimization*, volume 4972 of *Lecture Notes in Computer Science*, pages 254–265. Springer Berlin Heidelberg, 2008.

V. C. Hemmelmayr, J.-F. Cordeau, and T. G. Crainic. An adaptive large neighborhood search heuristic for Two-Echelon Vehicle Routing Problems arising in city logistics. *Computers & Operations Research*, 39(12):3215–3228, 2012.

H. Hernández-Pérez and J.-J. Salazar-González. The One-Commodity Pickup-and-Delivery Travelling Salesman Problem. In M. Jünger, G. Reinelt, and G. Rinaldi, editors, *Combinatorial Optimization — Eureka, You Shrink!*, volume 2570 of *Lecture Notes in Computer Science*, pages 89–104. Springer Berlin Heidelberg, 2003.

H. Hernández-Pérez, I. Rodríguez-Martín, and J.-J. Salazar-González. A hybrid GRASP/VND heuristic for the one-commodity pickup-and-delivery traveling salesman problem. *Computers & Operations Research*, 36(5):1639–1645, 2009.

S. C. Ho and D. Haugland. A tabu search heuristic for the Vehicle Routing Problem with time windows and split deliveries. *Computers & Operations Research*, 31:1947–1964, 2002.

W. Ho, G. T. Ho, P. Ji, and H. C. Lau. A hybrid genetic algorithm for the Multi-depot Vehicle Routing Problem. *Engineering Applications of Artificial Intelligence*, 21(4): 548–557, 2008.

A. Hoff, I. Gribkovskaia, G. Laporte, and A. Løkketangen. Lasso solution strategies for the vehicle routing problem with pickups and deliveries. *European Journal of Operational Research*, 192(3):755–766, 2009.

J. Homberger and H. Gehring. A two-phase hybrid metaheuristic for the vehicle routing problem with time windows. *European Journal of Operational Research*, 162(1):220–238, 2005.

G. Ioannou, M. Kritikos, and G. Prastacos. A greedy look-ahead heuristic for the Vehicle Routing Problem with time windows. *Journal of the Operational Research Society*, 52: 523–537, 2001.

G. Ioannou, M. Kritikos, and G. Prastacos. A problem generator-solver heuristic for vehicle routing with soft time windows. *Omega*, 31(1):41–53, 2003.

H. Ishibuchi and K. Narukawa. Some issues on the implementation of local search in evolutionary multiobjective optimization. In *Proceedings of Genetic and Evolutionary Computation Conference, LNCS*, pages 1246–1258, 2004.

C. Jacobs-Blecha and M. Goetschalckx. The Vehicle Routing Problem with backhauls: Properties and solution algorithms. Technical report, School of Industrial and Systems Engineering, Georgia Institute of Technology, Atlanta, Georgia, 1993.

B.-I. Kim, S. Kim, and S. Sahoo. Waste collection vehicle routing problem with time windows. *Computers & Operations Research*, 33(12):3624–3642, 2006.

J. Knowles and D. Corne. Memetic Algorithms for Multiobjective Optimization: Issues, Methods and Prospects. In N. Krasnogor, J. E. Smith, and W. E. Hart, editors, *Recent advances in Memetic algorithms*, pages 325–332. Springer, 2000.

G. Laporte. What you should know about the Vehicle Routing Problem. *Naval Research. Logistics*, 54:811–819, 2007.

G. Laporte and F. Semet. Classical Heuristics for the Vehicle Routing Problem. In Toth, P. and Vigo, D., editors, *The Vehicle Routing Problem*, SIAM Monographs on Discrete Mathematics and Applications, pages 109–128. SIAM, Philadelphia, PA, 2002.

G. Laporte, M. Gendreau, J.-Y. Potvin, and F. Semet. Classical and Modern Heuristics for the Vehicle Routing Problem. *International Transactions in Operational Research*, 7(4/5):285–300, 2000.

H. C. Lau, S. Melvyn, and T. K. Meng. Vehicle Routing Problem with time windows and a limited number of vehicles. *European Journal of Operational Research*, 148(3): 559–569, 2003.

A. Le Bouthillier and T. G. Crainic. A cooperative parallel meta-heuristic for the Vehicle Routing Problem with time windows. *Computers & Operations Research*, 32(7):1685–1708, 2005.

A. Le Bouthillier, T. G. Crainic, and P. Kropf. A Guided Cooperative Search for the Vehicle Routing Problem with Time Windows. *IEEE Intelligent Systems*, 20(4):36–42, 2005.

Y. H. Lee, J. W. Jung, and K. M. Lee. Vehicle routing scheduling for cross-docking in the supply chain. *Computers & Industrial Engineering*, 51(2):247–256, 2006.

J. K. Lenstra and A. H. G. Rinnooy Kan. Complexity of vehicle routing and scheduling problems. *Networks*, 11(2):221–227, 1981.

H. Li and A. Lim. Local search with annealing-like restarts to solve the VRPTW. *European Journal of Operational Research*, 150(1):115–127, 2003. O.R. Applied to Health Services.

C.-J. Liao, Y. Lin, and S. C. Shih. Vehicle routing with cross-docking in the supply chain. *Expert Systems with Applications*, 37(10):6868–6873, 2010.

C. Lima, M. Goldbarg, and E. Goldbarg. A memetic algorithm for the Heterogeneous fleet Vehicle Routing Problem. *Electronic Notes in Discrete Mathematics*, 18:171–176, 2004. Latin-American Conference on Combinatorics, Graphs and Applications.

S. Lin. Computer solutions of the traveling salesman problem. *Bell System Technical Journal*, 44:2245–2269, 1965.

Z. Lü, F. Glover, and J.-K. Hao. A hybrid metaheuristic approach to solving the UBQP problem. *European Journal of Operational Research*, 207(3):1254–1262, 2010.

F. Meisel and H. Kopfer. Synchronized routing of active and passive means of transport. *OR Spectrum*, pages 1–26, 2012.

J. E. Mendoza, B. Castanier, C. Guéret, A. L. Medaglia, and N. Velasco. A memetic algorithm for the Multi-compartment Vehicle Routing Problem with stochastic demands. *Computers & Operations Research*, 37(11):1866–1898, 2010.

P. Merz and K. Katayama. Memetic Algorithms for the Unconstrained Binary Quadratic Programming Problem. *BioSystems*, 2004:99–118, 2004.

H. Min. The multiple vehicle routing problem with simultaneous delivery and pick-up points. *Transportation Research Part A: General*, 23(5):377–386, 1989.

S. Mitrović-Minić. Pickup and delivery problem with time windows: A survey. Technical report, Technical report TR 1998-12, School of Computing Science, Simon Fraser University, Burnaby, BC, Canada, 1998.

F. A. T. Montané and R. D. Galvao. A tabu search algorithm for the Vehicle Routing Problem with simultaneous pick-up and delivery service. *Computer & Operations Research*, 33(3):595–619, 2006.

G. Nagy and S. Salhi. Heuristic algorithms for single and multiple depot vehicle routing problems with pickups and deliveries. *European Journal of Operational Research*, 162 (1):126–141, 2005.

G. Nagy, N. A. Wassan, M. G. Speranza, and C. Archetti. The Vehicle Routing Problem with Divisible Deliveries and Pickups. *Transportation Science*, 0(0):1–24, 2013.

L. Nai-Wen and L. Chang-Shi. A Hyrid Tabu Search for the Vehicle Routing Problem with Soft Time Windows. In G. Yang, editor, *Proceedings of the 2012 International*

Conference on Communication, Electronics and Automation Engineering, volume 181 of *Advances in Intelligent Systems and Computing*, pages 507–512. Springer Berlin Heidelberg, 2013.

P. K. Nguyen, T. G. Crainic, and M. Toulouse. A tabu search for Time-dependent Multi-zone Multi-trip Vehicle Routing Problem with Time Windows. *European Journal of Operational Research*, 231(1):43–56, 2013.

A. Olivera and O. Viera. Adaptive memory programming for the Vehicle Routing Problem with multiple trips. *Computer & Operations Research*, 34(1):28–47, 2007.

B. M. Ombuki-Berman, A. Runka, and F. T. Hanshar. Waste collection vehicle routing problem with time windows using multi-objective genetic algorithms. In *Proceedings of the Third IASTED International Conference on Computational Intelligence*, pages 91–97. ACTA Press, 2007.

I. Or. *Traveling Salesman-type Combinatorial Problems and their relation to the Logistics of Blood Banking*. PhD thesis, Department of Industrial Engineering and Management Science, Northwestern University, Evanston, IL, 1976.

I. H. Osman. Metastrategy simulated annealing and tabu search algorithms for the Vehicle Routing Problem. *Annals of Operations Research*, 41:421–452, 1993.

I. H. Osman and N. Wassan. A reactive tabu search meta-heuristic for the vehicle routing problem with back-hauls. *Journal of Scheduling*, 5(4):263–285, 2002.

J. Park and B.-I. Kim. The school bus routing problem: A review. *European Journal of Operational Research*, 202(2):311–319, 2010.

S. Parragh, K. Doerner, and R. Hartl. A survey on pickup and delivery problems. Part I: Transportation between customers and depot. *Journal für Betriebswirtschaft*, 58:21–51, 2008a.

S. Parragh, K. Doerner, and R. Hartl. A survey on pickup and delivery problems. Part II: Transportation between pickup and delivery locations. *Journal für Betriebswirtschaft*, 58:81–117, 2008b.

R. J. Petch and S. Salhi. A multi-phase constructive heuristic for the Vehicle Routing Problem with multiple trips. *Discrete Applied Mathematics*, 133(1-3):69–92, 2003.

S. Pirkwieser and G. R. Raidl. A variable neighborhood search for the periodic Vehicle Routing Problem with time windows. In *Proceedings of the 9th EU/MEeting on Metaheuristics for Logistics and Vehicle Routing*, Troyes, France, 2008.

S. Pirkwieser and G. R. Raidl. Boosting a variable neighborhood search for the periodic vehicle routing problem with time windows by ILP techniques. In *Proceedings of the 8th Metaheuristic International Conference (MIC 2009)*, Hamburg, Germany, 2009a.

S. Pirkwieser and G. R. Raidl. Multiple variable neighborhood search enriched with ILP techniques for the periodic vehicle routing problem with time windows. In *Proceedings of Hybrid Metaheuristics - Sixth International Workshop*, volume 5818 of LNCS, pages 45–59, Udine, Italy, 2009b. Springer.

S. Pirkwieser and G. R. Raidl. Matheuristics for the periodic vehicle routing problem with time windows. In *Proceedings of the 3rd International Workshop on model-based metaheuristics (Metaheuristics 2010)*, Vienna, Austria, 2010.

D. Pisinger and S. Ropke. A general heuristic for vehicle routing problems. *Computers & Operations Research*, 34(8):2403–2435, 2007.

M. Polacek, R. F. Hartl, K. Doerner, and M. Reimann. A variable neighborhood search for the multi depot Vehicle Routing Problem with time windows. *Journal of Heuristics*, 10 (6):613–627, 2004.

J.-Y. Potvin. State-of-the Art Review - Evolutionary Algorithms for Vehicle Routing. *INFORMS Journal on computing*, 21(3):568–581, 2009.

J.-Y. Potvin and S. Bengio. The Vehicle Routing Problem with Time Windows - Part II: Genetic Search. *INFORMS Journal on Computing*, 8:165–172, 1996.

J.-Y. Potvin and J.-M. Rousseau. A parallel route building algorithm for the vehicle routing and scheduling problem with time windows. *European Journal of Operational Research*, 66(3):331–340, 1993.

J.-Y. Potvin and J.-M. Rousseau. An exchange heuristic for routing problems with time windows. *Journal of the Operational Reasearch Society*, 46:1433–1446, 1995.

J.-Y. Potvin, C. Duhamel, and F. Guertin. A genetic algorithm for vehicle routing with backhauling. *Applied Intelligence*, 6(4):345–355, 1996a.

J.-Y. Potvin, T. Kervahut, B.-L. Garcia, and J.-M. Rousseau. The vehicle routing problem with time windows, Part I: Tabu search. *INFORMS Journal on Computing*, 8:157–164, 1996b.

C. Rego and C. Roucairol. A Parallel Tabu Search Algorithm Using Ejection Chains for the Vehicle Routing Problem. In I. Osman and J. Kelly, editors, *Meta-Heuristics*, pages 661–675. Springer US, 1996.

M. Reimann and H. Ulrich. Comparing backhauling strategies in vehicle routing using Ant Colony Optimization. *Central European Journal of Operations Research*, 14(2): 105–123, 2006.

M. Reimann, K. Doerner, and R. Hartl. Insertion Based Ants for Vehicle Routing Problems with Backhauls and Time Windows. In M. Dorigo, G. Caro, and M. Sampels, editors, *Ant Algorithms*, volume 2463 of *Lecture Notes in Computer Science*, pages 135–148. Springer Berlin Heidelberg, 2002.

Y. Rochat and E. D. Taillard. Probabilistic diversification and intensification in local search for vehicle routing. *Journal of Heuristics*, 1:147–167, 1995.

S. Ropke and D. Pisinger. A unified heuristic for a large class of vehicle routing problems with backhauls. *European Journal of Operational Research*, 2004:750–775, 2006.

L.-M. Rousseau, M. Gendreau, and G. Pesant. Using constraint-based operators to solve the Vehicle Routing Problem with time windows. *Journal of Heuristics*, 8(1):43–58, 2002.

R. A. Russell and D. Gribbin. A multiphase approach to the period routing problem. *Networks*, 21(7):747–765, 1991.

S. Salhi and G. Nagy. A Cluster Insertion Heuristic for Single and Multiple Depot Vehicle Routing Problems with Backhauling. *Journal of the Operational Research Society*, 50 (10):1034–1042, 1999.

S. Salhi and R. Petch. A GA Based Heuristic for the Vehicle Routing Problem with Multiple Trips. *Journal of Mathematical Modelling and Algorithms*, 6:591–613, 2007.

S. Salhi, N. Wassan, and M. Hajarat. The Fleet Size and Mix Vehicle Routing Problem with Backhauls: Formulation and Set Partitioning-based Heuristics. *Transportation Research Part E: Logistics and Transportation Review*, 56(0):22–35, 2013.

M. W. P. Savelsbergh. The Vehicle Routing Problem with time windows: Minimizing route duration. *Journal on Computing*, 4:146–154, 1992.

M. W. P. Savelsbergh and M. M. Solomon. The general pickup and delivery problem. *Transportation Science*, 29:17–29, 1995.

J. Schulze and T. Fahle. A parallel algorithm for the Vehicle Routing Problem with time window constraints. *Annals of Operations Research*, 86:585–607, 1999.

M. M. Solomon. Algorithms for the vehicle routing and scheduling problems with time window constraints. *Operations Research*, 35:254–265, 1987.

M. M. Solomon and J. Desrosiers. Time Window Constrained Routing and Scheduling Problems. *Transportation Science*, 2(1):1–13, 1988.

A. Subramanian, L. Drummond, C. Bentes, L. Ochi, and R. Farias. A parallel heuristic for the Vehicle Routing Problem with Simultaneous Pickup and Delivery. *Computers & Operations Research*, 37(11):1899–1911, 2010.

E. D. Taillard, G. Laporte, and M. Gendreau. Vehicle Routing With Multiple Use Of Vehicles. *Journal of the Operational Research Society*, 47(8):1065–1070, 1995.

E. D. Taillard, P. Badeau, M. Gendreau, F. Guertin, and J.-Y. Potvin. A tabu search heuristic for the Vehicle Routing Problem with soft time windows. *Transportation Science*, 31: 170–186, 1997.

C. C. R. Tan and J. E. Beasley. A heuristic algorithm for the Period Vehicle Routing Problem. *Omega*, 12(5):497–504, 1984.

K. C. Tan, L. H. Lee, Q. L. Zhu, and K. Ou. Heuristic methods for vehicle routing problem with time windows. In *Proceeding 6th International Sympos. Artificial Intelligence Math.*, Ft. Lauderdale, FL, 2000.

K. C. Tan, L. H. Lee, and K. Ou. Artificial intelligence heuristics in solving vehicle routing problems with time window constraints. *Engineering Applications of Artificial Intelligence*, 14(6):825–837, 2001.

C. D. Tarantilis. Adaptive multi-restart Tabu Search algorithm for the vehicle routing problem with cross-docking. *Optimization Letters*, pages 1–14, 2012.

C. D. Tarantilis, E. E. Zachariadis, and C. T. Kiranoudis. A Hybrid Guided Local Search for the Vehicle-Routing Problem with Intermediate Replenishment Facilities. *INFORMS Journal on computing*, 20(1):154–168, 2008.

A. S. Tasan and M. Gen. A genetic algorithm based approach to vehicle routing problem with simultaneous pick-up and deliveries. *Computers & Industrial Engineering*, 62(3): 755–761, 2012.

S. R. Thangiah, K. E. Nygard, and P. L. Juell. GIDEON: a genetic algorithm system for vehicle routing with time windows. In *Proceedings of the Seventh IEEE Conference on Artificial Intelligence Applications, Miami, Florida.*, volume i, pages 322–328, 1991.

S. R. Thangiah, R. Vinayagamoorthy, and A. Gubbi. Vehicle routing with time deadlines using genetic and local algorithms. In *Proceedings of the Fifth International Conference on Genetic Algorithms, 506-513, Morgan Kaufman, New York*, 1993.

S. R. Thangiah, J.-Y. Potvin, and T. Sun. Heuristic approaches to vehicle routing with backhauls and time windows. *Computers & Operations Research*, 23(11):1043–1057, 1996.

P. Toth and A. Tramontani. An integer linear programming local search for Capacitated Vehicle Routing Problems. In B. Golden, S. Raghavan, and E. Wasil, editors, *The Vehicle Routing Problem: Latest Advances and New Challenges*, volume 43 of *Operations Research/Computer Science Interfaces*, pages 143–169. Springer US, 2008.

P. Toth and D. Vigo. A Heuristic Algorithm for the Vehicle Routing Problem with Backhauls. In L. Bianco and P. Toth, editors, *Advanced Methods in Transportation Analysis*, Transportation Analysis, pages 585–608. Springer Berlin Heidelberg, 1996.

P. Toth and D. Vigo. An Exact Algorithm for the Vehicle Routing Problem with Backhauls. *Transportation Science*, 31(4):372–385, 1997.

P. Toth and D. Vigo. *The Vehicle Routing Problem.* Society for Industrial and Applied Mathematics, Philadelphia, PA, 2002.

S. Tripathi and B. Minocha. Vehicle Routing Problem with Time Windows: An Evolutionary Algorithmic Approach. *Algorithmic Operations Research*, 1(2):104–118, 2006.

T. Vidal, T. G. Crainic, M. Gendreau, N. Lahrichi, and W. Rei. A hybrid genetic algorithm for multi-depot and periodic vehicle routing problems. *Operations Research*, 60(3): 611–624, 2012.

T. Vidal, T. G. Crainic, M. Gendreau, and C. 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):475–489, 2013a.

T. Vidal, T. G. Crainic, M. Gendreau, and C. Prins. Heuristics for multi-attribute vehicle routing problems: A survey and synthesis. *European Journal of Operational Research*, 231(1):1–21, 2013b.

N. Wassan. Reactive Tabu Adaptive Memory Programming Search for the Vehicle Routing Problem with Backhauls. *Journal of the Operational Research Society*, 58(12):1630–1641, 2007.

N. Wassan, A. Wassan, and G. Nagy. A reactive tabu search algorithm for the vehicle routing problem with simultaneous pickups and deliveries. *Journal of Combinatorial Optimization*, 15(4):368–386, 2008.

H. Wee-Kit, J. C. Ang, and A. Lim. A hybrid search algorithm for the Vehicle Routing Problem with time windows. *International Journal on Artificial Intelligence Tools*, 10 (3):431–449, 2001.

M. Wen, J. Larsen, J. Clausen, J.-F. Cordeau, and G. Laporte. Vehicle Routing with Cross-Docking. *Journal of the Operational Research Society*, 60:1708–1718, 2008.

J. Xu and J. P. Kelly. A Network Flow-Based Tabu Search Heuristic For The Vehicle Routing Problem. *Transportation Science*, 30:379–393, 1996.

J. Xu, M. Sohoni, M. McCleery, and T. G. Bailey. A dynamic neighborhood based tabu search algorithm for real-world flight instructor scheduling problems. *European Journal of Operational Research*, 169(3):978–993, 2006.

Y. Xu, L. Wang, and Y. Yang. A New Variable Neighborhood Search Algorithm for the Multi Depot Heterogeneous Vehicle Routing Problem with Time Windows. *Electronic Notes in Discrete Mathematics*, 39(0):289–296, 2012.

P. C. Yellow. A computational modification to the savings method of vehicle scheduling. *Operations Research Quarterly*, 21:281–283, 1970.

E. E. Zachariadis and C. T. Kiranoudis. A local search metaheuristic algorithm for the vehicle routing problem with simultaneous pick-ups and deliveries. *Expert Systems with Applications*, 38(3):2717–2726, 2011.

E. E. Zachariadis and C. T. Kiranoudis. An effective local search approach for the Vehicle Routing Problem with Backhauls. *Expert Systems with Applications*, 39(3):3174–3184, 2012.

E. E. Zachariadis, C. D. Tarantilis, and C. T. Kiranoudis. A hybrid metaheuristic algorithm for the vehicle routing problem with simultaneous delivery and pick-up service. *Expert Systems with Applications*, 36(2, Part 1):1070–1081, 2009.

F. Zhao, S. Li, J. Sun, and D. Mei. Genetic algorithm for the one-commodity pickup-and-delivery traveling salesman problem. *Computers & Industrial Engineering*, 56(4): 1642–1648, 2009.

Y. Zhong and M. H. Cole. A vehicle routing problem with backhauls and time windows: a guided local search solution. *Transportation Research Part E: Logistics and Transportation Review*, 41(2):131–144, 2005.

Annex A

SUPPLEMENTARY MATERIAL FOR THE PVRPTW

A.1 *The UTS and RVNS Meta-heuristics*

This Annex briefly presents the two meta-heuristics that have been implemented for the offspring education procedure.

A.1.1 Unified Tabu Search

UTS (Cordeau et al., 2004) uses its own penalty coefficients, α_1, α_2, and α_3, for violations of vehicle capacity, route duration and customer service time window constraints, respectively. The cost function of a solution s then becomes $f(s) = c(s) + \alpha_1 q(s) + \alpha_2 d(s) + \alpha_3 w(s)$.

An attribute set $B(s) = \{(i,k,l) : \text{customer } i \text{ is visited by vehicle } k \text{ on day } l\}$ is associated to each solution s. Let b_{rl} be a binary constant equal to 1 if and only if day l belongs to pattern r, for each pattern $r \in R$ and every day $l \in \mathscr{T}$ of solution s. The neighborhood $N(s)$ of a solution s is then defined by two transformations to (1) relocate a customer within a day (routing modification), and (2) replace the pattern of a customer (pattern modification):

1. Remove customer i from route k on day l and insert it into another route k'.

2. Replace pattern r currently assigned to customer i with another pattern $r' \in R_i$; Then, for $l = 1, \dots, t$ do

   - If $b_{rl} = 1$ and $b_{r'l} = 0$, remove customer i from its route of day l;

   - If $b_{rl} = 0$ and $b_{r'l} = 1$, insert customer i into the route of day l minimizing the increase in fitness $f(s)$.

UTS starts from a given offspring s and chooses, at each iteration, the best non-tabu solution in $N(s)$. After each iteration, the values of the parameters α_1, α_2, and α_3 are

Algorithm 5 UTS(s)

```
1: $\alpha_1 = 1$, $\alpha_2 = 1$, $\alpha_3 = 1$
2: $s_2 \leftarrow s$, $f(s_2) = f(s)$
3: **if** $s$ is feasible **then**
4:     $s_1 \leftarrow s$
5:     $c(s_1) = c(s)$
6: **else**
7:     $c(s_1) = \infty$
8: **end if**
9: $flag = 0$
10: **for** $it = 1, \ldots, UTS_{it}$ **do**
11:     Choose a solution $\bar{s} \in N(s)$ that minimizes $f(\bar{s}) + p(\bar{s})$ and is not tabu or satisfies the
        aspiration criteria.
12:     **if** solution $\bar{s}$ is feasible and $c(\bar{s}) < c(s_1)$ **then**
13:         $s_1 \leftarrow \bar{s}$
14:         $c(s_1) = c(\bar{s})$
15:         $flag = 1$
16:     **else if** $f(\bar{s}) < f(s_2)$ **then**
17:         $s_2 \leftarrow \bar{s}$
18:         $f(s_2) = f(\bar{s})$
19:     **end if**
20:     Compute $q(\bar{s})$, $d(\bar{s})$ and $w(\bar{s})$ and update $\alpha_1$, and $\alpha_2$, $\alpha_3$ accordingly
21:     $s \leftarrow \bar{s}$
22: **end for**
23: **if** $flag$ **then**
24:     return $s_1$
25: **else**
26:     return $s_2$
27: **end if**
```

modified by a factor $1+\delta$; multiplied by the factor if the solution is feasible with respect to the respective constraint, divided, otherwise. We set $\delta = 0.5$, the best value reported by the authors (Cordeau et al., 2004).

To diversify the search, any solution $\bar{s} \in N(s)$ such that $f(\bar{s}) \geq f(s)$ is penalized by a factor $p(\bar{s})$ proportional to the addition frequency of its attributes, $p(\bar{s}) = \lambda c(\bar{s})\sqrt{nm}\sum_{(i,k,l)\in B(\bar{s})} \rho_{ikl}$, where ρ_{ikl} is the number of times attribute (i,k,l) has been added to the solution during the search process and $\lambda = 0.015$.

The tabu length was adjusted to a smaller number of iterations performed and set to θ = $1.5log_{10}(n)$. The post-optimization heuristic is not implemented in the education procedure. The UTS procedure returns the best feasible solution s_1 if it exists, the best infeasible solution s_2, otherwise. The procedure is summarized in Algorithm 5.

A.1.2 Random Variable Neighborhood Search (RVNS)

RVNS (Pirkwieser and Raidl, 2008) uses three different neighborhood structures. For each of these structures, the authors defined six moves, hence resulting in a total of 18 neighborhoods: (1) randomly changing patterns of customers, (2) moving a random segment of customers of a route to another route on the same day, and (3) exchanging two random segments of customers between two routes on the same day. In the latter two cases, the segments are reversed with a small probability, $p_{rev} = 0.1$. The neighborhoods are summarized in Table A.1.

Table A.1: Neighborhood structures of RVNS

Neighborhood structure	Value of δ	Neighborhood
Change pattern up to	δ = [1,6] times	N_1,...,N_6
Move segment of maximal length	$\delta = [1,5]$	$N_7,\ldots,N_{11}$
	bounded by the route size	N_{12}
Exchange segment of maximal length	δ = [1,5]	$N_{13},\ldots,N_{17}$
	bounded by corresponding route size	N_{18}

RVNS starts with a random neighborhood ordering and generates a new ordering each time a full VNS iteration is completed. For intensification, RVNS applies 2-opt in a best-improvement fashion. Additionally, each new incumbent solution is subject to a 2-opt*. RVNS accepts solutions which degrade the objective value under the Metropolis criterion, like in simulated annealing. Thus, an inferior solution s' is accepted with probability $e^{-(f(s')-f(s))/T}$, depending on the cost difference to the current solution s relative to the temperature T. A linear cooling scheme is applied, T being decreased every τ iterations by an amount of $(T * \tau)/\tau_{max}$, where τ_{max} denoted the maximal VNS iterations. The value of τ was adjusted to the smaller number of iterations performed and set to τ=10, the initial temperature value $T_0 = 10$, and τ_{max} = [100, 800]. The detailed description of the implementation is given in Algorithms 6 and 7.

Algorithm 6 Shaking(solution s, int k, double p_{rev})

```
1: if ($1 \leq k \leq 6$) then
2:     $s' \leftarrow$ ShakingPattern($s$, $k$) (i.e., neighborhood $N_k$)
3: else if ($7 \leq k \leq 12$) then
4:     $s' \leftarrow$ ShakingMoveSegment($s$, $k$-6, $p_{rev}$) (neighborhood $N_{k-6}$)
5: else if ($13 \leq k \leq 18$) then
6:     $s' \leftarrow$ ShakingExchangeSegments($s$, $k$-12, $p_{rev}$) (neighborhood $N_{k-12}$)
7: end if
8: return $s'$
```

A.2 Detailed Results for the PVRPTW

Table A.2 summarizes the HGGA results for the Pirkwieser and Raidl (2009a) instances with and without travel cost truncation. The algorithm was run 10 times per instance. Best results, average results, standard deviations, and computation time are reported. Vidal et al. (2013a) are the only one to provide the best solutions for their algorithm (VCGP13), and for the truncation case only. We therefore also include these best results of the truncation case in the column VCGP13 for comparison purposes. The "GAP Best-to-Best" column displays the gaps of the best solutions obtained by HGGA with respect to those of VCGP13.

Algorithm 7 RVNS(solution s, int max_{iter})

```
1:  $p_{rev}$ = 0.1; $T = 10$ {initial temperature}
2:  $s^* \leftarrow s$
3:  $numNeighbor = 18$ {number of neighborhoods}
4:  $curIter = 1$ {current iteration}
5:  **repeat**
6:      RandPermutation(ar, $numNeighbor$) {random permutation of neighborhood sequence}
7:      $curNeighbor = 1$ {index of current neighborhood}
8:      **while** ($curNeighbor \leq numNeighbor$ and $curIter \leq max_{iter}$) **do**
9:          $k$ = ar[$curNeighbor$]
10:         $s'$ = Shaking($s$, $k$, $p_{rev}$)
11:         2-opt($s'$)
12:         **if** ($s'$ better than $s$) **then**
13:             2-opt*($s'$)
14:             $s \leftarrow s'$
15:             **if** ($s$ better than $s^*$) **then**
16:                 $s^* \leftarrow s$
17:             **end if**
18:         **else**
19:             {
20:             $prob_{accept}$ = $e^{-(f(s')-f(s))/T}$
21:             **if** (accept $s'$ with probability $prob_{accept}$) {accept worse solution} **then**
22:                 $s \leftarrow s'$
23:             **else**
24:                 $curNeighbor + = 1$
25:             **end if**
26:             }
27:         **end if**
28:         $curIter + = 1$
29:         Update temperature $T$ if needed
30:     **end while**
31: **until** ($curIter > max_{iter}$) {exceed maximum number of iterations}
32: return $s^*$
```

Table A.2: HGGA results on Pirkwieser and Raidl (2009a) instances with and without travel cost truncation

Instances	Without truncation			With truncation					
	HGGA			HGGA				VCGP13	GAP (%)
No	Best	Avg	Std	Best	Avg	Std	Time (min)	Best	Best-to-Best
p4r101	4162.90	4169.59	7.44	4082.6	4084.59	2.63	25.73	4082.0	0.015
p4r102	3738.16	3742.76	4.29	3724.5	3728.98	3.52	32.81	3724.3	0.005
p4r103	3166.02	3172	9.1	3153.1	3158.86	5.8	31.92	3153.1	0
p4r104	2578.06	2585.99	8.75	2566.3	2576.77	7.82	33.99	2566.0	0.012
p4r105	3659.17	3667.67	8.82	3638.8	3647.34	10.37	32.83	3638.9	-0.003
p4c101	2913.81	2913.81	0	2907.4	2907.4	0	34.26	2907.4	0
p4c102	2888.31	2894.39	7.7	2882.9	2887.48	4.77	35.67	2882.9	0
p4c103	2735.20	2752.55	12.68	2734.5	2745	7.34	41.02	2734.5	0
p4c104	2425.30	2437.63	15.15	2419.0	2425.96	11.08	43.33	2419.0	0
p4c105	2888.30	2902.44	10.94	2884.1	2886.62	9.17	34.67	2884.1	0
p4rc101	3975.53	3980.43	9.89	3955.3	3960.53	11.58	30.43	3956.4	-0.028
p4rc102	3765.02	3777.95	11.5	3755.7	3755.72	0.04	31.06	3755.7	0
p4rc103	3469.81	3482.64	14.63	3450.1	3461.38	16.07	32.46	3449.9	0.006
p4rc104	2999.94	3013.01	11.52	2991.6	2995.49	5.02	35.11	2991.5	0.003
p4rc105	3952.03	3966.77	15.49	3933.1	3936.27	2.92	32.94	3932.6	0.013
Average	3287.84	3297.31	9.86	3271.93	3277.23	6.54	33.88	3271.89	0.002
p6r101	5393.83	5398.83	6.6	5376.4	5380.55	6.65	38.76	5376.1	0.006
p6r102	5298.10	5303.3	10.37	5201.3	5211.37	7.3	37.51	5201.6	-0.006
p6r103	3955.42	3975.68	14.42	3940.6	3956.29	13.71	44.75	3940.5	0.003
p6r104	3341.60	3375.2	14.72	3335.9	3344.35	10.5	47.57	3335.8	0.003
p6r105	4288.44	4300.96	10.64	4273.0	4285.02	13.45	38.73	4272.9	0.002
p6c101	3984.30	3998.94	9.97	3981.2	3991.09	9.12	37.93	3981.2	0
p6c102	3843.20	3856.08	8.76	3841.7	3841.7	0	44.31	3841.7	0
p6c103	3523.30	3542.28	18.48	3523.3	3529.66	8.14	52.27	3523.6	-0.009
p6c104	3210.58	3230.69	13.43	3217.4	3225.104	8.92	52.81	3206.3	0.346
p6c105	4052.10	4072.3	11.9	4052.1	4052.1	0	42.82	4052.1	0
p6rc101	5792.86	5806.72	9.57	5780.6	5787.24	7.75	37.97	5781.5	-0.016
p6rc102	5353.53	5380.13	18.28	5333.2	5342.72	11.01	41.93	5333.3	-0.002
p6rc103	4289.15	4310.79	20.2	4273.3	4288.97	13.9	41.98	4273.1	0.005
p6rc104	4076.84	4092.66	18.11	4062.1	4078.36	15.29	48.53	4062.0	0.002
p6rc105	5241.73	5264.98	24.12	5227.2	5241.86	11.71	40.11	5227.1	0.002
Average	4376.33	4393.97	13.97	4361.29	4370.43	9.16	43.19	4360.59	0.022
p8r101	6492.00	6505.19	11.8	6469.6	6483.93	9.83	44.09	6471.3	-0.026
p8r102	6158.62	6163.74	7.88	6098.1	6108	11.55	45.23	6097.9	0.003
p8r103	4717.23	4760.88	18.7	4687.1	4696.72	13.91	51.02	4687.0	0.002
p8r104	4402.72	4430.98	15.54	4353.1	4366.52	14.87	52.31	4355.8	-0.062
p8r105	5491.93	5505.27	19.38	5476.9	5484.21	14.65	45.31	5476.5	0.007
p8c101	4687.93	4713.2	16.1	4679.1	4693.37	10.42	49.6	4679.1	0
p8c102	4942.33	4966.82	15	4933.3	4953.35	16.28	51.22	4933.3	0
p8c103	4673.79	4689.97	16.94	4664.3	4676.4	15.47	61.28	4664.0	0.006
p8c104	4606.24	4626.79	20.81	4591.7	4603.21	14.57	46.33	4591.6	0.002
p8c105	5146.48	5169.32	21.27	5134.2	5134.2	0	47.62	5134.2	0
p8rc101	6883.01	6898.79	19.19	6846.9	6858.81	12.82	41.24	6847.2	-0.004
p8rc102	5778.46	5799.03	20.91	5763.4	5769.08	7.53	48.09	5763.3	0.002
p8rc103	5447.04	5464.45	20.28	5425.1	5433.06	8.19	47.53	5424.9	0.004
p8rc104	4942.07	4960.9	16.39	4929.6	4941.31	7.77	53	4929.5	0.002
p8rc105	6224.69	6255.93	29.08	6203.6	6216.2	12.31	47.43	6203.4	0.003
Average	5372.97	5394.08	17.95	5350.4	5361.22	11.34	48.75	5350.6	-0.004

Annex B

SUPPLEMENTARY MATERIAL FOR THE TMZT-VRPTW

B.1 Neighborhood Selection Strategies

This Annex briefly presents the neighborhood selection strategies that have been studied to determine how neighborhoods should be combined in our algorithm and how they are intertwined. First, to verify the necessity of both types of neighborhood, i.e., vehicle-to-supply-point assignment and routing neighborhoods, we developed a variant (identified as **O**) using routing neighborhoods only. In this variant, the three routing neighborhoods are evaluated at each iteration, and the best move is selected among all the moves in all three neighborhoods. Second, to select the neighborhood selection strategy to be included into the proposed tabu search (Section 4.5.5), we tried two variants of a greedy strategy (called **L** and **M**), three variants of fixing the probability of selecting leg neighborhoods (called **N1**, **N2**, and **N3**), and three variants of a two-level strategy. The performance of all variants is compared to that of the *Control* procedure embedded into the proposed tabu search, where the selection of the neighborhood is driven dynamically.

Greedy strategy L. All five neighborhoods are evaluated at each iteration. The best move is selected among all the moves in all neighborhoods.

Greedy strategy M. One type of neighborhood, leg or routing, is first selected at each iteration based on the value of r. Then, the neighborhoods of the selected type are evaluated. The best move is finally selected.

Fixing the probability of selecting neighborhoods. A single neighborhood is evaluated at each iteration selected randomly by a fixed probability. The **Strategy N1** allocates the same probability to all neighborhood ($r = 1$). **Strategy N2** sets the probability of selecting routing neighborhoods greater than the probability of selecting leg neighborhoods ($r > 1$) to give more time to optimizing routes following a leg move. **Strategy N3** allows the search to freely explore the solution space at the beginning by allowing all neighborhoods to share the same probability of selection, while restricting the selection of leg moves (and encouraging the optimization of routes following such a move) afterwards. It thus applies Strategy N1 for the first T_1 iterations, and Strategy N2 for the last T_2 iterations.

The same number of iterations, 1 million, was performed for all strategies on each instance. In addition, we ran three times the M, N2, N3 strategies for three values of $r = \{15, 30, 45\}$. Table B.1 displays the average best- solution results for all strategies and problem types. The last column (GAP to Control (%)) shows the average gap of each strategy relative to the results obtained by the *Control* neighborhood selection strategy used in the implementation studied in the main sections of the paper.

Table B.1: Performance comparison between neighborhood selection strategies

Strategy		Problem set						GAP to
		A1	A2	B1	B2	C1	C2	Control (%)
O		18602.03	15015.79	56502.56	48764.74	122616.80	108965.20	5.91
L		18593.07	14958.53	56089.99	48058.11	117755.10	107118.90	4.43
M	$r = 15$	18246.81	14744.71	54524.40	46218.93	116762.10	100245.40	1.40
	$r = 30$	18253.74	14740.81	54408.71	46526.30	116008.10	100474.13	1.41
	$r = 45$	18268.07	14729.11	54100.25	46295.93	117025.80	100810.93	1.43
N1		18255.39	14660.38	54109.26	46122.47	116200.50	100056.02	1.04
N2	$r = 15$	18198.15	14664.80	53927.76	46536.96	115836.10	99894.28	0.94
	$r = 30$	18198.24	14636.68	54029.69	46535.03	115298.60	99982.87	0.94
	$r = 45$	18200.22	14663.08	53865.71	46330.53	115812.40	99978.64	0.92
N3	$r = 15$	18143.10	14657.21	54364.18	46320.67	116379.40	99931.71	1.09
	$r = 30$	18158.00	14646.96	53869.52	46282.02	115768.70	99957.79	0.83
	$r = 45$	18142.19	14646.21	53741.44	46128.00	115595.40	99973.40	0.70
Control		18102.02	14549.13	53539.38	46000.00	114260.90	98503.27	

Examining the experimental results, as expected, the variant O using routing neighborhoods only produced the worst solutions, emphasizing the importance of the including both types of neighborhoods. Furthermore, among variants using both types of neighborhoods, i.e., L, M, N1, N2, and N3, we observe that the performance of Strategy L is the worst. The main reason is that leg moves, which are necessary to diversify the search, are not selected sufficiently often in this strategy. Indeed, one observes the better performance of the greedy Strategy M where leg moves are given a higher probability of selection. The best performance in this group of strategies is offered by Strategy N3, however, adjusting the type of exploration as the search progresses, from equal probabilities at the beginning of the search, to increased route-optimization neighborhoods in later stages.

Yet, fixing a priori the selection probabilities may limit the exploration capability of the search. Indeed, the *Control* procedure, implementing a selection strategy driven dynamically by variations in solution quality, outperforms all the other strategies.

Two-level strategy. It consists in running TS using one type of neighborhood until no further improvement can be found, and then switching to using the other type of neighborhood.

The algorithm starts at the "high level" applying the two leg neighborhoods in randomized order. In each case, all pairs of vehicles are searched sequentially, and the search stops either at the first improvement, or with the best improvement once all pairs have been evaluated. When both neighborhoods are completely evaluated but no improving solution has been identified, the best move is selected and becomes the current solution. "Low-level" search then starts from this solution using the three routing neighborhoods in randomized order with either the first or the best improvement criterion. Low-level search continues until no further improving solution can be found. The algorithm stops when the maximum number of iterations is reached, otherwise it switches back to the high level.

In the variant described above, only one leg move is implemented before making the transition from the high level to the low level, no mater whether improved moves have been performed or not. It may cause the two-level search to be trapped in local optima, since employing only one leg move may not change significantly the solution. Thus, we define the second variant where the search continues for a small number of leg moves before switching to the low level when no improving leg move is found in a high-level TS sequence.

We run each variant for 1 million iterations. Table B.2 reports comparison results between these variants of the two-level neighborhoods strategy. Corresponding average gaps to the solutions obtained by the tabu search with the *Control* neighborhood-selection strategy and average computation times are displayed in Columns *GAP to Control* and *Time (min)*, respectively.

For the first variant, best improvement produces higher quality solutions with an average error gap of 1.25% compared to 2.87% for the first improvement strategy. However, first improvement requires less computation time. For the second variant, the number of leg moves (high level search) is selected randomly in the integer interval [2,5]. The result is reported in the two last columns of Table B.2. One observes that this variant performs better than the two previous variants. However, given a same total number of iterations (1 million), compared to strategies N2 and N3 where neighborhoods are mixed liberally or their selections are driven by a fixed number, the two-level strategy does not perform better.

Table B.2: Comparative performance of two level neighborhood-search strategies

Problem set	First improvement		Best improvement		Best improvement & random leg moves	
	GAP to Control	Time (min)	GAP to Control	Time (min)	GAP to Control	Time (min)
A1	1.96%	32	1.51%	52	1.35%	64
A2	2.27%	25	1.62%	47	1.24%	53
B1	4.95%	59	2.59%	102	2.21%	128
B2	1.93%	41	0.74%	83	0.58%	94
C1	3.01%	105	0.65%	171	0.21%	193
C2	3.12%	93	0.39%	121	0.01%	138
Average	2.87%	59.17	1.25%	96	0.93%	111.67

To further illustrate the performance of our neighborhood selection strategy, Table B.3 reports the average move value for each of the five neighborhoods, as well as the proportion (in %) of each move application. To compare fairly, we compute the move value as the change in the routing cost and the fixed cost of using vehicles, without including penalties for constraint violation. Negative values indicate improvement. It can be seen that all three routing move types yield improvements (on average), while leg moves do not always do.

Routing moves and leg moves are respectively applied about 90% and 10% on average, ensuring adequate optimization of work assignments as desired.

Table B.3: Comparative performance of neighborhood types

Problem	Average value of routing move			Average value of leg move		Routing move applications	Leg move applications
set	Relocation	Exchange	2-opt	Relocation	Exchange	%	
A1	-0.77	-0.50	-0.53	1.92	-2.35	91.78	8.22
A2	-0.96	-0.59	-0.57	-0.62	-0.22	91.88	8.12
B1	-1.59	-0.85	-0.63	-0.97	-1.83	93.11	6.89
B2	-1.07	-0.65	-0.59	-0.76	-0.35	90.49	9.51
C1	-1.20	-0.45	-0.56	0.84	-1.76	88.52	11.48
C2	-1.34	-0.82	-0.73	0.55	0.34	89.85	10.15

B.2 Detailed Results for the TMZT-VRPTW

Tables B.4, B.5, and B.6 display comparison results on the Crainic et al. (2012b) instances. The first group of columns displays the results of Crainic et al. (2012b), the best solution values (*Best* column), the number of vehicles (*#Vehicles* column), the number of times vehicles move directly from one customer zone to another without passing through waiting stations (*DM* column), and the number of times waiting stations are used for moving between customer zones (*MWS* column). The next group of columns displays the same information for the proposed tabu search, plus the average values (*Avg 10* column) and standard deviations (*Std* column) over 10 runs, the range of route legs per vehicle (*RLPV* column) and the average number of route legs per vehicle (*ALPV* column), as well as the corresponding gap to the previous BKS (last column).

Table B.4: Best performance comparison between our algorithm and Crainic et al. (2012b) on problem sets A

Instances	Crainic et al. (2012b)				TS								GAP
No	Best	#Vehicles	DM	MWS	Avg 10	Std	Best 10	#Vehicles	DM	MWS	RLPV	ALPV	%
A1-1	17767	19	0	44	17152.7	24.36	17118.40	18	0	44	[2,4]	3.44	-3.65
A1-2	18783	24	0	35	18392.85	18.29	18371.20	24	0	35	[1,4]	2.46	-2.19
A1-3	16114	19	0	41	15749.17	14.10	15731.10	19	0	41	[2,4]	3.11	-2.38
A1-4	20936	31	0	30	20798.3	17.71	20780.00	31	0	29	[1,3]	1.94	-0.75
A1-5	15850	18	11	34	15633.8	17.46	15613.70	18	13	32	[3,4]	3.44	-1.49
A1-6	17456	20	0	40	16760.83	27.52	16720.80	19	0	40	[2,4]	3.11	-4.21
A1-7	19169	24	0	37	18790.99	15.08	18760.10	23	0	37	[1,4]	2.54	-2.13
A1-8	16790	20	2	40	16598.9	23.40	16553.30	20	4	38	[2,4]	3.11	-1.41
A1-9	21570	32	0	31	19361.32	43.02	19318.10	26	4	31	[1,4]	2.37	-10.44
A1-10	21316	32	0	26	21199.57	42.77	21138.50	32	0	26	[1,3]	1.81	-0.83
Average	18575.1	23.9	1.3	35.8	18043.84	24.37	18010.52	23	2.1	35.3	-	2.73	-3.04
A2-1	16380	21	1	41	15864.90	29.61	15823.40	20	2	41	[1,5]	3.15	-3.40
A2-2	18433	24	2	36	17140.57	35.63	17085.00	21	10	35	[1,5]	3.10	-7.31
A2-3	14654	18	4	42	13785.34	37.68	13717.80	16	6	41	[3,6]	3.94	-6.39
A2-4	13056	13	0	49	12828.75	31.65	12774.90	13	0	48	[4,6]	4.69	-2.15
A2-5	14256	14	3	46	13622.56	33.07	13579.70	13	3	46	[2,7]	4.77	-4.74
A2-6	17480	24	7	30	15430.33	36.16	15347.70	20	15	27	[1,5]	3.10	-12.16
A2-7	13955	15	1	49	13325.39	31.88	13287.90	14	0	49	[3,5]	4.5	-4.78
A2-8	14975	18	0	46	14815.85	29.67	14765.50	18	0	46	[1,5]	3.56	-1.40
A2-9	14430	16	0	45	13149.59	45.10	13103.00	13	4	46	[2,7]	4.77	-9.20
A2-10	16490	22	6	35	14988.99	53.43	14915.60	19	11	35	[2,6]	3.42	-9.55
Average	15410.9	18.5	2.4	41.9	14495.23	36.39	14440.05	16.8	5.1	41.3	-	3.90	-6.29

Table B.5: Best performance comparison between our algorithm and Crainic et al. (2012b) on problem sets B

Instances	Crainic et al. (2012b)				TS								GAP
No	Best	#Vehicles	DM	MWS	Avg 10	Std	Best 10	#Vehicles	DM	MWS	RLPV	ALPV	%
B1-1	71007	89	4	153	65660.43	81.26	65570.2	76	30	140	[1,7]	3.22	-7.66
B1-2	53419	44	22	180	50886.42	80.38	50778.1	39	42	160	[3,8]	6.16	-4.94
B1-3	51175	41	20	186	48328.92	33.47	48284.8	36	36	170	[3,10]	6.72	-5.65
B1-4	54331	42	25	176	53547.07	64.46	53439.8	42	32	168	[3,9]	5.77	-1.64
B1-5	54072	45	12	186	51435.01	56.81	51309.7	37	31	171	[3,10]	6.44	-5.11
B1-6	54593	50	16	182	51604.48	61.61	51577.5	41	45	161	[3,10]	6.01	-5.52
B1-7	53322	46	15	180	51843.15	67.97	51785	44	26	170	[3,7]	5.44	-2.88
B1-8	55423	48	5	193	53941.65	70.38	53820.5	45	22	183	[2,10]	5.55	-2.89
B1-9	55208	53	19	168	52555.43	71.93	52427.7	48	32	162	[2,9]	5.03	-5.04
B1-10	53979	48	5	195	51437.39	75.74	51368	42	14	188	[3,9]	5.80	-4.84
Average	55652.9	50.6	14.3	179.9	53124.00	66.40	53036.13	45.00	31.00	167.30	-	5.61	-4.62
B2-1	44889	32	21	193	43492.67	79.24	43318.3	27	43	175	[7,13]	9.06	-3.50
B2-2	50427	47	22	178	46697.50	109.51	46506.1	39	54	152	[3,11]	6.28	-7.78
B2-3	48941	40	25	188	46839.89	121.45	46582	36	52	168	[3,12]	7.11	-4.82
B2-4	45894	28	19	206	44627.32	80.86	44531.5	28	34	193	[6,13]	9.10	-2.97
B2-5	46523	41	17	195	44538.50	89.31	44345.1	37	29	182	[3,10]	6.70	-4.68
B2-6	46441	36	16	199	45374.68	91.52	45066.5	35	21	190	[4,11]	7.01	-2.96
B2-7	44894	34	16	199	43601.74	61.52	43450.6	31	32	183	[6,10]	7.93	-3.22
B2-8	44549	28	23	201	43041.03	101.17	42745.1	24	42	183	[8,13]	10.37	-4.05
B2-9	46801	37	21	198	44543.77	62.17	44408.8	30	58	165	[4,12]	8.43	-5.11
B2-10	54606	62	16	172	49972.49	96.31	49787.6	50	33	171	[2,9]	5.08	-8.82
Average	47396.5	38.5	19.6	192.9	45272.96	89.31	45074.16	33.70	39.80	176.20	-	7.71	-4.79

Table B.6: Best performance comparison between our algorithm and Crainic et al. (2012b) on problem sets C

Instances	Crainic et al. (2012b)				TS								GAP
No	Best	#Vehicles	DM	MWS	Avg 10	Std	Best 10	#Vehicles	DM	MWS	RLPV	ALPV	%
C1-1	115967	86	67	394	111243.00	62.10	111196	75	114	359	[4,12]	7.30	-4.11
C1-2	113176	92	43	411	109077.00	98.18	108993	81	85	388	[2,10]	6.82	-3.69
C1-3	114773	80	62	409	109013.30	118.31	108758	68	101	374	[4,14]	7.98	-5.24
C1-4	114310	83	30	435	110412.90	109.96	110269	79	61	404	[4,12]	6.88	-3.54
C1-5	121245	100	25	423	115635.50	116.06	115335	84	55	406	[3,10]	6.48	-4.87
C1-6	115324	87	39	426	111761.60	180.11	111558	81	64	400	[3,12]	6.72	-3.27
C1-7	120443	86	31	431	117028.30	98.70	116869	83	59	401	[3,10]	6.54	-2.97
C1-8	115473	78	40	433	111768.80	97.67	111565	72	76	394	[4,13]	7.52	-3.38
C1-9	126060	89	34	427	121740.30	112.01	121607	84	74	387	[3,14]	6.40	-3.53
C1-10	117493	85	18	442	112079.20	120.98	111959	74	47	426	[4,12]	7.39	-4.71
Average	117426.4	86.6	38.9	423.1	112975.99	111.41	112810.90	78.10	73.00	393.90	-	7.01	-3.92
C2-1	101232	69	70	423	97566.43	101.30	97350.2	67	101	391	[4,12]	8.34	-3.83
C2-2	98289	53	85	433	95340.22	105.34	95262.7	48	146	372	[8,19]	11.79	-3.08
C2-3	106180	65	71	426	102038.30	85.31	101758	62	102	388	[5,14]	8.90	-4.16
C2-4	97387	74	35	450	93426.91	96.49	93217.2	64	79	410	[5,13]	8.64	-4.28
C2-5	101090	53	74	442	96744.10	103.29	96581.4	52	98	414	[6,15]	10.84	-4.46
C2-6	106847	85	54	426	101813.60	88.09	101665	75	90	392	[3,12]	7.42	-4.85
C2-7	98471	62	76	424	95197.20	98.25	95096.4	60	99	395	[5,13]	11.08	-3.43
C2-8	99948	55	73	442	96761.28	82.87	96675.2	49	123	388	[7, 17]	11.42	-3.27
C2-9	102301	57	79	432	97949.94	92.90	97691.2	56	134	373	[5,14]	10.05	-4.51
C2-10	103950	63	67	437	100633.80	94.98	100561	60	108	387	[6,15]	9.25	-3.26
Average	101569.5	63.6	68.4	433.5	97747.18	94.88	97585.83	59.30	108.00	391.00	-	9.77	-3.91

Annex C

SUPPLEMENTARY MATERIAL FOR THE MZT-PDTWS

Tables C.1, C.2, C.3, C.4, C.5, and C.6 display the detailed results obtained by the proposed tabu search, the average values (*Avg10* column), standard deviations (*Std* column), and the best solution values (*Best10* column) over 10 runs, the number of vehicles (*#Vehicles* column), the number of times vehicles move directly to a supply point without passing through waiting stations (*DM* column), the number of times waiting stations are used for moving between customer zones (*MWS* column), the number of times vehicles do both 'unload and load' operation once they arrive at supply points (*PD* column), the number of legs (*#Legs* column).

Table C.1: Detailed results on problem instances set A1

Instance	Avg10	Std	Best10	#Vehicles	DM	MWS	PD	#Legs
A1-1	19125.65	35.52	19052.70	18	3	42	8	70
A1-2	20627.00	41.79	20532.30	24	2	35	9	68
A1-3	17555.37	77.06	17438.78	18	0	42	9	68
A1-4	24232.20	95.98	24027.82	31	0	31	9	69
A1-5	17826.23	44.93	17741.77	18	16	30	9	72
A1-6	19768.53	40.51	19694.70	18	17	33	27	89
A1-7	21709.69	67.39	21572.62	24	9	30	25	85
A1-8	18536.57	87.90	18292.37	18	13	33	26	85
A1-9	25565.77	100.09	25382.80	31	1	32	26	87
A1-10	19457.42	92.58	19328.76	19	19	28	27	90
A1-11	21572.33	79.16	21399.10	18	43	19	57	123
A1-12	24223.69	93.21	23978.20	24	28	31	46	119
A1-13	20797.80	70.05	20652.90	18	31	32	55	118
A1-14	28375.16	112.27	28136.80	31	18	40	55	118
A1-15	22390.35	139.26	22061.10	20	39	29	57	125
Average	21450.92	78.51	21286.18	22.00	15.93	32.47	29.67	92.40

Table C.2: Detailed results on problem instances set A2

Instance	Avg10	Std	Best10	#Vehicles	DM	MWS	PD	#Legs
A2-1	18577.68	98.24	18410.01	19	6	40	12	76
A2-2	19725.88	87.47	19590.37	21	12	30	8	70
A2-3	15885.33	87.89	15767.75	16	9	40	10	74
A2-4	14923.72	42.97	14859.18	13	1	49	11	72
A2-5	15307.64	91.77	15156.95	13	7	43	10	72
A2-6	20286.66	97.82	20113.70	19	16	32	30	95
A2-7	21459.12	99.36	21321.10	21	21	25	29	92
A2-8	17541.52	97.37	17339.61	16	17	35	28	91
A2-9	16378.68	80.03	16209.82	13	6	47	28	89
A2-10	16646.85	88.86	16494.59	13	20	33	28	90
A2-11	22422.22	91.86	22195.60	20	24	41	55	127
A2-12	23920.24	56.28	23823.10	21	38	27	55	124
A2-13	21312.70	97.21	21079.20	18	23	39	58	126
A2-14	19055.70	90.23	18901.30	14	22	40	57	121
A2-15	19043.00	84.86	18906.10	13	37	27	55	124
Average	18832.46	86.15	18677.89	16.67	17.27	36.53	31.60	96.20

Table C.3: Detailed results on problem instances set B1

Instance	Avg10	Std	Best10	#Vehicles	DM	MWS	PD	#Legs
B1-1	83561.38	35.75	83498.7	77	30	151	30	282
B1-2	62675.24	70.27	62513.8	40	50	158	33	277
B1-3	59566.89	103.07	59281.3	36	45	171	30	276
B1-4	66674.47	119.30	66454.1	41	39	168	31	273
B1-5	62420.99	139.67	62071.1	40	33	174	29	271
B1-6	94815.97	105.30	94634.4	77	47	171	104	358
B1-7	70117.71	123.55	69773.4	43	72	150	104	349
B1-8	69328.12	113.24	69139.9	37	82	148	106	356
B1-9	72630.66	74.17	72449.4	40	79	150	103	352
B1-10	71042.78	32.74	70994	42	69	147	104	347
B1-11	115479.33	105.64	115313.5	80	70	205	219	495
B1-12	90142.91	85.73	90053.1	46	120	146	211	484
B1-13	93478.07	73.38	93389.9	49	119	152	213	492
B1-14	99444.83	72.84	99349.3	46	87	176	216	485
B1-15	97233.69	97.95	97024	53	114	142	211	480
Average	80574.20	90.17	80395.99	49.80	70.40	160.60	116.27	371.80

Table C.4: Detailed results on problem instances set B2

Instance	Avg10	Std	Best10	#Vehicles	DM	MWS	PD	#Legs
B2-1	57606.41	97.85	57406.8	31	48	174	40	291
B2-2	64177.54	68.39	64048	45	47	165	36	285
B2-3	61249.92	90.45	61096.2	36	43	182	38	292
B2-4	58321.89	108.80	58286.9	32	35	194	33	289
B2-5	57787.64	77.15	57749.5	38	38	181	36	288
B2-6	69084.69	68.22	68991.4	38	45	178	104	353
B2-7	77033.95	85.71	76810.3	48	68	169	112	366
B2-8	73010.21	70.96	72821.9	36	59	174	106	360
B2-9	70535.79	58.06	70448.2	36	52	174	103	353
B2-10	70829.02	96.03	70656	42	46	177	109	361
B2-11	91315.56	98.60	91199.3	41	86	180	211	487
B2-12	100857.73	60.85	100728.3	60	95	177	220	493
B2-13	97624.36	102.62	97410.2	41	85	192	209	493
B2-14	89699.15	131.32	89332.2	42	91	191	219	504
B2-15	90624.95	47.31	90522.6	46	92	187	220	500
Average	75317.25	84.16	75167.19	40.80	62.00	179.67	119.73	381.00

Table C.5: Detailed results on problem instances set C1

Instance	Avg10	Std	Best10	#Vehicles	DM	MWS	PD	#Legs
C1-1	154322.30	72.45	154127	93	89	374	65	616
C1-2	150028.70	142.39	149835	92	82	374	65	607
C1-3	152287.90	135.71	152100	83	95	386	65	619
C1-4	154935.90	57.31	154805	97	68	393	63	616
C1-5	155101.20	161.33	154665	87	78	395	65	619
C1-6	203099.00	120.09	202994	101	116	372	225	786
C1-7	196721.20	165.82	196335	101	100	397	228	786
C1-8	200641.20	33.40	200559	91	120	395	231	787
C1-9	198211.80	110.56	197954	107	106	375	232	792
C1-10	201688.80	162.98	201402	97	99	404	227	799
C1-11	293262.00	153.85	293078	123	156	426	501	1106
C1-12	285020.80	81.31	284801	121	203	391	484	1086
C1-13	293493.00	155.59	293226	112	221	370	491	1100
C1-14	286054.90	91.54	285829	119	162	423	497	1098
C1-15	302333.50	48.99	302249	124	172	433	469	1112
Average	215146.81	112.89	214930.60	103.20	124.47	393.87	260.53	835.27

Table C.6: Detailed results on problem instances set C2

Instance	Avg10	Std	Best10	#Vehicles	DM	MWS	PD	#Legs
C2-1	142736.00	109.07	142511	77	114	391	78	655
C2-2	141224.60	55.47	141095	77	97	414	80	663
C2-3	147322.30	61.62	147219	82	86	426	80	657
C2-4	134630.60	42.95	134567.2	73	83	416	79	642
C2-5	139177.10	196.80	138623	72	83	426	80	655
C2-6	188064.70	202.55	187501	89	144	392	232	817
C2-7	224601.60	118.17	224486	103	222	408	236	900
C2-8	191738.70	210.74	191220	98	113	433	232	817
C2-9	183440.30	166.22	183034	92	100	428	233	812
C2-10	190020.60	87.56	189789	90	106	440	235	833
C2-11	290027.30	75.65	289826	117	198	451	484	1142
C2-12	265985.50	210.68	265391	88	148	479	494	1132
C2-13	276053.30	107.45	275884	122	177	455	477	1126
C2-14	274952.90	137.77	274742	110	148	482	485	1132
C2-15	284755.10	160.20	284448	97	167	463	524	1157
Average	204982.04	129.53	204689.08	92.47	132.40	433.60	268.60	876.00