MODELS AND METAHEURISTICS FOR VEHICLE ROUTING PROBLEMS UNDER UNCERTAINTY

by

Chenlian Hu

A THESIS SUBMITTED

IN FULFILMENT OF THE REQUIREMENTS

FOR THE DEGREE OF

Doctor of Philosophy

School of Computer Science

Faculty of Engineering and Information Technology (FEIT)

University of Technology Sydney

July 2021

Certificate of Authorship/Originality

I, Chenlian Hu declare that this thesis, is submitted in fulfilment of the requirements for the award of Doctor of Philosophy, in the School of Computer Science, Faculty of Engineering and Information Technology at the University of Technology Sydney.

This thesis is wholly my own work unless otherwise referenced or acknowledged. In addition, I certify that all information sources and literature used are indicated in the thesis.

I certify that the work in this thesis has not previously been submitted for a degree nor has it been submitted as part of the requirements for a degree at any other academic institution except as fully acknowledged within the text. This thesis is the result of a Collaborative Doctoral Research Degree program with the Shanghai Jiao Tong University.

This research is supported by the Australian Government Research Training Program.

July 2021

Acknowledgements

Looking back at the past several years, I have been privileged to study and work with the smart and creative colleagues of the Decision Systems & e-Service Intelligence (DeSI) Lab in the Australian Artificial Intelligence Institute (AAII, formerly the Centre for Artificial Intelligence (CAI)). This thesis would not have been possible without their help and support.

First of all, I would like to express my deeply-felt thanks to my principal supervisor Distinguished Professor Jie Lu. I owe my sincere gratitude to Professor Lu, for her patience, assistance, and encouragement. Professor Lu is knowledgeable, humorous, and kind. She taught me how to conduct in-depth research and gave me freedom to pursue my own research interests. Without her guidance and support, this thesis would not have become a reality.

I would also like to thank my co-supervisor A./Professor Guangquan Zhang, for his selfless love and support on my research and life in Sydney. The discussions with him gave me a lot of insights and ideas for my research. His decisiveness and sharp insights motivated me when I felt lost or afraid about the future.

I am grateful to all the members of the DeSI Lab. They gave me a lot of valuable suggestions to improve this thesis. It was really a great experience to work with these dedicated researchers.

Last but not least, I would like to owe my sincere gratitude to my family. My father and mother have been continuously supporting me and giving me love and encouragement to overcome all the adversities I have faced.

Abstract

Within the logistics and transportation industry, the vehicle routing problem (VRP) bears significant importance in many real-life logistics activities. As one of the most important and widely studied combinatorial optimization problems in the past sixty years, the VRP, also known as the capacitated VRP (CVRP), focuses on minimizing transportation costs: it concerns how to serve a set of geographically dispersed customers with a fleet of homogeneous vehicles at minimum cost. Given the potentially substantial savings from optimizing routing strategies in practical logistics activities, various complex extensions of the CVRP inspired from real-life applications have increasingly received attention. In the CVRP and most of its extensions, a common assumption is that the values of all problem parameters are readily available and can be precisely known in advance. However, this assumption does not invariably hold in many practical routing problems due to uncertainty, which could be secondary to factors such as imprecise information on customer demands, unfixed service times for customers, and varying travel times for vehicles. Thus, routing strategies generated without considering uncertainty may ultimately be found infeasible in real-life applications.

This thesis aims to study several important extensions of the CVRP under uncertainty. To model these problems, we adopt the robust optimization paradigm which is an effective framework for optimization problems with uncertain data. Given their complexity, we focus on developing efficient metaheuristic solution approaches. Our investigations are threefold. Firstly, we study the vehicle routing problem with time windows considering uncertainty in customer demands, service times, and travel times. To capture these different types of uncertainty, novel route-dependent uncertainty sets are defined. The problem is modelled through a robust mathematical formulation with the route-dependent uncertainty sets and solved via a metaheuristic based on the adaptive variable neighbourhood search method. Secondly, we study the vehicle routing problem with simultaneous pickup and delivery and time windows under uncertainty in pickup demands and travel times. A robust mathematical formulation with two route-dependent uncertainty sets is presented to model the problem and a metaheuristic based on the adaptive large neighbourhood search method is proposed to solve it. Finally, we study the two-echelon multiple-trip vehicle routing problem with time windows and satellite synchronization under customer demand uncertainty. This problem considers a two-echelon transportation system and a number of practical features commonly observed in city logistics. A robust mathematical formulation with a novel demand uncertainty set and a metaheuristic based on the variable neighbourhood search framework are accordingly proposed. We conduct extensive numerical experiments which employ benchmark instances from the literature. The computational results show that the proposed solution approaches can generate high-quality deterministic and robust solutions for large-sized instances within a reasonable running time. In addition, Monte Carlo simulation tests are designed to evaluate the robustness of the obtained solutions. Useful managerial insights for decision-makers in the logistics and transportation industry are derived from a comprehensive analysis of the computational results.

Table of Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Background and Motivations

The logistics and transportation industry plays a central role in contemporary society, given its coordinated and systemic transport of goods across geographical locations and the resultant economic nexus. In Australia, it has been estimated that the industry contributed \$131.6 billion to the economy in 2013, accounting for 8.6% of the national GDP (Australian Logistics Council, 2014). Among different traditional transport modes, road transport is the most common and popular in Australia for non-bulk freight because of its advantages in price, convenience, reliability, and speed (Ferrier Hodgson, 2014). Moreover, for road transport tasks in Australia, a 75% growth has been projected over the next 20 years (Ferrier Hodgson, 2017). Thus, a small decrease in cost incurred by road transport translates into substantial profits for the whole industry. To reduce road freight transportation costs in real-life applications, researchers have paid considerable attention to the vehicle routing problem (VRP) and its numerous extensions over the past sixty years.

The VRP, also known as the capacitated VRP (CVRP), was first introduced by Dantzig and Ramser (1959) in the late fifties. The CVRP concerns how to serve a set of geographically dispersed customers with a fleet of homogeneous vehicles at a central depot. Its objective is to find a set of routes, each executed by a vehicle starting from and returning to the depot, such that each customer is served exactly once, the vehicle capacity constraints are satisfied, and the total routing cost is minimized. An example for the CVRP with 16 customers is shown in Figure 1.1. Since the CVRP is classified as an NP-hard problem (Lenstra and Kan, 1981), various solution approaches involving both exact and heuristic methods have been developed over the past six decades. Section 2.1 comprehensively reviews the solution methods for the CVRP in the literature.

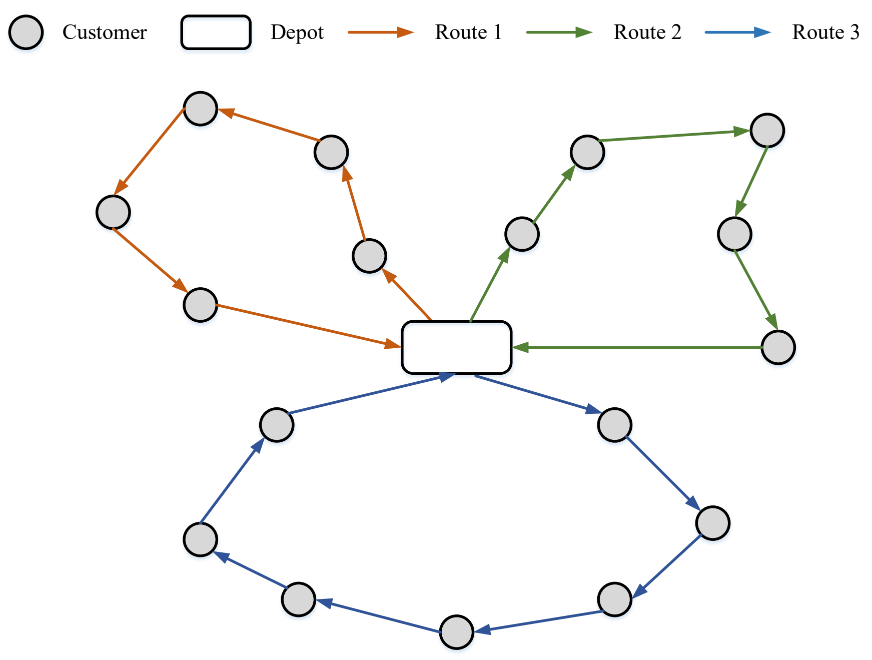


Figure 1.1 Example for the CVRP.

Given the development of efficient and effective solution methods for the CVRP, large reduction of travel distances based on test instances have been reported in the literature. Therefore, logistics and transportation companies have become increasingly interested in addressing real-world VRPs through operations research techniques. However, VRPs encountered in practical applications usually exhibit many problem-specific features, such as service time windows, heterogeneous vehicles, requirements of simultaneous pickup and delivery, and multi-echelon transportation networks. These practical features make real-life VRPs much more challenging than the standard CVRP. Thus, various extensions of the CVRP over the decades have been studied, one of which is the vehicle routing problem with time windows (VRPTW). In the VRPTW, each customer needs to be served within a specified time interval (called a time window). The time window restrictions are common in real-life logistics operations, such as parcel delivery, school bus routing, and municipal solid waste collection (Desaulniers et al., 2014). Another important extension is the vehicle routing problem with simultaneous pickup and delivery (VRPSPD). In the VRPSPD, each customer may have both a delivery demand and a pickup demand to be satisfied simultaneously. The practical applications of the VRPSPD are frequently encountered in reverse logistics, such as the distribution of returnable beverage packaging and home healthcare services. A third popular extension is the two-echelon capacitated vehicle routing problem (2E-CVRP). In the 2E-CVRP, the transportation network consists of two echelons with intermediate facilities (called satellites) (Perboli et al., 2011). In a typical two-echelon distribution system, the freight is initially transported from the depot to the intermediate facilities in the first echelon, and then from such facilities to the customers in the second echelon. City logistics, postal and parcel delivery, and multi-modal transportation are typical real-world applications of the 2E-CVRP (Cuda et al., 2015). Comprehensive surveys on the CVRP and its extensions can be found in Toth and Vigo (2014) and Braekers et al. (2016).

In the CVRP and most of its extensions, the common assumption is that the values of all problem parameters are deterministic and can be precisely known in advance. However, this assumption does not invariably hold in real-life applications due to uncertainty. In practical routing problems, a lag time usually exists between planning the routing strategy and executing it (Gendreau et al., 2014). Thus, uncertainty can seriously affect the values of critical problem parameters. Uncertainty may result from different factors, such as imprecise information on customer demands, unpredictable travel times for vehicles, and varying service times for customers. Routing plans generated without considering uncertainty may ultimately be found infeasible and thus cannot be perfectly executed in real-life applications. For example, if the travel times between any two customers are substantially variable and the service times for customers can be known only after they are served, a planned parcel delivery route performed by a vehicle on a given day may turn out to be infeasible if the vehicle misses the time windows of several customers due to service and travel time uncertainty. It follows that the vehicle may have to be rescheduled for the unserved customers on a different day, incurring additional costs. Accordingly, routing strategies overlooking uncertainty are undesirable and may jeopardize both the profits and reputation of logistics and transportation companies.

Motivated by the uncertainty issue in many real-life VRP applications, researchers have become more and more interested in studying the CVRP and its several basic extensions under uncertainty over the past two decades. Specifically, stochastic vehicle routing problems (SVRPs) have been widely studied in the literature (see Gendreau et al. (2016) and Oyola et al. (2018)). The SVRPs, an important family of extensions of the CVRP, encompass some key problem parameters assumed to be stochastic, such as customer demands and vehicle travel times. To address these problems, the stochastic programming paradigm (Birge and Louveaux, 2011) is widely adopted, under which uncertain parameters are either described as random variables that follow exact probability distributions or approximated through a large number of scenarios. Based on the stochastic programming paradigm, two basic modelling approaches - stochastic programming with recourse (SPR) and chance-constrained programming (CCP) - are often used to model the SVRPs. The paradigm, however, suffers from several limitations. Firstly, the exact probability distributions of the uncertain parameters may not be readily discerned or estimated in real-life VRP applications. Secondly, SPR models rely crucially on the definitions of the recourse actions. The design of effective and practical recourse actions could be challenging. Thirdly, many SPR and CCP models developed for the SVRPs are hard to solve. Their practical utility may thus be limited in addressing large-scale real-world VRP applications with the consideration of multiple types of uncertainty.

To overcome some of the said limitations, robust optimization has emerged as a novel framework to address optimization problems with uncertain data (Ben-Tal et al., 2009). In classical robust optimization, uncertainty is captured by uncertainty sets (Bertsimas et al., 2011). The idea of robust optimization is to generate solutions that are feasible for all realizations of the uncertain parameters in the uncertainty sets (Gabrel et al., 2014). Compared to the stochastic programming paradigm, robust optimization offers several advantages. Firstly, the robust counterparts are computationally tractable for many types of optimization problems and several types of uncertainty sets. Secondly, the probability distributions of the uncertain parameters need not be known or estimated. Uncertainty sets can be easily derived from rough historical data or decision-makers' experiences. Over the last decade, researchers have started to adopt the robust optimization framework to address the CVRP under uncertainty (Gounaris et al., 2013; Sungur et al., 2008) and some of its extensions under uncertainty (Agra et al., 2013; Lee et al., 2012; Lu and Gzara, 2019; Munari et al., 2019; Zhang et al., 2021). However, the focus has mainly been on the robust versions of the CVRP and the VRPTW considering one type of uncertainty. Moreover, most of the solution methods for these problems are exact algorithms which can solve only small- or medium-sized instances. Therefore, to bridge this research gap, adopting the robust optimization framework and designing heuristic solution approaches to address complex extensions of the CVRP under multiple types of uncertainty are worth studying.

This thesis aims to study three important and complex extensions of the CVRP under uncertainty, which can be considered as the mathematical basis of many real-life routing applications. To model these problems, we adopt the (adjustable) robust optimization framework (Ben-Tal et al., 2009, 2004). Specifically, we define novel uncertainty sets to capture uncertainty and develop robust mathematical formulations based on the defined uncertainty sets. Given the complexity of solving these mathematical formulations, we focus on designing efficient metaheuristics that can solve instances with large sizes. The proposed mathematical formulations and metaheuristics are envisioned to not only enrich the literature on VRPs under uncertainty, but also aid logistics practitioners in generating reliable and cost-effective routing strategies for real-life VRP applications.

1.2 Research Problems

In view of the aforementioned backgrounds and motivations, this thesis studies three important extensions of the CVRP under uncertainty. The research problems are briefly discussed as follows.

PROBLEM 1. *The vehicle routing problem with time windows under uncertainty.*

As one of the most important extension of the CVRP, the VRPTW involves customers who must be served within specified time windows. Due to the fast pace of the modern society and the busy schedules of individuals, customers expect their demands to be satisfied within specified time slots. Given its incorporation of time window constraints, the VRPTW can be considered as the mathematical basis of many real-life logistics activities such as parcel delivery, bank delivery, and school bus routing. However, in many practical VRPTW applications, uncertainty is inherent to important parameters such as travel times for vehicles and service times for customers. Such uncertainty may cause vehicles to miss the customers' time windows when executing their planned routing strategies. Thus, this thesis studies the VRPTW under multiple types of uncertainty in the parameters: customer demands, service times, and travel times. However, addressing the VRPTW under multiple types of uncertainty is not trivial. The characterization of different types of uncertainty and the development of a mathematical formulation to accordingly model the problem are challenging. Moreover, solving the VRPTW under different types of uncertainty with large-sized instances could present difficulties.

PROBLEM 2. *The vehicle routing problem with simultaneous pickup and delivery and time windows under uncertainty.*

The vehicle routing problem with simultaneous pickup and delivery and time windows (VRPSPDTW) can be considered as the generalization of the VRPTW and the VRPSPD. In the VRPSPDTW, each customer has a delivery demand and a pickup demand that must be satisfied simultaneously within a given time window. Due to its practical relevance to reverse logistics, the VRPSPDTW can be used to model real-world applications such as the distribution of returnable beverage packaging and some simultaneous milk pickup and diesel delivery arrangement. However, uncertainty is frequently observed in these VRPSPDTW applications, as illustrated by some simultaneous milk pickup and diesel delivery arrangement: uncertainty can be noted for pickup demands of milk farms due to limited information and for travel times of trucks due to variable road conditions and drivers' driving skills. Thus, this thesis studies the VRPSPDTW with the consideration of uncertainty in pickup demands and travel times. By comparison, the VRPSPDTW under uncertainty is more challenging than the VRPTW under uncertainty: for the former, the vehicle loads vary substantially and frequently owing to the concurrence of pickup and delivery of freight whereas, for the latter, any decrease (delivery) in the vehicle loads or any increase (pickup) is monotonic. Thus, the development of a mathematical formulation to model the problem and the design of a solution method to solve it could prove challenging.

PROBLEM 3. *The two-echelon multiple-trip vehicle routing problem with time windows and satellite synchronization under uncertainty.*

The two-echelon multiple-trip vehicle routing problem with time windows and satellite synchronization (2E-MTVRPTWSS) is a complex extension of the 2E-CVRP. In this problem, the freight is transported via satellites (intermediate facilities) in a two-echelon network. Each customer's demanded freight must be delivered (or collected) within a time window using two fleets of homogeneous vehicles. Specifically, the second-echelon vehicles are allowed to perform multiple trips. In addition, since the satellites are considered to have no storage capacity, vehicles in both echelons need to synchronize at satellites. Many practical applications in city logistics can be modelled as the 2E-MTVRPTWSS such as two-echelon municipal solid waste collection and parcel delivery with vans and bikes in urban areas. However, customer demands can be uncertain when determining effective routing strategies for such activities in city logistics because of possibly incomplete information on customer demands. Thus, this thesis studies the 2E-MTVRPTWSS under customer demand uncertainty. Given the complexity of the 2E-MTVRPTWSS, uncertainty in travel times and service times are not considered, despite its presence in many practical applications. Compared to the 2E-CVRP, the 2E-MTVRPTWSS is more difficult to solve. The 2E-MTVRPTWSS with extra consideration of uncertainty in customer demands further compounds its complexity. One challenge lies in the capture of such uncertainty and the development of a mathematical formulation to model the problem. Another lies in solving the 2E-MTVRPTWSS under such uncertainty with large-sized instances.

1.3 Research Contributions

To address the aforesaid research problems, this thesis develops mathematical formulations based on the adjustable robust optimization framework (Ben-Tal et al., 2009, 2004) and designs efficient metaheuristic solution approaches. The main contributions of this research are as follows.

- To model the VRPTW under uncertainty, a robust mathematical formulation is constructed. The robust optimization model simultaneously incorporates three types of uncertainty in customer demands, service times, and travel times. Each type of uncertainty is captured by a novel route-dependent uncertainty set. A metaheuristic solution approach based on the adaptive variable neighbourhood search (AVNS) method (Stenger et al., 2013) is proposed to solve the problem.

- To model the VRPSPDTW under uncertainty, a robust mathematical formulation is constructed. Two types of important problem parameters including pickup demands and travel times are assumed to be uncertain. These two types of uncertainty are captured by two route-dependent uncertainty sets. A metaheuristic solution approach based on the adaptive large neighbourhood search (ALNS) method (Stenger et al., 2013) is proposed to solve the problem.

- To model the 2E-MTVRPTWSS under customer demand uncertainty, a robust mathematical formulation is constructed. A novel uncertainty set is defined to capture the uncertainty in customer demands based on vehicle routes in two echelons. Additionally, a novel route generation strategy is proposed, in which a set of first-echelon vehicle routes are derived from a set of feasible second-echelon ones. Moreover, a metaheuristic solution approach which adopts the novel route generation strategy and the variable neighbourhood search (VNS) framework (Mladenović and Hansen, 1997) is proposed to solve the problem.

- Extensive numerical experiments are conducted which employ various benchmark instances from the literature. Computational results show that the proposed metaheuristics can generate high-quality deterministic and robust solutions for the problems of interest with large-sized instances within a reasonable running time. In addition, Monte Carlo simulation tests are designed to evaluate the robustness of the obtained solutions. Moreover, the impacts of different types of uncertainty on deriving robust solutions are investigated. Useful managerial insights for practitioners in the logistics and transportation industry are derived from a comprehensive analysis of the computational results.

1.4 Research Significance

The significance of this research is twofold: theoretical and practical.

Theoretical significance: Research on VRPs under uncertainty is still limited in the existing literature due to the difficulty of properly modelling uncertainty and the complexity of solving VRPs with large-sized instances. Therefore, this research crucially addresses three extensions of the CVRP (i.e., the VRPTW, the VRPSPDTW, and the 2E-MTVRPTWSS) under uncertainty. The proposed mathematical formulations and metaheuristic solution approaches are of theoretical significance as they can enrich the current methodologies for modelling and solving different VRPs under uncertainty. Moreover, these methods can be easily adapted to address other complex extensions of the CVRP inspired from real-life routing problems.

Practical significance: Uncertainty is prevalent in practical logistics and transportation activities. Routing plans generated without considering uncertainty may ultimately be found infeasible and thus unusable in real-life applications. The proposed mathematical formulations and metaheuristic solution approaches are of practical significance as they can generate robust routing strategies that are not easily affected by uncertainty in real-life routing problems. Moreover, these strategies generally have a good balance between costs and robustness. Thus, they have the potential for reducing the overall transportation costs and improving customer satisfaction in practical logistics activities.

1.5 Thesis Structure

This thesis comprises six chapters. Following the introduction in Chapter 1, Chapter 2 reviews the literature on the CVRP and its solution methods as well as the literature related to its three extensions (i.e., the VRPTW, the VRPSPDTW, and the 2E-MTVRPTWSS) under uncertainty. Chapter 3 studies the VRPTW with the consideration of uncertainty in customer demands, service times, and travel times. Each type of uncertainty is captured by a novel route-dependent uncertainty set. A robust mathematical formulation is presented to model the problem and an AVNS-based metaheuristic is designed to solve it. Extensive numerical experiments are conducted which adopt the Solomon's benchmark instances (Solomon, 1987). The computational results show that both deterministic and robust solutions can be generated for large-sized instances within a reasonable running time. To further assess the robustness of the obtained solutions, Monte Carlo simulation tests are designed and performed. A detailed analysis of the results generates useful managerial insights for practical VRPTW applications under multiple types of uncertainty.

Chapter 4 studies the VRPSPDTW under uncertainty in customer pickup demands and travel times. The considered two types of uncertainty are captured by two route-dependent uncertainty sets. The problem is modelled through a robust mathematical formulation and solved via an ALNS-based metaheuristic. An extensive numerical study is conducted which adopts the benchmark instances of Wang and Chen (2012) for the standard VRPSPDTW. Both deterministic and robust solutions are generated for the medium- and large-sized instances and their robustness is further evaluated by using Monte Carlo simulation tests. A comprehensive analysis of the computational results highlights the features of the proposed methods and provides routing suggestions for real-life VRPSPDTW applications under pickup demand and travel time uncertainty.

Chapter 5 studies the 2E-MTVRPTWSS under uncertainty, in which practical features in city logistics are considered, e.g., a two-echelon transportation system, time windows, multiple trips, vehicle synchronization, and uncertainty in customer demands. To capture customer demand uncertainty, a novel uncertainty set is defined based on vehicle routes in two echelons. A robust mathematical formulation with the defined uncertainty set is presented to model the problem and a VNS-based metaheuristic is proposed to solve it. Numerical experiments are conducted on adapted benchmark instances and the computational results illustrate the effectiveness of the formulation and the metaheuristic.

Finally, Chapter 6 concludes this thesis by summarizing the whole research, discussing the main findings, and identifying directions for future research. Figure 1.2 shows the structure and the main research contents of the thesis.

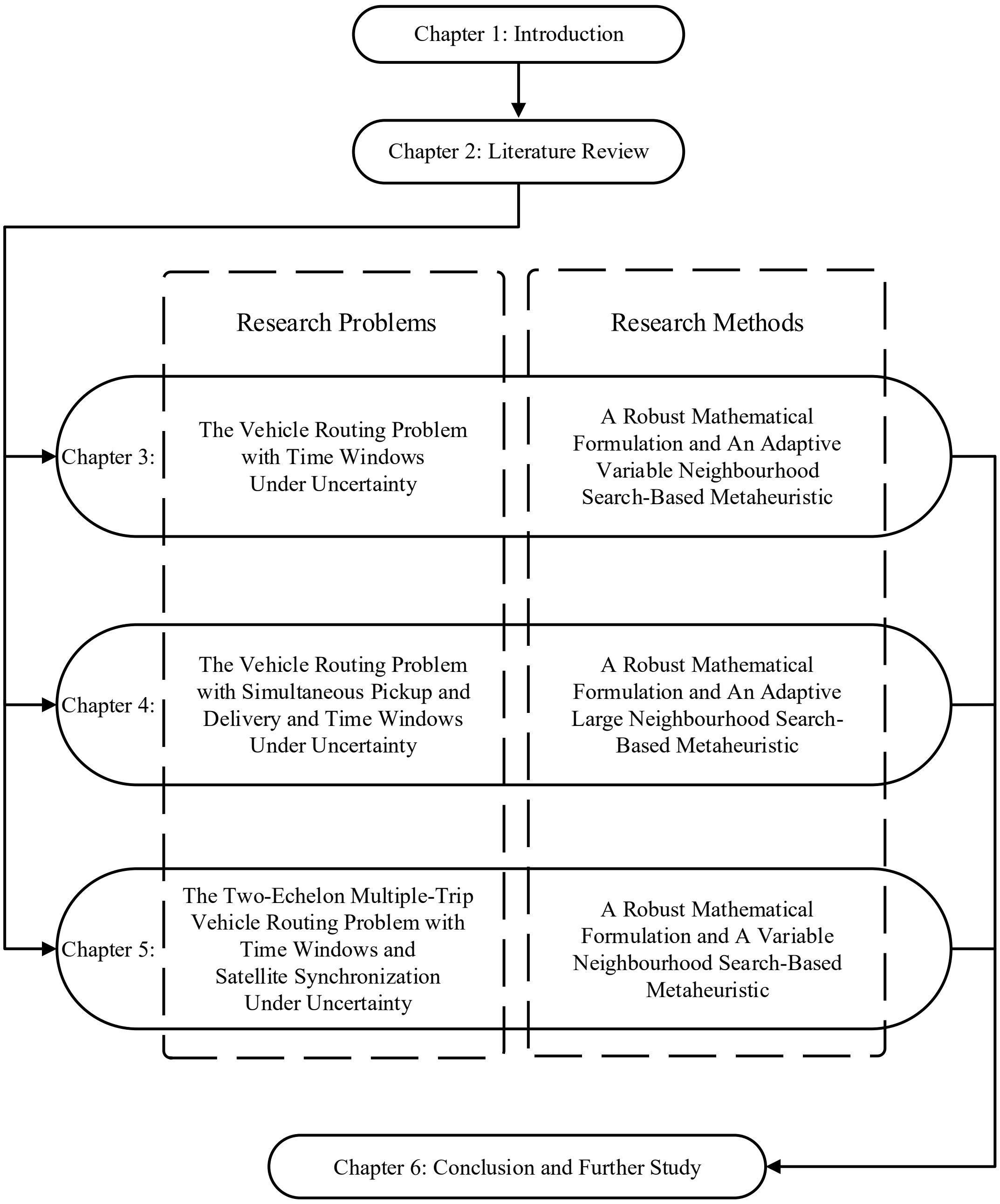

Chapter 1: Introduction
Chapter 2: Literature Review
Research Problems
Research Methods
Chapter 3:
The Vehicle Routing Problem with Time Windows Under Uncertainty
A Robust Mathematical Formulation and An Adaptive Variable Neighbourhood Search-Based Metaheuristic
Chapter 4:
The Vehicle Routing Problem with Simultaneous Pickup and Delivery and Time Windows Under Uncertainty
A Robust Mathematical Formulation and An Adaptive Large Neighbourhood Search-Based Metaheuristic
Chapter 5:
The Two-Echelon Multiple-Trip Vehicle Routing Problem with Time Windows and Satellite Synchronization Under Uncertainty
A Robust Mathematical Formulation and A Variable Neighbourhood Search-Based Metaheuristic
Chapter 6: Conclusion and Further Study

Figure 1.2 Thesis structure and main research contents.

1.6 Publications

Journal papers:

J-1. **Hu, C.**, Liu, X., and Lu, J. (2017). A bi-objective two-stage robust location model for waste-to-energy facilities under uncertainty. *Decision Support Systems*, 99: 37-50. (ERA Rank A*)

J-2. **Hu, C.**, Lu, J., Liu, X., and Zhang, G. (2018). Robust vehicle routing problem with hard time windows under demand and travel time uncertainty. *Computers & Operations Research*, 94: 139-153. (ERA Rank A)

J-3. **Hu, C.**, Liu, X., Lu, J., and Wang, C. H. (2020). Operations scheduling of waste-to-energy plants under uncertainty. *Journal of Cleaner Production*, 253: 119953. (ERA Rank A)

J-4. **Hu, C.**, Liu, X., Lu, J., and Wang, C. H. (2020). Distributionally robust optimization for power trading of waste-to-energy plants under uncertainty. *Applied Energy*, 276: 115509. (ERA Rank A)

J-5. **Hu, C.**, Lu, J., and Zhang, G. (2021). The vehicle routing problem with simultaneous pickup and delivery and time windows under multiple types of uncertainty. Working Paper. Prepare to submit to *Transportation Research Part B: Methodological*. (ERA Rank A*)

J-6. **Hu, C.**, Lu, J., and Zhang, G. (2021). The two-echelon multiple-trip vehicle routing problem with time windows and satellite synchronization under demand uncertainty. Working Paper. Prepare to submit to *Computers & Operations Research*. (ERA Rank A)

Conference papers:

C-1. **Hu, C.**, Liu, X., and Lu, J. (2019). Robust trading strategies for a waste-to-energy combined heat and power plant in a day-ahead electricity market. IFAC-PapersOnLine, 52(13):1108–1113. 9th IFAC Conference on Manufacturing Modelling, Management and Control MIM 2019.

Chapter 2

Literature Review

This chapter provides a detailed review on the related literature. Section 2.1 introduces the CVRP and its numerous exact and heuristic solution methods. Section 2.2 reviews the literature related to the VRPTW under uncertainty, including not only the standard VRPTW and its solution methods, but also the stochastic and robust versions of the VRPTW. Section 2.3 reviews the literature related to the VRPSPDTW under uncertainty. We first introduce the standard VRPSPD (alongside its various solution methods) and then the VRPSPDTW and several stochastic versions of the VRPSPD. Section 2.4 reviews the literature related to the 2E-MTVRPTWSS under uncertainty. The basic 2E-CVRP and its solution methods are reviewed, alongside several complex extensions of the 2E-CVRP with or without uncertainty. Section 2.5 summarizes this chapter and identifies research gaps.

2.1 The CVRP

The CVRP is the most studied version of the VRP. It was introduced by Dantzig and Ramser (1959) who studied a practical truck dispatching problem concerning the delivery of gasoline to gas stations. In the CVRP, a fleet of homogeneous vehicles, which are based at a central depot, must be routed to provide services for a set of customers with known locations and demands. Each vehicle should start from and return to the central depot and the load of a vehicle cannot exceed its capacity along its route. Solving the CVRP is to find a set of routes for the vehicles, such that each customer is served exactly once by only one vehicle and the total travel distance is minimized. The CVRP is classified as an NP-hard problem (Lenstra and Kan, 1981). To address the CVRP, researchers have developed many classical mathematical formulations and proposed various exact and heuristic solution methods over the last sixty years.

2.1.1 Exact Methods

Since the CVRP is a direct extension of the travelling salesman problem, many early exact solution methods for the CVRP are derived from the branch and bound (BB) algorithms designed for the travelling salesman problem (Semet et al., 2014). The basic idea of the BB scheme is to partition the set of feasible solutions into several subsets of solutions. Then, the lower and upper bounds for the subsets of solutions are systematically evaluated until the best solution is found. Christofides and Eilon (1969) developed the first BB algorithm for the CVRP and the algorithm solved two small problems with 6 and 13 customers. Later, BB algorithms were developed based on different bounding procedures. Examples of early effective BB algorithms developed for the CVRP can be found in Christofides et al. (1981), Laporte et al. (1986), and Fisher (1994).

To further improve the performance of the BB algorithms, researchers have designed branch and cut (BC) algorithms for the CVRP. Laporte et al. (1985) proposed the first BC algorithm for the CVRP based on a two-index formulation. Later, different versions of BC algorithms were developed based on new valid inequalities and separation procedures. For example, Augerat (1995) proposed a BC algorithm based on four families of valid inequalities and different branching schemes. Lysgaard et al. (2004) developed a BC algorithm that improves the algorithm of Augerat (1995). Baldacci et al. (2004) presented a BC algorithm based on a two-commodity network flow formulation. This algorithm optimally solved instances with more than 100 customers.

The BC algorithms that separate complex families of valid inequalities were the best performing algorithms for the CVRP until the introduction of the branch and cut and price (BCP) algorithms. Fukasawa et al. (2006) developed the first BCP algorithm for the CVRP, which combines cut and column generation based on a set partition and a two-index formulations. The algorithm optimally solved the benchmark instances with up to 135 vertices from the literature. Since then, the most effective exact algorithms for the CVRP are based on the combination of cut and column generation. These exact solution methods can be further classified as the BCP algorithms (Pecin et al., 2017b; Pessoa et al., 2008; Røpke, 2012) and the set partitioning-based algorithms (Baldacci et al., 2008, 2011). Baldacci et al. (2008) proposed a new exact solution method based on a set partitioning formulation with capacity and clique inequalities. Baldacci et al. (2011) improved upon Baldacci et al. (2008) by using a novel route relaxation, called ng-route, and a new strategy to effectively solve the pricing subproblem. Pecin et al. (2017b) developed a new BCP algorithm which improves and combines several important elements including route enumeration and strong branching strategies used in previous effective algorithms. The BCP algorithm can solve all benchmark instances with up to 199 customers optimally and some large-sized instances with up to 360 customers effectively. For comprehensive surveys on the exact solution methods developed for the CVRP, readers may refer to Toth and Vigo (2002), Semet et al. (2014), and Poggi and Uchoa (2014).

2.1.2 Heuristic Methods

Although effective exact algorithms have been developed for the CVRP based on different mathematical formulations, most of them have high computational complexity and only can solve instances with around 200 customers optimally. In many real-life applications, the sizes of the routing problems are very large and high-quality solutions are required in a short period of time. Thus, efficient and flexible heuristic algorithms are well suited for practical routing problems. Since Dantzig and Ramser (1959) introduced the CVRP with a simple heuristic, numerous heuristic methods have been proposed. According to Laporte et al. (2014), the heuristic solution methods for the CVRP can be broadly divided into three categories: constructive heuristics, improvement heuristics, and metaheuristics. Most of the early heuristics for the CVRP are constructive heuristics. Owing to the simplicity of the constructive rules, constructive heuristics are very fast. Several classical constructive heuristics developed for the CVRP can be found in Clarke and Wright (1964), Gillett and Miller (1974), and Renaud et al. (1996). Generally, constructive heuristics are not able to produce high-quality solutions. Thus, they are often used to generate initial solutions for improvement heuristics or metaheuristics. Improvement heuristics aim to improve initial solutions using a set of operators (moves). Researchers have designed a variety of improvement operators, such as swapping customers between different routes and exchanging the positions of customers in the same route. The existing operators can be divided into two groups: intra-route and inter-route operators. Classical intra-route operators include λ-opt (Lin, 1965) and or-opt (Or, 1976). Classical inter-route operators include relocate (Savelsbergh, 1992), 2-opt* (Potvin and Rousseau, 1995), and cyclic transfer (Thompson and Psaraftis, 1993). The operators associated with the destroy and repair scheme proposed in Shaw (1998) also can be considered as inter-route operators. These operators have been used in complex metaheuristics and proved to be very effective (Pisinger and Ropke, 2007). For more details on the constructive and improvement heuristics developed for the CVRP, readers can refer to Laporte and Semet (2002).

In addition to the constructive and improvement heuristics, effective metaheuristics have been developed for the CVRP over the last three decades. Metaheuristics are complex solution procedures that can embed classical constructive and improvement heuristics and perform a deep exploration of the solution space based on different search strategies. According to Laporte et al. (2014), the existing metaheuristics developed for the CVRP can be divided into two groups: local search metaheuristics and population-based metaheuristics. Local search metaheuristics attempt to improve a single solution by making modifications to it based on neighbourhood operators. Classical local search metaheuristics for the CVRP include tabu search (TS) (Gendreau et al., 1994; Toth and Vigo, 2003), simulated annealing (SA) (Lin et al., 2009; Osman, 1993), iterated local search (ILS) (Chen et al., 2010), variable neighbourhood search (VNS) (Kytöjoki et al., 2007), and large neighbourhood search (LNS) (Pisinger and Ropke, 2007). Population-based metaheuristics, on the other hand, attempt to improve multiple candidate solutions by selecting and combining a pool of solutions. Classical population-based metaheuristics for the CVRP include genetic algorithms (GA) (Mester and Bräysy, 2007; Nagata and Bräysy, 2009; Prins, 2004), ant colony optimization (ACO) (Bullnheimer et al., 1999; Reimann et al., 2004), particle swarm optimization (PSO) (Ai and Kachitvichyanukul, 2009a), and path relinking (PR) (Ho and Gendreau, 2006).

To further improve the performance of the metaheuristics, a variety of hybrid metaheuristics have been designed for the CVRP and its variants during the last twenty years. These hybrid metaheuristics often combine components from several standalone metaheuristics or crossbreed exact algorithms and metaheuristics to make different optimization strategies cooperate in synergy (Labadie et al., 2016). Laporte et al. (2014) discussed several important hybridization schemes including hybridization between population-based search and local search, meta-meta hybridization, hybridization with large neighbourhoods, and hybridization with exact algorithms (mathheuristics). We introduce several efficient and effective hybrid metaheuristics developed for the CVRP as follows. Osman (1993) designed a metaheuristic which combines SA with TS. Tarantilis (2005) proposed an adaptive memory programming procedure which complements local search methods by using population-based search concepts. The proposed algorithm was tested on two sets of benchmark instances and it obtained high-quality solutions within a short computation time. Pisinger and Ropke (2007) developed a well-known ALNS algorithm which employs LNS, SA, and an adaptive mechanism. Perboli et al. (2008) presented a hybrid metaheuristic which involves both GA and TS. The proposed metaheuristic is easy to implement and its performance is comparable to the previous state-of-the-art algorithms. Prins (2009) introduced a hybrid metaheuristic based on a greedy randomized adaptive search procedure (GRASP) and a local search method. Marinakis and Marinaki (2010) presented a hybrid GA-PSO algorithm. Groër et al. (2011) proposed a parallel algorithm which combines a serial metaheuristic with integer programming. The algorithm produced high-quality solutions for instances with up to 1200 nodes by using as many as 129 processors.

In recent years, due to the emergence of novel VRPs inspired from real-life applications, unified heuristic methods which can address the CVRP as well as its variants with small modifications have drawn researchers' attention. Unified heuristic methods are sufficiently flexible and can produce high-quality solutions for the CVRP and its basic variants. Examples of such methods include the unified TS heuristic (Cordeau et al., 2001), the parallel iterated TS heuristic (Cordeau and Maischberger, 2012), the ALNS heuristic (Pisinger and Ropke, 2007), and the unified hybrid genetic search heuristic (Vidal et al., 2014). Comprehensive surveys on the metaheuristics developed for the CVRP and its various extensions can be found in Laporte et al. (2014) and Labadie et al. (2016).

2.2 Literature Related to the VRPTW Under Uncertainty

2.2.1 The VRPTW

The VRPTW is one of the most important extensions of the CVRP, in which each customer needs to be served within a specified time window. Time windows can be considered as hard or soft constraints. A customer can be served before or after its time window with a penalty if soft time windows are considered. However, if time windows are hard constraints, customers must be served within their corresponding time windows. The early works on the VRPTW can date back to the 1960s (Knight and Hofer, 1968; Pullen and

Webb, 1967). Pullen and Webb (1967) studied a transport scheduling problem for the bulk conveyance of mails in the London Postal Region. Due to the need to meet trains and to schedule sorting office works, each service requirement has a time window. Knight and Hofer (1968) studied a vehicle scheduling problem with timed and connected calls for a contract transport undertaking company in the London area. As discussed in Savelsbergh (1985), it is an NP-complete problem to find a feasible solution for the VRPTW with a fixed number of vehicles. Thus, designing effective and efficient solution methods for the VRPTW has been a hot research topic over the last three decades.

Various exact solution methods have been developed for the VRPTW and most of the successful ones are based on the column generation technique (Pecin et al., 2017a). The column generation technique is an iterative procedure, which decomposes the VRPTW into a restricted master problem and a pricing subproblem. In each iteration of the procedure, new routes are first generated by solving the subproblem and then added to the restricted master problem to derive the best routes from a subset of routes. Desrochers et al. (1992) embedded the column generation technique in a BB framework and developed the first branch and price (BP) algorithm for the VRPTW. Other effective BP algorithms can be found in Feillet et al. (2004), Chabrier (2006), and Feillet et al. (2007). To further improve the lower bounds in the BB search tree, researchers have extended the BP algorithms to the BCP algorithms by dynamically adding valid inequalities to strengthen the linear relaxations used in the search tree. Kohl et al. (1999) introduced the first BCP algorithm for the VRPTW with 2-path valid inequalities. Jepsen et al. (2008) developed a BCP algorithm with novel subset-row inequalities. Recently, Pecin et al. (2017a) proposed a BCP algorithm with two enhancements including a new family of inequalities and a new form of the rank-1 inequalities. The proposed algorithm optimally solved all 56 benchmark instances with 100 customers and 50 of 61 benchmark instances with 200 customers. Other classes of exact solution methods have also been developed for the VRPTW, which include the BC algorithms (Bard et al., 2002; Lysgaard, 2006) and the set partitioning-based algorithms (Baldacci et al., 2010, 2011). For comprehensive surveys on the exact solution methods developed for the VRPTW, readers may refer to Desaulniers et al. (2010), Baldacci et al. (2012), and Desaulniers et al. (2014).

In addition to the exact algorithms, heuristic methods have been proposed to solve the VRPTW with large-sized instances. Generally, effective heuristic methods can generate good-quality solutions for large-sized instances within a short running time. Thus, they are very popular in tackling real-life VRPTW applications. Due to the close relationship between the CVRP and the VRPTW, most of the heuristic methods developed for the CVRP can be extended to address the VRPTW. Thus, the existing heuristic methods for the VRPTW also can be divided into three groups: constructive heuristics, improvement heuristics, and metaheuristics (Bräysy and Gendreau, 2005a,b). Several well-known constructive heuristics can be found in Solomon (1987), Potvin and Rousseau (1993), and Ioannou et al. (2001). Specially, Solomon (1987) introduced 56 benchmark instances with 100 customers for the VRPTW and proposed four types of constructive heuristics. However, constructive heuristics often produce low-quality solutions. Thus, improvement heuristics are widely used to refine the solutions generated by the constructive heuristics with the help of neighbourhood operators. Besides the classical intra-route and inter-route operators designed for the CVRP, other effective operators have been developed for the VRPTW. These operators include the GENI-exchange (Gendreau et al., 1992), the λ-interchange (Osman, 1993), the CROSS-exchange (Taillard et al., 1997), and the ejection chains (Glover, 1992). For a comprehensive review on the constructive and improvement heuristics developed for the VRPTW, readers can refer to Bräysy and Gendreau (2005a).

In the past two decades, most effective heuristic methods for the VRPTW are metaheuristics. Similar to the classification of the metaheuristics for the CVRP, the existing metaheuristics for the VRPTW also can be classified as local search metaheuristics and population-based metaheuristics. The classical local search metaheuristics mainly include SA (Chiang and Russell, 1996), TS (Badeau et al., 1997; Chiang and Russell, 1997; Garcia et al., 1994; Potvin et al., 1996), VNS (Bräysy, 2003), ILS (Hashimoto et al., 2008; Ibaraki et al., 2005), and LNS (Pisinger and Ropke, 2007). The classical population-based metaheuristics mainly include GA (Berger and Barkaoui, 2004; Ho et al., 2001; Homberger and Gehring, 1999; Potvin and Bengio, 1996), ACO (Gambardella et al., 1999), PSO (Marinakis et al., 2019), PR (Hashimoto and Yagiura, 2008), and scatter search (SS) (Russell and Chiang, 2006). However, the state-of-the-art heuristic methods for the VRPTW are hybrid metaheuristics which combine concepts from classical metaheuristics and exact algorithms. We briefly introduce several effective hybrid metaheuristics as follows. Bent and Van Hentenryck (2004) developed a two-stage hybrid metaheuristic for the VRPTW. Specifically, the first stage reduces the number of vehicle routes by using the basic SA and the second stage minimizes the total travel distance by using a modified LNS algorithm. Mester and Bräysy (2005) also developed a two-stage hybrid metaheuristic based on guided local search and some evolutionary strategies. The proposed algorithm was tested on an extended set of 302 benchmark instances and found the best-known solutions for most of the test instances. Hashimoto et al. (2008) presented a metaheuristic which utilizes ILS and dynamic programming. Prescott-Gagnon et al. (2009) embedded TS and a heuristic version of the exact BP technique into a LNS framework to address the VRPTW. The proposed solution method was tested on 356 benchmark instances and found 145 new best solutions. Nagata et al. (2010) presented an effective memetic algorithm which hybridizes GA with local search and a solution repair procedure. Vidal et al. (2013) developed a hybrid genetic algorithm with complex diversity control techniques for the VRPTW and some of its extensions. The algorithm outperformed all previous effective approaches on the medium-sized benchmark instances. It also solved the large-sized instances with 1000 customers efficiently. Recently, Marinakis et al. (2019) proposed a metaheuristic which combines three adaptive strategies with the PSO method. The proposed multi-adaptive PSO can produce high-quality solutions for instances with up to 1000 customers. In addition to the hybrid metaheuristics, several unified algorithms can solve the VRPTW effectively.

These algorithms can be found in Cordeau et al. (2001), Cordeau and Maischberger (2012), Pisinger and Ropke (2007), and Vidal et al. (2014). For comprehensive surveys on the metaheuristics developed for the VRPTW, readers may refer to Bräysy and Gendreau (2005b), Gendreau and Tarantilis (2010), and Desaulniers et al. (2014).

2.2.2 Stochastic Versions of the VRPTW

As uncertainty is prevalent in practical VRPTW applications, researchers have studied stochastic versions of the VRPTW under uncertainty during the last fifteen years. In most stochastic versions of the VRPTW, the critical problem parameters such as customer demands and travel times are assumed to be stochastic variables. To model these problems, two basic modelling approaches - stochastic programming with recourse (SPR) and chance-constrained programming (CCP) - are often used based on the stochastic programming paradigm (Birge and Louveaux, 2011). For instance, Li et al. (2010) studied the VRPTW with stochastic travel and service times. Both a SPR formulation and a CCP formulation were presented to model the problem. A TS-based heuristic was also designed to solve it efficiently. Russell and Urban (2008) studied a stochastic version of the VRPTW, in which soft time windows are considered and stochastic travel times for vehicles are assumed to follow the Erlang distribution. Taş et al. (2013) developed a TS metaheuristic for a VRPTW considering soft time windows and stochastic travel times. Specifically, the stochastic travel times were assumed to follow a gamma distribution in the problem. Later, the same problem was solved by an exact BP algorithm in Taş et al. (2014). Ehmke et al. (2015) studied a stochastic version of the VRPTW, in which travel times for vehicles are assumed to be stochastic and obey a normal distribution. To model the problem, a CCP formulation was proposed. Miranda and Conceição (2016) also presented a CCP formulation for the VRPTW with stochastic travel and service times. An ILS-based metaheuristic was proposed to generate high-quality solutions for the problem. Errico et al. (2016) extended the VRPTW considering only stochastic service times. A SPR formulation and a BCP algorithm were developed to address the problem. Later, the same problem was modelled with a CCP formulation and solved via a BCP algorithm in Errico et al. (2018). For a comprehensive survey on the stochastic versions of the VRPTW, readers can refer to Oyola et al. (2017) and Oyola et al. (2018).

2.2.3 Robust Versions of the VRPTW

Although stochastic versions of the VRPTW have been widely studied in the literature, these problems always make a strong assumption that the probability distributions of the stochastic parameters can be precisely known or approximated through a large number of scenarios. However, in many practical VRPTW applications, it is very difficult or even impossible to know the exact probability distributions of the critical uncertain parameters. Thus, robust optimization (Ben-Tal et al., 2009) has emerged as an alternative method to deal with the VRPTW under uncertainty. In classical robust optimization, uncertain parameters are captured by uncertainty sets, which can be derived from rough historical data or experts' experiences. Thus, the probability distributions of the uncertain parameters need not be known or estimated.

Over the last decade, researchers have paid much attention to different robust versions of the VRPTW under uncertainty. Agra et al. (2012) studied a robust VRPTW with the consideration of travel time uncertainty. A budget uncertainty set was defined to capture the uncertainty and a layered formulation was proposed to model the problem. Lee et al. (2012) considered a robust VRP with deadlines and uncertainty in customer demands and travel times. A column generation based exact solution approach was proposed for the problem. Agra et al. (2013) constructed two novel formulations for the robust VRPTW introduced in Agra et al. (2012). One extended the well-known resource inequalities formulation based on the adjustable robust optimization framework (Ben-Tal et al., 2004). The other was developed based on a path inequalities formulation. Unfortunately, both formulations can be optimally solved with only medium-sized instances. To address the robust VRPTW with large-sized instances, Braaten et al. (2017) proposed an ALNS metaheuristic which can generate good-quality solutions for instances with up to 100 customers. Hu et al. (2018) introduced a robust version of the VRPTW under demand and travel time uncertainty. Both types of uncertainty were captured by novel budget uncertainty sets and the problem was solved via a modified AVNS heuristic. Lu and Gzara (2019) studied a robust VRPTW considering only customer demand uncertainty. An exact BCP algorithm was designed for the problem and it optimally solved the test instances with up to 50 customers. Munari et al. (2019) studied a robust VRPTW considering uncertainty in travel times and customer demands. A novel compact flow-based formulation with budget uncertainty sets was developed to model the problem. An exact BCP algorithm was proposed which can optimally solve the adapted benchmark instances with up to 100 customers. De La Vega et al. (2020) considered a robust VRPTW with multiple deliverymen under three types of uncertainty. The authors extended the mathematical formulation proposed in Munari et al. (2019) to model the problem and designed a BCP algorithm to solve it. Bartolini et al. (2021) studied a robust travelling salesman problem with time windows considering uncertain travel times. A novel knapsack-constrained uncertainty set was designed to capture the uncertainty. In addition, an exact algorithm which adopts dynamic programming and column generation was proposed to solve the problem. Zhang et al. (2021) studied a robust data-driven VRPTW under travel time uncertainty. To address the problem, a distributionally robust optimization model with a Wasserstein distance-based ambiguity set was developed. In addition, both an exact BC algorithm and a VNS metaheuristic were designed.

2.3 Literature Related to the VRPSPDTW Under Uncertainty

2.3.1 The VRPSPD

The VRPSPD is another important extension of the CVRP. It was introduced by Min (1989) thirty years ago. Min (1989) studied the VRPSPD which is inspired from a library material distribution problem facing a public library in Franklin County, Ohio. In the VRPSPD, each customer has a delivery demand and a pickup demand which need to be satisfied simultaneously. Compared to the CVRP and the VRPTW, the VRPSPD has received less attention in the literature. However, due to its relevance to reverse logistics and importance in real-life transportation systems, the VRPSPD has been extensively studied over the last fifteen years. Recently, Koç et al. (2020) provided a comprehensive survey on the VRPSPD and its important variants.

As an extension of the CVRP, the VRPSPD is also an NP-hard problem. Only a few researchers have built mathematical formulations and developed exact solution methods to address the VRPSPD in the literature. Dell'Amico et al. (2006) developed a BP algorithm based on a two-index one-commodity flow formulation. Subramanian et al. (2010b) presented both an undirected and a directed two-index two-commodity flow formulations for the VRPSPD. BC algorithms were also designed based on these two formulations. Subramanian et al. (2011) proposed a BC algorithm based on a two-index vehicle-flow formulation. The proposed algorithm was tested on benchmark instances for the VRPSPD and it generated several best-known lower bounds. Subramanian et al. (2013b) later proposed a BCP algorithm based on the same formulation presented in Subramanian et al. (2011). The algorithm was extensively tested on the benchmark instances of Salhi and Nagy (1999), Nagy and Salhi (2005), and Montané and Galvao (2006). Computational results showed that it improved some best-known lower bounds for instances with up to

200 customers and identified the optimal solutions for instances with up to 100 customers. Rieck and Zimmermann (2013) developed a commodity-flow formulation and a vehicle-flow formulation along with a BC algorithm to address the VRPSPD.

In addition to the exact methods, researchers have designed efficient heuristic methods to solve the VRPSPD with large-sized instances. According to Koç et al. (2020), the heuristic methods for the VRPSPD can be divided into three categories: classical constructive and improvement heuristics, local search metaheuristics, and population-based metaheuristics. Early research works of the VRPSPD have mainly focused on the constructive and improvement heuristics. For instance, Salhi and Nagy (1999) extended several insertion-based heuristics and proposed a cluster insertion heuristic. Dethloff (2001) proposed an insertion heuristic which combines different insertion criteria including radial surcharge, residual capacity, and travel distance. Nagy and Salhi (2005) developed a four-phase heuristic which modifies and integrates the popular heuristics designed for the CVRP.

Compared to constructive and improvement heuristics, metaheuristics generally yield better solutions as they can employ effective strategies to guide the search process. Most existing local search metaheuristics for the VRPSPD are based on pure or hybrid version of the TS framework. For example, Montané and Galvao (2006) proposed a TS metaheuristic which utilizes four types of neighbourhood operators in the local search. Chen and Wu (2006) designed a hybrid TS algorithm based on a record-to-record travel method and several classical route improvement procedures. Bianchessi and Righini (2007) developed a new solution approach which integrates VNS into the TS framework. Wassan et al. (2008) later proposed a reactive TS heuristic, in which neighbourhood moves can be checked efficiently and the local search process can be guided effectively. The proposed method was tested on a number of benchmark instances ranging from 50 to 199 customers and it produced 10 new best solutions. Zachariadis et al. (2009) developed a hybrid algorithm which combines TS and guided local search. Besides the TS-based metaheuristics, other effective local search metaheuristics have been proposed for the VRPSPD. Subramanian et al. (2010a) designed a parallel heuristic which employs a multi-start scheme and an ILS procedure. The proposed method obtained 15 new best solutions for the benchmark instances of Salhi and Nagy (1999) and Montané and Galvao (2006). Zachariadis and Kiranoudis (2011) proposed a local search metaheuristic, which can efficiently examine rich solution neighbourhoods and effectively explore solution space by using special data structures and diversification strategies. Subramanian et al. (2013a) proposed a hybrid metaheuristic for a class of VRPs including the VRPSPD. The metaheuristic combines the ILS framework with an exact procedure based on a set partition formulation and several enhanced strategies. Avci and Topaloglu (2015) presented a metaheuristic which combines SA and the variable neighbourhood descend method. An adaptive threshold function was designed to make the algorithm self-tuning. Recently, Hof and Schneider (2019) developed a metaheuristic which hybridizes the ALNS framework with a PR procedure for the VRPSPD and its several variants. The proposed method was tested on extensive benchmark instances from the literature. The computational results showed that the metaheuristic was able to identify the current best-known solutions for most of the benchmark instances and even generate new best solutions for many instances.

Population-based metaheuristics are another important type of metaheuristics developed for the VRPSPD. Effective population-based metaheuristics mainly include ACO (Çatay, 2010; Gajpal and Abad, 2009), PSO (Ai and Kachitvichyanukul, 2009b; Gan et al., 2012), and GA (Tasan and Gen, 2012; Vidal et al., 2014). To improve the performance of the population-based metaheuristics, researchers often hybridize them with local search metaheuristics. For example, Gajpal and Abad (2009) designed an ACO algorithm that utilizes the nearest neighbour heuristic in the initial solution construction process and two multi-route local search schemes in the route improvement process. The proposed method outperformed several previous effective algorithms and found new best solutions for some benchmark instances of Salhi and Nagy (1999) and Dethloff (2001). Çatay (2010) presented a hybrid metaheuristic that combines ACO and a local search procedure. Kalayci and Kaya (2016) also presented a hybrid metaheuristic that combines ACO and VNS. High-quality solutions were generated for the benchmark instances using the proposed algorithm. Ai and Kachitvichyanukul (2009b) designed a PSO algorithm which adopts a novel solution representation and decoding procedure. Zachariadis et al. (2010) proposed an adaptive memory programming approach for the VRPSPD. This approach is very effective as it can collect and combine promising solution features in the search process. Tasan and Gen (2012) presented a GA with direct genetic representation for encoding. Vidal et al. (2014) proposed a unified hybrid genetic search heuristic for multi-attribute VRPs including the VRPSPD. The best-known solutions for most of the benchmark instances of Salhi and Nagy (1999) and Dethloff (2001) were identified by the unified heuristic.

2.3.2 The VRPSPDTW

If customers have time windows in the VRPSPD, the VRPSPD becomes the VRPSPDTW. Compared to the VRPSPD, the VRPSPDTW has received less attention in the literature. Angelelli and Mansini (2002) were the first to study the VRPSPDTW and proposed the only exact solution approach based on a set covering mathematical formulation. Unfortunately, the proposed algorithm can solve instances only with small sizes. To solve the VRPSPDTW with large-sized instances, a variety of heuristic methods have been developed. Mingyong and Erbao (2010) designed an improved differential evolution algorithm with a new decimal coding strategy. The proposed algorithm can generate good-quality solutions for instances with 40 customers. Wang and Chen (2012) introduced the benchmark instances for the VRPSPDTW by adapting the well-known Solomon's instances (Solomon, 1987). A co-evolution GA was developed to solve all benchmark instances. Liu et al. (2013) developed a hybrid metaheuristic which combines GA and TS to address a VRPSPDTW in the context of home healthcare services. Kassem and Chen (2013) studied a VRPSPDTW in a closed-loop logistics network. A solution approach was designed which employs a sequential route construction technique and the SA mechanism. Wang et al. (2013)

presented a SA-based heuristic with a randomized local search method and a slow cooling schedule for the VRPSPDTW. Later, Wang et al. (2015) proposed a parallel SA heuristic and designed 30 new large-sized instances for the VRPSPDTW. The heuristic showed a good performance on the benchmark instances of Wang and Chen (2012) as well as the newly-designed instances. Recently, Shi et al. (2020) presented a lexicographical-based two-stage algorithm for the VRPSPDTW. In the algorithm, a modified VNS heuristic is designed to minimize the number of vehicle routes in the first stage and a bi-structure based TS heuristic is proposed to minimize the travel distance in the second stage. The two-stage algorithm was tested on benchmark instances from the literature. The computational results showed that it was able to find the best-known solutions or better ones for most of the benchmark instances.

2.3.3 Stochastic Versions of the VRPSPD

In many practical VRPSPD applications, important parameters such as pickup demands of customers and travel times for vehicles can be uncertain. Compared to the standard VRPSPD, the VRPSPD under uncertainty has seldom been studied in the literature. Only a few researchers have addressed several stochastic versions of the VRPSPD. Hou and Zhou (2010) studied a stochastic version of the VRPSPD, in which the delivery demands of customers and the travel times between any two customers for vehicles are assumed to follow normal distributions. Wang and Qiu (2011) presented a cross-entropy method for the VRPSPD with stochastic delivery demands. Zhang et al. (2012) studied a stochastic version of the VRPSPD, in which the stochastic travel time associated with each arc is assumed to follow a normal distribution. A SS-based solution approach with a new constrained programming method was proposed to solve the problem. Shi et al. (2018) studied a VRPSPD in the context of home healthcare services considering stochastic travel and service times. A SPR model and a SA-based heuristic solution method were proposed to address the problem. Shi et al. (2019) further studied a robust version of the VRPSPD considering uncertainty in service and travel times. Three heuristic methods including a SA algorithm, a TS algorithm, and a VNS algorithm were designed to solve the problem.

2.4 Literature Related to the 2E-MTVRPTWSS Under Uncertainty

2.4.1 The 2E-CVRP

Due to the popularity of two-echelon transportation systems in supply chains and city logistics, researchers have extended the CVRP to the 2E-CVRP, in which the freight transportation passes through intermediate facilities (called satellites) in a two-echelon network. The literature related to the 2E-CVRP dates back to Jacobsen and Madsen (1980) who studied a problem of distributing newspapers with a two-echelon network and intermediate transfer points. Three fast heuristics designed for a location-routing problem were employed and modified to address the problem. Later, Crainic et al. (2004) introduced a two-echelon transportation system with satellite platforms to optimize freight transportation in congested urban areas. Due to the continuously increasing activities in city logistics, Crainic et al. (2009) introduced a two-echelon, multi-depot, multi-tour, synchronized, heterogeneous vehicle routing problem with time windows to evaluate and plan city logistics systems. The problem can be considered as a complex extension of the standard 2E-CVRP. Perboli et al. (2011) formally introduced the 2E-CVRP with a three-index flow-based mathematical formulation. Since then, both exact and heuristic methods have been developed to address the problem.

Owing to its complexity, only a few researchers have attempted to solve the 2E-CVRP via exact methods. Gonzalez-Feliu (2008) proposed a BC algorithm based on a commodity-flow formulation. However, the proposed algorithm cannot generate high-quality solutions for instances with more than 50 customers. Perboli et al. (2011) improved the BC algorithm proposed in Gonzalez-Feliu (2008) based on a flow-based formulation and two families of valid inequalities. However, according to Jepsen et al. (2013), this formulation may fail to provide the correct upper bounds for solutions when instances have more than three satellites. Jepsen et al. (2013) designed a BC algorithm based on an edge flow formulation. The algorithm incorporated an effective branching scheme to obtain feasible solutions and was tested on instances with up to 50 customers and five satellites. The computational results showed that it was able to provide tight lower bounds for the obtained solutions. Baldacci et al. (2013) developed an exact algorithm for the 2E-CVRP based on a novel mathematical formulation. The algorithm first decomposes the 2E-CVRP into a limited set of multi-depot VRPs with side constraints and then solves the resulting problems to derive the optimal solution for the 2E-CVRP. Extensive computational experiments were conducted on benchmark instances. The results showed that the proposed algorithm outperformed the previous exact methods for the 2E-CVRP in terms of solution quality, computation time, and instance size. Santos et al. (2014) developed a BCP algorithm for the 2E-CVRP based on a mathematical formulation with several classes of valid inequalities. Perboli et al. (2018) presented a BC algorithm based on the mathematical formulation in Perboli et al. (2011) with new valid inequalities. Computational results showed that the BC algorithm optimally solved most of the benchmark instances adopted from the literature and successfully reduced the optimality gaps for the remaining instances. Recently, Dellaert et al. (2019) developed two path-based mathematical formulations for a basic variant of the 2E-CVRP, which incorporates time window constraints. BP-based algorithms were designed based on the two formulations and they were tested on a comprehensive set of instances. The results showed that the BP-based algorithms were able to generate the optimal solutions for the instances with up to five satellites and 100 customers.

Although the current best exact algorithms can optimally solve instances with up to 100 customers and five satellites, they are often time consuming and may be limited in generating high-quality solutions for instances with real-world sizes. Thus, various efficient and effective heuristic methods have been developed for the 2E-CVRP. Crainic et al. (2008) proposed several metaheuristics based on a two-phase procedure where a feasible solution is generated in the first phase and then improved by several heuristics in the second phase. Specifically, the procedure separates the first- and second-echelon routing problems and solves them sequentially in the first phase to obtain feasible solutions. Following the concept of separating the routing problems in two echelons introduced in Crainic et al. (2008), Crainic et al. (2011) designed a family of multi-start metaheuristics. In each iteration of the multi-start procedure, an initial solution is first generated by sequentially solving the two resulting problems and then improved by a local search method. Perboli et al. (2011) proposed a mathematical formulation and designed two heuristic solution methods for the 2E-CVRP. Hemmelmayr et al. (2012) introduced an ALNS algorithm that employs different kinds of destroy and repair operators. The algorithm was tested on three sets of benchmark instances for the 2E-CVRP and a new set of adapted large-scale instances. The results showed that the ALNS algorithm produced good-quality solutions for the instances with up to 10 satellites and 100 customers. Crainic et al. (2013) proposed a hybrid metaheuristic which combines a GRASP with the PR approach for the 2E-CVRP. However, its performance is not as good as that of the ALNS algorithm proposed in Hemmelmayr et al. (2012). Zeng et al. (2014) also presented a hybrid metaheuristic that combines a GRASP for the initial solution construction process and a variable neighbourhood descend method for the solution improvement process. The algorithm was tested on three sets of benchmark instances and it outperformed the previous effective heuristic methods for the 2E-CVRP in terms of both solution quality and computation time. Breunig et al. (2016) developed a simple and fast metaheuristic for several classes of two-echelon routing problems including the 2E-CVRP. The algorithm employed a local search procedure, LNS methods, and some tailored operators. Extensive computational experiments were conducted and the proposed algorithm found the best-known solutions or better ones for almost all benchmark instances adopted from the literature.

2.4.2 Complex Extensions of the 2E-CVRP

Since real-life routing problems with two-echelon transportation networks always exhibit practical features such as multiple trips and vehicle synchronization requirements, various extensions of the standard 2E-CVRP have been studied in the literature over the last decade. Perboli et al. (2011) introduced five basic variants of the 2E-CVRP, which are the 2E-CVRP with time windows, the 2E-CVRP with satellite synchronization, the multi-depot 2E-CVRP, the 2E-CVRP with pickups and deliveries, and the 2E-CVRP with taxi services. In this section, we review several complex extensions of the 2E-CVRP, which consider special features such as time windows, multiple trips, vehicle synchronization, and uncertainty. Crainic et al. (2015) studied a very complex extension of the 2E-CVRP that simultaneously considers multiple depots, multiple trips, vehicle synchronization, demand uncertainty, and time windows. Customer demands were assumed to be stochastic parameters and a two-stage SPR formulation was developed to model the problem. However, the formulation can be solved with only small-sized instances owing to its complexity. Grangier et al. (2016) extended the 2E-CVRP by considering time windows, synchronization constraints, and multiple trips. An ALNS algorithm with several newly-designed destroy and repair operators was developed and it solved the problem with large-sized instances. Anderluh et al. (2016) studied a 2E-CVRP with satellite synchronization arising in a two-echelon freight distribution network where vans and cargo bikes need to synchronize at intermediate transfer points. A metaheuristic which combines a GRASP and the PR technique was proposed to solve the problem. Li et al. (2016b) studied a two-echelon time-constrained CVRP in linehaul-delivery systems. A heuristic method was proposed which adopts the Clarke and Wright saving heuristic for the initial solution construction process and a local search method for the solution improvement process. Li et al. (2016a) later extended the problem studied in Li et al. (2016b) by considering carbon dioxide emissions. Masson et al. (2017) addressed a 2E-CVRP with satellite synchronization arising in a novel distribution system with mixed passengers and goods. An ALNS algorithm was developed and tested on a real-world instance in the city of La Rochelle in France. Liu et al. (2017) extended the 2E-CVRP by considering stochastic customer demands. Monte Carlo sampling was adopted to deal with stochastic demands and a TS algorithm was designed to solve the problem. Wang et al. (2017) extended the 2E-CVRP by considering stochastic demands and split delivery. A two-stage SPR formulation and a GA-based solution approach were developed to address the problem. Li et al. (2018) studied a 2E-CVRP with time windows and varying real-time transshipment capacity. A two-stage heuristic solution approach was designed based on the Clarke and Wright saving heuristic and the VNS framework. Recently, Anderluh et al. (2019) employed Monte Carlo simulation to evaluate the impact of travel time uncertainty in a 2E-CVRP with synchronization constraints. Li et al. (2020) introduced a 2E-CVRP with time windows and mobile satellites arising in parcel deliveries with unmanned aerial vehicles and human-driven vans. To address the problem, a vehicle-flow formulation and an ALNS algorithm were developed. For a comprehensive survey on the 2E-CVRP and its extensions, readers may refer to Cuda et al. (2015).

2.5 Summary

This chapter comprehensively reviews the literature on the CVRP and its solution methods as well as the research works related to its three extensions (i.e., the VRPTW, the VRP-SPDTW, and the 2E-MTVRPTWSS) under uncertainty. For the CVRP, progress has been made over the past two decades. The state-of-the-art exact algorithms for the CVRP can optimally solve the benchmark instances with up to 200 customers while the corresponding best metaheuristics are able to produce high-quality solutions for instances with around 500 customers. Specifically, unified heuristic methods that can address different extensions of the CVRP with slight modifications have grown increasingly popular.

As one of the most important extensions of the CVRP, the VRPTW has received considerable attention, for which numerous exact and heuristic methods have been developed.

In the literature, the state-of-the-art exact methods for the VRPTW can optimally solve the Solomon's benchmark instances within a reasonable computing time. However, it is still computationally expensive for the exact algorithms to solve instances with more than 200 customers. Metaheuristics have been established as the most effective for the VRPTW during the last two decades. The state-of-the-art metaheuristics for the VRPTW are able to generate high-quality solutions for instances with up to 500 customers. Compared to the VRPTW itself, the stochastic and robust versions of the VRPTW under uncertainty have received little attention in the literature. Most stochastic and robust versions of the VRPTW consider only one or two types of uncertainty. In addition, the proposed solution methods for these problems are often unable to handle large-sized instances. Thus, to address the VRPTW under multiple types of uncertainty, novel mathematical models and solution methods are warranted.

The VRPSPDTW is the generalization of the VRPTW and the VRPSPD. As another important extension of the CVRP, the VRPSPD has likewise garnered attention. Most of its solution methods are heuristics and metaheuristics. Metaheuristics often generate superior solutions though at the expense of additional computational time. However, to the best of our knowledge, little literature has focused on the VRPSPDTW and stochastic versions of the VRPSPD, and almost none on the VRPSPDTW under multiple types of uncertainty. Thus, this thesis aims to address the VRPSPDTW under different types of uncertainty by developing mathematical formulations and designing metaheuristic solution methods.

The 2E-MTVRPTWSS is a complex extension of the 2E-CVRP. For the 2E-CVRP, the literature has focused on its standard version alongside its exact and metaheuristic solution methods. Most exact algorithms for the 2E-CVRP can solve instances with up to only 50 customers and five satellites, while the metaheuristics are able to address relatively large instances. This therefore warrants the design of metaheuristics that can handle instances with real-world sizes. Notwithstanding the focus on extensions of the 2E-CVRP such as the 2E-CVRP with time windows and the 2E-CVRP with satellite synchronization, few studies have extended the 2E-CVRP by simultaneously considering practical features including vehicle synchronization, multiple trips, time windows, and uncertainty in customer demands. Thus, this thesis aims to develop mathematical formulations and design metaheuristic approaches to address the 2E-MTVRPTWSS under demand uncertainty.

Chapter 3

Vehicle Routing Problem with Time Windows Under Uncertainty

3.1 Introduction

In contemporary logistics and transportation services, customers' expectation that their demands are satisfied within specific time slots translates into practical time window constraints in activities such as parcel delivery, bank delivery, and municipal solid waste collection. Accordingly, the VRPTW which extends the CVRP by incorporating time windows constraints has been a hot research topic over the last several decades. A crucial assumption in the VRPTW is that the values of all parameters are deterministic and can be precisely known in advance. However, this assumption does not always hold in many real-life VRPTW applications due to uncertainty, which could be secondary to factors such as imprecise information on customer demands and varying travel times for vehicles. For instance, when planning the vehicle routes for municipal solid waste collection, it is difficult to anticipate the amount of waste at each collection point. In addition, uncertainty in travel times may arise from variable traffic conditions and drivers' driving skills. Uncertainty is thus prevalent in practical VRPTW applications. Routing plans generated without considering uncertainty may ultimately be found infeasible, resulting in poor service and customer dissatisfaction.

In this chapter, we study the VRPTW under three types of uncertainty. Specifically, we consider uncertainty in customer demands, service times, and travel times. To model these three types of uncertainty, three uncertainty sets are defined with novel route-dependent uncertainty polytopes. Based on such sets, we present a robust mathematical formulation to model the problem and propose an AVNS-based metaheuristic to solve it. To demonstrate the effectiveness of the proposed methods, we adopt the well-known Solomon's benchmark instances (Solomon, 1987) and conduct extensive numerical experiments.

The rest of this chapter is organized as follows. Section 3.2 first introduces the VRPTW and presents a deterministic mathematical formulation. Then, the VRPTW under three types of uncertainty is described and the robust mathematical formulation with the novel route-dependent uncertainty sets is presented. Section 3.3 describes the details of the AVNS-based metaheuristic. Section 3.4 presents the computational results from the numerical experiments with a comprehensive analysis. Finally, Section 3.5 summarizes the whole chapter.

3.2 Problem Statement and Model Formulation

3.2.1 The VRPTW

The VRPTW can be described as follows. There are a set of geographically dispersed customers and a fleet of homogeneous vehicles based at a central depot. Each customer has a given demand which must be satisfied by a vehicle within a specified time window. Each vehicle starts and ends at the central depot. Moreover, the load of a vehicle cannot exceed its capacity along its route. Solving the VRPTW is to find a set of routes for the vehicles, such that each customer is served exactly once, the capacity and time window constraints for the vehicles are satisfied, and the total routing cost (in terms of the number of vehicles used and the total travel distance) is minimized.

The above described problem can be defined on a directed graph $\mathcal{G} = (\mathcal{N}, \mathcal{A})$, where $\mathcal{N} = \{0\} \cup \mathcal{C} \cup \{n+1\}$ is a set of nodes and $\mathcal{A} = \{(i,j)|i,j \in \mathcal{C}, i \neq j\} \cup \{(0,j)|j \in \mathcal{C}\} \cup \{(i,n+1)|i \in \mathcal{C}\}$ is a set of arcs. Set $\mathcal{C}$ represents a set of customers and $\mathcal{C} = \{1, \cdots, n\}$. Nodes 0 and $n+1$ denote the same central depot for modelling purposes. A fleet of homogeneous vehicles is denoted by set $\mathcal{K}$. Each vehicle $k \in \mathcal{K}$ has a capacity Q. A travel time t_{ij} and a travel distance d_{ij} are associated with each arc $(i,j) \in \mathcal{A}$. Each node $i \in \mathcal{N}$ has a demand q_i, such that $q_i > 0$ for each $i \in \mathcal{C}$ and $q_0 = q_{n+1} = 0$. In addition, each node $i \in \mathcal{N}$ has a time window $[a_i, b_i]$. a_i and b_i respectively denote the earliest and latest start-of-service times of node i. If a vehicle arrives at node i before a_i, it must wait until a_i to start servicing the node. If it arrives after b_i, node i cannot be served. A service time s_i is also associated with each node $i \in \mathcal{N}$, such that $s_i > 0$ for each $i \in \mathcal{C}$ and $s_0 = s_{n+1} = 0$. A hierarchical objective is considered in the VRPTW. It consists of minimizing the number of vehicles used (primary criterion) and the total travel distance (secondary criterion).

To model the VRPTW, we present a mathematical formulation with two types of variables based on the mixed-integer programming formulation for the VRPTW introduced in Desaulniers et al. (2014). The first are binary variables x_{ij}^k for all $(i,j) \in \mathcal{A}$ and $k \in \mathcal{K}$. $x_{ij}^k = 1$ if vehicle $k \in \mathcal{K}$ traverses arc $(i,j) \in \mathcal{A}$ and 0 otherwise. The second are continuous variables y_i^k for all $i \in \mathcal{N}$ and $k \in \mathcal{K}$. y_i^k specifies the arrival time or the start of service time of vehicle $k \in \mathcal{K}$ at node $i \in \mathcal{N}$. Based on the defined variables, the deterministic mathematical formulation for the VRPTW can be expressed as follows:

$$\text{(VRPTW)} \quad \textit{lex-}\min \left(\sum_{k \in \mathcal{K}} \sum_{j \in \mathcal{C}} x_{0j}^k, \sum_{k \in \mathcal{K}} \sum_{(i,j) \in \mathcal{A}} d_{ij} x_{ij}^k \right) \tag{3.1}$$

$$\text{s.t.} \sum_{j \in \mathcal{C}} x_{0j}^k \leq 1 \qquad \forall k \in \mathcal{K}, \tag{3.2}$$

$$\sum_{j \in \mathcal{C}} x_{0j}^k = \sum_{i \in \mathcal{C}} x_{i(n+1)}^k \qquad \forall k \in \mathcal{K}, \tag{3.3}$$

$$\sum_{i:(i,j)\in\mathcal{A}} x_{ij}^k - \sum_{i:(j,i)\in\mathcal{A}} x_{ji}^k = 0 \qquad \forall k \in \mathcal{K}, j \in \mathcal{C}, \tag{3.4}$$

$$\sum_{k\in\mathcal{K}} \sum_{i:(i,j)\in\mathcal{A}} x_{ij}^k = 1 \qquad \forall j \in \mathcal{C}, \tag{3.5}$$

$$\sum_{(i,j)\in\mathcal{A}} q_i x_{ij}^k \leq Q \qquad \forall k \in \mathcal{K}, \tag{3.6}$$

$$y_i^k + t_{ij} + s_i \leq y_j^k + M(1 - x_{ij}^k) \qquad \forall k \in \mathcal{K}, (i,j) \in \mathcal{A}, \tag{3.7}$$

$$a_i \leq y_i^k \leq b_i \qquad \forall k \in \mathcal{K}, i \in \mathcal{N}, \tag{3.8}$$

$$x_{ij}^k \in \{0,1\} \qquad \forall k \in \mathcal{K}, (i,j) \in \mathcal{A}. \tag{3.9}$$

Objective function (3.1) lexicographically minimizes the number of vehicles used and the overall travel distance. Constraints (3.2)-(3.3) ensure that each used vehicle starts from and returns to its depot. Constraints (3.4) are the flow conservation constraints. Constraints (3.5) guarantee that each customer is served exactly once by only one vehicle. Constraints (3.6) denote the vehicle capacity constraints. Constraints (3.7) and (3.8) ensure that the time window constraints associated with the customers and the depot nodes are satisfied. Note that the value of y_i^k is meaningless whenever vehicle $k \in \mathcal{K}$ does not visit node $i \in \mathcal{N}$. In addition, M is an arbitrary large constant. According to Desaulniers et al. (2014), a reasonable value for M can be set to $\max_{(i,j)\in\mathcal{A}}\{b_i - a_j + s_i + t_{ij}\}$. A larger value for M will not affect the optimal solution of the above deterministic formulation for the VRPTW. However, the optimal solution of the formulation may be affected if M is set less than the reasonable value $\max_{(i,j)\in\mathcal{A}}\{b_i - a_j + s_i + t_{ij}\}$. Finally, constraints (3.9) impose the domain of the decision variables x_{ij}^k.

3.2.2 The VRPTW Under Uncertainty

As introduced in Section 3.1, data uncertainty is very popular in many real-life VRPTW applications. Routing strategies derived from the above deterministic mathematical formulation without considering uncertainty may ultimately be found infeasible and cannot be perfectly executed in practical situations. Thus, we study the VRPTW with the consideration of uncertainty in three critical parameters including customer demands, service times, and travel times. To model the problem, we adopt the adjustable robust optimization framework (Ben-Tal et al., 2009, 2004) and develop a robust mathematical formulation. Novel route-dependent uncertainty sets are defined to capture the three types of uncertainty. Before introducing the details of the uncertainty sets and the robust formulation, a short discussion is needed.

In practical VRPTW applications, each vehicle is very likely to visit and serve a different subset of customers. Thus, the level of uncertainty experienced by each vehicle is different and may be related to the number of customers on its planned route. For instance, if a vehicle needs to visit more customers based on its planned route, it may suffer higher uncertainty in customer demands and service times. Similarly, if a vehicle needs to travel a longer distance to serve all customers on its route, it may subject to higher uncertainty in travel times. Moreover, it is unlikely that every customer will have large demand variations and every route segment between two customers will have substantial travel time variations on a planned route. In real-life applications, it is more likely that only some customers have large variations in demands and few route segments have large variations in travel times while others do not. Thus, we can restrict the number of customers with large variations in demands and service times and the number of route segments with substantial variations in travel times on a planned route to control its robustness.

Based on the above discussion, we define three types of uncertainty polytopes associated with the route r_k of each used vehicle $k \in \mathcal{K}'$: the demand uncertainty polytope $\mathcal{U}_q^k$, the service time uncertainty polytope $\mathcal{U}_s^k$, and the travel time uncertainty polytope $\mathcal{U}_t^k$. $\mathcal{K}' \subseteq \mathcal{K}$ denotes the set of used vehicles. These polytopes are budget uncertainty polytopes which were initially introduced in Bertsimas and Sim (2003, 2004). To capture these three types of uncertainty, we define the customer demand uncertainty set $\mathcal{U}_q$ with polytopes $\mathcal{U}_q^k$, the service time uncertainty set $\mathcal{U}_s$ with polytopes $\mathcal{U}_s^k$, and the travel time uncertainty set

$\mathcal{U}_t$ with polytopes $\mathcal{U}_t^k$ in the following equations:

$$\mathcal{U}_q = \bigcap_{k \in \mathcal{K}'} \mathcal{U}_q^k \tag{3.10}$$

with

$$\mathcal{U}_q^k = \left\{ \tilde{q} \in \mathbb{R}^{|\mathcal{C}|} | \tilde{q}_i = \overline{q}_i + \alpha_i^k \hat{q}_i, |\alpha_i^k| \leq 1, \forall i \in \mathcal{C}; \sum_{i \in \mathcal{C}^k} |\alpha_i^k| \leq \Gamma_q^k, \Gamma_q^k = \lceil \theta_q |\mathcal{C}^k| \rceil \right\} \tag{3.11}$$

$$\mathcal{U}_s = \bigcap_{k \in \mathcal{K}'} \mathcal{U}_s^k \tag{3.12}$$

with

$$\mathcal{U}_s^k = \left\{ \tilde{s} \in \mathbb{R}^{|\mathcal{C}|} | \tilde{s}_i = \overline{s}_i + \beta_i^k \hat{s}_i, |\beta_i^k| \leq 1, \forall i \in \mathcal{C}; \sum_{i \in \mathcal{C}^k} |\beta_i^k| \leq \Gamma_s^k, \Gamma_s^k = \lceil \theta_s |\mathcal{C}^k| \rceil \right\} \tag{3.13}$$

$$\mathcal{U}_t = \bigcap_{k \in \mathcal{K}'} \mathcal{U}_t^k \tag{3.14}$$

with

$$\mathcal{U}_t^k = \left\{ \tilde{t} \in \mathbb{R}^{|\mathcal{A}|} | \tilde{t}_{ij} = \overline{t}_{ij} + \gamma_{ij}^k \hat{t}_{ij}, |\gamma_{ij}^k| \leq 1, \forall (i,j) \in \mathcal{A}; \sum_{(i,j) \in \mathcal{A}^k} |\gamma_{ij}^k| \leq \Gamma_t^k, \Gamma_t^k = \lceil \theta_t |\mathcal{A}^k| \rceil \right\}. \tag{3.15}$$

In equation (3.10), the demand uncertainty set $\mathcal{U}_q$ is the intersection of polytopes $\mathcal{U}_q^k$ for all $k \in \mathcal{K}'$. Uncertainty polytope $\mathcal{U}_q^k$ in equation (3.11) is defined based on the route r_k of vehicle $k \in \mathcal{K}'$. It captures the possible demand uncertainty experienced by a used vehicle. In polytope $\mathcal{U}_q^k$, $\tilde{q}_i$ denotes the uncertain demand of customer i and vector $\tilde{q}$ subsumes $\tilde{q}_i$ for all $i \in \mathcal{C}$. $\tilde{q}_i$ is expressed as $\overline{q}_i + \alpha_i^k \hat{q}_i$, where $\overline{q}_i$ denotes the nominal value of $\tilde{q}_i$ and $\hat{q}_i$ denotes the maximum deviation of $\tilde{q}_i$ from $\overline{q}_i$. α_i^k is an auxiliary variable associated with $\tilde{q}_i$ and it takes values in the interval $[-1,1]$. If $\alpha_i^k = -1$, $\tilde{q}_i = \overline{q}_i - \hat{q}_i$. If $\alpha_i^k = 1$, $\tilde{q}_i = \overline{q}_i + \hat{q}_i$. Thus, uncertain customer demand $\tilde{q}_i$ actually takes values in the interval $[\overline{q}_i - \hat{q}_i, \overline{q}_i + \hat{q}_i]$ for each $i \in \mathcal{C}$. However, due to the existence of the inequality $\sum_{i\in\mathcal{C}^k} |\alpha_i^k| \leq \Gamma_q^k$ in polytope $\mathcal{U}_q^k$, the number of auxiliary variables α_i^k $(i \in \mathcal{C}^k)$ which can simultaneously take their maximum values of 1 is bounded by Γ_q^k. Γ_q^k denotes the uncertainty budget which actually restricts the number of customers who can simultaneously have the largest possible demands $\overline{q}_i + \hat{q}_i$ on route r_k. Γ_q^k equals $\lceil\theta_q|\mathcal{C}^k|\rceil$ and $\lceil\theta_q|\mathcal{C}^k|\rceil$ denotes the least integer that is greater than or equal to $\theta_q|\mathcal{C}^k|$. $\mathcal{C}^k$ denotes the set of customers served by vehicle k based on route r_k and $|\mathcal{C}^k|$ denotes the number of customers on route r_k. θ_q denotes the uncertainty budget coefficient. It is set by decision-makers and can have a value between 0 and 1. If $\theta_q = 0$, $\Gamma_q^k = 0$. Thus, $\alpha_i^k = 0$ and $\tilde{q}_i = \overline{q}_i$ for all $i \in \mathcal{C}$. No demand uncertainty is considered in polytope $\mathcal{U}_q^k$. If $\theta_q = 1$, $\Gamma_q^k = |\mathcal{C}^k|$. Thus, α_i^k can take any value in the interval $[-1,1]$ and uncertain customer demand $\tilde{q}_i$ can take any value in the interval $[\overline{q}_i - \hat{q}_i, \overline{q}_i + \hat{q}_i]$ for each $i \in \mathcal{C}$. In real-life VRPTW applications, θ_q reflects decision-makers' attitudes towards customer demand uncertainty. If decision-makers do not worry about the impact of customer demand uncertainty on the feasibility of the planned routing strategies, θ_q can be set close to 0. On the contrary, θ_q can be set close to 1 if they are seriously concerned about the impact of demand uncertainty.

The service time uncertainty set $\mathcal{U}_s$ in equation (3.12) is the intersection of polytopes $\mathcal{U}_s^k$ for all $k \in \mathcal{K}'$. The notations in uncertainty polytope $\mathcal{U}_s^k$ in equation (3.13) have similar meanings to those in polytope $\mathcal{U}_q^k$. Specifically, $\tilde{s}_i$ denotes the uncertain service time of customer i and vector $\tilde{s}$ subsumes $\tilde{s}_i$ for all $i \in \mathcal{C}$. $\overline{s}_i$ denotes the nominal value of $\tilde{s}_i$ and $\hat{s}_i$ denotes the maximum deviation of $\tilde{s}_i$ from $\overline{s}_i$. β_i^k is the auxiliary variable associated with uncertain service time $\tilde{s}_i$ and $\tilde{s}_i = \overline{s}_i + \beta_i^k \hat{s}_i$ for all $i \in \mathcal{C}$. As the mathematical restriction of β_i^k is $[-1,1]$, the uncertain service time $\tilde{s}_i$ of customer i actually takes values in the interval $[\overline{s}_i - \hat{s}_i, \overline{s}_i + \hat{s}_i]$ for each $i \in \mathcal{C}$. In uncertainty polytope $\mathcal{U}_s^k$, Γ_s^k denotes the uncertainty budget which equals $\lceil \theta_s |\mathcal{C}^k| \rceil$. Similar to the uncertainty budget Γ_q^k in polytope $\mathcal{U}_q^k$, Γ_s^k also imposes an upper bound on the number of customers who can simultaneously have the longest possible service times $\bar{s}_i + \hat{s}_i$ on the route r_k of vehicle k. θ_s denotes the uncertainty budget coefficient and it can be set to any value in the interval $[0,1]$ by decision-makers. Note that no service time uncertainty is considered in polytope $\mathcal{U}_s^k$ if $\theta_s = 0$. In practical VRPTW applications, θ_s reflects decision-makers' attitudes towards service time uncertainty. If decision-makers are concerned about the impact of service time uncertainty on the feasibility of the planned routing strategies, θ_s can be set close to 1. On the contrary, θ_s can be set close to 0.

The travel time uncertainty set $\mathcal{U}_t$ in equation (3.14) is the intersection of polytopes $\mathcal{U}_t^k$ for all $k \in \mathcal{K}'$. Uncertainty polytope $\mathcal{U}_t^k$ in equation (3.15) captures the possible travel time uncertainty experienced by vehicle $k \in \mathcal{K}'$ based on its route r_k. Specifically, $\tilde{t}_{ij}$ denotes the uncertain travel time on arc $(i,j) \in \mathcal{A}$ and vector $\tilde{t}$ subsumes $\tilde{t}_{ij}$ for all $(i,j) \in \mathcal{A}$. $\bar{t}_{ij}$ represents the nominal value of $\tilde{t}_{ij}$ and $\hat{t}_{ij}$ represents the maximum deviation of $\tilde{t}_{ij}$ from $\bar{t}_{ij}$. γ_{ij}^k denotes the auxiliary variable associated with uncertain travel time $\tilde{t}_{ij}$ and $\tilde{t}_{ij} = \bar{t}_{ij} + \gamma_{ij}^k \hat{t}_{ij}$ for each $(i,j) \in \mathcal{A}$. As the mathematical restriction of γ_{ij}^k is $[-1,1]$, the uncertain travel time $\tilde{t}_{ij}$ on arc $(i,j) \in \mathcal{A}$ actually takes values in the interval $[\bar{t}_{ij} - \hat{t}_{ij}, \bar{t}_{ij} + \hat{t}_{ij}]$. However, according to the inequality $\sum_{(i,j)\in\mathcal{A}^k} |\gamma_{ij}^k| \leq \Gamma_t^k$ in uncertainty polytope $\mathcal{U}_t^k$, at most Γ_t^k auxiliary variables γ_{ij}^k $((i,j) \in \mathcal{A}^k)$ can simultaneously take their maximum values of 1. Γ_t^k denotes the uncertainty budget and it restricts the number of arcs (route segments) which can simultaneously have the longest possible travel times $\bar{t}_{ij} + \hat{t}_{ij}$ on route r_k. Γ_t^k equals $\lceil \theta_t |\mathcal{A}^k| \rceil$, where $\mathcal{A}^k$ denotes the set of arcs on route r_k and θ_t denotes the uncertainty budget coefficient. θ_t reflects decision-makers' attitudes towards travel time uncertainty in practical VRPTW applications and it can be set to a value between 0 and 1. Similar to uncertainty polytope $\mathcal{U}_q^k$, no travel time uncertainty is considered in polytope $\mathcal{U}_t^k$ if $\theta_t = 0$. Decision-makers can set θ_t close to 0 if they are not concerned about the impact of travel time uncertainty on the feasibility of the planned routing strategies. If they are very concerned about the impact of travel time uncertainty, θ_t can be set close to 1.

To model the VRPTW under three types of uncertainty, we present a two-stage robust formulation based on the resource inequalities formulation introduced in Agra et al. (2013). Agra et al. (2013) studied a robust VRPTW under travel time uncertainty and proposed two formulations to model the problem. One is the resource inequalities formulation which was built based on the adjustable robust optimization framework (Ben-Tal et al., 2004). In such framework, the decisions are divided into two stages. First-stage decision variables have to be determined before the uncertainty is revealed, while second-stage (adjustable) decision variables can adjust after the uncertainty has been disclosed. Based on the defined uncertainty sets and the adjustable robust optimization framework, we extend the deterministic formulation (3.1)-(3.9) to the following two-stage robust formulation:

$$\text{(R-VRPTW)} \quad \textit{lex-}\min \left(\sum_{k\in\mathcal{K}} \sum_{j\in\mathcal{C}} x_{0j}^k, \sum_{k\in\mathcal{K}} \sum_{(i,j)\in\mathcal{A}} d_{ij} x_{ij}^k \right) \tag{3.16}$$

$$\text{s.t.} \qquad (3.2), (3.3), (3.4), (3.5), (3.9),$$

$$\sum_{(i,j)\in\mathcal{A}} \tilde{q}_i x_{ij}^k \le Q \quad \forall k \in \mathcal{K}, \tilde{q} \in \mathcal{U}_q, \tag{3.17}$$

$$y_i^k(\tilde{s},\tilde{t}) + \tilde{t}_{ij} + \tilde{s}_i \le y_j^k(\tilde{s},\tilde{t}) + M(1 - x_{ij}^k) \quad \forall k \in \mathcal{K}, (i,j) \in \mathcal{A}, \tilde{s} \in \mathcal{U}_s, \tilde{t} \in \mathcal{U}_t, \tag{3.18}$$

$$a_i \le y_i^k(\tilde{s},\tilde{t}) \le b_i \quad \forall k \in \mathcal{K}, i \in \mathcal{N}, \tilde{s} \in \mathcal{U}_s, \tilde{t} \in \mathcal{U}_t. \tag{3.19}$$

In the above formulation, binary variables x_{ij}^k for all $k \in \mathcal{K}$ and $(i,j) \in \mathcal{A}$ become the first-stage decision variables. They have to be determined before the values of the uncertain parameters $(\tilde{q}, \tilde{s}, \tilde{t})$ are revealed. Continuous variables y_i^k for all $k \in \mathcal{K}$ and $i \in \mathcal{N}$ are extended to the second-stage (adjustable) decision variables $y_i^k(\tilde{s},\tilde{t})$ given every realization of uncertain parameters $\tilde{s} \in \mathcal{U}_s$ and $\tilde{t} \in \mathcal{U}_t$. They are determined after the values of the uncertain parameters are revealed given the determined first-stage decisions. Thus, the second-stage decision variables are dependent on the uncertain parameters. Note that constraints (3.6)-(3.8) in the deterministic formulation are replaced by constraints (3.17)-(3.19) in the robust formulation. In addition, the reasonable value for M should be accordingly adjusted in constraints (3.18) due to the consideration of uncertainty.

To make the above robust formulation easier to understand, a few remarks are made. Firstly, a feasible first-stage solution consisting of first-stage decisions x_{ij}^k must ensure that constraints (3.17)-(3.19) are satisfied for every possible realization of the uncertain parameters $\tilde{q} \in \mathcal{U}_q$, $\tilde{s} \in \mathcal{U}_s$, and $\tilde{t} \in \mathcal{U}_t$. Secondly, the uncertainty sets $\mathcal{U}_q$, $\mathcal{U}_s$, and $\mathcal{U}_t$ are dependent on the first-stage solution because the first-stage decisions x_{ij}^k determine the specific route for each vehicle. Thirdly, the values of the hierarchical objective function (3.16) are only affected by the first-stage decisions. Fourthly, the robust formulation has an infinite number of second-stage decision variables $y_i^k(\tilde{s}, \tilde{t})$ and constraints (3.17)-(3.19) because there are an infinite number of possible vectors in uncertainty sets $\mathcal{U}_q$, $\mathcal{U}_s$, and $\mathcal{U}_t$. As discussed in Agra et al. (2013), these uncertainty sets can be restricted to sets $ext(\mathcal{U}_q)$, $ext(\mathcal{U}_s)$, and $ext(\mathcal{U}_t)$ which contain only the extreme points of the corresponding uncertainty sets. Based on such sets, exact solution methods can be designed according to Agra et al. (2013). However, these methods may be able to tackle only small- or medium-sized instances. Thus, to solve the above two-stage robust formulation with large-sized instances, we focus on developing efficient metaheuristics.

3.3 An AVNS-Based Metaheuristic

To solve the VRPTW under uncertainty introduced in Section 3.2.2, we develop a metaheuristic solution approach based on the AVNS algorithm proposed in Stenger et al. (2013). The AVNS algorithm replaces the random shaking step in the classical VNS paradigm (Mladenović and Hansen, 1997) with a guided shaking step which relies on a number of route and node sequence selection methods and an adaptive mechanism. Due to its strong diversification possibilities and high searching efficiency, the AVNS algorithm has been employed and adapted to solve many complex extensions of the CVRP in the literature (Hof et al., 2017; Schneider et al., 2015). Next, we introduce the details of the AVNS-based metaheuristic developed for the VRPTW under uncertainty in the following sections.

3.3.1 Overview of the Metaheuristic

This section gives an overview of the AVNS-based metaheuristic. The main components and steps of the metaheuristic are presented in pseudocode in Algorithm 3.1. To describe the AVNS-based metaheuristic with simplicity and clarity, we refer to $\mathcal{R}$ as a solution to the VRPTW under uncertainty in the metaheuristic. $\mathcal{R}$ consists of a set of vehicle routes and $|\mathcal{R}|$ denotes the number of vehicles (routes) used.

Algorithm 3.1 An AVNS-based metaheuristic for the VRPTW under uncertainty.

```
1: $\mathcal{R}^0 \leftarrow$ InitialSolutionAVNS();
2: $\mathcal{R} \leftarrow \mathcal{R}^0$;
3: while $|\mathcal{R}| > V_{min}$ do
4:     $\mathcal{R}' \leftarrow$ RemoveOneRoute($\mathcal{R}$);
5:     $\mathcal{R}' \leftarrow$ AVNSFeasibilityRecovery($\mathcal{R}'$);
6:     if $\mathcal{R}'$ is feasible then
7:         $\mathcal{R} \leftarrow \mathcal{R}'$;
8:     else
9:         break;
10:    end if
11: end while
12: $\mathcal{R}^* \leftarrow$ AVNSDistanceMinimization($\mathcal{R}$);
```

As shown in Algorithm 3.1, the AVNS-based metaheuristic can be divided into three phases. In the first phase, function InitialSolutionAVNS() generates a feasible initial solution $\mathcal{R}^0$ for the VRPTW under uncertainty (line 1). The details of function InitialSolutionAVNS() are described in Section 3.3.3. The number of vehicles (routes) used in the initial solution is minimized by functions RemoveOneRoute() and AVNSFeasibilityRecovery() in the second phase (lines 3-11). In function RemoveOneRoute(), the route with the minimum number of customers is first removed from the current solution $\mathcal{R}$. Then, the customers on the removed route are randomly reinserted into the remaining routes to generate a new solution $\mathcal{R}'$. Note that solution $\mathcal{R}'$ is generally infeasible because of the random reinsertion process. Infeasible solutions are allowed in the AVNS-based metaheuristic and an augment cost function $Cost()$ is defined in equation (3.20) to evaluate both feasible and infeasible solutions. Function AVNSFeasibilityRecovery() tries to recover the feasibility of solution $\mathcal{R}'$ based on the AVNS framework. Its detailed procedure is shown in pseudocode in Algorithm 3.2. The vehicle (route) reduction phase stops when the feasibility of solution $\mathcal{R}'$ cannot be recovered or the current feasible solution $\mathcal{R}$ has the minimum required number of vehicles $V_{min} = \sum_{i \in \mathcal{C}} \overline{q}_i / Q$. Given the resulting solution $\mathcal{R}$ from the second phase, its total travel distance is minimized by function AVNSDistanceMinimization() in the third phase (line 12). Function AVNSDistanceMinimization() is shown in pseudocode in Algorithm 3.3. Finally, the metaheuristic returns the best found solution $\mathcal{R}^*$.

In Algorithm 3.2, function AVNSFeasibilityRecovery() adopts the AVNS framework and tries to recover the feasibility of the input solution $\mathcal{R}^0$. In the beginning, two sets of neighbourhood structures $\mathcal{N}_{shake}$ and $\mathcal{N}_{search}$ are defined (line 1) and solution $\mathcal{R}^0$ is improved by function LocalSearch() (line 2). Function LocalSearch() is a local search procedure which employs six widely used neighbourhood structures in the VRP literature. We introduce the local search procedure and the neighbourhood structures $\mathcal{N}_{search}$ in Section 3.3.4.4. Note that the local search procedure immediately stops once a feasible solution is found whenever function LocalSearch() is called in function AVNSFeasibilityRecovery(). In the main loop of the AVNS framework (lines 5-32), function AdaptiveShaking() first perturbs the current solution $\mathcal{R}$ using one of the predefined shaking neighbourhood structures $\mathcal{N}_{shake}^{(i)}$ (line 7). $\mathcal{N}_{shake}^{(i)}$ denotes the ith neighbourhood structure in set $\mathcal{N}_{shake}$. In function AdaptiveShaking(), a number of route and node sequence selection methods are designed and used to help choosing the routes and node sequences of the current solution to be involved in the perturbation (shaking) process. The shaking neighbourhood structures $\mathcal{N}_{shake}$ and the selection methods are introduced in Section 3.3.4.1. The resulting solution $\mathcal{R}'$ from function AdaptiveShaking() is immediately improved by function LocalSearch()

Algorithm 3.2 AVNS for solution feasibility recovery.

```
Function AVNSFeasibilityRecovery($\mathcal{R}^0$)
1: Define two sets of neighbourhood structures $\mathcal{N}_{shake}$ and $\mathcal{N}_{search}$;
2: $\mathcal{R}^0 \leftarrow$ LocalSearch($\mathcal{R}^0$,$\mathcal{N}_{search}$);
3: $\mathcal{R} \leftarrow \mathcal{R}^0$; $\mathcal{R}^* \leftarrow \mathcal{R}^0$;
4: $Iter \leftarrow 0$; $NoImp \leftarrow 0$; $i \leftarrow 1$;
5: while $\mathcal{R}^*$ is infeasible and $Iter < MaxIter1$ and $NoImp < MaxNoImp1$ do
6:     $Iter \leftarrow Iter + 1$;
7:     $\mathcal{R}' \leftarrow$ AdaptiveShaking($\mathcal{R}$,$\mathcal{N}^{(i)}_{shake}$);
8:     $\mathcal{R}' \leftarrow$ LocalSearch($\mathcal{R}'$,$\mathcal{N}_{search}$);
9:     if $\mathcal{R}'$ is feasible then
10:        $\mathcal{R}^* \leftarrow \mathcal{R}'$;
11:        break;
12:    end if
13:    if Accept($\mathcal{R}'$,$\mathcal{R}$) then
14:        $\mathcal{R} \leftarrow \mathcal{R}'$;
15:        $i \leftarrow 1$;
16:        if $Cost(\mathcal{R}') < Cost(\mathcal{R}^*)$ then
17:            $\mathcal{R}^* \leftarrow \mathcal{R}'$;
18:            $NoImp \leftarrow 0$;
19:        else
20:            $NoImp \leftarrow NoImp + 1$;
21:        end if
22:    else
23:        $NoImp \leftarrow NoImp + 1$; $i \leftarrow (i \bmod |\mathcal{N}_{shake}|) + 1$;
24:    end if
25:    UpdateSelectionMethodScore($\mathcal{R}'$);
26:    if $0 \equiv Iter \pmod{\eta^{aw}}$ then
27:        AdaptSelectionMethodWeight();
28:    end if
29:    if $NoImp > 0$ and $0 \equiv NoImp \pmod{\eta^{sr}}$ then
30:        $\mathcal{R} \leftarrow$ SolutionReset($\mathcal{R}^*$);
31:    end if
32: end while
33: return $\mathcal{R}^*$;
```

(line 8). Function Accept() employs a SA-based acceptance mechanism to decide whether solution $\mathcal{R}'$ replaces $\mathcal{R}$ as the current solution for the subsequent iteration. The details of the SA-based acceptance mechanism are given in Section 3.3.4.5. Functions UpdateSelectionMethodScore() and AdaptSelectionMethodWeight() are employed in an adaptive mechanism which helps to choose effective route and node sequence selection methods

Algorithm 3.3 AVNS for distance minimization.

```
Function AVNSDistanceMinimization($\mathcal{R}^0$)
1:  Define two sets of neighbourhood structures $\mathcal{N}_{shake}$ and $\mathcal{N}_{search}$;
2:  $\mathcal{R}^0 \leftarrow$ LocalSearch($\mathcal{R}^0$,$\mathcal{N}_{search}$);
3:  $\mathcal{R} \leftarrow \mathcal{R}^0$; $\mathcal{R}^* \leftarrow \mathcal{R}^0$;
4:  if $\mathcal{R}^0$ is feasible then
5:      $\mathcal{R}^*_f \leftarrow \mathcal{R}^0$;
6:  end if
7:  $Iter \leftarrow 0$; $NoImp \leftarrow 0$; $i \leftarrow 1$;
8:  while $Iter < MaxIter2$ and $NoImp < MaxNoImp2$ do
9:      $Iter \leftarrow Iter + 1$;
10:     $\mathcal{R}' \leftarrow$ AdaptiveShaking($\mathcal{R}$,$\mathcal{N}^{(i)}_{shake}$);
11:     $\mathcal{R}' \leftarrow$ LocalSearch($\mathcal{R}'$,$\mathcal{N}_{search}$);
12:     if Accept($\mathcal{R}'$, $\mathcal{R}$) then
13:         $\mathcal{R} \leftarrow \mathcal{R}'$;
14:         $i \leftarrow 1$;
15:         if $Cost(\mathcal{R}') < Cost(\mathcal{R}^*)$ then
16:             $\mathcal{R}^* \leftarrow \mathcal{R}'$;
17:             $NoImp \leftarrow 0$;
18:         else
19:             $NoImp \leftarrow NoImp + 1$;
20:         end if
21:         if $\mathcal{R}'$ is feasible and $Cost(\mathcal{R}') < Cost(\mathcal{R}^*_f)$ then
22:             $\mathcal{R}^*_f \leftarrow \mathcal{R}'$;
23:         end if
24:     else
25:         $NoImp \leftarrow NoImp + 1$; $i \leftarrow (i \bmod |\mathcal{N}_{shake}|) + 1$;
26:     end if
27:     UpdatePenaltyFactor($\mathcal{R}'$);
28:     UpdateSelectionMethodScore($\mathcal{R}'$);
29:     if $0 \equiv Iter \pmod{\eta^{aw}}$ then
30:         AdaptSelectionMethodWeight();
31:     end if
32:     if $NoImp > 0$ and $0 \equiv NoImp \pmod{\eta^{sr}}$ then
33:         $\mathcal{R} \leftarrow$ SolutionReset($\mathcal{R}^*$);
34:     end if
35: end while
36: return $\mathcal{R}^*_f$;
```

in function AdaptiveShaking() (lines 25-28). The details of these two functions and the adaptive mechanism are given in Section 3.3.4.3. Function SolutionReset() is a restarting mechanism which resets the current solution $\mathcal{R}$ to the best solution $\mathcal{R}^*$ after a certain number of iterations η^{sr} without an overall improvement (lines 29-31). It aims to help the search process escape from deep infeasible regions. We terminate the feasibility recovery procedure when the best solution $\mathcal{R}^*$ becomes feasible or a maximum number of iterations $MaxIter1$ or no overall improvement iterations $MaxNoImp1$ is reached.

In Algorithm 3.3, function AVNSDistanceMinimization() also employs the AVNS framework to minimize the total travel distance of the resulting solution from the vehicle (route) minimization phase. The main components and functions used in function AVNS-DistanceMinimization() are the same as those used in function AVNSFeasibilityRecovery(). However, function AVNSDistanceMinimization() adopts an additional component Update-PenaltyFactor() which is a dynamic penalty mechanism. The dynamic penalty mechanism is similar to those used in Hiermann et al. (2016) and Hof and Schneider (2019). The details of the mechanism are described in Section 3.3.2. Following the idea in Hiermann et al. (2016), two types of best solutions are kept in function AVNSDistanceMinimization(). The first is the best feasible solution $\mathcal{R}_f^*$ and the second is the best solution $\mathcal{R}^*$ with the current penalty setting. We adapt the cost function values of the current solution $\mathcal{R}$ and the best solution $\mathcal{R}^*$ every time the penalty factors are updated in function UpdatePenaltyFactor(). However, this may lead to a situation that the cost function value $Cost(\mathcal{R}^*)$ of the best solution is worse than the cost function value $Cost(\mathcal{R}_f^*)$ of the best feasible solution. Thus, we reset the best solution $\mathcal{R}^*$ to the best feasible solution $\mathcal{R}_f^*$ when this situation happens.

3.3.2 Solution Evaluation

For the purpose of exploring large search space and generating high-quality solutions, infeasible solutions are allowed in the AVNS-based metaheuristic. To evaluate a given solution $\mathcal{R}$, we use an augment cost function $Cost(\mathcal{R})$ which includes penalty costs for the possible violations of all operational constraints. Specifically, the penalty costs for violations of vehicle capacity and time window constraints are respectively calculated by multiplying the corresponding amount of violations with a penalty factor. We express the augment cost function $Cost(\mathcal{R})$ of a given solution $\mathcal{R}$ in the following equation (3.20):

$$Cost(\mathcal{R}) = Distance(\mathcal{R}) + \rho^{cap}V^{cap}(\mathcal{R}) + \rho^{tw}V^{tw}(\mathcal{R}) \tag{3.20}$$

where $Distance(\mathcal{R})$ denotes the total travel distance of solution $\mathcal{R}$. ρ^{cap} represents the penalty factor for vehicle capacity violations $V^{cap}(\mathcal{R})$ and ρ^{tw} represents the penalty factor for time window violations $V^{tw}(\mathcal{R})$.

As shown in Algorithm 3.3, function AVNSDistanceMinimization() incorporates an important component UpdatePenaltyFactor(). It is a dynamic penalty mechanism which is similar to those used in Hiermann et al. (2016) and Hof and Schneider (2019). We briefly introduce the mechanism as follows. The penalty factors ρ^{cap} and ρ^{tw} are initially set to ρ^0 and they are dynamically updated within the interval $[\rho^{min}, \rho^{max}]$ based on the following rules in the main loop of function AVNSDistanceMinimization(). If the candidate solution $\mathcal{R}'$ from function LocalSearch() stays infeasible after η^{ins} consecutive AVNS iterations, both penalty factors are multiplied by ρ^{update}. Conversely, both penalty factors are divided by ρ^{update} if solution $\mathcal{R}'$ stays feasible after η^{des} consecutive AVNS iterations. As shown in Algorithm 3.2, the goal of function AVNSFeasibilityRecovery() is to recover the feasibility of an infeasible solution. Thus, the dynamic penalty mechanism is not considered in function AVNSFeasibilityRecovery() and we set both penalty factors ρ^{cap} and ρ^{tw} to the maximum value ρ^{max} when evaluating a given solution in it.

In the VRPTW under uncertainty introduced in Section 3.2.2, three important types of problem parameters including customer demands $\tilde{q}_i$ $(\forall i \in \mathcal{C})$, service times $\tilde{s}_i$ $(\forall i \in \mathcal{C})$, and travel times $\tilde{t}_{ij}$ $(\forall (i, j) \in \mathcal{A})$ are assumed to be uncertain. Moreover, these three types of uncertain parameters respectively take values in the route-dependent uncertainty set $\mathcal{U}_q$ (3.10) defined with polytopes $\mathcal{U}_q^k$ (3.11), set $\mathcal{U}_s$ (3.12) defined with polytopes $\mathcal{U}_s^k$ (3.13), and set $\mathcal{U}_t$ (3.14) defined with polytopes $\mathcal{U}_t^k$ (3.15). Next, we introduce how to calculate the time window violations $V^{tw}(\mathcal{R})$ and the vehicle capacity violations $V^{cap}(\mathcal{R})$ when evaluating a given solution $\mathcal{R}$ based on the defined uncertainty sets and polytopes.

$$Y^k(v_i^k,\Gamma_s^k,\Gamma_t^k)=\begin{cases}0, & \text{if } i=1;\\ \max\Big\{a_{v_i^k}, Y^k(v_{i-1}^k,\Gamma_s^k,\Gamma_t^k)+\bar{s}_{v_{i-1}^k}+\bar{t}_{v_{i-1}^k,v_i^k}\Big\}, & \\ & \text{if } 2\le i\le m+1, \Gamma_s^k=\Gamma_t^k=0;\\ \max\Big\{a_{v_i^k}, Y^k(v_{i-1}^k,\Gamma_s^k,\Gamma_t^k)+\bar{s}_{v_{i-1}^k}+\bar{t}_{v_{i-1}^k,v_i^k}, & \\ \quad Y^k(v_{i-1}^k,\Gamma_s^k-1,\Gamma_t^k)+\bar{s}_{v_{i-1}^k}+\hat{s}_{v_{i-1}^k}+\bar{t}_{v_{i-1}^k,v_i^k}\Big\}, & \\ & \text{if } 2\le i\le m+1, 1\le\Gamma_s^k\le i-1, \Gamma_t^k=0;\\ \max\Big\{a_{v_i^k}, Y^k(v_{i-1}^k,\Gamma_s^k,\Gamma_t^k)+\bar{s}_{v_{i-1}^k}+\bar{t}_{v_{i-1}^k,v_i^k}, & \\ \quad Y^k(v_{i-1}^k,\Gamma_s^k,\Gamma_t^k-1)+\bar{s}_{v_{i-1}^k}+\bar{t}_{v_{i-1}^k,v_i^k}+\hat{t}_{v_{i-1}^k,v_i^k}\Big\}, & \\ & \text{if } 2\le i\le m+1, \Gamma_s^k=0, 1\le\Gamma_t^k\le i-1;\\ \max\Big\{a_{v_i^k}, Y^k(v_{i-1}^k,\Gamma_s^k,\Gamma_t^k)+\bar{s}_{v_{i-1}^k}+\bar{t}_{v_{i-1}^k,v_i^k}, & \\ \quad Y^k(v_{i-1}^k,\Gamma_s^k-1,\Gamma_t^k)+\bar{s}_{v_{i-1}^k}+\hat{s}_{v_{i-1}^k}+\bar{t}_{v_{i-1}^k,v_i^k}, & \\ \quad Y^k(v_{i-1}^k,\Gamma_s^k,\Gamma_t^k-1)+\bar{s}_{v_{i-1}^k}+\bar{t}_{v_{i-1}^k,v_i^k}+\hat{t}_{v_{i-1}^k,v_i^k}, & \\ \quad Y^k(v_{i-1}^k,\Gamma_s^k-1,\Gamma_t^k-1)+\bar{s}_{v_{i-1}^k}+\hat{s}_{v_{i-1}^k}+\bar{t}_{v_{i-1}^k,v_i^k}+\hat{t}_{v_{i-1}^k,v_i^k}\Big\}, & \\ & \text{if } 2\le i\le m+1, 1\le\Gamma_s^k\le i-1, 1\le\Gamma_t^k\le i-1;\\ Y^k(v_i^k,\Gamma_s^k-1,\Gamma_t^k), & \text{if } 2\le i\le\Gamma_s^k;\\ Y^k(v_i^k,\Gamma_s^k,\Gamma_t^k-1), & \text{if } 2\le i\le\Gamma_t^k.\end{cases} \tag{3.21}$$

Suppose there is a given solution $\mathcal{R}$ which consists of h vehicle routes and $\mathcal{R}=\{r_1,\cdots,r_h\}$. Let $r_k=\{v_1^k,v_2^k,\cdots,v_m^k,v_{m+1}^k\}\in\mathcal{R}$ be the route of vehicle k. v_1^k and v_{m+1}^k are nodes 0 and $n+1$, respectively. They denote the same central depot in the VRPTW introduced in Section 3.2.1. As discussed in Section 3.2.2, both service time uncertainty polytope $\mathcal{U}_s^k$ (3.13) and travel time uncertainty polytope $\mathcal{U}_t^k$ (3.15) are defined based on the route r_k of vehicle k. To determine the time violations with route r_k, we can first calculate the latest arrival time of vehicle k at every node on route r_k considering all possible realizations of uncertain service times $\tilde{s}_i$ ($\forall i \in \mathcal{C}^k$) in polytope $\mathcal{U}_s^k$ and uncertain travel times $\tilde{t}_{ij}$ ($\forall (i,j) \in \mathcal{A}^k$) in polytope $\mathcal{U}_t^k$. $\mathcal{C}^k$ and $\mathcal{A}^k$ respectively denote the sets of customers and route segments (arcs) on route r_k. As shown in the service time uncertainty polytope $\mathcal{U}_s^k$, the uncertain service time $\tilde{s}_i$ of each customer $i \in \mathcal{C}^k$ takes values in the interval $[\overline{s}_i - \hat{s}_i, \overline{s}_i + \hat{s}_i]$. However, due to the restriction of the uncertainty budget Γ_s^k, there are at most Γ_s^k customers who can simultaneously have the longest service times $\overline{s}_i + \hat{s}_i$ on route r_k. Similarly, as shown in polytope $\mathcal{U}_t^k$, the uncertain travel time $\tilde{t}_{ij}$ on each arc $(i,j) \in \mathcal{A}^k$ takes values in the interval $[\overline{t}_{ij} - \hat{t}_{ij}, \overline{t}_{ij} + \hat{t}_{ij}]$ and there are at most Γ_t^k arcs which can simultaneously have the longest travel times $\overline{t}_{ij} + \hat{t}_{ij}$ on route r_k. To calculate the latest vehicle arrival time $Y^k(v_i^k, \Gamma_s^k, \Gamma_t^k)$ at the ith node v_i^k on route $r_k = \{v_1^k, v_2^k, \cdots, v_m^k, v_{m+1}^k\}$, we use equation (3.21) which is a recursive function of the nodes v_i^k on route r_k, the uncertainty budget Γ_s^k in polytope $\mathcal{U}_s^k$, and the uncertainty budget Γ_t^k in polytope $\mathcal{U}_t^k$. Note that this function was originally proposed by Shi et al. (2019) who studied a home healthcare routing and scheduling problem under service and travel time uncertainty. By comparing $Y^k(v_i^k, \Gamma_s^k, \Gamma_t^k)$ with the latest start-of-service time of each node on route r_k, we can calculate the time window violations with route r_k and further determine the total time window violations with the given solution $\mathcal{R}$.

However, using function $Y^k(v_i^k, \Gamma_s^k, \Gamma_t^k)$ in equation (3.21) to help determining the time window violations with route r_k can be computational expensive especially when the route consists of many nodes. In addition, both functions AVNSFeasibilityRecovery() and AVNSDistanceMinimization() in the AVNS-based metaheuristic rely heavily on function LocalSearch(), which requires a large number of solution and route evaluations. To reduce the computational time of evaluating a solution $\mathcal{R}$ in function LocalSearch(), we overcalculate the time window violations with each route $r_k \in \mathcal{R}$ if it is time window infeasible even without considering service and travel time uncertainty. Specifically, if route $r_k \in \mathcal{R}$ is infeasible due to the existence of time window violations considering only the nominal values $\overline{s}_i$ of uncertain service times $\tilde{s}_i$ for all $i \in \mathcal{C}^k$ and the nominal values $\overline{t}_{ij}$ of uncertain travel times $\tilde{t}_{ij}$ for all $(i,j) \in \mathcal{A}^k$, we ignore the restrictions of the uncertainty budgets Γ_s^k and Γ_t^k in polytopes $\mathcal{U}_s^k$ and $\mathcal{U}_t^k$ and approximately determine the latest arrival time of vehicle k at every node on route r_k and the time window violations with route r_k under the assumption $\tilde{s}_i = \overline{s}_i + \hat{s}_i$ for all $i \in \mathcal{C}^k$ and $\tilde{t}_{ij} = \overline{t}_{ij} + \hat{t}_{ij}$ for all $(i,j) \in \mathcal{A}^k$.

To determine the vehicle capacity violations with route r_k, we can calculate the largest vehicle load on the route considering all possible realizations of uncertain customer demands $\tilde{q}_i$ $(\forall i \in \mathcal{C}^k)$ in uncertainty polytope $\mathcal{U}_q^k$ (3.11). As discussed in Section 3.2.2, the customer demand uncertainty polytope $\mathcal{U}_q^k$ is defined based on route r_k. Moreover, the uncertain demand $\tilde{q}_i$ of customer i on route r_k can take values in the interval $[\overline{q}_i - \hat{q}_i, \overline{q}_i + \hat{q}_i]$ for each $i \in \mathcal{C}^k$. However, owing to the restriction of the uncertainty budget Γ_q^k in polytope $\mathcal{U}_q^k$, there are at most Γ_q^k customers who can simultaneously have the largest demands $\overline{q}_i + \hat{q}_i$ on route r_k. Thus, to calculate the largest vehicle load on route $r_k = \{v_1^k, v_2^k, \cdots, v_m^k, v_{m+1}^k\}$, we first sort the customers on route r_k in descending order based on the maximum deviations of their uncertain demands and select the first Γ_q^k customers to generate a new ordered set $\mathcal{C}_{\Gamma_q^k}^k = \{v'^k_1, \cdots, v'^k_{\Gamma_q^k}\}$. In set $\mathcal{C}_{\Gamma_q^k}^k$, v'^k_j denotes the customer with the jth largest demand deviation $\hat{q}_{v'^k_j}$ on route r_k, $j = 1, \cdots, \Gamma_q^k$. Then, the largest vehicle load $Z^k(\Gamma_q^k)$ on route r_k can be calculated using equation (3.22). By comparing $Z^k(\Gamma_q^k)$ with the vehicle capacity Q, the vehicle capacity violations with route r_k can be calculated and the total vehicle capacity violations with the given solution $\mathcal{R}$ can be determined.

$$Z^k(\Gamma_q^k) = \sum_{i=1}^{m+1} \overline{q}_{v_i^k} + \sum_{i=1}^{\Gamma_q^k} \hat{q}_{v'^k_i}. \tag{3.22}$$

To further reduce the computational time of evaluating a solution $\mathcal{R}$ in function LocalSearch(), we also overcalculate the vehicle capacity violations with each route $r_k \in \mathcal{R}$ if it is capacity infeasible even without considering customer demand uncertainty. Specifically, if route $r_k \in \mathcal{R}$ is infeasible due to the existence of vehicle capacity violations considering only the nominal values $\overline{q}_i$ of uncertain customer demands $\tilde{q}_i$ for all $i \in \mathcal{C}^k$, we ignore the restriction of the uncertainty budget Γ_q^k in polytope $\mathcal{U}_t^q$ and approximately calculate the largest vehicle load on route r_k and the vehicle capacity violations with route r_k under the assumption $\tilde{q}_i = \overline{q}_i + \hat{q}_i$ for all $i \in \mathcal{C}^k$. Note that the vehicle capacity and time window violations with a solution may be overcalcuated only when it is evaluated in function LocalSearch(). In addition, if the constraint violations with the output solution from function LocalSearch() are overcalcuated, we need to recalcuate its exact cost function value based on equations (3.21) and (3.22) whenever function LocalSearch() is called in functions AVNSFeasibilityRecovery() and AVNSDistanceMinimization().

3.3.3 Initial Solution

As shown in Algorithm 3.1, function InitialSolutionAVNS() generates a feasible initial solution for the VRPTW under uncertainty in the AVNS-based metaheuristic. In function InitialSolutionAVNS(), we adopt a sequential insertion heuristic to construct a set of feasible routes given a sufficient number of vehicles. The heuristic is an iterative procedure which contains three main steps in each iteration. In the first step, we select a vehicle that has not been used before and construct an empty route. In the second step, we select a seed customer and insert it into the empty route. The seed customer is chosen as the customer with the smallest upper bound of its time window (latest start-of-service time) among all unrouted customers. In the third step, we sequentially insert the unrouted customers into the current route based on the cheapest insertion criterion. If the remaining unrouted customers cannot be inserted at any position in the current route without violating any time window and vehicle capacity constraints, we go back to the first step and repeat the whole procedure. Note that the feasibility of inserting an unrouted customer into a route can be determined by calculating the largest vehicle load and the latest vehicle arrival time at every node on the new route after inserting the customer based on equations (3.21) and (3.22). The sequential insertion procedure stops when all customers are routed.

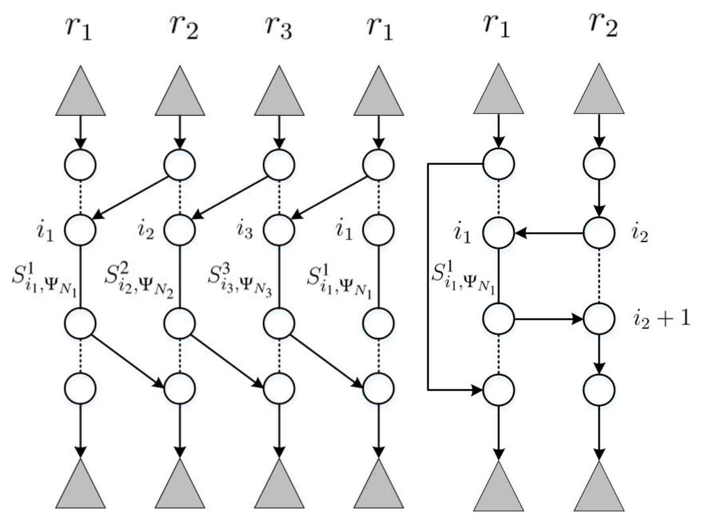
3.3.4 Key Components of the AVNS Framework

As discussed in Section 3.3.1, the AVNS framework is adopted in both functions AVNS-FeasibilityRecovery() and AVNSDistanceMinimization() to do the optimization tasks. In the AVNS framework, functions AdaptiveShaking(), LocalSearch(), Accept() are the three most important components. Next, we first introduce the details of function AdaptiveShaking() in the following three sections.

3.3.4.1 Shaking Neighbourhoods

In function AdaptiveShaking(), a set of shaking neighbourhood structures $\mathcal{N}_{shake}$ are defined to perturb the current solution in the AVNS framework. Similar to Stenger et al. (2013) and Schneider et al. (2015), we adopt two types of operators to generate the shaking neighbourhood structures: a cyclic exchange operator and a sequence reinsertion operator. The cyclic exchange operator was originally introduced by Thompson and Psaraftis (1993), which moves nodes among routes in a cyclic way. It is associated with two important parameters: the number of routes to be perturbed Ψ_R and the maximum number of nodes to be moved Ψ_N. For each route r_k involved in the perturbation process, the cyclic exchange operator moves the node sequence $S^k_{i_k,\Psi_{N_k}}$ starting with node i_k at a length of Ψ_{N_k} to route r_{k+1} at the former position of sequence $S^{k+1}_{i_{k+1},\Psi_{N_{k+1}}}$ (Ibaraki et al., 2005). Figure 3.1(a) gives an example of a cyclic exchange operation with three routes. Note that we reduce Ψ_R accordingly if the total number of existing routes in the current solution is less than the number of routes to be perturbed. Similarly, we adjust Ψ_N if it is greater than the number of customer nodes on route r_k. In addition, empty routes are not allowed in the perturbation process. The sequence reinsertion operator is a simplified version of the cyclic exchange operator, in which a node sequence is reinserted from one route to another. Figure 3.1(b) depicts a sequence reinsertion operation with two routes.

The set of shaking neighbourhood structures $\mathcal{N}_{shake}$ used in function AdaptiveShaking() are shown in Table 3.1.Note that set $\mathcal{N}_{shake}$ is an ordered set. We observe that the



(a) Example of a cyclic exchange with three routes (b) Example of a sequence reinsertion with two routes

Figure 3.1 Examples for the cyclic exchange and sequence reinsertion operators.

Table 3.1. The shaking neighbourhood structures adopted in the AVNS-based metaheuristic.

NO.	Operator	Ψ_R	Ψ_N	NO.	Operator	Ψ_R	Ψ_N
1	sequence reinsertion	2	1	16	cyclic exchange	3	1
2	sequence reinsertion	2	2	17	cyclic exchange	3	2
3	sequence reinsertion	2	3	18	cyclic exchange	3	3
4	sequence reinsertion	2	4	19	cyclic exchange	3	4
5	sequence reinsertion	2	5	20	cyclic exchange	3	5
6	cyclic exchange	2	1	21	cyclic exchange	3	6
7	cyclic exchange	2	2	22	cyclic exchange	3	7
8	cyclic exchange	2	3	23	cyclic exchange	3	8
9	cyclic exchange	2	4	24	cyclic exchange	3	9
10	cyclic exchange	2	5	25	cyclic exchange	3	10
11	cyclic exchange	2	6	26	cyclic exchange	4	1
12	cyclic exchange	2	7	27	cyclic exchange	4	2
13	cyclic exchange	2	8	28	cyclic exchange	4	3
14	cyclic exchange	2	9	29	cyclic exchange	4	4
15	cyclic exchange	2	10	30	cyclic exchange	4	5

neighbourhood structures can exchange two to four routes and move up to 10 nodes among them. Given a shaking neighbourhood, we first need to select Ψ_R routes to be involved in the shaking process from the current solution. Then, the actual length of the node sequence to be moved on each selected route is randomly generated in the interval $[0, \Psi_N]$ if $\Psi_N \leq 5$. If $\Psi_N > 5$, length of the node sequence is assumed to be fixed and equals Ψ_N.

3.3.4.2 Selection Methods

In function AdaptiveShaking(), a number of selection methods are also defined, which are used to select some routes and node sequences of the current solution to be involved in the shaking process. Specifically, we adopt several effective route and node sequence selection methods introduced in Stenger et al. (2013) and design some new methods based on the special features of the VRPTW under uncertainty. Next, we describe the details of the route and node sequence selection process and the corresponding selection methods.

Route selection. Given a shaking neighbourhood structure, the route selection process determines Ψ_R routes of the current solution which will be involved in the shaking process. We determine the first of the Ψ_R routes using one of the following six selection methods.

Random. The probability of being selected is the same for each route.

Distance. The probability of selecting a route is proportional to the travel distance of the route. This method aims to remove customers on the long routes and reinsert them into shorter routes to reduce the total travel distance.

Inefficiency. The probability of selecting a route is proportional to the inefficiency of the route. The inefficiency of a route is defined as the ratio between the travel distance and the largest vehicle load on the route. Note that the largest vehicle load on a route is calculated using equation (3.22) considering customer demand uncertainty. This method aims to help shortening the inefficient routes.

Cost. The probability of selecting a route is proportional to the cost function value of the route. The cost function value of a route is the sum of its travel distance and the penalty cost for the possible violations of the vehicle capacity and time window constraints. It can be calculated based on the augment cost function defined in equation (3.20). This method aims to help recovering the feasibility of the possible infeasible routes.

Vehicle load. The probability of selecting a route is proportional to the largest vehicle load on the route considering customer demand uncertainty. This method aims to reduce the number of routes at risk of exceeding vehicle capacity.

Lateness. The probability of selecting a route is proportional to the lateness value of the route. The lateness value of a node on a route is defined as the difference between the latest vehicle arrival time at the node considering service and travel time uncertainty and the latest start-of-service time of the node if the latest vehicle arrival time is larger than the latest start-of-service time and zero otherwise. Thus, the lateness value of a route can be calculated as the sum of the lateness values of all nodes on the route. Note that a route is chosen at random if the lateness values of all existing routes are zero.

After determining the first route to be involved in the shaking process, the remaining $\Psi_R - 1$ routes are selected based on an iterative procedure. Before laying out the details of the procedure, we first define the customer relatedness measure which is used to reflect the similarity and interchangeability of any two customers. Following the customer relatedness measure proposed in Hof and Schneider (2019), we define the relatedness measure $R(i,j)$ of customers i and j based on the travel distance d_{ij} separating them, the absolute difference between their nominal demands $|\overline{q}_i - \overline{q}_j|$, and the absolute difference between their earliest start-of-service times $|a_i - a_j|$. $R(i,j)$ can be calculated using the following equation (3.23):

$$\begin{aligned} R(i,j) = \psi^d \frac{d_{ij}}{\max\limits_{(i,j)\in\mathcal{A}} (d_{ij})} + \psi^q \frac{|\overline{q}_i - \overline{q}_j|}{\max\limits_{i\in\mathcal{C}}(\overline{q}_i) - \min\limits_{i\in\mathcal{C}}(\overline{q}_i)} \\ + \psi^a \frac{|a_i - a_j|}{\max\limits_{i\in\mathcal{C}}(a_i) - \min\limits_{i\in\mathcal{C}}(a_i)} \end{aligned} \tag{3.23}$$

where ψ^d, ψ^q, and ψ^a denote the weights of the normalized partial relatedness measures. Based on the customer relatedness measure $R(i,j)$, the relatedness value of two routes r_1 and r_2 can be calculated as the sum of the relatedness values $R(i,j)$ of each customer i on route r_1 and each customer j on route r_2. Next, we introduce the iterative procedure to choose the remaining $\Psi_R - 1$ routes given the first selected route. The procedure contains two steps in each iteration. In the first step, we sort all unselected routes in ascending order based on their relatedness values with respect to the selected route in the last iteration and store these routes in an ordered set $\mathcal{S}$. In the second step, we choose the route at position $\lfloor |\mathcal{S}|\xi^{\nu_{rs}} \rfloor$ in set $\mathcal{S}$. Parameter ξ is randomly chosen in the interval $[0, 1)$. Parameter ν_{rs} introduces some randomness in the procedure and we set $\nu_{rs} = 3$. The above two steps are repeated until Ψ_R routes are selected.

Node sequence selection. After determining the routes to be involved in the shaking process, we determine the node sequence to be moved on each selected route using one of the following five selection methods.

Random. Each node sequence is selected with equal probability.

Distance to the next route. The probability of selecting a node sequence is inversely proportional to the distance between the centre of gravity of the node sequence and that of the route they will be inserted into.

Relatedness to the next route. The probability of selecting a node sequence is inversely proportional to the relatedness value between the node sequence and the route they will be inserted into. The method to calculate the relatedness value of a node sequence and a route is similar the procedure of calculating the relatedness value of two routes discussed above.

Distance saving. The probability of selecting a node sequence is proportional to the distance savings of removing the node sequence from the route. The distance savings of removing a node sequence is defined as the difference in the travel distance of the route when the node sequence is included in the route and the node sequence is removed from the route.

Cost saving. The probability of selecting a node sequence is proportional to the cost savings of removing the node sequence from the route. The cost savings of removing a node sequence is defined as the difference in the cost function value of the route when the node sequence is included in the route and the node sequence is removed from the route. The cost function value of a route can be calculated based on the augment cost function defined in equation (3.20).

3.3.4.3 Adaptive Mechanism

In each iteration of the AVNS main loops in functions AVNSFeasibilityRecovery() and AVNSDistanceMinimization(), function AdaptiveShaking() first perturbs the current solution with the help of a route selection method and a node sequence selection method. Then, function LocalSearch() improves the perturbed solution and generates a new candidate solution. In function AdaptiveShaking(), the route and node sequence selection methods are chosen independently based on a roulette wheel selection method used in Ropke and Pisinger (2006). An adaptive mechanism proposed in Stenger et al. (2013) is also adopted to bias the selection of these methods. In the mechanism, each route or node sequence selection method i is associated with a weight ω_i and a score π_i. In the beginning, all selection methods are assigned the same weights and their scores are set to zero. Then, the scores of all selection methods are updated based on a scoring system in function UpdateSelectionMethodScore(). The details of the scoring system are described as follows. If the new candidate solution is an overall best solution, the current scores of the route and node sequence selection methods used in the current AVNS iteration are increased by σ_1. If the new solution is better than the current solution, the current scores of the just used selection methods are increased by σ_2. If the new solution is worse than the current solution but can be accepted based on the SA-based acceptance mechanism introduced in Section 3.3.4.5, the current scores of the just used selection methods are increased by σ_3. As shown in Algorithms 3.2 and 3.3, the weights of all selection methods are updated based on their scores after every η^{aw} AVNS iterations in function AdaptSelectionMethodWeight(). Let ϕ_i denote the number of times selection method i has been used since the last weight update. The new weight of selection method i can be calculated as $\omega_i = \omega_i(1 - \varpi) + \varpi\pi_i/\phi_i$. Reaction factor ϖ takes values in the interval $[0, 1]$ and it controls the inertia of the weight adjustment. The values of π_i and ϕ_i are reset to zero after every weight update.

3.3.4.4 Local Search

As mentioned in the above sections, function LocalSearch() improves the perturbed solution from function AdaptiveShaking() in every iteration of the AVNS main loops in functions AVNSFeasibilityRecovery() and AVNSDistanceMinimization(). Function LocalSearch() is a local search procedure, which employs six popular neighbourhood structures in the VRP literature. The neighbourhood structures are defined based on the following operators.

- **Intra-Route-Reinsert**. This operator first removes a node from a route and then reinserts it after a different node on the same route.
- **Intra-Route-Swap**. This operator exchanges the positions of two nodes on a route.
- **Intra-Route-2opt**. This operator first selects two nodes on a route and then reverses the sequence from one selected node to the other.
- **Inter-Route-Reinsert**. This operator first removes a node from a route and then reinserts it after a node on a different route.
- **Inter-Route-Swap**. This operator exchanges the positions of two nodes on different routes.
- **Inter-Route-2opt**. This operator first selects two routes and splits each route into two sequences from a node on the route. Then, it reconnects the first sequence of each route with the second sequence of the other route.

In the local search procedure, the six neighbourhood structures are selected in random order. We use the first improvement strategy in the neighbourhood search for the purpose of improving computational efficiency. A neighbourhood structure is reused until the current solution cannot be further improved. The local search procedure is terminated when none of the six neighbourhood structures can further improve the current solution.

As shown in functions AVNSFeasibilityRecovery() and AVNSDistanceMinimization(), the local search procedure which employs a number of neighbourhoods is performed in every iteration of the AVNS main loop. Thus, it can be computationally expensive when dealing with large-sized instances. To speed up the local search procedure, we adapt a pruning approach introduced in Vidal et al. (2013). In this approach, we generate a set of so-called related customers for each customer based on the relatedness measure $R(i,j)$ defined in equation (3.23). Specifically, for each customer $i \in \mathcal{C}$, we select its Ω closest customers $j \in \mathcal{C}$ with respect to the relatedness measure $R(i,j)$ and generate a set of Ω related customers. This set is used to prohibit neighbourhood moves associated with customer i and other customers who are not in the set. Thus, the number of neighbours considered in each neighbourhood can be reduced. We set $\Omega = 40$ when performing local search in all six neighbourhoods. In addition, we observe that most route changes are associated with two nodes (customers) based on the defined operators. Thus, a change tracking method proposed in Benjamin and Beasley (2013) is adopted and slightly extended in the local search procedure to reduce "unnecessary" route evaluations. For the details of the change tracking method, readers can refer to Benjamin and Beasley (2013).

3.3.4.5 Acceptance Criterion

In functions AVNSFeasibilityRecovery() and AVNSDistanceMinimization(), function Accept() determines whether a new candidate solution $\mathcal{R}'$ generated from function LocalSearch() should be accepted or not. In function Accept(), a SA-based acceptance mechanism is designed and its details are as follows. If the cost function value $Cost(\mathcal{R}')$ of the candidate solution $\mathcal{R}'$ is less than the cost function value $Cost(\mathcal{R})$ of the current solution $\mathcal{R}$, solution $\mathcal{R}'$ is always accepted. Otherwise, it is accepted with a probability $exp^{[-(Cost(\mathcal{R}')-Cost(\mathcal{R}))/T]}$, where $T > 0$ denotes the temperature parameter. We initialize T with a value of T_0 and multiply it by a cooling rate $\tau \in (0,1)$ at each main AVNS iteration. We set T_0 to a value such that a solution $\mathcal{R}$ with the cost function value

$Cost(\mathcal{R}) = \chi \cdot Cost(\mathcal{R}^0)$ is accepted with probability 0.5. $\mathcal{R}^0$ denotes the improved input solution in functions AVNSFeasibilityRecovery() and AVNSDistanceMinimization(). χ is a given parameter which helps to determine the initial temperature T_0. For the cooling factor τ, we set it to a value such that the temperature parameter T is below 0.0001 for the last 20% of the maximum allowed iterations in the AVNS main loops in functions AVNSFeasibilityRecovery() and AVNSDistanceMinimization().

3.4 Computational Experiments

This section conducts extensive computational experiments to investigate the effectiveness of the proposed methods and to derive useful managerial insights for real-life VRPTW applications under uncertainty. Section 3.4.1 describes the details of the test instances and discusses the parameter settings of the proposed AVNS-based metaheuristic. Section 3.4.2 presents the computational results along with a detailed analysis.

3.4.1 Experiment Description and Parameter Setting

The computational experiments employ the well-known Solomon's benchmark instances (Solomon, 1987) for the VRPTW. Based on the geographic distribution of customers, the lengths of time windows, and the vehicle capacity, the Solomon's instances can be grouped into six classes, referred to as C1, C2, R1, R2, RC1, and RC2. The instances of classes C1 and C2 contain customers in clusters; those of classes R1 and R2 contain randomly-distributed customers; and those of classes RC1 and RC2 comprise a mixture of them. Moreover, the instances of classes C1, R1, and RC1 have narrow time windows and small-capacity vehicles whereas those of classes C2, R2, and RC2 exhibit the converse. Note that the experiments focus on medium-sized instances with 50 customers and large-sized instances with 100 customers. Each medium-sized instance contains the first 50 customers in the corresponding large-sized instance.

In the VRPTW under uncertainty introduced in Section 3.2.2, customer demands $\tilde{q}_i\,(\forall i \in \mathcal{C})$, service times $\tilde{s}_i\,(\forall i \in \mathcal{C})$, and travel times $\tilde{t}_{ij}\,(\forall (i,j) \in \mathcal{A})$ are uncertain parameters. Moreover, these three types of uncertain parameters respectively take values in the route-dependent uncertainty set $\mathcal{U}_q$ (3.10) defined with polytopes $\mathcal{U}_q^k$ (3.11), set $\mathcal{U}_s$ (3.12) defined with polytopes $\mathcal{U}_s^k$ (3.13), and set $\mathcal{U}_t$ (3.14) defined with polytopes $\mathcal{U}_t^k$ (3.15). As shown in uncertainty polytopes $\mathcal{U}_q^k$, $\mathcal{U}_s^k$, and $\mathcal{U}_t^k$, the nominal values $(\overline{q}_i, \overline{s}_i, \overline{t}_{ij})$ of the uncertain parameters $(\tilde{q}_i, \tilde{s}_i, \tilde{t}_{ij})$ are critical and they must be set by decision-makers (route planners). Thus, we assume that the nominal values of the uncertain customer demands $\tilde{q}_i$, uncertain service times $\tilde{s}_i$, and uncertain travel times $\tilde{t}_{ij}$ in each medium- or large-sized instance are the original values of the deterministic parameters in the corresponding Solomon's instance. In addition, there are two important types of parameters which help to define uncertainty polytopes $\mathcal{U}_q^k$, $\mathcal{U}_s^k$, and $\mathcal{U}_t^k$. The first are the maximum deviations $(\hat{q}_i, \hat{s}_i, \hat{t}_{ij})$ of the uncertain parameters $(\tilde{q}_i, \tilde{s}_i, \tilde{t}_{ij})$, which determine the maximum variation ranges of the uncertain parameters. The second are the uncertainty budget coefficients $(\theta_q, \theta_s, \theta_t)$, which help to determine the uncertainty budgets $(\Gamma_q^k, \Gamma_s^k, \Gamma_t^k)$ in the uncertainty polytopes. In the experiments, we first run the medium-sized instances with different settings for these two types of parameters. For the maximum deviations $(\hat{q}_i, \hat{s}_i, \hat{t}_{ij})$, we consider two different cases. Specifically, we consider a low uncertainty case, in which $\hat{q}_i = 0.2\overline{q}_i$ and $\hat{s}_i = 0.2\overline{s}_i$ for all $i \in \mathcal{C}$ and $\hat{t}_{ij} = 0.2\overline{t}_{ij}$ for all $(i,j) \in \mathcal{A}$. In addition, we consider a high uncertainty case, in which $\hat{q}_i = 0.4\overline{q}_i$ and $\hat{s}_i = 0.4\overline{s}_i$ for all $i \in \mathcal{C}$ and $\hat{t}_{ij} = 0.4\overline{t}_{ij}$ for all $(i,j) \in \mathcal{A}$. For the uncertainty budget coefficients $(\theta_q, \theta_s, \theta_t)$, we consider a total of nine combinations of coefficient values including $(0,0,0)$, $(0.1,0,0)$, $(0.2,0,0)$, $(0,0.1,0)$, $(0,0.2,0)$, $(0,0,0.1)$, $(0,0,0.2)$, $(0.1,0.1,0.1)$, and $(0.2,0.2,0.2)$. These nine combinations can be further classified into five cases: 1) no uncertainty, in which case $(\theta_q, \theta_s, \theta_t) = (0,0,0)$; 2) uncertainty in customer demands, in which case $(\theta_q, \theta_s, \theta_t) = (0.1,0,0)$ or $(0.2,0,0)$; 3) uncertainty in service times, in which case $(\theta_q, \theta_s, \theta_t) = (0,0.1,0)$ or $(0,0.2,0)$; 4) uncertainty in travel times, in which case $(\theta_q, \theta_s, \theta_t) = (0,0,0.1)$ or $(0,0,0.2)$; 5) uncertainty in all three, in which case $(\theta_q, \theta_s, \theta_t) = (0.1,0.1,0.1)$ or $(0.2,0.2,0.2)$. As discussed in Section 3.2.2, no uncertainty is considered in all uncertainty polytopes when the uncertainty budget coefficients $(\theta_q, \theta_s, \theta_t) = (0,0,0)$. Thus, the solution derived with $(\theta_q, \theta_s, \theta_t) = (0,0,0)$ is referred to as the deterministic solution for each medium-sized instance.

To further investigate the efficiency and effectiveness of the proposed AVNS-based metaheuristic, we generate the robust solutions for the large-sized instances with 100 customers based on the following settings in the uncertainty polytopes $\mathcal{U}_q^k$, $\mathcal{U}_s^k$, and $\mathcal{U}_t^k$. Specifically, we only consider the high uncertainty case for the maximum deviations $(\hat{q}_i, \hat{s}_i, \hat{t}_{ij})$ of the uncertain parameters $(\tilde{q}_i, \tilde{s}_i, \tilde{t}_{ij})$, in which $\hat{q}_i = 0.4\overline{q}_i$ and $\hat{s}_i = 0.4\overline{s}_i$ for all $i \in \mathcal{C}$ and $\hat{t}_{ij} = 0.4\overline{t}_{ij}$ for all $(i,j) \in \mathcal{A}$. In addition, we set the uncertainty budget coefficients $(\theta_q, \theta_s, \theta_t) = (0.2,0.2,0.2)$ for the C1, R1, and RC1 instances and $(\theta_q, \theta_s, \theta_t) = (0.1,0.1,0.1)$ for the C2, R2, and RC2 instances. We also solve the large-sized instances with $(\theta_q, \theta_s, \theta_t) = (0,0,0)$ to generate the corresponding deterministic solutions without considering uncertainty. Thus, solutions are derived with different parameter settings in the uncertainty polytopes for both medium- and large-sized instances in the experiments.

All experiments are conducted on a 3.4 GHz Inter core i7 machine with 8G of RAM. The proposed AVNS-based metaheuristic is coded in Matlab 2018b and each test instance is run with the parameter settings for the metaheuristic described below. Note that the parameter values used in the AVNS-based metaheuristic are calibrated based on the corresponding ones chosen from the literature and some preliminary experiments with a small number of test instances. As the two main functions AVNSFeasibilityRecovery() and AVNSDistanceMinimization() of the metaheuristic are based on the AVNS framework, we first describe the parameter settings in the framework. The parameter values which need to be set in the AVNS framework are those associated with the adaptive mechanism, the dynamic penalty mechanism, the solution reset mechanism, and the SA-based acceptance mechanism. The adaptive mechanism introduced in Section 3.3.4.3 is controlled by the scoring values $(\sigma_1, \sigma_2, \sigma_3)$, the reaction factor ϖ, the learning period of η^{aw} iterations. We set these five parameters $(\sigma_1, \sigma_2, \sigma_3, \varpi, \eta^{aw}) = (9, 3, 1, 0.5, 50)$. The dynamic penalty mechanism introduced in Section 3.3.2 is controlled by six parameters including the minimum penalty value ρ^{min}, the maximum penalty value ρ^{max}, the initial penalty value ρ^0, the penalty update factor ρ^{update}, and the penalty update indicators (η^{ins}, η^{des}). We set these six parameters $(\rho^{min}, \rho^{max}, \rho^0, \rho^{update}, \eta^{ins}, \eta^{des}) = (0.1, 10000, 10, 1.1, 2, 2)$. The solution reset mechanism in function SolutionReset() is associated with parameter η^{sr} and we set $\eta^{sr} = 200$. The SA-based acceptance mechanism introduced in Section 3.3.4.5 is controlled by parameter χ and we set $\chi = 1.01$. For the stopping criteria of the VNS main loop, we set a limit of $MaxIter1 = 500$ iterations and $MaxNoImp1 = 200$ iterations without an overall improvement in function AVNSFeasibilityRecovery() and $MaxIter2 = 2500$ and $MaxNoImp2 = 1000$ in function AVNSDistanceMinimization(). Finally, we set a time limit of 300 seconds for the first two phases of the AVNS-based metaheuristic and a time limit of 1800 seconds for implementing the whole solution approach.

To evaluate the robustness of the obtained solutions for all test instances, we design Monte Carlo simulation tests. In the simulation tests, 1000 random and independent scenarios are generated for uncertain customer demands $\tilde{q}_i \, (\forall i \in \mathcal{C})$, service times $\tilde{s}_i \, (\forall i \in \mathcal{C})$, and travel times $\tilde{t}_{ij} \, (\forall (i,j) \in \mathcal{A})$. Since the exact distributions of the uncertain parameters are unknown in the VRPTW under uncertainty, uniform distributions are assumed for the uncertain parameters to generate these scenarios following the ideas used in Munari et al. (2019) and De La Vega et al. (2020). In each scenario, a customer demand vector is generated and its component values are uniformly drawn from the intervals $[\overline{q}_i - \hat{q}_i, \overline{q}_i + \hat{q}_i]$ for all $i \in \mathcal{C}$. In addition, we generate a service time vector consisting of component values which are uniformly drawn from the intervals $[\overline{s}_i - \hat{s}_i, \overline{s}_i + \hat{s}_i]$ for all $i \in \mathcal{C}$. Moreover, a travel time matrix is generated and its component values are also uniformly drawn from the intervals $[\overline{t}_{ij} - \hat{t}_{ij}, \overline{t}_{ij} + \hat{t}_{ij}]$ for all $(i,j) \in \mathcal{A}$. Accordingly, we then design performance indicators to reflect the robustness of a solution from different perspectives.

Feasibility ratio (FR). This indicator reflects the robustness of a given solution in terms of its feasibility ratio. It is calculated as the percentage of the scenarios in which the solution remains feasible over the 1000 randomly generated ones. Given the customer demand vector, the service time vector, and the travel time matrix generated in each scenario, the feasibility of the evaluated solution can be simply determined by examining the vehicle capacity and time window constraints based on the vehicle routes in it.

Average customer dissatisfaction rate (ACDR). This indicator reflects the robustness of a given solution in terms of the customer dissatisfaction rate. The customer dissatisfaction rate associated with a solution is calculated as the percentage of dissatisfied customers over all customers in each generated scenario. Based on the vehicle routes in the evaluated solution, a customer is considered dissatisfied in circumstances as follows: its time window is missed; its demand is unfulfilled or partially fulfilled; and a vehicle having picked up the customer's goods fails to deliver them to the depot within the allowable time window (customer demands are assumed to be pickup demands in the simulation tests). Based on the 1000 generated scenarios, the average customer dissatisfaction rate associated with the evaluated solution can accordingly be computed.

Average unfulfilled demand (AUD). This indicator reflects the robustness of a given solution in terms of the total unfulfilled demands of all customers. In each generated scenario, the unfulfilled demand of a customer can be determined by calculating the load and the arrival time of a vehicle at the customer's location based on the vehicle routes in the evaluated solution. Note that customer demands are assumed to be pickup demands in the simulation tests. Thus, if a vehicle fails to return to the depot within the allowable time window, the demands of the customers who are served by the vehicle are also considered to be unfulfilled. Based on the 1000 generated scenarios, the average unfulfilled demand associated with the evaluated solution can accordingly be computed.

Note that robustness indicators FR and ACDR will be used with similar definitions in the Monte Carlo simulation tests designed for the VRPSPDTW under uncertainty in

Section 4.4.1 and indicators FR and AUD will be used with similar definitions in the simulation tests designed for the 2E-MTVRPTWSS under uncertainty in Section 5.4.1.

3.4.2 Computational Results

In this section, we first present the computational results for the medium-sized instances with 50 customers. Interesting findings are observed and the impacts of different types of uncertainty are investigated based on the solutions derived with different parameter settings in the uncertainty polytopes. Next, we show the computational results for the large-sized instances with 100 customers. The advantages of the proposed formulation and the uncertainty sets are highlighted by comparing the derived robust solutions with their deterministic counterparts. In addition, the effectiveness of the AVNS-based metaheuristic is demonstrated by comparing the obtained best deterministic solutions with the current best-known solutions for the Solomon's benchmark instances in the literature.

3.4.2.1 Results for Medium-Sized Instances

As introduced in Section 3.4.1, the Solomon's instances of classes C1, R1, and RC1 have narrow time windows, short planning horizons, and small-capacity vehicles. Thus, we first detail the average results of the solutions generated with different parameter settings in the uncertainty polytopes for the medium-sized instances of classes C1, R1, and RC1 in Tables 3.2, 3.3, and 3.4, respectively. In these tables, average results of the obtained solutions are presented in terms of the number of vehicles used (NV), the travel distance (TD), and three robustness indicators (FR, ACDR, and AUD) from the Monte Carlo simulation tests. As introduced in Section 3.4.1, we consider nine value combinations for the uncertainty budget coefficients $(\theta_q, \theta_s, \theta_t)$ and two cases for the maximum deviations $(\hat{q}_i, \hat{s}_i, \hat{t}_{ij})$ of the uncertain parameters $(\tilde{q}_i, \tilde{s}_i, \tilde{t}_{ij})$ in the uncertainty polytopes. Given every setting for the uncertainty budget coefficients $(\theta_q, \theta_s, \theta_t)$ and the maximum deviations $(\hat{q}_i, \hat{s}_i, \hat{t}_{ij})$, each instance (e.g., instance C101) of a class (e.g., class C1) is tested ten times and the best solution from ten runs is selected for the instance. The average results for all instances of a class is calculated based on the selected best solutions for all instances of that class under each considered setting for the uncertainty budget coefficients $(\theta_q, \theta_s, \theta_t)$ and the maximum deviations $(\hat{q}_i, \hat{s}_i, \hat{t}_{ij})$. Note that instances R101, R102, R103, and R104 of class R1 and instance RC105 of class RC1 are discarded. Because feasible robust solutions do not exist when solving these instances with some considered settings for the uncertainty budget coefficients $(\theta_q, \theta_s, \theta_t)$ and the maximum deviations $(\hat{q}_i, \hat{s}_i, \hat{t}_{ij})$.

Table 3.2. Average results of the solutions derived with different parameter settings in the uncertainty polytopes for the C1 instances with 50 customers.

$(\theta_q, \theta_s, \theta_t)$	Low uncertainty $(\hat{q}_i = 0.2\overline{q}_i, \hat{s}_i = 0.2\overline{s}_i, \hat{t}_{ij} = 0.2\overline{t}_{ij})$					High uncertainty $(\hat{q}_i = 0.4\overline{q}_i, \hat{s}_i = 0.4\overline{s}_i, \hat{t}_{ij} = 0.4\overline{t}_{ij})$				
	NV	TD	FR (%)	ACDR (%)	AUD	NV	TD	FR (%)	ACDR (%)	AUD
(0,0,0)	5.00	362.52	21.77	5.80	44.10	5.00	362.52	9.31	14.37	117.09
(0.1,0,0)	5.00	394.98	36.39	5.84	47.84	5.22	433.31	11.17	15.92	125.15
(0.2,0,0)	5.22	432.67	39.71	5.22	43.08	5.22	498.18	15.40	10.92	88.58
(0,0.1,0)	5.22	386.35	27.20	3.12	21.79	5.78	432.17	20.18	6.12	47.03
(0,0.2,0)	5.67	409.70	39.23	1.79	7.54	6.22	506.75	50.92	2.29	13.81
(0,0,0.1)	5.00	362.52	21.77	5.80	44.10	5.00	362.52	9.31	14.37	117.09
(0,0,0.2)	5.00	362.52	21.77	5.80	44.10	5.11	372.70	10.26	13.21	106.36
(0.1,0.1,0.1)	5.22	424.11	62.20	1.52	9.68	6.22	484.18	37.99	3.87	29.62
(0.2,0.2,0.2)	5.67	458.46	90.46	0.29	2.19	6.78	525.37	79.13	0.69	4.52

Table 3.3. Average results of the solutions derived with different parameter settings in the uncertainty polytopes for the R1 instances with 50 customers.

$(\theta_q, \theta_s, \theta_t)$	Low uncertainty $(\hat{q}_i = 0.2\overline{q}_i, \hat{s}_i = 0.2\overline{s}_i, \hat{t}_{ij} = 0.2\overline{t}_{ij})$					High uncertainty $(\hat{q}_i = 0.4\overline{q}_i, \hat{s}_i = 0.4\overline{s}_i, \hat{t}_{ij} = 0.4\overline{t}_{ij})$				
	NV	TD	FR (%)	ACDR (%)	AUD	NV	TD	FR (%)	ACDR (%)	AUD
(0,0,0)	6.50	766.01	11.03	13.99	94.56	6.50	766.01	4.05	24.09	168.61
(0.1,0,0)	6.50	766.01	11.03	13.99	94.56	6.50	766.77	4.05	24.34	167.87
(0.2,0,0)	6.50	766.01	11.03	13.99	94.56	6.50	766.46	4.05	24.36	172.96
(0,0.1,0)	6.75	765.24	45.03	6.94	51.54	6.88	776.93	17.79	13.14	96.12
(0,0.2,0)	6.88	772.27	49.74	4.38	31.23	7.13	773.74	42.44	5.21	34.64
(0,0,0.1)	7.00	762.12	65.23	1.93	12.46	7.25	781.08	47.19	4.39	30.86
(0,0,0.2)	7.13	767.04	89.60	0.50	3.78	7.38	789.23	75.09	1.29	7.96
(0.1,0.1,0.1)	7.13	777.14	88.70	0.52	3.68	7.50	782.24	80.89	1.30	8.97
(0.2,0.2,0.2)	7.38	774.30	98.36	0.09	0.61	8.13	808.67	97.29	0.19	1.38

Table 3.4. Average results of the solutions derived with different parameter settings in the uncertainty polytopes for the RC1 instances with 50 customers.

$(\theta_q,\theta_s,\theta_t)$	Low uncertainty $(\hat{q}_i = 0.2\overline{q}_i, \hat{s}_i = 0.2\overline{s}_i, \hat{t}_{ij} = 0.2\overline{t}_{ij})$					High uncertainty $(\hat{q}_i = 0.4\overline{q}_i, \hat{s}_i = 0.4\overline{s}_i, \hat{t}_{ij} = 0.4\overline{t}_{ij})$				
	NV	TD	FR (%)	ACDR (%)	AUD	NV	TD	FR (%)	ACDR (%)	AUD
(0,0,0)	6.29	713.76	9.33	10.32	84.20	6.29	713.76	1.29	20.46	185.88
(0.1,0,0)	6.57	799.94	13.03	11.46	107.24	6.57	813.95	3.57	19.03	179.14
(0.2,0,0)	6.57	813.95	21.49	9.21	85.90	6.71	829.14	5.13	16.37	149.66
(0,0.1,0)	6.29	719.63	11.99	8.00	62.43	6.29	731.77	2.69	19.26	172.71
(0,0.2,0)	6.29	731.54	13.96	7.78	57.65	6.71	749.29	11.50	12.11	103.53
(0,0,0.1)	6.71	774.05	31.39	3.15	16.68	7.57	852.14	32.09	5.38	47.83
(0,0,0.2)	7.00	779.67	50.56	1.81	6.25	7.86	881.48	55.99	1.81	15.79
(0.1,0.1,0.1)	7.14	821.05	80.41	0.83	6.11	8.43	930.16	76.34	1.68	14.40
(0.2,0.2,0.2)	7.57	854.62	99.23	0.02	0.04	8.71	960.14	94.84	0.19	1.68

Based on the results in Tables 3.2, 3.3, and 3.4, interesting findings are observed and useful managerial insights for real-life VRPTW applications under uncertainty are derived.

1) The deterministic solutions generated without considering uncertainty ($(\theta_q,\theta_s,\theta_t) = (0,0,0)$) for the medium-sized instances of classes C1, R1, and RC1 are fragile. Note that no uncertainty is considered in all uncertainty polytopes when the uncertainty budget coefficients $(\theta_q,\theta_s,\theta_t) = (0,0,0)$ and the solution derived with $(\theta_q,\theta_s,\theta_t) = (0,0,0)$ is referred to as the deterministic solution for each medium-sized instance. As shown in Table 3.4, the average FR of the deterministic solutions for the RC1 instances is only 9.33% and the average ACDR with these solutions is more than 10% in the Monte Carlo simulation tests even considering the low uncertainty case. Similar observations are made for the deterministic solutions derived for the C1 and R1 instances in the simulation tests considering both low and high uncertainty cases. Thus, deterministic routing strategies are very likely to become infeasible for real-life VRPTW applications with narrow time windows and small-capacity vehicles under multiple types of uncertainty. Moreover, these strategies may lead to high customer dissatisfaction rates and large unfulfilled customer demands.

2) The robust solutions generated considering all three types of uncertainty ($(\theta_q,\theta_s,\theta_t) = (0.1,0.1,0.1)$ or $(0.2,0.2,0.2)$) for the C1, R1, and RC1 instances with 50 customers attain relatively high robustness in terms of the indicators FR, ACDR, and AUD in the Monte Carlo simulation tests. Consider the robust solutions derived for the R1 instances with $(\theta_q,\theta_s,\theta_t) = (0.2,0.2,0.2)$ in the low uncertainty case (Table 3.3). Their average FR is 98.36% and their average ACDR is only 0.09%. Similar observations are made for the robust solutions generated with $(\theta_q,\theta_s,\theta_t) = (0.2,0.2,0.2)$ for the C1 and RC1 instances in both low and high uncertainty cases. However, compared to their deterministic counterparts, the average number of vehicles used and the average travel distance in the robust solutions for the C1, R1, and RC1 instances have a large increase especially in the high uncertainty case. As shown in Table 3.2, the average number of vehicles used increases from 5.00 to 6.78 and the average travel distance increases from 362.52 to 525.37 in the robust solutions generated with $(\theta_q,\theta_s,\theta_t) = (0.2,0.2,0.2)$ for the C1 instances. Thus, robust routing strategies can be very reliable for practical VRPTW applications with narrow time windows and small-capacity vehicles under multiple types of uncertainty. Moreover, these strategies generally lead to low customer dissatisfaction rates and small unfulfilled customer demands. However, they may incur a large additional cost in terms of the number of vehicles used and the travel distance under high uncertainty, compared to their deterministic counterparts.

3) Uncertainty in customer demands and service times have an obvious impact on deriving robust solutions for the C1 instances. Look at the average results of the robust solutions generated considering only service time uncertainty ($(\theta_q,\theta_s,\theta_t) = (0,0.1,0)$ or $(0,0.2,0)$) in Table 3.2. These robust solutions exhibit greater robustness than their deterministic counterparts in terms of the average FR, ACDR, and AUD in both low and high uncertainty cases. Similar observations are made for the robust solutions generated considering only demand uncertainty ($(\theta_q,\theta_s,\theta_t) = (0.1,0,0)$

or $(0.2, 0, 0)$). However, both the deterministic and robust solutions derived for the C1 instances are not sensitive to travel time uncertainty. As shown in Table 3.2, the average results of the robust solutions generated considering only travel time uncertainty ($(\theta_q, \theta_s, \theta_t) = (0, 0, 0.1)$ or $(0, 0, 0.2)$) are very close to those of their deterministic counterparts in both low and high uncertainty cases. The reason is that the C1 instances contain customers in clusters. The customers in the same cluster are located close to each other and they are usually on the similar routes in both the deterministic and robust solutions. Thus, uncertainty in customer demands and service times are key factors when generating robust routing strategies for real-life VRPTW applications which are similar to the C1 instances in uncertain environments.

4) Uncertainty in service times and travel times have an obvious impact on deriving robust solutions for the R1 instances. However, the obtained deterministic and robust solutions for the R1 instances are almost insensitive to customer demand uncertainty. As shown in Table 3.3, the average results of the robust solutions generated considering only demand uncertainty are almost the same as those of their deterministic counterparts in both low and high uncertainty cases. This is because narrow time windows and short scheduling horizons are commonly seen in the R1 instances. Thus, fewer customers are served on each route in the obtained solutions and demand uncertainty can be offset by vehicle capacity. The deterministic solutions for the R1 instances are very sensitive to travel time uncertainty. However, the robust solutions generated considering only travel time uncertainty exhibit substantial robustness in terms of the indicators FR, ACDR, and AUD. Similar observations are made for the robust solutions generated considering only service time uncertainty. Moreover, uncertainty in travel times has a larger impact on deriving robust solutions for the R1 instances than that in service times. Thus, decision-makers should pay more attention to travel and service time uncertainty when deriving robust routing

strategies for practical VRPTW applications which are similar to the R1 instances in uncertain environments.

5) All three types of uncertainty have an impact on deriving robust solutions for the RC1 instances. However, uncertainty in travel times has a larger impact than that in customer demands and service times. As shown in Table 3.4, the robust solutions generated considering only travel time uncertainty exhibit higher robustness than those generated considering only demand or service time uncertainty. For example, the average ACDR of the robust solutions generated with $(\theta_q, \theta_s, \theta_t) = (0, 0, 0.2)$ is only 1.81% in the high uncertainty case. The values of the same robustness indicator are respectively 16.37% and 12.11% associated with the robust solutions generated with $(\theta_q, \theta_s, \theta_t) = (0.2, 0, 0)$ and $(\theta_q, \theta_s, \theta_t) = (0, 0.2, 0)$ in the high uncertainty case. Thus, all three types of uncertainty should be considered when deriving robust routing strategies for real-life VRPTW applications which are similar to the RC1 instances in uncertain environments. In addition, decision-makers may need to pay more attention to travel time uncertainty.

6) Increasing the values of the uncertainty budget coefficients $(\theta_q, \theta_s, \theta_t)$ generally can improve the robustness of the robust solutions for the C1, R1, and RC1 instances. However, it may also increase their routing costs in terms of the number of vehicles used and the travel distance. Consider the robust solutions derived for the C1 instances (Table 3.2). When the uncertainty budget coefficients $(\theta_q, \theta_s, \theta_t)$ increase from $(0.1, 0.1, 0.1)$ to $(0.2, 0.2, 0.2)$, their average FR increases from 37.99% to 79.13% and their average ACDR decreases from 3.87% to 0.69% in the high uncertainty case. However, the average number of vehicles used increases from 6.22 to 6.78 and the average travel distance increases from 484.18 to 525.37 in the robust solutions. Thus, decision-makers may need to select appropriate values for the uncertainty budget coefficients to generate robust routing strategies which have a

good balance between robustness and routing cost for practical VRPTW applications with narrow time windows and small-capacity vehicles under uncertainty.

7) Given the same uncertainty budget coefficients $(\theta_q, \theta_s, \theta_t)$, the robust solutions generated in the high uncertainty case can be much more expensive than those generated in the low uncertainty case for the C1, R1, and RC1 instances. Consider the robust solutions generated with $(\theta_q, \theta_s, \theta_t) = (0.2, 0.2, 0.2)$ for the RC1 instances in both low and high uncertainty cases (Table 3.4). The average number of vehicles used increases from 7.57 to 8.71 and the average travel distance increases from 854.62 to 960.14 in such solutions. Thus, robust routing strategies for real-life VRPTW applications with narrow time windows and small-capacity vehicles may become very expensive when the level of uncertainty is high.

Contrary to the instances of classes C1, R1, and RC1, the instances of classes C2, R2, and RC2 have wide time windows, long scheduling horizons, and large-capacity vehicles. Next, we present the average results of the solutions generated with different parameter settings in the uncertainty polytopes for the C2, R2, and RC2 instances with 50 customers in Tables 3.5, 3.6, and 3.7, respectively. The headings NV, TD, FR, ACDR, and AUD have the same meaning as those in Tables 3.2-3.4. Several different findings are observed and some practical routing suggestions are provided.

1) The deterministic solutions generated without considering uncertainty ($(\theta_q, \theta_s, \theta_t) = (0, 0, 0)$) for the medium-sized instances of classes C2, R2, and RC2 are not very fragile. The average FRs of the deterministic solutions for the C2, R2, and RC2 instances are respectively 43.54%, 53.20%, and 33.14% in the Monte Carlo simulation tests even considering the high uncertainty case. Moreover, the average ACDRs of such solutions for the R2 and RC2 instances are only 1.93% and 3.71%, respectively. Thus, deterministic routing strategies for real-life VRPTW applications with wide

time windows and large-capacity vehicles are not easily to be affected by multiple types of uncertainty.

2) The robust solutions generated considering all three types of uncertainty for the C2, R2, and RC2 instances with 50 customers can attain high robustness at a small additional cost in terms of the travel distance and the number of vehicles used. Consider the robust solutions generated with $(\theta_q, \theta_s, \theta_t) = (0.2, 0.2, 0.2)$ for the R2 instances in the high uncertainty case (Table 3.6). Their average FR is 100% and their average ACDR and AUD are all 0. Moreover, compared to their deterministic counterparts, the average number of vehicles used only increases from 2.00 to 2.18 and the average travel distance only increases from 662.74 to 673.74 in the robust solutions. Similar observations are made for the robust solutions generated with $(\theta_q, \theta_s, \theta_t) = (0.1, 0.1, 0.1)$ or $(0.2, 0.2, 0.2)$ for the C2 and RC2 instances in both low and high uncertainty cases. Thus, highly robust routing strategies can be obtained at only a small additional cost for practical VRPTW applications with wide time windows and large-capacity vehicles under multiple types of uncertainty.

3) Uncertainty in customer demands does not have an obvious impact on deriving robust solutions for the instances of classes C2, R2, and RC2. As shown in Table 3.5, the average results of the robust solutions derived for the C2 instances considering only demand uncertainty ($(\theta_q, \theta_s, \theta_t) = (0.1, 0, 0)$ or $(0.2, 0, 0)$) are almost the same as those of their deterministic counterparts in both low and high uncertainty cases. Similar observations are made for the robust solutions generated considering only demand uncertainty for the R2 and RC2 instances (Tables 3.6 and 3.7). Thus, decision-makers may neglect demand uncertainty when generating robust routing strategies for real-life VRPTW applications with wide time windows and large-capacity vehicles in uncertain environments.

4) Uncertainty in service times and travel times have an impact on deriving robust solutions for the C2, R2, and RC2 instances. Specifically, uncertainty in service times has a larger impact on deriving robust solutions for the C2 instances than that in travel times. As shown in Table 3.5, the robust solutions generated for the C2 instances considering only service time uncertainty exhibit greater robustness than those generated considering only travel time uncertainty in both low and high uncertainty cases. On the contrary, uncertainty in travel times has a larger impact on deriving robust solutions for the R2 and RC2 instances than that in service times. As shown in Tables 3.6 and 3.7, the robust solutions generated considering only travel time uncertainty for the R2 and RC2 instances exhibit higher level of robustness than those generated considering only service time uncertainty. Thus, decision-makers should pay more attention to service time uncertainty when generating robust routing strategies for practical VRPTW applications which are similar to the C2 instances. In addition, they should pay more attention to travel time uncertainty when deriving such strategies for VRPTW applications which are similar to the R2 and RC2 instances in uncertain environments.

Table 3.5. Average results of the solutions derived with different parameter settings in the uncertainty polytopes for the C2 instances with 50 customers.

$(\theta_q, \theta_s, \theta_t)$	Low uncertainty $(\hat{q}_i = 0.2\overline{q}_i, \hat{s}_i = 0.2\overline{s}_i, \hat{t}_{ij} = 0.2\overline{t}_{ij})$					High uncertainty $(\hat{q}_i = 0.4\overline{q}_i, \hat{s}_i = 0.4\overline{s}_i, \hat{t}_{ij} = 0.4\overline{t}_{ij})$				
	NV	TD	FR (%)	ACDR (%)	AUD	NV	TD	FR (%)	ACDR (%)	AUD
(0,0,0)	2.00	398.98	66.35	3.95	37.03	2.00	398.98	43.54	12.50	119.86
(0.1,0,0)	2.00	398.98	66.35	3.95	37.03	2.00	398.98	43.54	12.50	119.86
(0.2,0,0)	2.00	398.98	66.35	3.95	37.03	2.00	406.81	42.49	10.65	100.19
(0,0.1,0)	2.00	402.26	87.21	1.95	19.37	2.00	414.27	81.88	4.16	40.64
(0,0.2,0)	2.00	412.94	99.61	0.02	0.18	2.13	450.34	98.41	0.14	1.26
(0,0,0.1)	2.00	401.84	86.38	2.27	22.39	2.00	401.84	54.64	10.34	100.27
(0,0,0.2)	2.00	401.84	86.38	2.27	22.39	2.00	401.84	54.64	10.34	100.27
(0.1,0.1,0.1)	2.00	409.92	98.04	0.15	1.50	2.00	416.93	85.00	2.94	28.78
(0.2,0.2,0.2)	2.00	414.80	99.63	0.02	0.18	2.25	435.64	98.06	0.08	0.66

Table 3.6. Average results of the solutions derived with different parameter settings in the uncertainty polytopes for the R2 instances with 50 customers.

$(\theta_q, \theta_s, \theta_t)$	Low uncertainty $(\hat{q}_i = 0.2\overline{q}_i, \hat{s}_i = 0.2\overline{s}_i, \hat{t}_{ij} = 0.2\overline{t}_{ij})$					High uncertainty $(\hat{q}_i = 0.4\overline{q}_i, \hat{s}_i = 0.4\overline{s}_i, \hat{t}_{ij} = 0.4\overline{t}_{ij})$				
	NV	TD	FR (%)	ACDR (%)	AUD	NV	TD	FR (%)	ACDR (%)	AUD
(0,0,0)	2.00	662.74	67.73	1.02	7.89	2.00	662.74	53.20	1.93	14.92
(0.1,0,0)	2.00	662.74	67.73	1.02	7.89	2.00	662.74	53.20	1.93	14.92
(0.2,0,0)	2.00	662.74	67.73	1.02	7.89	2.00	662.74	53.20	1.93	14.92
(0,0.1,0)	2.00	667.18	92.20	0.17	1.65	2.00	673.41	84.43	0.54	3.90
(0,0.2,0)	2.00	669.75	94.81	0.12	1.17	2.00	676.54	94.61	0.13	1.00
(0,0,0.1)	2.00	673.83	98.41	0.03	0.23	2.09	670.17	98.45	0.04	0.32
(0,0,0.2)	2.00	676.07	99.85	0.00	0.03	2.09	674.18	99.74	0.01	0.06
(0.1,0.1,0.1)	2.00	683.03	99.99	0.00	0.00	2.09	686.66	99.78	0.00	0.05
(0.2,0.2,0.2)	2.09	667.79	100.0	0.00	0.00	2.18	673.74	100.0	0.00	0.00

Table 3.7. Average results of the solutions derived with different parameter settings in the uncertainty polytopes for the RC2 instances with 50 customers.

$(\theta_q, \theta_s, \theta_t)$	Low uncertainty $(\hat{q}_i = 0.2\overline{q}_i, \hat{s}_i = 0.2\overline{s}_i, \hat{t}_{ij} = 0.2\overline{t}_{ij})$					High uncertainty $(\hat{q}_i = 0.4\overline{q}_i, \hat{s}_i = 0.4\overline{s}_i, \hat{t}_{ij} = 0.4\overline{t}_{ij})$				
	NV	TD	FR (%)	ACDR (%)	AUD	NV	TD	FR (%)	ACDR (%)	AUD
(0,0,0)	2.13	723.34	48.01	2.06	17.54	2.13	723.34	33.14	3.71	33.21
(0.1,0,0)	2.13	723.34	48.01	2.06	17.54	2.13	723.34	33.14	3.71	33.21
(0.2,0,0)	2.13	723.34	48.01	2.06	17.54	2.13	723.34	33.14	3.71	33.21
(0,0.1,0)	2.25	709.31	81.39	0.53	5.87	2.25	711.74	62.00	1.28	13.18
(0,0.2,0)	2.25	710.30	84.71	0.34	3.28	2.25	717.63	72.64	1.55	15.02
(0,0,0.1)	2.25	722.86	98.75	0.03	0.28	2.38	736.55	98.90	0.03	0.28
(0,0,0.2)	2.25	723.44	98.99	0.02	0.22	2.50	714.09	97.74	0.05	0.40
(0.1,0.1,0.1)	2.25	724.51	99.96	0.00	0.01	2.38	765.15	99.79	0.01	0.07
(0.2,0.2,0.2)	2.25	738.68	100.0	0.00	0.00	2.50	742.94	100.0	0.00	0.00

3.4.2.2 Results for Large-Sized Instances

To investigate the effectiveness of the AVNS-based metaheuristic, we generate the robust solutions for the large-sized Solomon's instances with 100 customers using the following parameter settings in the uncertainty polytopes. As introduced in Section 3.4.1, we set the uncertainty budget coefficients $(\theta_q, \theta_s, \theta_t) = (0.2, 0.2, 0.2)$ for the C1, R1, and RC1 instances and $(\theta_q, \theta_s, \theta_t) = (0.1, 0.1, 0.1)$ for the C2, R2, and RC2 instances. For the maximum deviations $(\hat{q}_i, \hat{s}_i, \hat{t}_{ij})$ of the uncertain parameters $(\tilde{q}_i, \tilde{s}_i, \tilde{t}_{ij})$, we only consider the high uncertainty case, in which $\hat{q}_i = 0.4\overline{q}_i$ and $\hat{s}_i = 0.4\overline{s}_i$ for all $i \in \mathcal{C}$ and $\hat{t}_{ij} = 0.4\overline{t}_{ij}$ for all $(i,j) \in \mathcal{A}$. We also solve the large-sized instances with $(\theta_q, \theta_s, \theta_t) = (0,0,0)$ by using the AVNS-based metaheuristic with slight modifications to generate the corresponding deterministic solutions. The average results of the deterministic and robust solutions for the Solomon's instances of each class with 100 customers are shown in Table 3.8. The headings NV, TD, FR, ACDR, and AUD have the same meaning as those in Tables 3.2-3.7. Note that every instance of a class is tested ten times and the best solution from ten runs is selected for the instance given the robust or deterministic setting. The average results of the robust (deterministic) solutions for all instances of each class are calculated based on the selected best robust (deterministic) solution for every instance of that class.

Table 3.8. Average results of the deterministic and robust solutions derived for the Solomon's instances with 100 customers.

Instance class	Solution type	NV	TD	FR (%)	ACDR (%)	AUD
C1	Deterministic	10.00	828.38	0.61	17.77	260.10
	Robust	13.11	1204.43	67.12	0.70	11.58
C2	Deterministic	3.00	589.86	19.98	23.26	416.52
	Robust	4.00	637.43	67.09	3.28	57.98
R1	Deterministic	10.63	1141.80	0.03	30.09	436.73
	Robust	12.88	1256.04	93.58	0.25	3.39
R2	Deterministic	2.73	953.56	11.45	8.38	122.92
	Robust	3.09	969.80	99.80	0.00	0.03
RC1	Deterministic	11.29	1350.00	0.01	27.87	475.43
	Robust	15.43	1603.55	93.40	0.16	2.31
RC2	Deterministic	3.25	1119.71	11.59	6.64	110.43
	Robust	3.38	1206.43	99.78	0.01	0.17

As shown in Table 3.8, the deterministic solutions derived for the large-sized instances of all classes are very fragile in terms of the robustness indicators FR, ACDR, and AUD in the Monte Carlo simulation tests considering the high uncertainty case. For example, the average FRs of such solutions for the R1 and RC1 instances are close to 0 and their average ACDRs are around 30%. Moreover, the average FR of the deterministic solutions for the RC2 instances with wide time windows and large-capacity vehicles is less than 12% and their average ACDR is more than 6%. Similar observations are made for the deterministic solutions derived for the C2 and R2 instances. Thus, deterministic routing strategies can be highly unreliable for practical VRPTW applications under multiple types of uncertainty. However, the robust solutions derived for the large-sized instances of most classes show greater robustness than their deterministic counterparts. For example, the average FR of the robust solutions for the RC1 instances is 93.40% and their average ACDR is only 0.16%. Similar observations are made for the robust solutions derived for the R1 and C1 instances. Despite their robustness for the C1, R1, and RC1 instances with narrow time windows and small-capacity vehicles, the robust solutions may incur significant additional routing costs in terms of the travel distance and the number of vehicles used, compared to their deterministic counterparts. Unlike the robust solutions for the C1, R1, and RC1 instances, such solutions for the C2, R2, and RC2 instances can attain relatively high robustness at only a small addition cost. For example, the average FR of the robust solutions for the RC2 instances reaches 99.78% and their average ACDR is almost 0. Compared to their deterministic counterparts, the average travel distance increases from 1119.71 to 1206.43 and the average number of vehicles used only increases from 3.25 to 3.38 in these robust solutions. Similar observations are made for the robust solutions derived for the C2 and R2 instances. Thus, robust routing strategies can be reliable and cost-effective for large-scale practical VRPTW applications with wide time windows and large-capacity vehicles. However, such strategies may be very expensive (e.g., more vehicles and much longer travel distances are required) to reach a high level of robustness for VRPTW applications with narrow time windows and small-capacity vehicles under high uncertainty.

To show the effectiveness and efficiency of the AVNS-based metaheuristic, we first report the average results of the best and average robust solutions among ten runs in terms of the number of vehicles used (NV) and the travel distance (TD) for the Solomon's instances of each class with 100 customers in Table 3.9. Then, we compare the above obtained best deterministic solutions with the current best-known solutions for the Solomon's instances in the literature. The best-known solutions are obtained by combining the best solutions published in Vidal et al. (2013) and Marinakis et al. (2019). The comparison results between the best solutions found by the AVNS-based metaheuristic and the current best-known solutions for the Solomon's instances are shown in Table 3.10.

Table 3.9. Average results of the best and average robust solutions derived for the Solomon's instances with 100 customers.

Instance class	Best robust solutions			Average robust solutions		
	NV	TD	Time (s)	NV	TD	Time (s)
C1	13.11	1204.43	1134.54	13.27	1192.75	1033.65
C2	4.00	637.43	1636.73	4.00	637.45	1663.94
R1	12.88	1256.04	866.10	13.09	1255.29	843.97
R2	3.09	969.80	1800.02	3.10	980.29	1800.02
RC1	15.43	1603.55	708.66	15.60	1608.12	684.54
RC2	3.38	1206.43	1770.15	3.46	1209.15	1785.21

Table 3.10. Best deterministic solutions generated by the AVNS-based metaheuristic in comparison to the current best-known solutions for the Solomon's instances with 100 customers.

Instance class	Best-known solutions		AVNS-based metaheuristic		Gap	
	NV	TD	NV	TD	NV	TD (%)
C1	10.00	828.38	10.00	828.38	0.00	0.00
C2	3.00	589.86	3.00	589.86	0.00	0.00
R1	11.92	1210.34	11.92	1214.34	0.00	0.33
R2	2.73	951.03	2.73	953.56	0.00	0.27
RC1	11.50	1384.16	11.50	1384.93	0.00	0.06
RC2	3.25	1119.24	3.25	1119.71	0.00	0.04

In Table 3.9, we observe that the average results of the best and average robust solutions derived for the large-sized instances of each class are very close. Moreover, the average computing times to derive such solutions for the instances of most classes are less than 1800 seconds. Thus, we can conclude that the performance of the AVNS-based metaheuristic is stable and it can generate good-quality robust solutions for the large-sized instances within a reasonable running time. In Table 3.10, the columns under "Gap" show the average gaps between the best solutions found by the AVNS-based metaheuristic and the current best-known solutions for the Solomon's instances of each class in terms of the number of vehicles used (NV) and the travel distance (TD) in percentage. We observe that the AVNS-based metaheuristic is able to find all best-known solutions for the C1 and C2 instances. Moreover, the average results of the best deterministic solutions generated by the AVNS-based metaheuristic are very close to the average results of the current best-known solutions for the R1, R2, RC1, and RC2 instances. Thus, the AVNS-based metaheuristic is comparable to the state-of-the-art heuristic solution methods proposed for the standard VRPTW in Vidal et al. (2013) and Marinakis et al. (2019).

3.5 Summary

In this chapter, we study the VRPTW with the consideration of uncertainty in customer demands, service times, and travel times. To capture these different types of uncertainty, novel uncertainty sets are defined with route-dependent uncertainty polytopes. We present a robust mathematical formulation with the defined uncertainty sets to model the problem and propose an AVNS-based metaheuristic to solve it. Extensive computational experiments are conducted which employ the Solomon's benchmark instances with 50 and 100 customers. Both deterministic and robust solutions are generated for the test instances and they are further evaluated through Monte Carlo simulation tests. To investigate the performance of the AVNS-based metaheuristic, we compare the obtained best deterministic solutions with the current best-known solutions for the Solomon's benchmark instances with 100 customers in the literature. The comparison results show that the AVNS-based metaheuristic is comparable to the state-of-the-art heuristic solution methods for the standard VRPTW. A detailed analysis of the computational results is also performed to highlight the features of the robust formulation and the defined uncertainty sets and polytopes. Moreover, useful managerial insights are derived to help generating effective routing strategies for real-life VRPTW applications under uncertainty.

Chapter 4

Vehicle Routing Problem with Simultaneous Pickup and Delivery and Time Windows Under Uncertainty

4.1 Introduction

Against the background of industry-wide competition and increasing environmental awareness, many logistics companies have incorporated reverse logistics into their operations to both increase operating profits and comply with stringent environmental regulations. Reverse logistics has found much utility in many real-life applications. In home healthcare services, the caregivers may need to finish a variety of logistical activities in a single day: they may first delivery drugs from a pharmacy to patients within specific time slots, and then collect patients' medical test samples and ship them to a laboratory. In some milk collection logistics, trucks travelling to farms in remote places may need to simultaneously collect raw milk from the farmers and deliver diesel fuel to them within required time slots. Accordingly, the VRPSPDTW can in general be considered as the mathematical basis of these real-life applications.

Uncertainty is prevalent in many real-life VRPSPDTW applications. For home healthcare services, uncertainty can be noted for travel times of caregivers due to dynamic traffic and weather conditions. For some simultaneous milk pickup and diesel delivery arrangement, the amount of raw milk to be collected from each farm is unknown until the truck arrives there. To help logistics practitioners generate effective routing strategies for these applications, we study the VRPSPDTW with the consideration of uncertainty in pickup demands and travel times. These two types of uncertainty are captured by two route-dependent uncertainty sets similar to those defined in Chapter 3. Given the complexity of the VRPSPDTW, uncertainty in service times is not considered, despite its presence in some real-life VRPSPDTW applications. Compared to the VRPTW under uncertainty in Chapter 3, the VRPSPDTW under uncertainty is more challenging because the concurrence of pickup and delivery of freight often leads to frequent variations in vehicle loads. We present a robust mathematical formulation to model the problem and propose an ALNS-based metaheuristic to solve it. Extensive numerical experiments are conducted which employ the benchmark instances introduced in Wang and Chen (2012) for the standard VRPSPDTW.

The rest of this chapter is organized as follows. Section 4.2 first introduces the VRPSPDTW and presents a deterministic mathematical formulation. Then, the VRPSPDTW under uncertainty is described and the robust mathematical formulation with the route-dependent uncertainty sets is presented to model the problem. Section 4.3 describes the detailed procedure of the ALNS-based metaheuristic. Section 4.4 presents the computational results from extensive numerical experiments. Finally, Section 4.5 summarizes the whole chapter.

4.2 Problem Statement and Model Formulation

4.2.1 The VRPSPDTW

The VRPSPDTW can be described as follows. A fleet of homogeneous vehicles at a central depot need to serve a set of geographically dispersed customers. Each customer has a pickup demand and a delivery demand for a certain amount of goods, which must be satisfied simultaneously by a vehicle within a specified time window. Each customer is allowed to be visited only once by exactly one vehicle. In addition, each vehicle should start from the depot with the goods it must delivery and return to the same depot with the goods it has picked up. Moreover, the load of a vehicle cannot exceed its capacity along its route. The solution of the VRPSPDTW needs to design a set of routes for the vehicles, such that all customers are served, the time window and vehicle capacity constraints are satisfied, and the total routing cost (in terms of the number of vehicles used and the total travel distance) is minimized.

The above described problem can be defined on a directed graph $\mathcal{G} = (\mathcal{N}, \mathcal{A})$ with the node set $\mathcal{N} = \{0\} \cup \mathcal{C} \cup \{n+1\}$ and the arc set $\mathcal{A} = \{(i,j)|i,j \in \mathcal{C}, i \neq j\} \cup \{(0,j)|j \in \mathcal{C}\} \cup \{(i,n+1)|i \in \mathcal{C}\}$. A set of customers is denoted by $\mathcal{C} = \{1, \cdots, n\}$. Nodes 0 and $n+1$ represent the same central depot for modelling purposes. Set $\mathcal{K}$ denotes a fleet of homogeneous vehicles. For each vehicle $k \in \mathcal{K}$, it has a capacity Q. For each arc $(i,j) \in \mathcal{A}$, it is associated with a travel time t_{ij} and a travel distance d_{ij}. Each node $i \in \mathcal{N}$ has a delivery demand q_i and a pickup demand p_i, such that $q_i > 0$ and $p_i > 0$ for each $i \in \mathcal{C}$ and $q_0 = q_{n+1} = p_0 = p_{n+1} = 0$. A time window $[a_i, b_i]$ is also associated with each node $i \in \mathcal{N}$. a_i and b_i respectively denote the earliest and latest start-of-service times of node i. Time windows are considered as hard constraints in the problem. Thus, node $i \in \mathcal{N}$ cannot be visited after b_i. However, a vehicle is allowed to arrive at node i before a_i and wait to start service until a_i. In addition, each node $i \in \mathcal{N}$ has a service time s_i, such that $s_i > 0$ for each $i \in \mathcal{C}$ and $s_0 = s_{n+1} = 0$. The VRPSPDTW considers a hierarchical objective which minimizes the number of vehicles used (primary criterion) and the total travel distance (secondary criterion).

To model the VRPSPDTW, we present a mathematical formulation based on those proposed in Dethloff (2001) and Wang et al. (2015). Three types of decision variables are defined in the formulation. Binary variable $x_{ij}^k = 1$ if vehicle $k \in \mathcal{K}$ traverses arc $(i,j) \in \mathcal{A}$ and 0 otherwise. Continuous variable y_i^k specifies the arrival time or the start of service time of vehicle $k \in \mathcal{K}$ at node $i \in \mathcal{N}$. Continuous variable z_i^k specifies the load of vehicle $k \in \mathcal{K}$ after visiting node $i \in \mathcal{N}$. Based on these decision variables, the deterministic mathematical formulation for the VRPSPDTW is presented below.

$$\text{(VRPSPDTW)} \quad \textit{lex}\text{-}\min \left(\sum_{k \in \mathcal{K}} \sum_{j \in \mathcal{C}} x_{0j}^k, \sum_{k \in \mathcal{K}} \sum_{(i,j) \in \mathcal{A}} d_{ij} x_{ij}^k \right) \tag{4.1}$$

$$\text{s.t.} \sum_{j \in \mathcal{C}} x_{0j}^k \leq 1 \qquad \forall k \in \mathcal{K}, \tag{4.2}$$

$$\sum_{j \in \mathcal{C}} x_{0j}^k = \sum_{i \in \mathcal{C}} x_{i(n+1)}^k \qquad \forall k \in \mathcal{K}, \tag{4.3}$$

$$\sum_{i:(i,j) \in \mathcal{A}} x_{ij}^k - \sum_{i:(j,i) \in \mathcal{A}} x_{ji}^k = 0 \qquad \forall k \in \mathcal{K}, j \in \mathcal{C}, \tag{4.4}$$

$$\sum_{k \in \mathcal{K}} \sum_{i:(i,j) \in \mathcal{A}} x_{ij}^k = 1 \qquad \forall j \in \mathcal{C}, \tag{4.5}$$

$$z_0^k = \sum_{(i,j) \in \mathcal{A}} q_j x_{ij}^k \qquad \forall k \in \mathcal{K}, \tag{4.6}$$

$$z_i^k - q_j + p_j \leq z_j^k + M(1 - x_{ij}^k) \qquad \forall k \in \mathcal{K}, (i,j) \in \mathcal{A}, \tag{4.7}$$

$$z_i^k \leq Q \qquad \forall k \in \mathcal{K}, i \in \mathcal{N}, \tag{4.8}$$

$$y_i^k + t_{ij} + s_i \leq y_j^k + M(1 - x_{ij}^k) \qquad \forall k \in \mathcal{K}, (i,j) \in \mathcal{A}, \tag{4.9}$$

$$a_i \leq y_i^k \leq b_i \qquad \forall k \in \mathcal{K}, i \in \mathcal{N}, \tag{4.10}$$

$$x_{ij}^k \in \{0,1\} \qquad \forall k \in \mathcal{K}, (i,j) \in \mathcal{A}. \tag{4.11}$$

Objective function (4.1) lexicographically minimizes the number of vehicles used and the total travel distance. Constraints (4.2) and (4.3) guarantee that each used vehicle starts from and returns to the same central depot. Constraints (4.4) denote the flow conservation constraints. Constraints (4.5) ensure that each customer must be visited exactly once by only one vehicle. Constraints (4.6) calculate the initial load of each vehicle $k \in \mathcal{K}$ at the starting depot 0. Constraints (4.7) calculate the load of vehicle $k \in \mathcal{K}$ after visiting node $i \in \mathcal{N}$. Constraints (4.8) ensure that the load of vehicle $k \in \mathcal{K}$ never exceeds its capacity. Constraints (4.9) calculate the arrival time of vehicle k at node $i \in \mathcal{N}$. Constraints (4.10) denote the time window constraints. Note that the values of y_i^k and z_i^k are meaningless whenever vehicle $k \in \mathcal{K}$ does not visit node $i \in \mathcal{N}$. Similar to constraints (3.7), constraints (4.7) and (4.9) also contain notation M which is an arbitrary large constant. A reasonable value for M can be set to the larger one between $\max_{i \in \mathcal{C}}\{p_i - q_i + Q\}$ and $\max_{(i,j) \in \mathcal{A}}\{b_i - a_j + s_i + t_{ij}\}$. A larger value for M will not affect the optimal solution of the above deterministic formulation for the VRPSPDTW. However, the optimal solution of the formulation may be affected if M is set less than the introduced reasonable value. Finally, constraints (4.11) impose the domain of the decision variables x_{ij}^k.

4.2.2 The VRPSPDTW Under Uncertainty

As discussed in Section 4.1, many real-life VRPSPDTW applications are subject to uncertainty. Thus, we assume that two important types of parameters including pickup demands and travel times in the VRPSPDTW are uncertain and study a robust version of the VRPSPDTW under uncertainty based on the adjustable robust optimization framework (Ben-Tal et al., 2004). To capture these two types of uncertainty, we adopt the concept of the route-dependent uncertainty sets proposed in Section 3.2.2. Specifically, for the route r_k of each used vehicle $k \in \mathcal{K}'$, we define two types of uncertainty polytopes: the pickup demand uncertainty polytope $\mathcal{U}_p^k$ and the travel time uncertainty polytope $\mathcal{U}_t^k$. $\mathcal{K}' \subseteq \mathcal{K}$ denotes the set of used vehicles. Based on the uncertainty polytopes, we define the pickup demand uncertainty set $\mathcal{U}_p$ and the travel time uncertainty set $\mathcal{U}_t$ in the following equations:

$$\mathcal{U}_p = \bigcap_{k \in \mathcal{K}'} \mathcal{U}_p^k \tag{4.12}$$

with

$$\mathcal{U}_p^k = \left\{ \tilde{p} \in \mathbb{R}^{|\mathcal{C}|} | \tilde{p}_i = \overline{p}_i + \alpha_i^k \hat{p}_i, |\alpha_i^k| \leq 1, \forall i \in \mathcal{C}; \sum_{i \in \mathcal{C}^k} |\alpha_i^k| \leq \Gamma_p^k, \Gamma_p^k = \lceil \theta_p |\mathcal{C}^k| \rceil \right\} \tag{4.13}$$

$$\mathcal{U}_t = \bigcap_{k \in \mathcal{K}'} \mathcal{U}_t^k \tag{4.14}$$

with

$$\mathcal{U}_t^k = \left\{ \tilde{t} \in \mathbb{R}^{|\mathcal{A}|} | \tilde{t}_{ij} = \overline{t}_{ij} + \gamma_{ij}^k \hat{t}_{ij}, |\gamma_{ij}^k| \leq 1, \forall (i,j) \in \mathcal{A}; \sum_{(i,j) \in \mathcal{A}^k} |\gamma_{ij}^k| \leq \Gamma_t^k, \Gamma_t^k = \lceil \theta_t |\mathcal{A}^k| \rceil \right\}. \tag{4.15}$$

In equation (4.12), $\mathcal{U}_p$ denotes the pickup demand uncertainty set. It is the intersection of polytopes $\mathcal{U}_p^k$ for all $k \in \mathcal{K}'$. Uncertainty polytope $\mathcal{U}_p^k$ in equation (4.13) captures the possible pickup demand uncertainty experienced by a vehicle $k \in \mathcal{K}'$ and it is defined based on the route r_k of vehicle k. In polytope $\mathcal{U}_p^k$, $\tilde{p}_i$ denotes the uncertain pickup demand of customer i and vector $\tilde{p}$ subsumes $\tilde{p}_i$ for all $i \in \mathcal{C}$. $\tilde{p}_i$ is expressed as $\overline{p}_i + \alpha_i^k \hat{p}_i$, where $\overline{p}_i$ denotes the nominal value of $\tilde{p}_i$ and $\hat{p}_i$ denotes the maximum deviation of $\tilde{p}_i$ from $\overline{p}_i$. Similar to the uncertainty polytope $\mathcal{U}_q^k$ defined in equation (3.11), α_i^k is also used in polytope $\mathcal{U}_p^k$ and it denotes the auxiliary variable associated with the uncertain pickup demand $\tilde{p}_i$ of customer i. As α_i^k takes values from the interval $[-1,1]$, $\tilde{p}_i$ actually takes values in the interval $[\overline{p}_i - \hat{p}_i, \overline{p}_i + \hat{p}_i]$ for each $i \in \mathcal{C}$. However, according to the inequality $\sum_{i \in \mathcal{C}^k} |\alpha_i^k| \leq \Gamma_p^k$ in polytope $\mathcal{U}_p^k$, at most Γ_p^k auxiliary variables α_i^k $(i \in \mathcal{C}^k)$ can simultaneously take their maximum values of 1. Γ_p^k denotes the uncertainty budget and it actually imposes an upper bound on the number of customers who can simultaneously have their largest possible pickup demands $\overline{p}_i + \hat{p}_i$ on route r_k. Γ_p^k equals $\lceil \theta_p |\mathcal{C}^k| \rceil$ and $\lceil \theta_p |\mathcal{C}^k| \rceil$ denotes the least integer that is greater than or equal to $\theta_p |\mathcal{C}^k|$. $\mathcal{C}^k$ represents the set of customers served by vehicle k and $|\mathcal{C}^k|$ denotes the number of customers on route r_k. θ_p denotes the uncertainty budget coefficient and it can be set to a value between 0 and 1 by decision-makers. If $\theta_p = 0$, $\Gamma_p^k = 0$. Thus, $\alpha_i^k = 0$ and $\tilde{p}_i = \overline{p}_i$ for all $i \in \mathcal{C}$. No pickup demand uncertainty is considered in polytope $\mathcal{U}_p^k$. If $\theta_p = 1$, $\Gamma_p^k = |\mathcal{C}^k|$. Thus, α_i^k can take any value in the interval $[-1, 1]$ and uncertain pickup demand $\tilde{p}_i$ can take any value in the interval $[\overline{p}_i - \hat{p}_i, \overline{p}_i + \hat{p}_i]$ for each $i \in \mathcal{C}$. In real VRPSPDTW applications, θ_p reflects decision-makers' attitudes towards pickup demand uncertainty. If decision-makers are not concerned about the impact of pickup demand uncertainty on the feasibility of the designed routing strategies, θ_p can be set close to 0. On the contrary, θ_p can be set close to 1 if they are seriously concerned about the impact of pickup demand uncertainty.

The travel time uncertainty set $\mathcal{U}_t$ in equation (4.14) with polytopes $\mathcal{U}_t^k$ in equation (4.15) are the same as those defined in equations (3.14) and (3.15) in Section 3.2.2. We rewrite the uncertainty set $\mathcal{U}_t$ with polytopes $\mathcal{U}_t^k$ in this section for the purpose of improving the readability of the robust formulation presented below.

To model the VRPSPDTW under uncertainty, we adopt the adjustable robust optimization framework (Ben-Tal et al., 2009, 2004) and extend the deterministic formulation (4.1)-(4.11) to a two-stage robust formulation based on the uncertainty sets $\mathcal{U}_p$ and $\mathcal{U}_t$. The robust formulation can be expressed as follows:

$$\text{(R-VRPSPDTW)} \quad \textit{lex-}\min \left(\sum_{k \in \mathcal{K}} \sum_{j \in \mathcal{C}} x_{0j}^k, \sum_{k \in \mathcal{K}} \sum_{(i,j) \in \mathcal{A}} d_{ij} x_{ij}^k \right) \tag{4.16}$$

$$\text{s.t.} \quad (4.2), (4.3), (4.4), (4.5), (4.11),$$

$$z_0^k(\tilde{p}) = \sum_{(i,j) \in \mathcal{A}} q_j x_{ij}^k \quad \forall k \in \mathcal{K}, \tilde{p} \in \mathcal{U}_p, \tag{4.17}$$

$$z_i^k(\tilde{p}) - q_j + \tilde{p}_j \leq z_j^k(\tilde{p}) + M(1 - x_{ij}^k) \quad \forall k \in \mathcal{K}, (i,j) \in \mathcal{A}, \tilde{p} \in \mathcal{U}_p, \tag{4.18}$$

$$z_i^k(\tilde{p}) \leq Q \quad \forall k \in \mathcal{K}, i \in \mathcal{N}, \tilde{p} \in \mathcal{U}_p, \tag{4.19}$$

$$y_i^k(\tilde{t}) + \tilde{t}_{ij} + s_i \leq y_j^k(\tilde{t}) + M(1 - x_{ij}^k) \quad \forall k \in \mathcal{K}, (i,j) \in \mathcal{A}, \tilde{t} \in \mathcal{U}_t, \tag{4.20}$$

$$a_i \leq y_i^k(\tilde{t}) \leq b_i \quad \forall k \in \mathcal{K}, i \in \mathcal{N}, \tilde{t} \in \mathcal{U}_t. \tag{4.21}$$

In the above two-stage robust formulation, the first-stage decision variables are binary variables x_{ij}^k for all $k \in \mathcal{K}$ and $(i,j) \in \mathcal{A}$. They should be determined before the values of the uncertain parameters $(\tilde{p}, \tilde{t})$ are revealed. The second-stage (adjustable) decision variables are $y_i^k(\tilde{t})$ and $z_i^k(\tilde{p})$. $y_i^k(\tilde{t})$ extends the continuous variable y_i^k for every realization of the uncertain parameter $\tilde{t} \in \mathcal{U}_t$. $z_i^k(\tilde{p})$ extends z_i^k for every realization of the uncertain parameter $\tilde{p} \in \mathcal{U}_p$. The second-stage decision variables are determined after the values of the uncertain parameters $(\tilde{p}, \tilde{t})$ are revealed given the determined first-stage decisions x_{ij}^k. Thus, they are dependent on the uncertain parameters. Note that constraints (4.6)-(4.10) in the deterministic formulation are replaced by constraints (4.17)-(4.21) in the robust formulation. In addition, the reasonable value for M should be accordingly adjusted in constraints (4.18) and (4.20) due to the consideration of uncertainty. The above two-stage robust formulation is similar to that proposed for the VRPTW under three types of uncertainty in Section 3.2.2. Hence, some common features can be observed in both formulations. For instance, the uncertainty sets are dependent on the first-stage solution because the first-stage decisions determine the specific route for each vehicle. In addition, there are an infinite number of second-stage decision variables and constraints in the robust formulations. Thus, it is challenging to design exact algorithms to solve these formulations with large-sized instances.

4.3 An ALNS-Based Metaheuristic

The VRPSPDTW is known as an NP-hard problem. As reviewed in Section 2.3, a variety of effective metaheuristic solution methods have been developed for the VRPSPDTW, such as the GA (Wang and Chen, 2012), the parallel SA (Wang et al., 2013), the ALNS with PR (Hof and Schneider, 2019), and the VNS with TS (Shi et al., 2020). To solve the VRPSPDTW under pickup demand and travel time uncertainty with large-sized instances, we develop a metaheuristic solution approach based on the well-known ALNS framework introduced in Stenger et al. (2013).

4.3.1 Overview of the Metaheuristic

An overview of the developed ALNS-based metaheuristic is provided in this section. Algorithm 4.1 shows the main components and steps of the metaheuristic. Note that we refer to $\mathcal{R}$ as a solution to the VRPSPDTW under pickup demand and travel time uncertainty in the metaheuristic. Solution $\mathcal{R}$ consists of a set of vehicle routes and $|\mathcal{R}|$ denotes the number of vehicles (routes) used.

Algorithm 4.1 An ALNS-based metaheuristic for the VRPSPDTW under uncertainty.

```
1:  $\mathcal{R}^0 \leftarrow$ InitialSolutionALNS();
2:  $\mathcal{R} \leftarrow \mathcal{R}^0$;
3:  while $|\mathcal{R}| > V_{min}$ do
4:      $\mathcal{R}' \leftarrow$ RemoveOneRoute($\mathcal{R}$);
5:      $\mathcal{R}' \leftarrow$ ALNSFeasibilityRecovery($\mathcal{R}'$);
6:      if $\mathcal{R}'$ is feasible then
7:          $\mathcal{R} \leftarrow \mathcal{R}'$;
8:      else
9:          break;
10:     end if
11: end while
12: $\mathcal{R}^* \leftarrow$ ALNSDistanceMinimization($\mathcal{R}$);
```

As shown in Algorithm 4.1, the ALNS-based metaheuristic mainly contains three phases. In the first phase, we generate a feasible initial solution $\mathcal{R}^0$ for the VRPSPDTW under uncertainty using function InitialSolutionALNS() (line 1). The details of function InitialSolutionALNS() are described in Section 4.3.3. Next, we minimize the number of vehicles (routes) used in the initial solution using functions RemoveOneRoute() and ALNS-FeasibilityRecovery() in the second phase (lines 3-11). In function RemoveOneRoute(), we first remove the route with the minimum number of customers from the current solution $\mathcal{R}$. Then, we randomly reinsert the customers on the removed route into the remaining routes to generate a new solution $\mathcal{R}'$. Due to the random insertion process of the removed customers, solution $\mathcal{R}'$ is generally infeasible. Thus, we allow infeasible solutions in the ALNS-based metaheuristic. Both feasible and infeasible solutions are evaluated using an augment cost function $Cost()$ defined in equation (4.22). Function ALNSFeasibilityRecovery() tries to recover the feasibility of solution $\mathcal{R}'$ based on the ALNS framework. Its detailed procedure is shown in pseudocode in Algorithm 4.2. The route (vehicle) reduction process stops when the feasibility of solution $\mathcal{R}'$ cannot be recovered or the current feasible solution $\mathcal{R}$ has the minimum required number of vehicles $V_{min} = \max(\sum_{i\in\mathcal{C}} q_i/Q, \sum_{i\in\mathcal{C}} \overline{p}_i/Q)$. Finally, we minimize the total travel distance of the resulting solution $\mathcal{R}$ from the second phase using function ALNSDistanceMinimization() in the third phase and the metaheuristic returns the best found solution $\mathcal{R}^*$ (line 12). Function ALNSDistanceMinimization() is shown in pseudocode in Algorithm 4.3.

Next, we introduce function ALNSFeasibilityRecovery() in Algorithm 4.2. Function ALNSFeasibilityRecovery() utilizes the ALNS framework to recover the feasibility of the input solution $\mathcal{R}^0$. Initially, we define a set of destroy operators $\mathcal{O}_{des}$, a set of repair operators $\mathcal{O}_{rep}$, and a set of neighbourhood structures $\mathcal{N}_{search}$ (line 1). The input solution $\mathcal{R}^0$ is first improved by function LocalSearch() with the defined neighbourhood structures $\mathcal{N}_{search}$ (line 2). Function LocalSearch() is a local search procedure and its details are given in Section 4.3.4.4. Note that the local search procedure immediately stops once a feasible solution is found whenever function LocalSearch() is called in function ALNS-FeasibilityRecovery(). In the main loop of the ALNS framework (lines 5-31), function DestroyAndRepair() first adopts the defined destroy and repair operators to modify the current solution $\mathcal{R}$ and generates an intermediate solution $\mathcal{R}'$ (line 7). We introduce the destroy and repair operators in Sections 4.3.4.1 and 4.3.4.2, respectively. Then, the intermediate solution $\mathcal{R}'$ is improved by function LocalSearch() (line 8). Function Accept() uses a

Algorithm 4.2 ALNS for solution feasibility recovery.

```
Function ALNSFeasibilityRecovery($\mathcal{R}^0$)
1: Define two sets of operators $\mathcal{O}_{des}$ and $\mathcal{O}_{rep}$ and a set of neighbourhood structures $\mathcal{N}_{search}$;
2: $\mathcal{R}^0 \leftarrow$ LocalSearch($\mathcal{R}^0$,$\mathcal{N}_{search}$);
3: $\mathcal{R} \leftarrow \mathcal{R}^0$; $\mathcal{R}^* \leftarrow \mathcal{R}^0$;
4: $Iter \leftarrow 0$; $NoImp \leftarrow 0$;
5: while $\mathcal{R}^*$ is infeasible and $Iter < MaxIter1$ and $NoImp < MaxNoImp1$ do
6:     $Iter \leftarrow Iter + 1$;
7:     $\mathcal{R}' \leftarrow$ DestroyAndRepair($\mathcal{R}, \mathcal{O}_{des}, \mathcal{O}_{rep}$);
8:     $\mathcal{R}' \leftarrow$ LocalSearch($\mathcal{R}'$,$\mathcal{N}_{search}$);
9:     if $\mathcal{R}'$ is feasible then
10:        $\mathcal{R}^* \leftarrow \mathcal{R}'$;
11:        break;
12:    end if
13:    if Accept($\mathcal{R}', \mathcal{R}$) then
14:        $\mathcal{R} \leftarrow \mathcal{R}'$;
15:        if $Cost(\mathcal{R}') < Cost(\mathcal{R}^*)$ then
16:            $\mathcal{R}^* \leftarrow \mathcal{R}'$;
17:            $NoImp \leftarrow 0$;
18:        else
19:            $NoImp \leftarrow NoImp + 1$;
20:        end if
21:    else
22:        $NoImp \leftarrow NoImp + 1$;
23:    end if
24:    UpdateOperatorScore($\mathcal{R}'$);
25:    if $0 \equiv Iter \pmod{\eta^{aw}}$ then
26:        AdaptOperatorWeight();
27:    end if
28:    if $NoImp > 0$ and $0 \equiv NoImp \pmod{\eta^{sr}}$ then
29:        $\mathcal{R} \leftarrow$ SolutionReset($\mathcal{R}^*$);
30:    end if
31: end while
32: return $\mathcal{R}^*$;
```

SA-based acceptance mechanism to decide whether solution $\mathcal{R}'$ should replace the current solution $\mathcal{R}$ for the subsequent iteration. We introduce the SA-based acceptance mechanism in Section 4.3.4.5. Functions UpdateOperatorScore() and AdaptOperatorWeight() are employed in an adaptive mechanism which helps selecting effective destroy and repair operators in function DestroyAndRepair() (lines 24-27). The details of these two functions

Algorithm 4.3 ALNS for distance minimization.

```
Function ALNSDistanceMinimization($\mathcal{R}^0$)
1:  Define two sets of operators $\mathcal{O}_{des}$ and $\mathcal{O}_{rep}$ and a set of neighbourhood structures $\mathcal{N}_{search}$;
2:  $\mathcal{R}^0 \leftarrow$ LocalSearch($\mathcal{R}^0$,$\mathcal{N}_{search}$);
3:  $\mathcal{R} \leftarrow \mathcal{R}^0$; $\mathcal{R}^* \leftarrow \mathcal{R}^0$;
4:  if $\mathcal{R}^0$ is feasible then
5:      $\mathcal{R}^*_f \leftarrow \mathcal{R}^0$;
6:  end if
7:  $Iter \leftarrow 0$; $NoImp \leftarrow 0$;
8:  while $Iter < MaxIter2$ and $NoImp < MaxNoImp2$ do
9:      $Iter \leftarrow Iter + 1$;
10:     $\mathcal{R}' \leftarrow$ DestroyAndRepair($\mathcal{R}, \mathcal{O}_{des}, \mathcal{O}_{rep}$);
11:     $\mathcal{R}' \leftarrow$ LocalSearch($\mathcal{R}'$,$\mathcal{N}_{search}$);
12:     if Accept($\mathcal{R}', \mathcal{R}$) then
13:         $\mathcal{R} \leftarrow \mathcal{R}'$;
14:         if $Cost(\mathcal{R}') < Cost(\mathcal{R}^*)$ then
15:             $\mathcal{R}^* \leftarrow \mathcal{R}'$;
16:             $NoImp \leftarrow 0$;
17:         else
18:             $NoImp \leftarrow NoImp + 1$;
19:         end if
20:         if $\mathcal{R}'$ is feasible and $Cost(\mathcal{R}') < Cost(\mathcal{R}^*_f)$ then
21:             $\mathcal{R}^*_f \leftarrow \mathcal{R}'$;
22:         end if
23:     else
24:         $NoImp \leftarrow NoImp + 1$;
25:     end if
26:     UpdatePenaltyFactor($\mathcal{R}'$);
27:     UpdateOperatorScore($\mathcal{R}'$);
28:     if $0 \equiv Iter \pmod{\eta^{aw}}$ then
29:         AdaptOperatorWeight();
30:     end if
31:     if $NoImp > 0$ and $0 \equiv NoImp \pmod{\eta^{sr}}$ then
32:         $\mathcal{R} \leftarrow$ SolutionReset($\mathcal{R}^*$);
33:     end if
34: end while
35: return $\mathcal{R}^*_f$;
```

and the adaptive mechanism are given in Section 4.3.4.3. Since infeasible solutions are allowed in function ALNSFeasibilityRecovery(), we employ function SolutionReset() to avoid searching too deep in infeasible regions. In function SolutionReset(), the current solution $\mathcal{R}$ is reset to the best solution $\mathcal{R}^*$ after a certain number of iterations η^{sr} with no overall improvement (lines 28-30). The feasibility recovery procedure terminates when the best solution $\mathcal{R}^*$ becomes feasible or a maximum number of iterations $MaxIter1$ or no overall improvement iterations $MaxNoImp1$ is reached.

In Algorithm 4.3, function ALNSDistanceMinimization() also adopts the ALNS framework to minimize the total travel distance of the resulting solution from the route (vehicle) minimization phase. It can be observed that most components (functions) used in function ALNSDistanceMinimization() are the same as those used in function ALNSFeasibilityRecovery(). However, function ALNSDistanceMinimization() adds a new component UpdatePenaltyFactor() which is a dynamic penalty mechanism. The dynamic penalty mechanism is similar to those used in Hiermann et al. (2016) and Hof and Schneider (2019). A detailed description of the mechanism is provided in Section 4.3.2. Following the idea in Hiermann et al. (2016), we keep two types of best solutions in function ALNSDistanceMinimization(). Specifically, we keep the best feasible solution $\mathcal{R}^*_f$ and the best solution $\mathcal{R}^*$ with the current penalty setting. Note that the cost function value $Cost(\mathcal{R})$ of the current solution and the cost function value $Cost(\mathcal{R}^*)$ of the best solution are adapted every time the values of the penalty factors are updated in function UpdatePenaltyFactor(). However, this might lead to a situation that the cost function value $Cost(\mathcal{R}^*)$ of the best solution is worse than the cost function value $Cost(\mathcal{R}^*_f)$ of the best feasible solution. Thus, the best solution $\mathcal{R}^*$ is reset to the best feasible solution $\mathcal{R}^*_f$ when this situation happens.

4.3.2 Solution Evaluation

As discussed in Section 4.3.1, infeasible solutions are allowed in the ALNS-based metaheuristic. To evaluate a given feasible or infeasible solution $\mathcal{R}$ in the metaheuristic, we use the same augment cost function $Cost(\mathcal{R})$ defined in equation (3.20) in Section 3.3.2. To improve the readability of this section, we rewrite the augment cost function $Cost(\mathcal{R})$ in equation (4.22):

$$Cost(\mathcal{R}) = Distance(\mathcal{R}) + \rho^{cap} V^{cap}(\mathcal{R}) + \rho^{tw} V^{tw}(\mathcal{R}) \tag{4.22}$$

where $Distance(\mathcal{R})$ denotes the total travel distance of solution $\mathcal{R}$. ρ^{cap} and ρ^{tw} respectively denote the penalty factors for vehicle capacity violations $V^{cap}(\mathcal{R})$ and time window violations $V^{tw}(\mathcal{R})$. It can be observed that the penalty costs for violations of the vehicle capacity or time window constraints are calculated by multiplying the amount of violations with the corresponding penalty factor.

To increase the effectiveness of the ALNS-based metaheuristic, a dynamic penalty mechanism which is similar to those used in Hiermann et al. (2016) and Hof and Schneider (2019) is adopted in the component UpdatePenaltyFactor() of function ALNSDistanceMinimization(). In the mechanism, we initially set both penalty factors ρ^{cap} and ρ^{tw} to ρ^0. Then, they are dynamically updated within the interval $[\rho^{min}, \rho^{max}]$ based on the following rules in the main loop of function ALNSDistanceMinimization(). If the candidate solution $\mathcal{R}'$ from function LocalSearch() stays feasible after η^{des} consecutive ALNS iterations, both penalty factors ρ^{cap} and ρ^{tw} are divided by ρ^{update}. Conversely, both penalty factors are multiplied by ρ^{update} if solution $\mathcal{R}'$ stays infeasible after η^{ins} consecutive ALNS iterations. Note that function ALNSFeasibilityRecovery() in Algorithm 4.2 does not adopt function UpdatePenaltyFactor(). The reason is that the main purpose of function ALNSFeasibilityRecovery() is to recover the feasibility of an infeasible solution. Thus, we set both penalty factors ρ^{cap} and ρ^{tw} to the maximum value ρ^{max} when evaluating a given solution in function ALNSFeasibilityRecovery().

In the VRPSPDTW under uncertainty introduced in Section 4.2.2, pickup demands $\tilde{p}_i\,(\forall i \in \mathcal{C})$ and travel times $\tilde{t}_{ij}\,(\forall (i,j) \in \mathcal{A})$ are uncertain parameters. To capture these two types of uncertainty, two route-dependent uncertainty sets are defined. They are set $\mathcal{U}_p$ (4.12) defined with polytopes $\mathcal{U}_p^k$ (4.13) and set $\mathcal{U}_t$ (4.14) defined with polytopes $\mathcal{U}_t^k$ (4.15). When evaluating a given solution $\mathcal{R}$, it is not trivial to determine the time window violations $V^{tw}(\mathcal{R})$ and the vehicle capacity violations $V^{cap}(\mathcal{R})$ based on the defined uncertainty sets and polytopes. Thus, we next introduce the ideas and equations which can help to calculate the constraint violations with a given solution.

Suppose there is a given solution $\mathcal{R}$ which consists of h vehicle routes and $\mathcal{R} = \{r_1, \cdots, r_h\}$. Let $r_k = \{v_1^k, v_2^k, \cdots, v_m^k, v_{m+1}^k\} \in \mathcal{R}$ be the route of vehicle k. v_1^k and v_{m+1}^k are nodes 0 and $n+1$, respectively. They represent the same central depot in the VRPSPDTW introduced in Section 4.2.1. Since the travel time uncertainty polytope $\mathcal{U}_t^k$ in equation (4.15) is defined based on the route r_k of vehicle k, we can first calculate the latest arrival time of vehicle k at every node on route r_k considering all possible realizations of uncertain travel times $\tilde{t}_{ij}$ $(\forall (i,j) \in \mathcal{A}^k)$ in polytope $\mathcal{U}_t^k$. $\mathcal{A}^k$ denotes the set of arcs (route segments) on route r_k. Then, the time window violations with route r_k can be determined by comparing the vehicle's latest arrival time at each node on the route with the latest start-of-service time of the corresponding node.

$$Y^k(v_i^k, \Gamma_t^k) = \begin{cases} 0, & \text{if } i = 1; \\ \max\left\{a_{v_i^k}, Y^k(v_{i-1}^k, \Gamma_t^k) + s_{v_{i-1}^k} + \bar{t}_{v_{i-1}^k, v_i^k}\right\}, & \\ & \text{if } 2 \le i \le m+1, \Gamma_t^k = 0; \\ \max\left\{a_{v_i^k}, Y^k(v_{i-1}^k, \Gamma_t^k) + s_{v_{i-1}^k} + \bar{t}_{v_{i-1}^k, v_i^k},\right. & \\ \quad \left. Y^k(v_{i-1}^k, \Gamma_t^k - 1) + s_{v_{i-1}^k} + \bar{t}_{v_{i-1}^k, v_i^k} + \hat{t}_{v_{i-1}^k, v_i^k}\right\}, & \\ & \text{if } 2 \le i \le m+1, 1 \le \Gamma_t^k \le i-1; \\ Y^k(v_i^k, \Gamma_t^k - 1), & \text{if } 2 \le i \le \Gamma_t^k. \end{cases} \tag{4.23}$$

As mentioned in Section 4.2.2, the travel time uncertainty polytope $\mathcal{U}_t^k$ in equation (4.15) is the same as that defined in equation (3.15) in Section 3.2.2. Thus, the uncertain travel time $\tilde{t}_{ij}$ on each arc $(i,j) \in \mathcal{A}^k$ takes values in the interval $[\bar{t}_{ij} - \hat{t}_{ij}, \bar{t}_{ij} + \hat{t}_{ij}]$ based on the discussion in Section 3.2.2. However, due to the restriction of the uncertainty budget Γ_t^k in polytope $\mathcal{U}_t^k$, there are at most Γ_t^k arcs (route segments) which can simultaneously have the longest travel times $\bar{t}_{ij} + \hat{t}_{ij}$ on route r_k. Thus, we use a recursive function in equation (4.23) to calculate the latest arrival time $Y^k(v_i^k, \Gamma_t^k)$ of vehicle k at the ith node v_i^k on route $r_k = \{v_1^k, v_2^k, \cdots, v_m^k, v_{m+1}^k\}$. Note that function $Y^k(v_i^k, \Gamma_t^k)$ in equation (4.23) was originally proposed in Agra et al. (2013) and it is a simplified version of function $Y^k(v_i^k, \Gamma_s^k, \Gamma_t^k)$ in equation (3.21) considering only travel time uncertainty. By comparing $Y^k(v_i^k, \Gamma_t^k)$ with the latest start-of-service time of each node on route r_k, we can calculate the time window violations with route r_k and further determine the total time window violations with the given solution $\mathcal{R}$.

Similar to function $Y^k(v_i^k, \Gamma_s^k, \Gamma_t^k)$ in equation (3.21), using function $Y^k(v_i^k, \Gamma_t^k)$ to determine the time window violations with route r_k also can be computational expensive. In addition, both functions ALNSFeasibilityRecovery() and ALNSDistanceMinimization() depend heavily on function LocalSearch() which requires a large number of solution and route evaluations. To reduce the computational time of evaluating a solution $\mathcal{R}$ in function LocalSearch(), we overcalculate the time window violations with each route $r_k \in \mathcal{R}$ if it is time window infeasible even without considering travel time uncertainty. Specifically, if route $r_k \in \mathcal{R}$ is infeasible due to the existence of time window violations considering only the nominal values $\bar{t}_{ij}$ of uncertain travel times $\tilde{t}_{ij}$ for all $(i,j) \in \mathcal{A}^k$, we ignore the restriction of the uncertainty budget Γ_t^k in polytope $\mathcal{U}_t^k$ and approximately determine the latest arrival time of vehicle k at every node on route r_k and the time window violations with route r_k under the assumption $\tilde{t}_{ij} = \bar{t}_{ij} + \hat{t}_{ij}$ for all $(i,j) \in \mathcal{A}^k$.

To determine the vehicle capacity violations with route r_k, we can calculate the largest load of vehicle k at each node on the route considering all possible realizations of uncertain pickup demands $\tilde{p}_i$ $(\forall i \in \mathcal{C}^k)$ in uncertainty polytope $\mathcal{U}_p^k$. As discussed in Section 4.2.2, the pickup demand uncertainty polytope $\mathcal{U}_p^k$ in equation (4.13) is defined based on route r_k and the uncertain pickup demand $\tilde{p}_i$ of customer i can take values in the interval $[\bar{p}_i - \hat{p}_i, \bar{p}_i + \hat{p}_i]$ for each $i \in \mathcal{C}^k$. $\mathcal{C}^k$ denotes the set of customers on route r_k. However, owing to the restriction of the uncertainty budget Γ_p^k in polytope $\mathcal{U}_p^k$, at most Γ_p^k customers can simultaneously have their largest pickup demands $\bar{p}_i + \hat{p}_i$ on route r_k. Thus, we use equation (4.24) to calculate the largest vehicle load $Z^k(v_i^k, \Gamma_p^k)$ of vehicle k at the ith node v_i^k on route $r_k = \{v_1^k, v_2^k, \cdots, v_m^k, v_{m+1}^k\}$. Note that equation (4.24) is similar to equation (4.23) and it is a recursive function of the nodes v_i^k on the route and the uncertainty budget Γ_p^k in polytope $\mathcal{U}_p^k$. By comparing the largest vehicle load $Z^k(v_i^k, \Gamma_p^k)$ at each node on route r_k with the vehicle capacity Q, we can calculate the vehicle capacity violations with route r_k and determine the total vehicle capacity violations with the given solution $\mathcal{R}$. To further reduce the computational time of evaluating a solution $\mathcal{R}$ in function LocalSearch(), we also overcalculate the vehicle capacity violations with each route $r_k \in \mathcal{R}$ if it is capacity infeasible even without considering pickup demand uncertainty. Specifically, if route r_k is infeasible due to the existence of vehicle capacity violations considering only the nominal values $\bar{p}_i$ of uncertain pickup demands $\tilde{p}_i$ for all $i \in \mathcal{C}^k$, we ignore the restriction of uncertainty budget Γ_p^k in polytope $\mathcal{U}_p^k$ and approximately determine the largest vehicle load at the every node on route r_k and the vehicle capacity violations with route r_k under the assumption $\tilde{p}_i = \bar{p}_i + \hat{p}_i$ for all $i \in \mathcal{C}^k$. Note that the vehicle capacity and time window violations with a solution may be overcalcuated only when it is evaluated in function LocalSearch(). Moreover, if the constraint violations with the output solution from function LocalSearch() are overcalcuated, we need to recalcuate its exact cost function value based on equations (4.23) and (4.24) whenever function LocalSearch() is called in functions ALNSFeasibilityRecovery() and ALNSDistanceMinimization().

$$Z^k(v_i^k, \Gamma_p^k) = \begin{cases} \sum_{j=1}^{m+1} q_{v_j^k}, & \text{if } i = 1; \\ Z^k(v_{i-1}^k, \Gamma_p^k) - q_{v_i^k} + \bar{p}_{v_i^k}, & \text{if } 2 \le i \le m+1, \Gamma_p^k = 0; \\ \max\left\{ Z^k(v_{i-1}^k, \Gamma_p^k) - q_{v_i^k} + \bar{p}_{v_i^k},\ Z^k(v_{i-1}^k, \Gamma_p^k - 1) - q_{v_i^k} + \bar{p}_{v_i^k} + \hat{p}_{v_i^k} \right\}, & \\ & \text{if } 2 \le i \le m+1, 1 \le \Gamma_p^k \le i-1; \\ Z^k(v_i^k, \Gamma_p^k - 1), & \text{if } 2 \le i \le \Gamma_p^k. \end{cases} \tag{4.24}$$

4.3.3 Initial Solution

As shown in Algorithm 4.1, function InitialSolutionALNS() generates a feasible initial solution in the ALNS-based metaheuristic. To construct a set of feasible vehicle routes for the VRPSPDTW under pickup demand and travel time uncertainty, function InitialSolutionALNS() adopts a sequential insertion heuristic. The main steps of the heuristic are the same as those of the sequential insertion heuristic proposed for the VRPTW under uncertainty in Section 3.3.3. However, since the VRPSPDTW under uncertainty considers simultaneous pickup and delivery constraints and uncertainty in pickup demands and travel times, we use equations (4.23) and (4.24) to help checking whether an unrouted customer can be feasibly inserted into the current route in the sequential insertion heuristic designed for the VRPSPDTW under uncertainty.

4.3.4 Key Components of the ALNS Framework

The ALNS framework is employed in both functions ALNSFeasibilityRecovery() and ALNSDistanceMinimization() as described in Section 4.3.1. In these two functions, functions DestroyAndRepair(), LocalSearch(), and Accept() are the three critical components. In each iteration of the ALNS main loop, function DestroyAndRepair() selects one destroy operator and one repair operator to modify the current solution $\mathcal{R}$ and generates an intermediate solution $\mathcal{R}'$ as shown in Algorithms 4.2 and 4.3. Next, we first introduce the destroy and repair operators used in function DestroyAndRepair().

4.3.4.1 Destroy Operators

A set of eight destroy operators $\mathcal{O}_{des}$ are employed in function DestroyAndRepair(). Some of them have been widely used in the literature, while others are designed based on the special features of the VRPSPDTW under uncertainty. A destroy operator removes at least n_r customers from the current solution. Parameter n_r is a random integer which is drawn from the interval $[\zeta_{min} \cdot |\mathcal{C}|, \zeta_{max} \cdot |\mathcal{C}|]$. Note that we allow empty routes in a partially destroyed solution if the customers on some routes are completely removed from the current solution. Thus, the number of vehicles (routes) in the partially destroyed solution keeps unchanged in the solution destruction process. The eight destroy operators are described as follows.

Random removal. This operator randomly removes n_r customers from the current solution with equal probability.

Related removal. This operator is inspired from the "Shaw Removal" operator introduced in Ropke and Pisinger (2006). Its idea is to remove customers that are considered to be similar and may be interchangeable. We adopt the customer relatedness measure proposed in Hof and Schneider (2019). Specifically, the relatedness measure $R(i, j)$ of two customers i and j is defined based on the absolute difference between their delivery demands $|q_i - q_j|$ and nominal pickup demands $|\overline{p}_i - \overline{p}_j|$, the absolute difference between their earliest allowable start-of-service times $|a_i - a_j|$, and the travel distance d_{ij} separating them. $R(i, j)$ can be calculated using the following equation (4.25):

$$\begin{aligned} R(i,j) = {} & \psi^q \frac{|q_i - q_j|}{\max\limits_{i \in \mathcal{C}}(q_i) - \min\limits_{i \in \mathcal{C}}(q_i)} + \psi^p \frac{|\overline{p}_i - \overline{p}_j|}{\max\limits_{i \in \mathcal{C}}(\overline{p}_i) - \min\limits_{i \in \mathcal{C}}(\overline{p}_i)} \\ & + \psi^a \frac{|a_i - a_j|}{\max\limits_{i \in \mathcal{C}}(a_i) - \min\limits_{i \in \mathcal{C}}(a_i)} + \psi^d \frac{d_{ij}}{\max\limits_{(i,j) \in \mathcal{A}}(d_{ij})} \end{aligned} \tag{4.25}$$

where ψ^q, ψ^p, ψ^a, and ψ^d denote the weights of the normalized partial relatedness measures. In the removal process based on this operator, we initially select a customer at random and remove it from the current solution. Then, we use the following three steps to remove the next $n_r - 1$ customers iteratively. In the first step, a customer is randomly selected from the removed customers. In the second step, we sort the remaining unremoved customers in ascending order based on their relatedness values with the customer selected in the first step and store them in an ordered set $\mathcal{S}$. In the third step, the customer at position $\lfloor |\mathcal{S}| \xi^{\nu_{re}} \rfloor$

in set $\mathcal{S}$ is selected and removed. Parameter ξ is randomly drawn in the interval $[0, 1)$ and parameter ν_{re} introduces some randomness in the selection process.

Neighbourhood removal. This operator was originally proposed in Pelletier et al. (2019). It aims to remove a set of customers who are located close to each other. This operator first randomly selects a customer and removes it from the current solution. Then, the remaining customers are sorted in ascending order based on their distances to the removed customer and they are stored in an order set $\mathcal{S}$. Finally, the first $n_r - 1$ customers in set $\mathcal{S}$ are removed from the current solution.

Worst cost removal. This operator was originally introduced in Ropke and Pisinger (2006). It aims to remove customers with high removal savings from the current solution. The savings of removing a customer is defined as the difference between the cost function value when the solution includes the customer and the cost function value when the solution excludes the customer. In the removal process based on this operator, we first sort all unremoved customers in descending order based on their removal savings and store them in an ordered set $\mathcal{S}$. Then, the customer at position $\lfloor |\mathcal{S}| \xi^{\nu_{wc}} \rfloor$ in set $\mathcal{S}$ is selected and removed. Parameter ξ is a random number in $[0, 1)$ and parameter ν_{wc} is used to randomize the removal process. We repeat the above removal process until n_r customers are removed from the current solution.

Worst distance removal. This operator is similar to the worst cost removal. The main difference lies in the definition of the removal savings. In the worst distance removal, the savings of removing a customer from the current solution is defined as the difference between the total travel distance when the solution includes the customer and the total travel distance when the solution excludes the customer. Parameters ξ and ν_{wd} are introduced to randomize the removal process as with the worst cost removal.

Distance-based route removal. This operator selects the longest route of the current solution and removes all customers on it. If the number of the already removed customers is less than n_r, the longest route among the remaining routes is again selected and all customers on it are removed. This removal process is repeated until at least n_r customers are removed from the current solution.

Lateness-based route removal. This operator is similar to the distance-based route removal. It selects the route with the largest lateness value and remove all its customers until at least n_r customers are removed from the current solution. The lateness value of a node on a route is defined as the difference between the latest vehicle arrival time at the node considering travel time uncertainty and the latest start-of-service time of the node if the latest vehicle arrival time is larger than the latest start-of-service time and zero otherwise. The lateness value of a route can thus be calculated as the sum of the lateness values of all nodes on the route. Note that if the lateness values of the remaining routes are all zero and the number of already removed customers is less than n_r, we randomly select one of the remaining routes and remove all its customers until at least n_r customers are removed.

Load-based route removal. This operator is similar to the lateness-based route removal. It selects the route with the largest average vehicle load and removes all its customers until at least n_r customers are removed from the current solution. The average vehicle load on a route is calculated based on the largest vehicle load at every node on the route considering pickup demand uncertainty.

4.3.4.2 Repair Operators

After the current solution is destroyed by one of the above introduced destroy operators, one of following repair operators is employed to reinsert the removed customers into the partially destroyed solution in function DestroyAndRepair() in each main ALNS iteration.

Greedy insertion. This operator was originally proposed in Ropke and Pisinger (2006). It aims to reinsert the removed customers into the partially destroyed solution based on their best insertion positions. The insertion process based on this operator is an iterative procedure which contains two steps in each iteration. In the first step, we determine the best insertion position in the current partial solution for each of the remaining removed customers. Note that a removed customer can be inserted at any position in a route since constraint violations are penalized. In the second step, the customer whose best insertion position brings the smallest increase in the cost function value of the current partial solution is selected and reinserted at its best insertion position. The above two steps are repeated until all removed customers are reinserted.

Randomized greedy insertion. This operator is a randomized version of the above greedy insertion. The insertion process based on this operator is also an iterative procedure. In each iteration, we first identify the best insertion position in the current partial solution for each remaining removed customer. Then, we sort all remaining removed customers in ascending order based on the increase in the cost function value of the current partial solution associated with the best insertion position of each removed customer and store these customers in an ordered set $\mathcal{S}$. Finally, the customer at position $\lfloor |\mathcal{S}|\xi^{\nu_{rg}} \rfloor$ in set $\mathcal{S}$ is selected and reinserted at its best insertion position in the current partial solution. Parameter ξ is randomly drawn in the interval $[0, 1)$ and parameter ν_{rg} is utilized to randomize the insertion process. The above insertion process is repeated until all removed customers are reinserted.

Regret insertion. This operator was also introduced in Ropke and Pisinger (2006). It aims to improve the basic greedy insertion operator by considering a regret value when selecting a removed customer to reinsert. The regret insertion operator considers the cheapest insertion position of a removed customer not only in its best route, but also in its second-best route, third-best route and so on. Specifically, the regret-k value of a removed customer is calculated as the difference in the cost function value of the partial solution resulting from inserting the customer at the cheapest insertion positions of its best route and its k-best route. The insertion process based on the regret insertion operator is also an iterative procedure. At each iteration, we first determine the regret value associated with each removed customer. It can be calculated as the total difference in the cost function value of the current partial solution between inserting a removed customer into its best route and its $k-1$ best routes. Then, we select the customer with the largest regret value and reinsert it at the cheapest insertion position in its best route. Note that we employ the Regret-2, Regret-3, and Regret-4 insertion operators in function DestroyAndRepair().

4.3.4.3 Adaptive Mechanism

In each iteration of the ALNS main loops in functions ALNSFeasibilityRecovery() and ALNSDistanceMinimization(), a destroy operator is first selected to destroy the current solution and a repair operator is then chosen to repair the partially destroyed solution in function DestroyAndRepair(). Moreover, the resulting solution from function DestroyAndRepair() is further improved by function LocalSearch() to generate a new candidate solution. The destroy and repair operators are selected independently based on a roulette wheel selection method used in Ropke and Pisinger (2006). An adaptive mechanism proposed in Ropke and Pisinger (2006) is also adopted to bias the selection of these operators. In the mechanism, each destroy or repair operator i is associated with a weight ω_i and a score π_i. In the beginning, all operators are weighted equally and their scores are set to zero. Then, the scores of all operators are dynamically updated based on a scoring system in function UpdateOperatorScore(). The details of the scoring system are described as follows. If the new candidate solution is an overall best solution, the current scores of the selected destroy and repair operators in the current ALNS iteration are increased by σ_1. If the new solution has not been accepted before and is better than the current solution, the current scores of the selected operators are increased by σ_2. If the new solution is worse than the current solution and has never been accepted before but can be accepted based on the acceptance criterion described in Section 4.3.4.5, the current scores of the selected operators are increased by σ_3. As shown in Algorithms 4.2 and 4.3, the weights of all operators are updated based on their performances (scores) after every η^{aw} iterations in function AdaptOperatorWeight(). Let ϕ_i denote the number of times of operator i has been used since the last weight update. We calculate the new weight of operator i as $\omega_i = \omega_i(1-\varpi) + \varpi\pi_i/\phi_i$. ϖ is a given parameter which takes values in the interval $[0,1]$. Parameter ϖ denotes the reaction factor which controls the inertia of the weight adjustment. The values of π_i and ϕ_i are reset to zero after each weight update.

4.3.4.4 Local Search

As shown in functions ALNSFeasibilityRecovery() and ALNSDistanceMinimization(), function LocalSearch() is used to improve the resulting solution from function DestroyAndRepair() in every ALNS iteration. Function LocalSearch() is a local search procedure, which adopts six neighbourhood structures. The adopted neighbourhood structures are designed based on the operators introduced in Section 3.3.4.4, which include the Intra-Route-Reinsert, the Intra-Route-Swap, the Intra-Route-2opt, the Inter-Route-Reinsert, the Inter-Route-Swap, and the Inter-Route-2opt. In the local search procedure, the neighbourhood structures are selected in random order. In addition, the first improvement strategy is considered in the search process to improve computational efficiency. Moreover, a neighbourhood structure is reused until the current solution cannot be further improved. When none of the six neighbourhood structures can further improve the current solution, the local search procedure terminates.

As function LocalSearch() is used in every ALNS iteration and the local search procedure employs a number of neighbourhoods, it can be computationally expensive when the sizes of the instances become large. To speed up the local search procedure, we use a neighbourhood pruning approach that is similar to the approach introduced in Section 3.3.4.4. In the neighbourhood pruning approach, for each customer $i \in \mathcal{C}$, we generate a set which consists of the Ω closest customers $j \in \mathcal{C}$ with respect to the relatedness measure $R(i,j)$ defined in equation (4.25). This set can be used to reduce the number of neighbours considered in each neighbourhood because neighbourhood moves with customer i and other customers who are not in this set are prohibited. We set $\Omega = 40$ when performing local search in all six neighbourhoods. Moreover, since most of the route changes are associated with two nodes (customers) based on the neighbourhood structures, a change tracking method proposed in Benjamin and Beasley (2013) is adopted and slightly extended to further reduce the computational complexity of the local search procedure. The details of the change tracking method can be found in Benjamin and Beasley (2013).

4.3.4.5 Acceptance Criterion

To increase the diversification of the ALNS framework used in functions ALNSFeasibilityRecovery() and ALNSDistanceMinimization(), a SA-based acceptance mechanism is adopted in function Accept(). The details of the mechanism are described as follows. If the new candidate solution $\mathcal{R}'$ generated from function LocalSearch() is better than the current solution $\mathcal{R}$ in terms of the cost function value, it is always accepted. Otherwise, solution $\mathcal{R}'$ is accepted with a probability $exp^{[-(Cost(\mathcal{R}')-Cost(\mathcal{R}))/T]}$, where $T > 0$ denotes the temperature parameter. T is initialized with a value of T_0 and it is multiplied by a cooling rate $\tau \in (0,1)$ at each ALNS iteration. T_0 is set to a value such that a solution $\mathcal{R}$ with the cost function value $Cost(\mathcal{R}) = \chi \cdot Cost(\mathcal{R}^0)$ is accepted with probability 0.5. $\mathcal{R}^0$ denotes the improved input solution in functions ALNSFeasibilityRecovery() and ALNSDistanceMinimization(). χ is a given parameter which helps to determine the initial temperature T_0. The cooling factor τ is set to a value such that the temperature T is below 0.0001 for the last 20% of the maximum allowed iterations in the ALNS main loops in functions ALNSFeasibilityRecovery() and ALNSDistanceMinimization().

4.4 Computational Experiments

Extensive computational experiments are conducted to show the effectiveness of the proposed methods in this section. The test instances for the VRPSPDTW under uncertainty and the parameter settings for the ALNS-based metaheuristic are described in Section 4.4.1.

Computational results and analyses follow in Section 4.4.2, alongside useful managerial insights derived for real-life VRPSPDTW applications with uncertain data.

4.4.1 Experiment Description and Parameter Setting

The computational experiments employ the benchmark instances introduced in Wang and Chen (2012) for the standard VRPSPDTW without considering uncertainty. These benchmark instances were adapted from the well-known Solomon's instances (Solomon, 1987). Similar to the Solomon's instances, the benchmark instances for the VRPSPDTW also can be divided into six classes, referred to as Cdp1, Cdp2, Rdp1, Rdp2, RCdp1, and RCdp2. Specifically, the Cdp1 and Cdp2 instances contain customers in clusters. The Rdp1 and Rdp2 instances contain randomly-distributed customers. The RCdp1 and RCdp2 instances comprise a mixture of randomly-distributed and clustered customers. Moreover, the Cdp1, Rdp1, and RCdp1 instances have narrow time windows and small-capacity vehicles whereas the Cdp2, Rdp2, and RCdp2 instances have wide time windows and large-capacity vehicles. We focus on medium-sized instances with 50 customers and large-sized instances with 100 customers in the experiments. Each medium-sized instance contains the first 50 customers in the corresponding large-sized instance.

In the VRPSPDTW under uncertainty introduced in Section 4.2.2, pickup demands $\tilde{p}_i\,(\forall i \in \mathcal{C})$ and travel times $\tilde{t}_{ij}\,(\forall (i,j) \in \mathcal{A})$ are uncertain parameters. Moreover, these two types of uncertain parameters take values in the route-dependent uncertainty set $\mathcal{U}_p$ (4.12) defined with polytopes $\mathcal{U}_p^k$ (4.13) and set $\mathcal{U}_t$ (4.14) defined with polytopes $\mathcal{U}_t^k$ (4.15). As shown in uncertainty polytopes $\mathcal{U}_p^k$ and $\mathcal{U}_t^k$, the nominal values $(\overline{p}_i, \overline{t}_{ij})$ of the uncertain parameters $(\tilde{p}_i, \tilde{t}_{ij})$ are critical and they need to be set by decision-makers (route planners). Thus, we set the nominal values $(\overline{p}_i, \overline{t}_{ij})$ of the uncertain pickup demands $\tilde{p}_i$ and uncertain travel times $\tilde{t}_{ij}$ in each medium- or large-sized instance to the original values of the deterministic parameters in the corresponding benchmark instance in Wang and Chen (2012). In addition to the nominal values $(\overline{p}_i, \overline{t}_{ij})$, there are two important types of parameters in the uncertainty polytopes $\mathcal{U}_p^k$ and $\mathcal{U}_t^k$, which are the maximum deviations $(\hat{p}_i, \hat{t}_{ij})$ of the uncertain parameters $(\tilde{p}_i, \tilde{t}_{ij})$ and the uncertainty budget coefficients (θ_p, θ_t). The maximum deviations $(\hat{p}_i, \hat{t}_{ij})$ restrict the variation ranges of the uncertain parameters $(\tilde{p}_i, \tilde{t}_{ij})$ and the uncertainty budget coefficients (θ_p, θ_t) help to determine the uncertainty budgets (Γ_p^k, Γ_t^k) in the uncertainty polytopes. To investigate the impacts of these two types of parameters on deriving robust solutions, we first test the medium-sized instances with different parameter settings in the uncertainty polytopes. Two cases are considered for the maximum deviations $(\hat{p}_i, \hat{t}_{ij})$ of the uncertain parameters $(\tilde{p}_i, \tilde{t}_{ij})$. Specifically, we consider a low uncertainty case, in which $\hat{p}_i = 0.2\overline{p}_i$ for all $i \in \mathcal{C}$ and $\hat{t}_{ij} = 0.2\overline{t}_{ij}$ for all $(i,j) \in \mathcal{A}$. We also consider a high uncertainty case, in which $\hat{p}_i = 0.4\overline{p}_i$ for all $i \in \mathcal{C}$ and $\hat{t}_{ij} = 0.4\overline{t}_{ij}$ for all $(i,j) \in \mathcal{A}$. For the uncertainty budget coefficients (θ_p, θ_t), we consider a total of seven combinations of coefficient values including (0,0), (0.1,0), (0.2,0), (0,0.1), (0,0.2), (0.1,0.1), and (0.2,0.2). These seven combinations can be further classified into four cases: 1) no uncertainty, in which case $(\theta_p, \theta_t) = (0,0)$; 2) uncertainty in pickup demands, in which case $(\theta_p, \theta_t) = (0.1,0)$ or $(0.2,0)$; 3) uncertainty in travel times, in which case $(\theta_p, \theta_t) = (0,0.1)$ or $(0,0.2)$; 4) uncertainty in both two, in which case $(\theta_p, \theta_t) = (0.1,0.1)$ or $(0.2,0.2)$. Based on the discussions in Section 4.2.2, no uncertainty is considered in all uncertainty polytopes when the uncertainty budget coefficients $(\theta_p, \theta_t) = (0,0)$. Thus, the solution derived with $(\theta_p, \theta_t) = (0,0)$ is referred to as the deterministic solution for each medium-sized instance.

To further assess the efficiency and effectiveness of the ALNS-based metaheuristic, we also derive the robust solutions for the large-sized instances with 100 customers using the following parameter settings in the uncertainty polytopes. Specifically, we set the uncertainty budget coefficients $(\theta_p, \theta_t) = (0.2,0.2)$ for the Cdp1, Rdp1, and RCdp1 instances and $(\theta_p, \theta_t) = (0.1,0.1)$ for the Cdp2, Rdp2, and RCdp2 instances. In addition, we only consider the high uncertainty case, in which $\hat{p}_i = 0.4\overline{p}_i$ for all $i \in \mathcal{C}$ and $\hat{t}_{ij} = 0.4\overline{t}_{ij}$ for all $(i,j) \in \mathcal{A}$. Moreover, we generate the corresponding deterministic solutions with $(\theta_p, \theta_t) = (0,0)$ for the large-sized instances without considering uncertainty. Thus, solutions are derived with different parameter settings in the uncertainty polytopes for both medium- and large-sized instances in the experiments.

All experiments are conducted on a desktop computer with an Intel Core i7 processor at 3.4 GHz and 8GB of RAM. The proposed ALNS-based metaheuristic is coded in Matlab 2018b and its detailed parameter settings are as follows. It is of note that the parameter values of the ALNS-based metaheuristic are calibrated based on the corresponding ones chosen from the literature and some preliminary experiments with a small number of test instances. As the ALNS framework is the core of the proposed metaheuristic, we first describe the settings for the parameters in the key steps and components of the framework. In each iteration of the ALNS main loop, at least n_r nodes need to be removed from the current solution and n_r is an integer which is randomly drawn from the interval $[\zeta_{min} \cdot |\mathcal{C}|, \zeta_{max} \cdot |\mathcal{C}|]$. To balance the computing time and solution quality, we set $\zeta_{min} = 0.01$ and $\zeta_{max} = 0.3$. The destroy and repair operators introduced in Sections 4.3.4.1 and 4.3.4.2 are critical in the ALNS framework. The related removal operator is controlled by five parameters $(\psi^q, \psi^p, \psi^a, \psi^d, \nu_{re})$ and we set $(\psi^q, \psi^p, \psi^a, \psi^d, \nu_{re}) = (1,1,1,2,5)$. The worst cost removal operator, the worst distance removal operator, and the randomized greedy insertion operator are respectively controlled by parameters ν_{wc}, ν_{wd}, and ν_{rg}. These parameters are introduced to control the randomness in the corresponding operations and we set $\nu_{wc} = 3$, $\nu_{wd} = 3$, and $\nu_{rg} = 2$. The other destroy and repair operators are parameter free. The adaptive mechanism, the dynamic penalty mechanism, the solution reset mechanism, and the SA-based acceptance mechanism are also important in the ALNS framework. We set the five control parameters $(\sigma_1, \sigma_2, \sigma_3, \varpi, \eta^{aw}) = (33, 9, 13, 0.5, 50)$ in the adaptive mechanism introduced in Section 4.3.4.3. We set the six control parameters $(\rho^{min}, \rho^{max}, \rho^0, \rho^{update}, \eta^{ins}, \eta^{des}) = (0.1, 10000, 10, 1.1, 2, 2)$ in the dynamic penalty mechanism introduced in Section 4.3.2. We set $\eta^{sr} = 200$ in the solution reset mechanism in function SolutionReset() and $\chi = 1.01$ in the SA-based acceptance mechanism introduced in Section 4.3.4.5. Next, we describe the stopping criteria of the ALNS main loops in functions ALNSFeasibilityRecovery() and ALNSDistanceMinimization(). In function ALNSFeasibilityRecovery(), we set a limit of $MaxIter1 = 500$ total iterations and $MaxNoImp1 = 200$ iterations without an overall improvement. In function ALNS-DistanceMinimization(), we set $MaxIter2 = 2500$ and $MaxNoImp2 = 1000$. Note that we also set a time limit of 300 seconds for the first two phases of the ALNS-based metaheuristic and a time limit of 1800 seconds for executing the whole algorithm.

Monte Carlo simulation tests are designed to evaluate the robustness of the obtained solutions for all test instances. In the simulation tests, we generate 1000 random and independent scenarios for uncertain pickup demands $\tilde{p}_i$ $(\forall i \in \mathcal{C})$ and travel times $\tilde{t}_{ij}$ $(\forall (i,j) \in \mathcal{A})$. Since the exact distributions of the uncertain parameters are unknown in the VRPSPDTW under uncertainty, uniform distributions are assumed for the uncertain parameters to generate these scenarios following the ideas used in Lu and Gzara (2019) and Munari et al. (2019). In each scenario, we generate a pickup demand vector whose component values are uniformly drawn from the intervals $[\overline{p}_i - \hat{p}_i, \overline{p}_i + \hat{p}_i]$ for all $i \in \mathcal{C}$. A travel time matrix is also generated and its component values are uniformly drawn from the intervals $[\overline{t}_{ij} - \hat{t}_{ij}, \overline{t}_{ij} + \hat{t}_{ij}]$ for all $(i,j) \in \mathcal{A}$. Given the 1000 generated scenarios, we assess the robustness of a solution using the following indicators.

Feasibility ratio (FR). This indicator reflects the robustness of a given solution in terms of its feasibility ratio. It is calculated as the percentage of the scenarios in which the solution remains feasible over the 1000 randomly generated ones. Given the pickup demand vector and the travel time matrix generated in each scenario, the feasibility of the evaluated solution can be simply determined by examining the vehicle capacity and time window constraints based on the vehicle routes in it.

Average customer dissatisfaction rate (ACDR). This indicator reflects the robustness of a given solution in terms of the customer dissatisfaction rate. In each scenario, the customer dissatisfaction rate associated with the evaluated solution is calculated as the percentage of dissatisfied customers over all customers. Based on the vehicle routes in the solution, a customer is considered dissatisfied in circumstances as follows: its time window is missed; its pickup demand is unfulfilled or partially fulfilled; and a vehicle having picked up the customer's goods fails to deliver them to the depot within the allowable time window. Given the 1000 generated scenarios, the average customer dissatisfaction rate associated with the evaluated solution can accordingly be computed.

Average unfulfilled delivery demand (AUDD). This indicator reflects the robustness of a given solution in terms of the total unfulfilled delivery demands of all customers. In each generated scenario, a customer's delivery demand is unfulfilled if its time window is missed. Thus, we can calculate the arrival time of a vehicle at a customer's location to determine its unfulfilled delivery demand based on the vehicle routes in the evaluated solution. Given the 1000 generated scenarios, the average unfulfilled delivery demand associated with the evaluated solution can accordingly be computed.

Average unfulfilled pickup demand (AUPD). This indicator reflects the robustness of a given solution in terms of the total unfulfilled pickup demands of all customers. In each generated scenario, the unfulfilled pickup demand of a customer can be determined by calculating the load and the arrival time of a vehicle at the customer's location based on the vehicle routes in the evaluated solution. Moreover, if a vehicle fails to return to the depot within the allowable time window, the pickup demands of the customers who are served by the vehicle are also considered to be unfulfilled. Given the 1000 generated scenarios, the average unfulfilled pickup demand associated with the evaluated solution can accordingly be computed.

4.4.2 Computational Results

This section first presents the computational results for the medium-sized instances with 50 customers. The impacts of different types of uncertainty are investigated and managerial insights for real-life VRPSPDTW applications under uncertainty are derived based on the solutions generated with different parameter settings in the uncertainty polytopes. Next, the computational results for the large-sized instances with 100 customers are presented. Both deterministic and robust solutions are generated for the large-sized instances. The performance of the ALNS-based metaheuristic is demonstrated by comparing the obtained best deterministic solutions with the current best-known solutions for the benchmark instances in Wang and Chen (2012).

4.4.2.1 Results for Medium-Sized Instances

As mentioned in Section 4.4.1, the benchmark instances introduced in Wang and Chen (2012) for the VRPSPDTW can be grouped into six classes based on the geographical locations of customers, the lengths of time windows, and the capacities of vehicles. Thus, we detail the average results of the solutions generated with different parameter settings in the uncertainty polytopes for the medium-sized instances of all six classes in Tables 4.1-4.6, respectively. In these tables, average results of the obtained solutions are presented in terms of the number of vehicles used (NV), the travel distance (TD), and the four robustness indicators (FR, ACDR, AUDD, and AUPD) from the Monte Carlo simulation tests. As introduced in Section 4.4.1, we consider seven value combinations for the uncertainty budget coefficients (θ_p, θ_t) and two cases for the maximum deviations $(\hat{p}_i, \hat{t}_{ij})$ of the uncertain parameters $(\tilde{p}_i, \tilde{t}_{ij})$ in the uncertainty polytopes. Given every setting for the uncertainty budget coefficients (θ_p, θ_t) and the maximum deviations $(\hat{p}_i, \hat{t}_{ij})$, each instance (e.g., instance Rdp101) of a class (e.g., class Rdp1) is tested ten times and the best solution from ten runs is selected for the instance. The average results for all instances of a class is calculated based on the selected best solutions for all instances of that class under each considered setting for the uncertainty budget coefficients (θ_p, θ_t) and the maximum deviations $(\hat{p}_i, \hat{t}_{ij})$. Note that instances Rdp101, Rdp102, and Rdp103 of class Rdp1 and instance RCdp105 of class RCdp1 are discarded. Because feasible robust solutions do not exist when solving these instances with some considered settings for the uncertainty budget coefficients (θ_p, θ_t) and the maximum deviations $(\hat{p}_i, \hat{t}_{ij})$.

Table 4.1. Average results of the solutions derived with different parameter settings in the uncertainty polytopes for the Cdp1 instances with 50 customers.

(θ_p, θ_t)	Low uncertainty $(\hat{p}_i = 0.2\overline{p}_i, \hat{t}_{ij} = 0.2\overline{t}_{ij})$						High uncertainty $(\hat{p}_i = 0.4\overline{p}_i, \hat{t}_{ij} = 0.4\overline{t}_{ij})$					
	NV	TD	FR (%)	ACDR (%)	AUDD	AUPD	NV	TD	FR (%)	ACDR (%)	AUDD	AUPD
(0,0)	5.11	409.03	21.06	4.68	0.12	266.45	5.11	409.03	13.12	6.26	2.69	300.32
(0.1,0)	5.33	439.40	69.20	0.96	0.77	53.36	5.56	469.36	64.20	1.28	1.85	63.90
(0.2,0)	5.44	465.31	86.19	0.38	2.32	8.54	5.89	453.74	84.87	0.38	0.82	20.16
(0,0.1)	5.11	409.08	21.67	4.40	0.04	255.65	5.11	409.46	15.61	5.65	0.08	285.80
(0,0.2)	5.11	409.46	22.44	4.37	0.00	252.95	5.22	418.12	17.00	5.81	0.00	286.87
(0.1,0.1)	5.44	433.24	74.38	0.83	0.13	51.43	5.67	470.97	71.37	0.99	0.02	60.28
(0.2,0.2)	5.56	452.40	97.62	0.06	0.00	4.16	5.89	457.12	90.76	0.22	0.00	18.52

Table 4.2. Average results of the solutions derived with different parameter settings in the uncertainty polytopes for the Rdp1 instances with 50 customers.

(θ_p, θ_t)	Low uncertainty $(\hat{p}_i = 0.2\overline{p}_i, \hat{t}_{ij} = 0.2\overline{t}_{ij})$						High uncertainty $(\hat{p}_i = 0.4\overline{p}_i, \hat{t}_{ij} = 0.4\overline{t}_{ij})$					
	NV	TD	FR (%)	ACDR (%)	AUDD	AUPD	NV	TD	FR (%)	ACDR (%)	AUDD	AUPD
(0,0)	6.44	751.31	15.14	11.51	27.48	91.35	6.44	751.31	4.86	20.60	49.86	159.66
(0.1,0)	6.44	751.41	16.50	11.41	28.24	84.33	6.56	746.37	6.78	16.88	41.92	124.21
(0.2,0)	6.56	743.95	19.10	10.01	25.68	68.84	6.56	745.38	10.54	16.68	45.45	114.90
(0,0.1)	6.89	748.03	80.53	0.98	2.39	8.91	7.33	759.96	70.86	1.81	4.70	19.71
(0,0.2)	7.00	752.61	96.80	0.13	0.22	2.41	7.33	779.40	89.86	0.48	0.64	10.75
(0.1,0.1)	6.89	748.03	80.53	0.98	2.39	8.91	7.33	760.29	72.02	1.72	4.69	17.93
(0.2,0.2)	7.00	754.34	98.03	0.11	0.21	0.60	7.33	781.31	94.20	0.32	0.64	2.90

Based on the results in Tables 4.1-4.6, meaningful findings are observed along with a detailed analysis. In addition, helpful managerial insights for real-life VRPSPDTW applications under pickup demand and travel time uncertainty are derived. Note that some of the findings are similar to those observed in Section 3.4.2.

1) The deterministic solutions generated without considering uncertainty $((\theta_p, \theta_t) = (0, 0))$ for the Cdp1, Rdp1, and RCdp1 instances generally attain low robust-

Table 4.3. Average results of the solutions derived with different parameter settings in the uncertainty polytopes for the RCdp1 instances with 50 customers.

(θ_p,θ_t)	Low uncertainty $(\hat{p}_i = 0.2\overline{p}_i, \hat{t}_{ij} = 0.2\overline{t}_{ij})$						High uncertainty $(\hat{p}_i = 0.4\overline{p}_i, \hat{t}_{ij} = 0.4\overline{t}_{ij})$					
	NV	TD	FR (%)	ACDR (%)	AUDD	AUPD	NV	TD	FR (%)	ACDR (%)	AUDD	AUPD
(0,0)	6.57	777.07	15.91	8.88	21.49	178.34	6.57	777.07	3.01	17.57	47.99	288.60
(0.1,0)	6.71	783.50	33.49	4.90	16.32	80.78	6.71	813.20	5.27	17.47	46.75	189.34
(0.2,0)	6.71	810.71	23.93	7.51	22.55	79.69	6.71	827.41	4.84	19.56	50.21	200.03
(0,0.1)	6.71	823.09	35.36	3.19	2.38	105.09	8.00	871.81	46.03	3.03	1.53	94.76
(0,0.2)	7.14	818.42	59.04	1.26	0.10	68.19	8.14	897.20	60.03	1.64	0.42	76.09
(0.1,0.1)	6.86	817.06	70.39	1.66	2.15	46.58	8.00	879.27	79.79	1.73	1.44	29.08
(0.2,0.2)	7.14	836.43	99.16	0.04	0.04	1.35	8.14	911.99	94.87	0.17	0.63	5.81

Table 4.4. Average results of the solutions derived with different parameter settings in the uncertainty polytopes for the Cdp2 instances with 50 customers.

(θ_p,θ_t)	Low uncertainty $(\hat{p}_i = 0.2\overline{p}_i, \hat{t}_{ij} = 0.2\overline{t}_{ij})$						High uncertainty $(\hat{p}_i = 0.4\overline{p}_i, \hat{t}_{ij} = 0.4\overline{t}_{ij})$					
	NV	TD	FR (%)	ACDR (%)	AUDD	AUPD	NV	TD	FR (%)	ACDR (%)	AUDD	AUPD
(0,0)	2.00	398.98	90.98	0.25	1.55	4.71	2.00	398.98	78.55	0.64	3.24	16.15
(0.1,0)	2.00	399.88	89.61	0.36	2.38	3.02	2.00	402.13	84.54	0.44	3.12	3.19
(0.2,0)	2.00	400.07	89.61	0.36	2.38	3.02	2.00	404.06	79.49	0.81	5.84	7.84
(0,0.1)	2.00	401.84	98.16	0.05	0.00	3.38	2.00	402.47	93.71	0.18	0.00	12.78
(0,0.2)	2.00	401.84	98.16	0.05	0.00	3.38	2.00	401.84	93.71	0.18	0.00	12.78
(0.1,0.1)	2.00	403.10	100.0	0.00	0.00	0.00	2.00	404.90	100.0	0.00	0.00	0.00
(0.2,0.2)	2.00	403.29	100.0	0.00	0.00	0.00	2.00	407.30	100.0	0.00	0.00	0.00

ness. Note that no uncertainty is considered in all uncertainty polytopes when the uncertainty budget coefficients $(\theta_p, \theta_t) = (0,0)$ and the solution derived with $(\theta_p, \theta_t) = (0,0)$ is referred to as the deterministic solution for each medium-sized instance. As shown in Table 4.3, the average FR of the deterministic solutions for the RCdp1 instances is only 15.91% in the Monte Carlo simulation tests even considering the low uncertainty case. Moreover, the average ACDR with the deterministic solutions for the RCdp1 instances is more than 17% in the simulation tests considering the high uncertainty case. Similar observations are made for the deterministic

Table 4.5. Average results of the solutions derived with different parameter settings in the uncertainty polytopes for the Rdp2 instances with 50 customers.

(θ_p, θ_t)	Low uncertainty $(\hat{p}_i = 0.2\overline{p}_i, \hat{t}_{ij} = 0.2\overline{t}_{ij})$						High uncertainty $(\hat{p}_i = 0.4\overline{p}_i, \hat{t}_{ij} = 0.4\overline{t}_{ij})$					
	NV	TD	FR (%)	ACDR (%)	AUDD	AUPD	NV	TD	FR (%)	ACDR (%)	AUDD	AUPD
(0,0)	2.00	663.34	67.29	0.96	7.43	4.95	2.00	663.34	53.23	1.81	13.91	10.39
(0.1,0)	2.00	663.34	67.29	0.96	7.43	4.95	2.00	663.34	53.23	1.81	13.91	10.39
(0.2,0)	2.00	663.34	67.29	0.96	7.43	4.95	2.00	663.34	53.23	1.81	13.91	10.39
(0,0.1)	2.00	674.16	99.70	0.01	0.04	0.01	2.09	670.78	99.83	0.00	0.03	0.01
(0,0.2)	2.00	676.07	100.0	0.00	0.00	0.00	2.09	674.79	100.0	0.00	0.00	0.00
(0.1,0.1)	2.00	673.83	99.70	0.01	0.04	0.01	2.09	670.78	99.83	0.00	0.03	0.01
(0.2,0.2)	2.00	676.07	100.0	0.00	0.00	0.00	2.09	674.79	100.0	0.00	0.00	0.00

Table 4.6. Average results of the solutions derived with different parameter settings in the uncertainty polytopes for the RCdp2 instances with 50 customers.

(θ_p, θ_t)	Low uncertainty $(\hat{p}_i = 0.2\overline{p}_i, \hat{t}_{ij} = 0.2\overline{t}_{ij})$						High uncertainty $(\hat{p}_i = 0.4\overline{p}_i, \hat{t}_{ij} = 0.4\overline{t}_{ij})$					
	NV	TD	FR (%)	ACDR (%)	AUDD	AUPD	NV	TD	FR (%)	ACDR (%)	AUDD	AUPD
(0,0)	2.13	723.34	49.23	1.90	15.94	19.00	2.13	723.34	34.45	3.50	31.00	36.52
(0.1,0)	2.13	723.34	49.23	1.90	15.94	19.00	2.13	723.34	34.45	3.50	31.00	36.52
(0.2,0)	2.13	723.34	49.23	1.90	15.94	19.00	2.13	723.34	34.45	3.50	31.00	36.52
(0,0.1)	2.25	722.86	100.0	0.00	0.00	0.00	2.38	737.10	99.85	0.00	0.05	0.03
(0,0.2)	2.25	723.44	100.0	0.00	0.00	0.00	2.50	714.09	100.0	0.00	0.00	0.00
(0.1,0.1)	2.25	722.86	100.0	0.00	0.00	0.00	2.38	737.10	99.85	0.00	0.05	0.03
(0.2,0.2)	2.25	723.44	100.0	0.00	0.00	0.00	2.50	714.09	100.0	0.00	0.00	0.00

solutions derived for the Cdp1 and Rdp1 instances in the simulation tests considering both low and high uncertainty cases. However, the deterministic solutions generated for the Cdp2, Rdp2, and RCdp2 instances attain relatively high robustness in terms of the indicators FR, ACDR, AUDD, and AUPD. As shown in Tables 4.4-4.6, the average FRs of the deterministic solutions for the Cdp2, Rdp2, and RCdp2 instances are respectively 78.55%, 53.23%, and 34.45% in the simulation tests even considering the high uncertainty case. Moreover, the average ACDRs of such solutions for the Cdp2 and Rdp2 instances are only 0.64% and 1.81%, respectively. Thus, deterministic routing strategies are not easily to be affected by uncertainty in pickup demands and travel times for real-life VRPSPDTW applications with wide time windows and large-capacity vehicles. However, such strategies are fragile under both types of uncertainty for VRPSPDTW applications with narrow time windows and small-capacity vehicles. Moreover, they may lead to high customer dissatisfaction rates and large unfulfilled pickup demands.

2) The robust solutions generated considering both types of uncertainty ($(\theta_p, \theta_t) = (0.1, 0.1)$ or $(0.2, 0.2)$) for the instances of most classes attain relatively substantial robustness in terms of the indicators FR, ACDR, AUPD, and AUDD. Consider the robust solutions generated with $(\theta_p, \theta_t) = (0.2, 0.2)$ for the Rdp1 instances in the low uncertainty case (Table 4.2). Their average FR is more than 95% and their average ACDR is only 0.11%. Moreover, the average AUDD and AUPD associated with these solutions are close to 0. Similar observations are made for the robust solutions generated with $(\theta_p, \theta_t) = (0.2, 0.2)$ for the Cdp1, RCdp1, Cdp2, Rdp2, and RCdp2 instances in both low and high uncertainty cases. Thus, robust routing strategies can be reliable for practical VRPSPDTW applications under uncertainty in pickup demands and travel times. Moreover, these strategies often lead to low customer dissatisfaction rates and small unfulfilled delivery and pickup demands.

3) Although the robust solutions generated considering both types of uncertainty for the Cdp1, Rdp1, and RCdp1 instances attain substantial robustness, they may incur large additional routing costs in terms of the travel distance and the number of vehicles used. Consider the robust solutions generated with $(\theta_p, \theta_t) = (0.2, 0.2)$ for the RCdp1 instances in the high uncertainty case (Table 4.3). Compared to their deterministic counterparts, the average travel distance increases from 777.07 to 911.99 and the average number of vehicles used increases from 6.57 to 8.14 in the robust solutions. Similar observations are made for the robust solutions derived

for the Rdp1 and Cdp1 instances in the high uncertainty case. However, the robust solutions derived for the Cdp2, Rdp2, and RCdp2 instances can reach a very high robustness level at almost no additional cost. Consider the robust solutions generated with $(\theta_p, \theta_t) = (0.2, 0.2)$ for the Cdp2 instances in the high uncertainty case (Table 4.4). Their average FR is 100.00% and their average ACDR is 0.00%. Moreover, the average AUDD and AUPD associated with these solutions are all 0. Compared to their deterministic counterparts, the average number of vehicles used stays the same and the average travel distance only increases from 398.98 to 407.30 in the robust solutions. Similar observations are made for the robust solutions generated with $(\theta_p, \theta_t) = (0.1, 0.1)$ or $(0.2, 0.2)$ for the Rdp2 and RCdp2 instances in both low and high uncertainty cases. Thus, highly robust routing strategies can be generated for practical VRPSPDTW applications with wide time windows and large-capacity vehicles at little additional cost. However, it can be much more expensive (e.g., more vehicles and longer travel distances are required) to derive such strategies for VRPSPDTW applications with narrow time windows and small-capacity vehicles under high pickup demand and travel time uncertainty.

4) Uncertainty in both pickup demands and travel times have an impact on deriving robust solutions for the Cdp1, Rdp1, and RCdp1 instances. Specifically, pickup demand uncertainty has a strong impact on deriving robust solutions for the Cdp1 instances. However, as shown in Table 4.1, both the deterministic and robust solutions for the Cdp1 instances are not very sensitive to travel time uncertainty. Thus, decision-makers should pay more attention to pickup demand uncertainty when deriving robust routing strategies for real-life VRPSPDTW applications which are similar to the Cdp1 instances in uncertain environments. Uncertainty in travel times has an obvious impact on deriving robust solutions for the Rdp1 instances. However, as shown in Table 4.2, uncertainty in pickup demands only has a small impact on deriving such solutions for the Rdp1 instances especially in the low

uncertainty case. Similar observations are made for the RCdp1 instances based on the results in Table 4.3. Thus, uncertainty in travel times is a critical factor when deriving robust routing strategies for practical VRPSPDTW applications which are similar to the Rdp1 and RCdp1 instances in uncertain environments.

5) Uncertainty in pickup demands only has a small impact on deriving robust solutions for the Cdp2, Rdp2, and RCdp2 instances. As shown in Table 4.6, the average results of the robust solutions generated considering only pickup demand uncertainty for the RCdp2 instances are the same as those of their deterministic counterparts in both low and high uncertainty cases. Similar observations are made for the Rdp2 instances. However, uncertainty in travel times has an obvious impact on deriving robust solutions for the Cdp2, Rdp2, and RCdp2 instances. Focus on the robust solutions generated with $(\theta_p, \theta_t) = (0, 0.2)$ considering only travel time uncertainty for the Rdp2 instances in the high uncertainty case (Table 4.5). Their average FR is 100% and their average ACDR is 0.00%. Similar observations are made for the Cdp2 and RCdp2 instances. Thus, decision-makers should pay more attention to travel time uncertainty when deriving robust routing strategies for practical VRPSPDTW applications with wide time windows and large-capacity vehicles in uncertain environments.

6) The AUPD is larger than the AUDD associated with both the deterministic and robust solutions derived for the instances of most classes in the Monte Carlo simulation tests, as shown in Tables 4.1-4.6. The reasons can be threefold. Firstly, each vehicle should start from the depot with the goods it must delivery in the VRPSPDTW under uncertainty. If a vehicle misses the time windows of several customers at the beginning of its route due to travel time uncertainty, the undelivered goods on the vehicle may cause large unfulfilled pickup demands of the customers at the end of the route. Secondly, pickup demand uncertainty may directly lead to unfulfilled

pickup demands while it will not cause unfulfilled delivery demands. Thirdly, if a vehicle fails to return to the depot within the allowable time window due to travel time uncertainty, the pickup demands of all customers who are served by the vehicle are considered as unfulfilled in the simulation tests.

4.4.2.2 Results for Large-Sized Instances

To investigate the effectiveness and efficiency of the ALNS-based metaheuristic, we derive the robust solutions for the large-sized instances of Wang and Chen (2012) with 100 customers using the following parameter settings in the uncertainty polytopes. As introduced in Section 4.4.1, we set the uncertainty budget coefficients $(\theta_p, \theta_t) = (0.2, 0.2)$ for the instances of classes Cdp1, Rdp1, and RCdp1 and $(\theta_p, \theta_t) = (0.1, 0.1)$ for the instances of classes Cdp2, Rdp2, and RCdp2. Moreover, we only consider the high uncertainty case for the maximum deviations $(\hat{p}_i, \hat{t}_{ij})$ of the uncertain parameters $(\tilde{p}_i, \tilde{t}_{ij})$, in which $\hat{p}_i = 0.4\overline{p}_i$ for all $i \in \mathcal{C}$ and $\hat{t}_{ij} = 0.4\overline{t}_{ij}$ for all $(i, j) \in \mathcal{A}$. In addition to the robust solutions, we solve the large-sized instances with $(\theta_p, \theta_t) = (0, 0)$ by using the ALNS-based metaheuristic with slight modifications to generate the corresponding deterministic solutions. We present the average results of the obtained deterministic and robust solutions for the instances of Wang and Chen (2012) of each class with 100 customers in Table 4.7. The headings NV, TD, FR, AUDD, and AUPD have the same meaning as those in Tables 4.1-4.6. Note that we run every instance of a class ten times and select the best solution from ten runs for the instance given the robust or deterministic setting. Then, we calculate the average results of the robust (deterministic) solutions for all instances of each class based on the selected best robust (deterministic) solution for every instance of that class.

In Table 4.7, we observe that the deterministic solutions derived for the large-sized instances of classes Cdp1, Rdp1, and RCdp1 attain extremely low robustness in terms of the indicators FR, ACDR, AUDD, and AUPD. Consider the deterministic solutions derived for the RCdp1 instances. Their average FR is nearly 0 and their average ACDR

Table 4.7. Average results of the deterministic and robust solutions derived for the instances of Wang and Chen (2012) with 100 customers.

Instance class	Solution type	NV	TD	FR (%)	ACDR (%)	AUDD	AUPD
Cdp1	Deterministic	10.11	947.71	0.11	9.06	10.29	632.26
	Robust	11.11	1109.01	84.38	0.20	0.04	30.56
Cdp2	Deterministic	3.00	589.86	30.74	15.40	5.16	427.11
	Robust	3.00	671.70	98.91	0.02	0.00	6.90
Rdp1	Deterministic	10.67	1119.69	0.08	25.04	79.17	403.15
	Robust	12.22	1184.27	90.42	0.49	0.79	9.36
Rdp2	Deterministic	2.73	954.12	15.05	6.41	35.07	88.68
	Robust	3.09	942.93	99.81	0.01	0.03	0.16
RCdp1	Deterministic	11.43	1368.94	0.03	27.64	126.34	562.05
	Robust	13.86	1525.39	91.41	0.23	0.50	8.04
RCdp2	Deterministic	3.25	1120.48	15.93	5.89	46.78	88.68
	Robust	3.38	1155.21	99.65	0.00	0.04	0.07

is more than 25%. Moreover, the average AUPD associated with these solutions is more than 500. Similar observations are made for the deterministic solutions derived for the Cdp1 and Rdp1 instances. The deterministic solutions for the Cdp2, Rdp2, and RCdp2 instances attain relatively low robustness in terms of the four indicators. For example, the average ACDR of the deterministic solutions for the Cdp2 instances is more than 15% and the average AUPD associated with these solutions is more than 400. Thus, deterministic routing strategies are unreliable for practical VRPSPDTW applications under high pickup demand and travel time uncertainty. However, the robust solutions derived for the large-sized instances of most classes attain substantial robustness even in the high uncertainty case. Consider the robust solutions derived for the Rdp2 instances. Their average FR is close to 100% and their average ACDR is only 0.01%. Moreover, the average AUDD and AUPD associated with these solutions are all close to 0. Similar observations are made for the robust solutions derived for the Cdp1, Cdp2, Rdp1, RCdp1, RCdp2 instances. Despite their high robustness, the robust solutions derived for the instances of some classes may lead to large additional routing costs in terms of the travel distance and the number of vehicles used. For example, compared to the deterministic solutions for the RCdp1 instances, the average number of vehicles used increases from 11.43 to 13.86 and the average distance travelled increases from 1368.94 to 1525.39 in the corresponding robust solutions. However, as shown in Table 4.7, the robust solutions for the instances of most classes attain high robustness at a small addition cost even in the high uncertainty case. Thus, we can conclude that robust routing strategies are reliable and cost-effective for practical VRPSPDTW applications under pickup demand and travel time uncertainty.

To show the performance of the ALNS-based metaheuristic, we first report the average results of the best and average robust solutions among ten runs in terms of the number of vehicles used (NV) and the travel distance (TD) for the benchmark instances of Wang and Chen (2012) of each class with 100 customers in Table 4.8. Next, we compare the above obtained best deterministic solutions with the current best-known solutions for the benchmark instances. As reviewed in Section 2.3, the ALNS-PR approach proposed in Hof and Schneider (2019) and the lexicographic-based two-stage algorithm proposed in Shi et al. (2020) have a much better performance than the previous solution methods for the VRPSPDTW in the literature. We combine all the best solutions reported in Hof and Schneider (2019) and Shi et al. (2020) to obtain the current best-known solutions for the benchmark instances of Wang and Chen (2012) with 100 customers. The comparison results between the best deterministic solutions found by the proposed ALNS-based metaheuristic and the current best-known solutions are detailed in Table 4.9.

Table 4.8. Average results of the best and average robust solutions derived for the instances of Wang and Chen (2012) with 100 customers.

Instance class	Best robust solutions			Average robust solutions		
	NV	TD	Time (s)	NV	TD	Time (s)
Cdp1	11.11	1109.01	547.84	11.20	1108.69	544.60
Cdp2	3.00	671.70	746.80	3.00	673.73	734.76
Rdp1	12.22	1184.27	405.37	12.44	1183.00	440.70
Rdp2	3.09	942.93	859.91	3.09	954.13	880.46
RCdp1	13.86	1525.39	441.60	14.11	1521.47	437.70
RCdp2	3.38	1155.21	883.73	3.38	1172.09	872.75

In Table 4.8, it can be observed that the average results of the best and average robust solutions derived for the large-sized instances of each class are very close. Moreover, the average computing time to generate the robust solutions for the instances of every class is less than 900 seconds. Thus, the performance of the ALNS-based metaheuristic is stable and it can generate robust solutions for the large-sized instances within a reasonable computing time. In Table 4.9, the columns under "Gap" show the difference between the best deterministic solution found by the ALNS-based metaheuristic and the current best-known solution for each benchmark instance in terms of the number of vehicles used (NV) and the travel distance (TD) in percentage. The metaheuristic updates the best-known solutions for 14 out of 56 instances and identifies the same best-known solutions for 30 out of 56 instances. The new best solutions are highlighted in bold in Table 4.9. Thus, we can conclude that the ALNS-based metaheuristic has a superior performance compared to the state-of-the-art solution methods for the VRPSPDTW in terms of solution quality.

Table 4.9. Best deterministic solutions generated by the ALNS-based metaheuristic in comparison to the current best-known solutions for the instances of Wang and Chen (2012).

Instance	Best-known solutions		ALNS-based metaheuristic			Gap	
	NV	TD	NV	TD	Time (s)	NV	TD (%)
Cdp101	11	976.04	11	977.57	120.75	0	0.16
Cdp102	10	941.49	10	941.49	148.29	0	0.00
Cdp103	10	892.98	10	892.98	167.28	0	0.00
Cdp104	10	871.40	10	871.40	173.32	0	0.00
Cdp105	10	1053.12	10	1053.12	171.65	0	0.00
Cdp106	10	963.45	10	963.45	193.52	0	0.00
Cdp107	10	987.64	10	987.64	223.73	0	0.00
Cdp108	10	932.88	10	932.49	161.15	0	**-0.04**
Cdp109	10	909.27	10	909.27	180.20	0	0.00
Cdp201	3	591.56	3	591.56	76.48	0	0.00
Cdp202	3	591.56	3	591.56	83.40	0	0.00
Cdp203	3	591.17	3	591.17	196.29	0	0.00
Cdp204	3	590.60	3	590.60	153.89	0	0.00
Cdp205	3	588.88	3	588.88	82.34	0	0.00
Cdp206	3	588.49	3	588.49	104.56	0	0.00
Cdp207	3	588.29	3	588.29	88.65	0	0.00
Cdp208	3	588.32	3	588.32	92.88	0	0.00

Table 4.9 Continued.

Instance	Best-known solutions		ALNS-based metaheuristic			Gap	
	NV	TD	NV	TD	Time (s)	NV	TD (%)
Rdp101	19	1650.80	19	1650.80	190.35	0	0.00
Rdp102	17	1486.12	17	1486.12	193.51	0	0.00
Rdp103	13	1294.75	13	1294.64	197.02	0	**-0.01**
Rdp104	10	984.81	10	984.81	210.77	0	0.00
Rdp105	14	1377.11	14	1377.11	171.15	0	0.00
Rdp106	12	1252.03	12	1252.03	163.77	0	0.00
Rdp107	10	1121.86	10	1117.19	210.20	0	**-0.42**
Rdp108	9	965.54	9	965.22	258.81	0	**-0.03**
Rdp109	11	1194.73	11	1194.73	221.39	0	0.00
Rdp110	10	1148.20	10	1131.04	192.08	0	**-1.49**
Rdp111	10	1098.84	10	1101.44	214.39	0	0.24
Rdp112	9	1010.42	10	953.63	235.50	1	-5.62
Rdp201	4	1258.09	4	1252.37	243.28	0	**-0.45**
Rdp202	3	1208.88	3	1191.70	314.90	0	**-1.42**
Rdp203	3	946.28	3	941.56	305.76	0	**-0.50**
Rdp204	2	833.09	2	833.65	407.21	0	0.07
Rdp205	3	994.43	3	994.43	342.72	0	0.00
Rdp206	3	913.68	3	918.33	321.91	0	0.51
Rdp207	2	890.61	2	898.33	406.92	0	0.87
Rdp208	2	726.82	2	726.82	305.64	0	0.00
Rdp209	3	909.16	3	910.55	269.32	0	0.15
Rdp210	3	939.37	3	941.85	404.07	0	0.26
Rdp211	2	904.44	2	885.71	485.16	0	**-2.07**
RCdp101	14	1708.21	14	1708.21	186.71	0	0.00
RCdp102	12	1583.62	12	1588.79	161.22	0	0.33
RCdp103	11	1283.62	11	1290.66	187.29	0	0.55
RCdp104	10	1171.65	10	1171.65	285.17	0	0.00
RCdp105	14	1548.38	13	1636.58	187.43	**-1**	5.70
RCdp106	12	1392.47	12	1392.47	172.34	0	0.00
RCdp107	11	1255.06	11	1252.79	181.06	0	**-0.18**
RCdp108	10	1198.36	10	1177.98	236.68	0	**-1.70**
RCdp201	4	1406.94	4	1406.94	268.93	0	0.00
RCdp202	3	1412.52	3	1368.27	343.92	0	**-3.13**
RCdp203	3	1050.64	3	1050.64	363.96	0	0.00
RCdp204	3	798.46	3	798.46	328.94	0	0.00
RCdp205	4	1297.65	4	1303.61	313.83	0	0.46
RCdp206	3	1146.32	3	1146.32	389.87	0	0.00
RCdp207	3	1061.84	3	1061.14	370.87	0	**-0.07**
RCdp208	3	828.14	3	828.44	298.47	0	0.04

4.5 Summary

In this chapter, we study the VRPSPDTW under two types of uncertainty. Specifically, uncertainty in pickup demands and travel times are considered and captured by the route-dependent uncertainty sets. Based on the uncertainty sets, we present a robust mathematical formulation to model the problem. Due to the complexity of the formulation, we design an ALNS-based metaheuristic which can solve it with large-sized instances. We adopt the benchmark instances for the standard VRPSPDTW introduced in Wang and Chen (2012) and conduct extensive computational experiments. The computational results show that both deterministic and robust solutions can be generated for the medium-sized instances with 50 customers and the large-sized instances with 100 customers within a reasonable computing time. The performance of the proposed metaheuristic is demonstrated by comparing the obtained best deterministic solutions with the current best-known solutions for the benchmark instances with 100 customers. The comparison results show that the ALNS-based metaheuristic finds more than half of the current best-known solutions and even generates new best solutions for one fourth of the benchmark instances. Monte Carlo simulation tests are designed to evaluate the robustness of all obtained solutions. A detailed analysis of the results is performed, which generates useful managerial insights for practical VRPSPDTW applications under pickup demand and travel time uncertainty.

Chapter 5

Two-Echelon Multiple-Trip Vehicle Routing Problem with Time Windows and Satellite Synchronization Under Uncertainty

5.1 Introduction

Due to increasing traffic congestion, road weight limits, and customer accessibility in urban areas, multi-echelon transportation systems, especially two-echelon ones, have been employed by both public administrations and private companies to improve operational efficiency and reduce transportation costs. In a two-echelon distribution system, the transportation network comprises two levels with intermediate facilities (called satellites). In the first level, the freight is transported from a distribution centre to a set of satellites in the proximity of customers. In the second level, the freight is transported from the satellites to the customers. Many city logistics activities such as express delivery, grocery distribution, and municipal solid waste collection have employed such two-echelon systems (Cuda et al., 2015).

Despite the popularity of these systems in city logistics, it is challenging to generate routing strategies for vehicles in both echelons because of practical constraints. For example, customers may expect their goods to be collected or delivered within specified time windows. Besides, customers may live at narrow streets which are difficult to manoeuvre or access for large first-echelon vehicles. Thus, small second-echelon vehicles may need to perform multiple trips to collect or distribute all the goods. Moreover, actual customer demands could be uncertain due to the limited information and rapidly-changing customer needs. Finally, it is expensive to manage a satellite in urban areas: transporters sometimes use reserved parking lots or bus depots as satellites, which given their lack of storage capacity warrants vehicle synchronization.

In this chapter, we study the 2E-MTVRPTWSS under customer demand uncertainty. The 2E-MTVRPTWSS was introduced by Grangier et al. (2016) based on a two-echelon distribution system. However, since uncertainty in customer demands is more common in freight collection activities such as municipal solid waste collection and parcel collection than in freight distribution, we study it under such uncertainty based on a two-echelon collection system. To capture the customer demand uncertainty, an uncertainty set with novel uncertainty polytopes is defined based on vehicle routes in both echelons. The problem is modelled by a robust mathematical formulation with the uncertainty set and solved by a VNS-based metaheuristic. Numerical experiments are conducted with the adapted benchmark instances to show the effectiveness of the proposed robust formulation and the solution approach.

The rest of this chapter is organized as follows. Section 5.2 first introduces the 2E-MTVRPTWSS based on a two-echelon collection system. Then, the 2E-MTVRPTWSS under customer demand uncertainty is described and the robust mathematical formulation with a novel demand uncertainty set is presented. Section 5.3 introduces the details of the

VNS-based metaheuristic. Section 5.4 presents the computational results from a series of numerical experiments. Finally, Section 5.5 summarizes the whole chapter.

5.2 Problem Statement and Model Formulation

5.2.1 The 2E-MTVRPTWSS

The 2E-MTVRPTWSS with a two-echelon collection system can be described as follows. A logistics centre adopts a two-echelon transportation network with intermediate transfer points to collect goods from a set of geographically dispersed customers. The intermediate transfer points are called satellites which are assumed to have no storage capacity. Each customer has a demand (quantity of goods to be collected) which must be satisfied within a specified time window. In each echelon of the network, a fleet of homogeneous vehicles are employed to perform the collection tasks. The capacity of a second-echelon vehicle is much smaller than that of a first-echelon vehicle. In the problem, the second-echelon vehicles initially collect the goods from the customers and transfer their loads to the first-echelon vehicles at the satellites. The first-echelon vehicles then transport all the goods from the satellites to the logistics centre. As the satellites have no storage capacity, a second-echelon vehicle needs to synchronize with a first-echelon vehicle at the same satellite (spatial synchronization) over the same period of time (temporal synchronization). Note that a first-echelon vehicle is assumed to be able to receive the loads from several second-echelon vehicles at a satellite simultaneously. After transferring their loads, the second-echelon vehicles can continue to collect goods from unserved customers. When they need to unload, they move again to satellites. Thus, multiple trips can be performed by the second-echelon vehicles. The first-echelon vehicles start from the logistics centre and the second-echelon vehicles start from a depot in the proximity of the customers. Vehicles in both echelons need to return to their depots before a maximum planning horizon. Direct shipping from the customers to the logistics centre is not allowed. A solution of the problem calls for the design of a set of first-echelon routes, a set of second-echelon routes, and a synchronization schedule, such that the capacity and temporal constraints are satisfied.

Figure 5.1 depicts a solution to a 2E-MTVRPTWSS instance with two satellites and seven customers. Second-echelon vehicle 1 starts from its depot, collects the goods of two customers, and transfers its load to first-echelon vehicle 1 at a satellite. Then, it visits two more customers and transfers its load to first-echelon vehicle 1 at another satellite. Second-echelon vehicle 2 also starts from its depot, visits three customers, and transfers its load to first-echelon vehicle 1 at a satellite. When all customers are served, second-echelon vehicles 1 and 2 return to their depot and first-echelon vehicle 1 finally ships all the goods to the logistics centre.

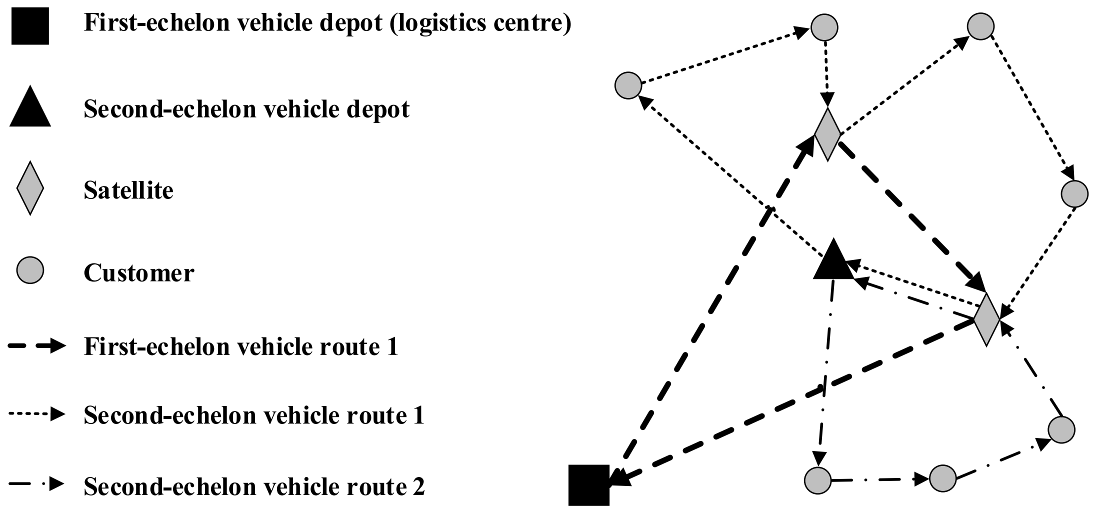

First-echelon vehicle depot (logistics centre)
Second-echelon vehicle depot
Satellite
Customer
First-echelon vehicle route 1
Second-echelon vehicle route 1
Second-echelon vehicle route 2

Figure 5.1 A solution to the 2E-MTVRPTWSS.

The above described problem can be defined on a directed graph $\mathcal{G} = (\mathcal{N}, \mathcal{A})$. $\mathcal{N}$ is a set of nodes and $\mathcal{N} = \{o_1\} \cup \{o_2\} \cup \mathcal{C} \cup \mathcal{S}$. Set $\mathcal{C}$ denotes a set of customers and set $\mathcal{S}$ denotes a set of satellites. Nodes o_1 and o_2 represent the depots of first- and second-echelon vehicles, respectively. For modelling purposes, we split the first-echelon vehicle depot o_1 into a start and an end depots $\{o_1, o_1'\}$. Similarly, we split the second-echelon vehicle depot o_2 into a start and an end depots $\{o_2, o_2'\}$. Moreover, set $\mathcal{S}$ is assumed to contain a sufficient number of replications of each satellite for the purpose of allowing multiple visits by different second-echelon vehicles. Since a two-echelon transportation network is considered in the problem, we define the first-echelon network on $\mathcal{G}_1 = (\mathcal{N}_1, \mathcal{A}_1)$ with $\mathcal{N}_1 = \{o_1\} \cup \mathcal{S} \cup \{o'_1\}$ and $\mathcal{A}_1 = \{(o_1, j) | j \in \mathcal{S}\} \cup \{(i,j) | i,j \in \mathcal{S}, i \neq j\} \cup \{(i, o'_1) | i \in \mathcal{S}\}$. We define the second-echelon network on $\mathcal{G}_2 = (\mathcal{N}_2, \mathcal{A}_2)$ with $\mathcal{N}_2 = \{o_2\} \cup \mathcal{C} \cup \mathcal{S} \cup \{o'_2\}$ and $\mathcal{A}_2 = \{(o_2, j) | j \in \mathcal{C}\} \cup \{(i,j) | i \in \mathcal{C}, j \in \mathcal{S}\} \cup \{(i,j) | i,j \in \mathcal{C}, i \neq j\} \cup \{(i,j) | i \in \mathcal{S}, j \in \mathcal{C}\} \cup \{(i, o'_2) | i \in \mathcal{S}\}$. $\mathcal{A}$ denotes a set of arcs and $\mathcal{A} = \mathcal{A}_1 \cup \mathcal{A}_2$. Each arc $(i,j) \in \mathcal{A}$ is associated with a travel distance d_{ij}. Each customer $i \in \mathcal{C}$ has a demand q_i, a service time s_i, and a time window $[a_i, b_i]$. Time windows are not associated with the satellites and the vehicle depots. However, vehicles in both echelons must return to their corresponding depots before a maximum planning horizon T_{max}. In addition, we assume that service times at the satellites and the depots can be integrated into the vehicle travel times from and to them. Thus, the service times at these locations are assumed to be zero in the problem and $s_i = 0$ for all $i \in \{o_1\} \cup \{o_2\} \cup \mathcal{S}$. We use set $\mathcal{K}_1$ to denote a fleet of first-echelon vehicles. For each vehicle $k_1 \in \mathcal{K}_1$, it has a capacity Q_1 and a travel time t^1_{ij} on arc $(i,j) \in \mathcal{A}_1$. We use set $\mathcal{K}_2$ to denote a fleet of second-echelon vehicles. For each vehicle $k_2 \in \mathcal{K}_2$, it has a capacity Q_2 and a travel time t^2_{ij} on arc $(i,j) \in \mathcal{A}_2$. The 2E-MTVRPTWSS considers a hierarchical objective which minimizes the number of first-echelon vehicles used (primary criterion), the number of second-echelon vehicles used (secondary criterion), and the total travel distance of the vehicle routes in two echelons (tertiary criterion).

To model the 2E-MTVRPTWSS, we present a mathematical formulation based on those proposed in Grangier et al. (2016) and Anderluh et al. (2016). The decision variables defined in the formulation are described as follows. Binary variable $x^{k_1}_{ij} = 1$ if first-echelon vehicle $k_1 \in \mathcal{K}_1$ traverses arc $(i,j) \in \mathcal{A}_1$ and 0 otherwise. Similarly, binary variable $x^{k_2}_{ij} = 1$ if second-echelon vehicle $k_2 \in \mathcal{K}_2$ traverses arc $(i,j) \in \mathcal{A}_2$ and 0 otherwise. Continuous variables $y^{k_1}_i$ and $w^{k_1}_i$ respectively specify the arrival time and the waiting time of first-echelon vehicle $k_1 \in \mathcal{K}_1$ at node $i \in \mathcal{N}_1$. Similarly, continuous variables

$y_i^{k_2}$ and $w_i^{k_2}$ respectively specify the arrival time and the waiting time of second-echelon vehicle $k_2 \in \mathcal{K}_2$ at node $i \in \mathcal{N}_2$. In addition, continuous variable $z_i^{k_1}$ specifies the load of first-echelon vehicle $k_1 \in \mathcal{K}_1$ before visiting node $i \in \mathcal{N}_1$. Continuous variable $z_i^{k_2}$ specifies the load of second-echelon vehicle $k_2 \in \mathcal{K}_2$ before visiting node $i \in \mathcal{N}_2$. Based on the above defined decision variables, the deterministic mathematical formulation for the 2E-MTVRPTWSS is presented as follows:

(2E-MTVRPTWSS)

$$\textit{lex-}\min\left(\sum_{k_1\in\mathcal{K}_1}\sum_{j\in\mathcal{S}} x_{o_1 j}^{k_1}, \sum_{k_2\in\mathcal{K}_2}\sum_{j\in\mathcal{C}} x_{o_2 j}^{k_2}, \sum_{k_1\in\mathcal{K}_1}\sum_{(i,j)\in\mathcal{A}_1} d_{ij}x_{ij}^{k_1} + \sum_{k_2\in\mathcal{K}_2}\sum_{(i,j)\in\mathcal{A}_2} d_{ij}x_{ij}^{k_2}\right) \tag{5.1}$$

$$\text{s.t.} \sum_{j\in\mathcal{S}} x_{o_1 j}^{k_1} \le 1 \quad \forall k_1 \in \mathcal{K}_1, \tag{5.2}$$

$$\sum_{j\in\mathcal{S}} x_{o_1 j}^{k_1} = \sum_{i\in\mathcal{S}} x_{io_1'}^{k_1} \quad \forall k_1 \in \mathcal{K}_1, \tag{5.3}$$

$$\sum_{i:(i,j)\in\mathcal{A}_1} x_{ij}^{k_1} - \sum_{i:(j,i)\in\mathcal{A}_1} x_{ji}^{k_1} = 0 \quad \forall k_1 \in \mathcal{K}_1, j \in \mathcal{S}, \tag{5.4}$$

$$\sum_{i:(i,j)\in\mathcal{A}_1} x_{ij}^{k_1} \le 1 \quad \forall k_1 \in \mathcal{K}_1, j \in \mathcal{S}, \tag{5.5}$$

$$y_i^{k_1} + w_i^{k_1} + t_{ij}^1 + s_i \le y_j^{k_1} + M(1 - x_{ij}^{k_1}) \quad \forall k_1 \in \mathcal{K}_1, (i,j) \in \mathcal{A}_1, \tag{5.6}$$

$$y_{o_1'}^{k_1} \le T_{max} \quad \forall k_1 \in \mathcal{K}_1, \tag{5.7}$$

$$z_i^{k_1} + \sum_{k_2\in\mathcal{K}_2} z_i^{k_2} \le z_j^{k_1} + M(1 - x_{ij}^{k_1}) \quad \forall k_1 \in \mathcal{K}_1, (i,j) \in \mathcal{A}_1, i \ne o_1, \tag{5.8}$$

$$z_j^{k_1} \le M(1 - x_{o_1 j}^{k_1}) \quad \forall k_1 \in \mathcal{K}_1, j \in \mathcal{S}, \tag{5.9}$$

$$0 \le z_i^{k_1} \le Q_1 \quad \forall k_1 \in \mathcal{K}_1, i \in \mathcal{N}_1, \tag{5.10}$$

$$\sum_{j\in\mathcal{C}} x_{o_2 j}^{k_2} \le 1 \quad \forall k_2 \in \mathcal{K}_2, \tag{5.11}$$

$$\sum_{j\in\mathcal{C}} x_{o_2 j}^{k_2} = \sum_{i\in\mathcal{S}} x_{io_2'}^{k_2} \quad \forall k_2 \in \mathcal{K}_2, \tag{5.12}$$

$$\sum_{i:(i,j)\in\mathcal{A}_2} x_{ij}^{k_2} - \sum_{i:(j,i)\in\mathcal{A}_2} x_{ji}^{k_2} = 0 \quad \forall k_2 \in \mathcal{K}_2, j \in \mathcal{C} \cup \mathcal{S}, \tag{5.13}$$

$$\sum_{i:(i,j)\in\mathcal{A}_2} x_{ij}^{k_2} \leq 1 \quad \forall k_2 \in \mathcal{K}_2, j \in \mathcal{S}, \tag{5.14}$$

$$\sum_{k_2\in\mathcal{K}_2}\sum_{i:(i,j)\in\mathcal{A}_2} x_{ij}^{k_2} = 1 \quad \forall k_2 \in \mathcal{K}_2, j \in \mathcal{C}, \tag{5.15}$$

$$y_i^{k_2} + w_i^{k_2} + t_{ij}^2 + s_i \leq y_j^{k_2} + M(1 - x_{ij}^{k_2}) \quad \forall k_2 \in \mathcal{K}_2, (i,j) \in \mathcal{A}_2, \tag{5.16}$$

$$a_i \leq y_i^{k_2} + w_i^{k_2} \leq b_i \quad \forall k_2 \in \mathcal{K}_2, i \in \mathcal{C}, \tag{5.17}$$

$$y_{o_2'}^{k_2} \leq T_{max} \quad \forall k_2 \in \mathcal{K}_2, \tag{5.18}$$

$$z_i^{k_2} + q_i \leq z_j^{k_2} + M(1 - x_{ij}^{k_2}) \quad \forall k_2 \in \mathcal{K}_2, (i,j) \in \mathcal{A}_2, i \notin \{o_2\} \cup \mathcal{S}, \tag{5.19}$$

$$z_j^{k_2} \leq M(1 - \sum_{i\in\{o_2\}\cup\mathcal{S}} x_{ij}^{k_2}) \quad \forall k_2 \in \mathcal{K}_2, j \in \mathcal{C}, \tag{5.20}$$

$$z_{o_2'}^{k_2} = 0 \quad \forall k_2 \in \mathcal{K}_2, \tag{5.21}$$

$$0 \leq z_i^{k_2} \leq Q_2 \quad \forall k_2 \in \mathcal{K}_2, i \in \mathcal{N}_2, \tag{5.22}$$

$$\sum_{k_1\in\mathcal{K}_1}\sum_{i:(i,j)\in\mathcal{A}_1} x_{ij}^{k_1} = \sum_{k_2\in\mathcal{K}_2}\sum_{i:(i,j)\in\mathcal{A}_2} x_{ij}^{k_2} \quad \forall j \in \mathcal{S}, \tag{5.23}$$

$$w_j^{k_1} \geq y_j^{k_2} - y_j^{k_1} + M(\sum_{i:(i,j)\in\mathcal{A}_1} x_{ij}^{k_1} + \sum_{i:(i,j)\in\mathcal{A}_2} x_{ij}^{k_2} - 2) \quad \forall k_1 \in \mathcal{K}_1, k_2 \in \mathcal{K}_2, j \in \mathcal{S}, \tag{5.24}$$

$$w_j^{k_2} \geq y_j^{k_1} - y_j^{k_2} + M(\sum_{i:(i,j)\in\mathcal{A}_1} x_{ij}^{k_1} + \sum_{i:(i,j)\in\mathcal{A}_2} x_{ij}^{k_2} - 2) \quad \forall k_1 \in \mathcal{K}_1, k_2 \in \mathcal{K}_2, j \in \mathcal{S}, \tag{5.25}$$

$$x_{ij}^{k_1} \in \{0,1\} \quad \forall k_1 \in \mathcal{K}_1, (i,j) \in \mathcal{A}_1, \tag{5.26}$$

$$y_i^{k_1}, w_i^{k_1} \geq 0 \quad \forall k_1 \in \mathcal{K}_1, i \in \mathcal{N}_1, \tag{5.27}$$

$$x_{ij}^{k_2} \in \{0,1\} \quad \forall k_2 \in \mathcal{K}_2, (i,j) \in \mathcal{A}_2, \tag{5.28}$$

$$y_i^{k_2}, w_i^{k_2} \geq 0 \quad \forall k_2 \in \mathcal{K}_2, i \in \mathcal{N}_2. \tag{5.29}$$

Objective function (5.1) lexicographically minimizes the number of first-echelon vehicles used, the number of second-echelon vehicles used, and the total travel distance. Constraints (5.2) and (5.3) guarantee that each used first-echelon vehicle starts from and returns to its depot. Constraints (5.4) are the flow conservation constraints for the first-echelon vehicles. Constraints (5.5) ensure that each satellite can be visited by one first-echelon vehicle at most. Constraints (5.6) calculate the arrival time and the waiting time of first-echelon vehicle $k_1 \in \mathcal{K}_1$ at node $i \in \mathcal{N}_1$. Constraints (5.7) guarantee that each first-echelon vehicle returns to its depot before the maximum planning horizon. Constraints (5.8) and (5.9) calculate the load of first-echelon vehicle $k_1 \in \mathcal{K}_1$ before visiting node $j \in \mathcal{N}_1$. Constraints (5.10) ensure that the load of each first-echelon vehicle is nonnegative and never exceeds its capacity. Constraints (5.11) and (5.12) guarantee that each used second-echelon vehicle starts from and returns to its depot. Constraints (5.13) are the flow conservation constraints for the second-echelon vehicles. Constraints (5.14) ensure that each satellite can be visited by one second-echelon vehicle at most. Constraints (5.15) ensure that each customer is visited by exactly one second-echelon vehicle. Constraints (5.16) calculate the arrival time and the waiting time of second-echelon vehicle $k_2 \in \mathcal{K}_2$ at node $i \in \mathcal{N}_2$. Constraints (5.17) ensure that each customer is served within its time window by a second-echelon vehicle. Constraints (5.18) guarantee that each second-echelon vehicle returns to its depot before the maximum planning horizon. Constraints (5.19) and (5.20) calculate the load of second-echelon vehicle $k_2 \in \mathcal{K}_2$ before visiting node $j \in \mathcal{C} \cup \mathcal{S}$. Constraints (5.21) ensure that each second-echelon vehicle is empty when it returns to its depot. Constraints (5.22) ensure that the load of each second-echelon vehicle is nonnegative and never exceeds its capacity. Constraints (5.23) ensure that each satellite visited by a second-echelon vehicle should also be visited by a first-echelon vehicle. Constraints (5.24) and (5.25) respectively calculate the waiting times of first-echelon vehicle $k_1 \in \mathcal{K}_1$ and second-echelon vehicle $k_2 \in \mathcal{K}_2$ if they synchronize at satellite $j \in \mathcal{S}$. Constraints (5.26)-(5.29) impose the domains of the decision variables. Note that notation M in several constraints of the deterministic formulation denotes an arbitrary large constant. Moreover, the values of $y_i^{k_1}$, $w_i^{k_1}$, and $z_i^{k_1}$ are meaningless whenever vehicle $k_1 \in \mathcal{K}_1$ does not visit node $i \in \mathcal{N}_1$. Similarly, the values of $y_i^{k_2}$, $w_i^{k_2}$, and $z_i^{k_2}$ are also meaningless whenever vehicle $k_2 \in \mathcal{K}_2$ does not visit node $i \in \mathcal{N}_2$.

5.2.2 The 2E-MTVRPTWSS Under Uncertainty

In some real-life logistics activities with two-echelon transportation systems, customer demands can be uncertain. For example, owing to the rapid development of e-commerce and express services in urban areas, many logistics companies employ vans and electric bikes to collect packages from online retailers and customers. However, the actual number of packages of some online retailers and customers is uncertain and can be known only after they are visited. Thus, we extend the 2E-MTVRPTWSS introduced in Section 5.2.1 by considering uncertain customer demands and study the 2E-MTVRPTWSS under demand uncertainty. In the problem, we assume that customer i has an uncertain demand $\tilde{q}_i$ for each $i \in \mathcal{C}$. To capture the customer demand uncertainty, a novel uncertainty set is defined based on the vehicle routes in two echelons. To model the problem, we adopt the adjustable robust optimization framework (Ben-Tal et al., 2004) and develop a robust mathematical formulation based on the defined uncertainty set. Before presenting the robust formulation and the uncertainty set, a short discussion is needed.

In the 2E-MTVRPTWSS under demand uncertainty, each customer is actually served by vehicles in both echelons because its goods are first collected by a second-echelon vehicle and finally shipped to the logistics centre by a first-echelon vehicle. In addition, each second-echelon vehicle visits a different subset of customers. Thus, the level of customer demand uncertainty experienced by each second-echelon vehicle is different and may be related to the number of customers it serves. Moreover, it is unlikely that every customer on a second-echelon trip to a satellite will have large demand variations. It is more likely that some customers have substantial demand variations while others do not. Thus, we can control the robustness of a second-echelon route by restricting the number of customers with large demand variations on every trip to a satellite on it.

Similar observations can be made for first-echelon vehicles and routes. Based on the above discussion and the concept of budget uncertainty sets proposed in Bertsimas and Sim (2003, 2004), we define the customer demand uncertainty set $\mathcal{U}_q$ in the following equations:

$$\mathcal{U}_q = \left(\bigcap_{k_1 \in \mathcal{K}_1'} \mathcal{U}_q^{k_1} \right) \bigcap \left(\bigcap_{k_2 \in \mathcal{K}_2'} \mathcal{U}_q^{k_2} \right) \tag{5.30}$$

with

$$\mathcal{U}_q^{k_1} = \left\{ \tilde{q} \in \mathbb{R}^{|\mathcal{C}|} | \tilde{q}_i = \overline{q}_i + \alpha_i^{k_1} \hat{q}_i, |\alpha_i^{k_1}| \leq 1, \forall i \in \mathcal{C}; \sum_{i \in \mathcal{C}^{k_1}} |\alpha_i^{k_1}| \leq \Gamma^{k_1}, \Gamma^{k_1} = \lceil \theta_1 |\mathcal{C}^{k_1}| \rceil \right\} \tag{5.31}$$

and

$$\mathcal{U}_q^{k_2} = \left\{ \tilde{q} \in \mathbb{R}^{|\mathcal{C}|} | \tilde{q}_i = \overline{q}_i + \alpha_i^{k_2} \hat{q}_i, |\alpha_i^{k_2}| \leq 1, \forall i \in \mathcal{C}; \right.$$
$$\left. \sum_{i \in \mathcal{C}_s^{k_2}} |\alpha_i^{k_2}| \leq \Gamma_s^{k_2}, \Gamma_s^{k_2} = \lceil \theta_2 |\mathcal{C}_s^{k_2}| \rceil, \forall s \in \mathcal{S}^{k_2} \right\}. \tag{5.32}$$

In equation (5.30), the customer demand uncertainty set $\mathcal{U}_q$ is the intersection of polytopes $\mathcal{U}_q^{k_1}$ for all $k_1 \in \mathcal{K}_1'$ and polytopes $\mathcal{U}_q^{k_2}$ for all $k_2 \in \mathcal{K}_2'$. $\mathcal{K}_1' \subseteq \mathcal{K}_1$ and $\mathcal{K}_2' \subseteq \mathcal{K}_2$ respectively denote the sets of used first- and second-echelon vehicles. Uncertainty polytope $\mathcal{U}_q^{k_1}$ in equation (5.31) captures the possible demand uncertainty experienced by first-echelon vehicle $k_1 \in \mathcal{K}_1'$ based on its route r_{k_1}. Similarly, uncertainty polytope $\mathcal{U}_q^{k_2}$ in equation (5.32) captures the possible demand uncertainty experienced by second-echelon vehicle $k_2 \in \mathcal{K}_2'$ based on its route r_{k_2}. In polytope $\mathcal{U}_q^{k_1}$, $\tilde{q} \in \mathbb{R}^{|\mathcal{C}|}$ denotes a vector which subsumes the uncertain demand $\tilde{q}_i$ of customer i for all $i \in \mathcal{C}$. $\tilde{q}_i$ is expressed as $\overline{q}_i + \alpha_i^{k_1} \hat{q}_i$, where $\overline{q}_i$ denotes the nominal value of $\tilde{q}_i$ and $\hat{q}_i$ denotes the maximum deviation of $\tilde{q}_i$ from $\overline{q}_i$. Similar to the auxiliary variable α_i^k used in uncertainty polytopes $\mathcal{U}_q^k$ (3.11) and $\mathcal{U}_p^k$ (4.13), auxiliary variable $\alpha_i^{k_1}$ is also used to help define uncertainty polytope $\mathcal{U}_q^{k_1}$. As $\alpha_i^{k_1}$

takes values in the interval $[-1,1]$, $\tilde{q}_i$ actually takes values in the interval $[\overline{q}_i - \hat{q}_i, \overline{q}_i + \hat{q}_i]$ for each $i \in \mathcal{C}$. However, according to the inequality $\sum_{i \in \mathcal{C}^{k_1}} |\alpha_i^{k_1}| \leq \Gamma^{k_1}$ in polytope $\mathcal{U}_q^{k_1}$, at most Γ^{k_1} auxiliary variables $\alpha_i^{k_1}$ $(i \in \mathcal{C}^{k_1})$ can simultaneously take their maximum values of 1. Γ^{k_1} denotes the uncertainty budget and $\mathcal{C}^{k_1}$ denotes the set of customers served by first-echelon vehicle k_1. $\mathcal{C}^{k_1}$ can be obtained based on the synchronization schedule of first-echelon vehicle k_1 with the second-echelon vehicles. Note that Γ^{k_1} actually restricts the number of customers who are served by first-echelon vehicle k_1 and can simultaneously have their largest possible demands $\overline{q}_i + \hat{q}_i$. Γ^{k_1} equals $\lceil \theta_1 |\mathcal{C}^{k_1}| \rceil$ and $\lceil \theta_1 |\mathcal{C}^{k_1}| \rceil$ denotes the least integer that is greater than or equal to $\theta_1 |\mathcal{C}^{k_1}|$. θ_1 denotes the uncertainty budget coefficient and it can be set to a value between 0 and 1 by decision-makers. In real-life 2E-MTVRPTWSS applications, θ_1 reflects decision-makers' attitudes towards customer demand uncertainty. If decision-makers are seriously concerned about the impact of demand uncertainty on the feasibility of the designed first-echelon routes, θ_1 can be set close to 1. On the contrary, θ_1 can be set close to 0.

Most notations in uncertainty polytope $\mathcal{U}_q^{k_2}$ (5.32) have similar meanings to those introduced in polytope $\mathcal{U}_q^{k_1}$ (5.31). $\tilde{q} \in \mathbb{R}^{|\mathcal{C}|}$ denotes a vector which subsumes the uncertain demand $\tilde{q}_i$ of customer i for all $i \in \mathcal{C}$. The auxiliary variable $\alpha_i^{k_2}$ is associated with the uncertain customer demand $\tilde{q}_i$ and helps to define uncertainty polytope $\mathcal{U}_q^{k_2}$. $\alpha_i^{k_2}$ takes values in the interval $[-1,1]$. In polytope $\mathcal{U}_q^{k_2}$, $\Gamma_s^{k_2}$ denotes the uncertainty budget and it equals $\lceil \theta_2 |\mathcal{C}_s^{k_2}| \rceil$ for every $s \in \mathcal{S}^{k_2}$. $\mathcal{S}^{k_2}$ denotes the set of satellites visited by second-echelon vehicle k_2 based on its route r_{k_2}. $\mathcal{C}_s^{k_2}$ denotes the set of customers who are served by second-echelon vehicle k_2 via satellite $s \in \mathcal{S}^{k_2}$. $\Gamma_s^{k_2}$ actually imposes an upper bound on the number of customers who can simultaneously have their largest possible demands $\overline{q}_i + \hat{q}_i$ on the trip of second-echelon vehicle k_2 to satellite $s \in \mathcal{S}^{k_2}$. θ_2 denotes the uncertainty budget coefficient which can be set to a value in the interval $[0,1]$ by decision-makers. Similar to θ_1 in polytope $\mathcal{U}_q^{k_1}$, θ_2 reflects decision-makers' attitudes towards the impact of customer demand uncertainty on the feasibility of the designed second-echelon routes in practical 2E-MTVRPTWSS applications. θ_2 can be set slightly greater than or equal to θ_1. Note that no demand uncertainty is considered in both polytopes $\mathcal{U}_q^{k_1}$ and $\mathcal{U}_q^{k_2}$ if $\theta_1 = \theta_2 = 0$. Since a first-echelon vehicle can receive the loads from several second-echelon vehicles, Γ^{k_1} should be greater than $\Gamma_s^{k_2}$ if first-echelon vehicle $k_1 \in \mathcal{K}_1'$ synchronizes with second-echelon vehicle $k_2 \in \mathcal{K}_2'$ at satellite $s \in \mathcal{S}$. However, $\Gamma^{k_1} = \lceil \theta_1 |\mathcal{C}^{k_1}| \rceil$ may be less than $\Gamma_s^{k_2} = \lceil \theta_2 |\mathcal{C}_s^{k_2}| \rceil$ if θ_1 is set much smaller than θ_2. Thus, we reset $\Gamma^{k_1} = \max_{s \in \mathcal{S}^{k_1}} \Gamma_s^{k_2}$ in polytope $\mathcal{U}_q^{k_1}$ if $\exists\, s \in \mathcal{S}^{k_1}$ such that $\lceil \theta_1 |\mathcal{C}^{k_1}| \rceil < \lceil \theta_2 |\mathcal{C}_s^{k_2}| \rceil$. $\mathcal{S}^{k_1}$ denotes the set of satellites visited by first-echelon vehicle k_1.

Based on the defined demand uncertainty set $\mathcal{U}_q$ and the concept of adjustable robust optimization (Ben-Tal et al., 2009, 2004), we extend the deterministic mathematical formulation (5.1)-(5.29) to the following two-stage robust formulation:

(R-2E-MTVRPTWSS)

$$\textit{lex-}\min\left(\sum_{k_1\in\mathcal{K}_1}\sum_{j\in\mathcal{S}} x_{o_1 j}^{k_1},\ \sum_{k_2\in\mathcal{K}_2}\sum_{j\in\mathcal{C}} x_{o_2 j}^{k_2},\ \sum_{k_1\in\mathcal{K}_1}\sum_{(i,j)\in\mathcal{A}_1} d_{ij}x_{ij}^{k_1} + \sum_{k_2\in\mathcal{K}_2}\sum_{(i,j)\in\mathcal{A}_2} d_{ij}x_{ij}^{k_2}\right) \tag{5.33}$$

$$\text{s.t.} \quad (5.2)-(5.7),\ (5.11)-(5.18),\ (5.23)-(5.29),$$

$$z_i^{k_1}(\tilde{q}) + \sum_{k_2\in\mathcal{K}_2} z_i^{k_2}(\tilde{q}) \le z_j^{k_1}(\tilde{q}) + M(1-x_{ij}^{k_1}) \quad \forall k_1\in\mathcal{K}_1, (i,j)\in\mathcal{A}_1, i\neq o_1, \tilde{q}\in\mathcal{U}_q, \tag{5.34}$$

$$z_j^{k_1}(\tilde{q}) \le M(1-x_{o_1 j}^{k_1}) \quad \forall k_1\in\mathcal{K}_1, j\in\mathcal{S}, \tilde{q}\in\mathcal{U}_q, \tag{5.35}$$

$$0 \le z_i^{k_1}(\tilde{q}) \le Q_1 \quad \forall k_1\in\mathcal{K}_1, i\in\mathcal{N}_1, \tilde{q}\in\mathcal{U}_q, \tag{5.36}$$

$$z_i^{k_2}(\tilde{q}) + \tilde{q}_i \le z_j^{k_2}(\tilde{q}) + M(1-x_{ij}^{k_2}) \quad \forall k_2\in\mathcal{K}_2, (i,j)\in\mathcal{A}_2, i\notin\{o_2\}\cup\mathcal{S}, \tilde{q}\in\mathcal{U}_q, \tag{5.37}$$

$$z_j^{k_2}(\tilde{q}) \le M(1-\sum_{i\in\{o_2\}\cup\mathcal{S}} x_{ij}^{k_2}) \quad \forall k_2\in\mathcal{K}_2, j\in\mathcal{C}, \tilde{q}\in\mathcal{U}_q, \tag{5.38}$$

$$z_{o_2'}^{k_2}(\tilde{q}) = 0 \quad \forall k_2\in\mathcal{K}_2, \tilde{q}\in\mathcal{U}_q, \tag{5.39}$$

$$0 \le z_i^{k_2}(\tilde{q}) \le Q_2 \quad \forall k_2\in\mathcal{K}_2, i\in\mathcal{N}_2, \tilde{q}\in\mathcal{U}_q. \tag{5.40}$$

In the above two-stage robust formulation, the first-stage decision variables are binary variables $x_{ij}^{k_1}$ and $x_{ij}^{k_2}$ and continuous variables $y_i^{k_1}$, $w_i^{k_1}$, $y_i^{k_2}$, and $w_i^{k_2}$. The first-stage decision variables should be determined before the uncertain demands of all customers become known. The second-stage decision variables are those related to the loads of first- and second-echelon vehicles. Thus, we respectively extend the continuous variables $z_i^{k_1}$ and $z_i^{k_2}$ to $z_i^{k_1}(\tilde{q})$ and $z_i^{k_2}(\tilde{q})$ for every realization of uncertain customer demands $\tilde{q} \in \mathcal{U}_q$. The second-stage decision variables can be determined after the uncertain customer demands of all customers are observed given the determined first-stage decisions. Thus, they are dependent on the demand vector $\tilde{q} \in \mathcal{U}_q$. In addition, the constraints which contain continuous variables $z_i^{k_1}$ and $z_i^{k_2}$ in the deterministic formulation (5.1)-(5.29) are replaced by constraints (5.34)-(5.40) in the robust formulation. Note that the uncertainty set $\mathcal{U}_q$ is dependent on the first-stage decisions because they determine the specific routes and the synchronization schedule for vehicles in both echelons. Moreover, there are an infinite number of second-stage decision variables and constraints because there are an infinite number of possible demand vectors $\tilde{q}$ in uncertainty set $\mathcal{U}_q$. Thus, it is difficult to design exact algorithms to solve the above two-stage robust formulation with large-sized instances.

5.3 A VNS-Based Metaheuristic

The 2E-CVRP is well known to be NP-hard. Being a complex extension of the 2E-CVRP, the 2E-MTVRPTWSS under customer demand uncertainty is more difficult to tackle. As reviewed in Section 2.3, a variety of efficient and effective metaheuristics have been developed for the 2E-CVRP and its complex extensions, such as the LNS (Breunig et al., 2016), the ALNS (Hemmelmayr et al., 2012), and the GRASP with PR (Anderluh et al., 2016). To solve the 2E-MTVRPTWSS under demand uncertainty with large-sized instances, we design a metaheuristic approach based on the classical VNS framework (Mladenović and Hansen, 1997).

5.3.1 Overview of the Metaheuristic

This section gives an overview of the designed VNS-based metaheuristic. To describe the metaheuristic with simplicity and clarity, we refer to $(\mathcal{R}_1, \mathcal{R}_2)$ as a complete solution to the 2E-MTVRPTWSS under demand uncertainty. $\mathcal{R}_1$ denotes a set of first-echelon routes and $\mathcal{R}_2$ denotes a set of second-echelon routes. In addition, we use $|\mathcal{R}_1|$ and $|\mathcal{R}_2|$ to respectively denote the number of first-echelon routes $\mathcal{R}_1$ and the number of second-echelon routes $\mathcal{R}_2$. As shown in Algorithm 5.1, the metaheuristic consists of three main functions and can thus be divided into three phases. In the first phase, function InitialSolutionConstruction() generates a feasible initial solution $(\mathcal{R}_1^0, \mathcal{R}_2^0)$ for the 2E-MTVRPTWSS under demand uncertainty. The details of function InitialSolutionConstruction() are described in Section 5.3.2. In the second phase, function RouteMinimization() minimizes the number of vehicle routes in two echelons based on the initial solution $(\mathcal{R}_1^0, \mathcal{R}_2^0)$ and returns an intermediate solution $(\mathcal{R}_1', \mathcal{R}_2')$. A thorough introduction of function RouteMinimization() is provided in Section 5.3.3. Given the solution $(\mathcal{R}_1', \mathcal{R}_2')$ obtained in the second phase, the total travel distance of the solution is minimized by function DistanceMinimization() in the third phase. Function DistanceMinimization() is introduced with details in Section 5.3.4. Finally, the VNS-based metaheuristic returns the best found solution $(\mathcal{R}_1, \mathcal{R}_2)$.

Algorithm 5.1 A VNS-based metaheuristic for the 2E-MTVRPTWSS under uncertainty.

```
1: $(\mathcal{R}_1^0, \mathcal{R}_2^0) \leftarrow$ InitialSolutionConstruction();
2: $(\mathcal{R}_1', \mathcal{R}_2') \leftarrow$ RouteMinimization($\mathcal{R}_1^0, \mathcal{R}_2^0$);
3: $(\mathcal{R}_1, \mathcal{R}_2) \leftarrow$ DistanceMinimization($\mathcal{R}_1', \mathcal{R}_2'$);
```

The classical VNS framework is adopted in all three phases of the metaheuristic. Mladenović and Hansen (1997) introduced the VNS framework, which contains four major components: (1) an initial solution construction component; (2) a shaking component with a set of perturbation neighbourhood structures; (3) a local search component with a set of improvement neighbourhood structures; (4) a solution acceptance component based on specific acceptance criteria. The performance of the VNS framework depends strongly on these components. For example, effective improvement neighbourhood structures in the local search component can improve solution quality and reduce computational efforts. In addition, powerful perturbation neighbourhood structures in the shaking component can help to explore larger search space without increasing too much computational complexity. Since the VNS framework is employed in different functions in all three phases of the metaheuristic, the details of these four components may be different when the framework is used. However, the shaking and local search components only adopt neighbourhood structures from sets $\mathcal{N}_{shake}$ and $\mathcal{N}_{search}$ whenever the VNS framework is employed. Next, we briefly introduce the set of shaking neighbourhood structures $\mathcal{N}_{shake}$ and the set of improvement neighbourhood structures $\mathcal{N}_{search}$ considered in the VNS-based metaheuristic.

We consider a set of 25 neighbourhood structures $\mathcal{N}_{shake}$ for the shaking component of the VNS framework in the VNS-based metaheuristic. They are generated based on the cyclic exchange operator introduced in Thompson and Psaraftis (1993). This operator moves node sequences among vehicle routes in a cyclic way. The cyclic exchange operator is also employed in the AVNS-based metaheuristic proposed for the VRPTW under uncertainty in Section 3.3. Two critical parameters need to be specified before applying the cyclic exchange operator: the number of routes Ψ_R to be perturbed and the maximum number of nodes Ψ_N to be exchanged. Note that set $\mathcal{N}_{shake}$ is an ordered set. The details of the 25 shaking neighbourhood structures in set $\mathcal{N}_{shake}$ are shown in Table 5.1.

Table 5.1. The shaking neighbourhood structures adopted in the VNS-based metaheuristic.

NO.	Ψ_R	Ψ_N	NO.	Ψ_R	Ψ_N	NO.	Ψ_R	Ψ_N	NO.	Ψ_R	Ψ_N	NO.	Ψ_R	Ψ_N
1	2	1	6	2	6	11	3	1	16	3	6	21	4	1
2	2	2	7	2	7	12	3	2	17	3	7	22	4	2
3	2	3	8	2	8	13	3	3	18	3	8	23	4	3
4	2	4	9	2	9	14	3	4	19	3	9	24	4	4
5	2	5	10	2	10	15	3	5	20	3	10	25	4	5

From Table 5.1, we observe that two to four routes can be perturbed simultaneously using these neighbourhood structures. Moreover, some of them can exchange up to 10 nodes among the routes involved in the shaking process. Given a shaking neighbourhood structure, we first randomly select Ψ_R routes from the solution which needs to be perturbed. Then, the actual length of the node sequence to be exchanged on each selected route is randomly generated in the interval $[0, \Psi_N]$ if $\Psi_N \leq 5$. If $\Psi_N > 5$, the sequence length is assumed to be fixed and equals Ψ_N. Note that the first- and second-echelon routes in a solution are perturbed independently in different functions due to the design of the VNS-based metaheuristic. In addition, empty routes are not allowed in the shaking process. Moreover, we adjust Ψ_R if it is greater than the number of existing first- or second-echelon routes in the solution to be perturbed. Similarly, we reduce Ψ_N if it is more than the number of the exchangeable nodes on a selected route. Specially, if there is only one first- or second-echelon route in a solution, we perturb the route by randomly permuting all the nodes on it except the starting and ending depots.

We consider a set of nine route improvement neighbourhood structures $\mathcal{N}_{search}$ for the local search component of the VNS framework in the VNS-based metaheuristic. Six of them are designed based on the popular intra-route and inter-route operators in the VRP literature and they will be used in the local search procedures for both first- and second-echelon routes. In addition, three new ones are designed based on several satellite movements on second-echelon routes. Thus, they will be used only in the local search procedure for second-echelon routes. We describe the details of the considered route improvement neighbourhood structures as follows.

- **Intra-Route-Reinsert**. For nodes i and j on the same route r, node i is first removed from route r and then immediately inserted after node j.
- **Intra-Route-Swap**. For nodes i and j on the same route r, the positions of nodes i and j are swapped.
- **Intra-Route-2opt**. For nodes i and j on the same route r, the node sequence from node i to node j is reversed.

- **Inter-Route-Reinsert**. For node i on route r_1 and node j on route r_2, node i is first removed from route r_1 and then immediately inserted after node j on route r_2.
- **Inter-Route-Swap**. For node i on route r_1 and node j on route r_2, the positions of nodes i and j are swapped.
- **Inter-Route-2opt**. For node i on route r_1 and node j on route r_2, route r_1 is first split into two sequences r_1^1 and r_1^2 from node i. Route r_2 is also split into two sequences r_2^1 and r_2^2 from node j. Then, sequence r_1^1 reconnects with sequence r_2^2 and sequence r_2^1 reconnects with sequence r_1^2.
- **Satellite-Reposition**. For satellite s on second-echelon route r (s is not the last satellite), it is first moved to a position between its previous and following non-customer nodes and then replaced by satellite s^* if $s^* \in \mathcal{S}$ brings the least travel distance of route r. $\mathcal{S}$ denotes the set of different satellites in the problem.
- **Satellite-Remove**. For satellite s on second-echelon route r (s is not the last satellite), it is removed from route r.
- **Satellite-Replace**. For satellite s on second-echelon route r, it is replaced by a different satellite $s' \in \mathcal{S}$.

5.3.2 Constructing an Initial Solution

In the first phase of the VNS-based metaheuristic, a feasible initial solution $(\mathcal{R}_1, \mathcal{R}_2)$ is generated for the 2E-MTVRPTWSS under demand uncertainty by function InitialSolutionConstruction(). We show function in pseudocode in Algorithm 5.2. It can be observed that the initial solution construction process contains two major steps. In the first step, a set of feasible second-echelon routes $\mathcal{R}_2'$ is generated by function CSER() (line 4). In the second step, a set of feasible first-echelon routes $\mathcal{R}_1'$ is generated by function CFER() given the obtained second-echelon routes $\mathcal{R}_2'$ (line 5). The details of functions CSER()

and CFER() are described in Sections 5.3.2.1 and 5.3.2.2, respectively. Ω is an ordered set of weight combinations $(\omega_d, \omega_u, \omega_w)$, which are used in function CSER() to generate different sets of feasible second-echelon routes. Since there are $|\Omega|$ weight combinations, we generate $|\Omega|$ initial solutions and the best initial solution $(\mathcal{R}_1, \mathcal{R}_2)$ is returned based on the hierarchical objective in equation (5.33) (lines 6-14). Note that $Distance(\mathcal{R}_1, \mathcal{R}_2)$ denotes the total travel distance of a complete solution $(\mathcal{R}_1, \mathcal{R}_2)$. In addition, the number of first- and second-echelon vehicles are assumed to be sufficient when implementing the whole VNS-based metaheuristic.

Algorithm 5.2 Constructing an initial solution for the 2E-MTVRPTWSS under uncertainty.

```
Function InitialSolutionConstruction($\Omega$)
1: $\mathcal{R}_1 \leftarrow \emptyset$; $\mathcal{R}_2 \leftarrow \emptyset$; $i \leftarrow 1$;
2: while $i \leq |\Omega|$ do
3:     $(\omega_d, \omega_u, \omega_w) \leftarrow$ the $i$th weight combination in set $\Omega$;
4:     $\mathcal{R}'_2 \leftarrow$ CSER($\omega_d, \omega_u, \omega_w$);
5:     $\mathcal{R}'_1 \leftarrow$ CFER($\mathcal{R}'_2$);
6:     if $|\mathcal{R}'_1| < |\mathcal{R}_1|$ or $i = 1$ then
7:         $(\mathcal{R}_1, \mathcal{R}_2) \leftarrow (\mathcal{R}'_1, \mathcal{R}'_2)$;
8:     else if $|\mathcal{R}'_1| = |\mathcal{R}_1|$ then
9:         if $|\mathcal{R}'_2| < |\mathcal{R}_2|$ then
10:            $(\mathcal{R}_1, \mathcal{R}_2) \leftarrow (\mathcal{R}'_1, \mathcal{R}'_2)$;
11:        else if $|\mathcal{R}'_2| = |\mathcal{R}_2|$ and $Distance(\mathcal{R}'_1, \mathcal{R}'_2) < Distance(\mathcal{R}_1, \mathcal{R}_2)$ then
12:            $(\mathcal{R}_1, \mathcal{R}_2) \leftarrow (\mathcal{R}'_1, \mathcal{R}'_2)$;
13:        end if
14:    end if
15:    $i \leftarrow i + 1$;
16: end while
17: return $(\mathcal{R}_1, \mathcal{R}_2)$;
```

5.3.2.1 Constructing a Set of Second-Echelon Vehicle Routes

As shown in Algorithm 5.2, Function CSER() constructs a set of feasible second-echelon routes. Before describing the details of Function CSER(), we first discuss how to check the vehicle capacity constraints associated with a second-echelon route under customer demand uncertainty. Consider a set of second-echelon routes $\mathcal{R}_2$. Let $r_{k_2} \in \mathcal{R}_2$ be the route of second-echelon vehicle k_2. As discussed in Section 5.2.2, customer demands are assumed to be uncertain and take values in the uncertainty set $\mathcal{U}_q$ (5.30) defined with polytopes $\mathcal{U}_q^{k_1}$ (5.31) and $\mathcal{U}_q^{k_2}$ (5.32). Thus, it is not trivial to check the vehicle capacity constraints associated with the second-echelon route r_{k_2} based on the defined uncertainty set and polytopes. However, since the demand uncertainty polytope $\mathcal{U}_q^{k_2}$ is defined based on route r_{k_2} for each $r_{k_2} \in \mathcal{R}_2$, we can calculate the largest possible load of vehicle k_2 at every satellite on route r_{k_2} to help checking the vehicle capacity constraints. In uncertainty polytope $\mathcal{U}_q^{k_2}$, the uncertain demand $\tilde{q}_i$ of customer i takes values in the interval $[\overline{q}_i - \hat{q}_i, \overline{q}_i + \hat{q}_i]$ for each $i \in \mathcal{C}$. However, due to the restriction of the uncertainty budget $\Gamma_s^{k_2}$ in polytope $\mathcal{U}_q^{k_2}$, there are at most $\Gamma_s^{k_2}$ customers who can simultaneously have the largest demands $\overline{q}_i + \hat{q}_i$ in set $\mathcal{C}_s^{k_2}$. $\mathcal{C}_s^{k_2}$ denotes the set of customers who are served by vehicle k_2 via satellite s on route r_{k_2}. Let $Z^{k_2}(s, \Gamma_s^{k_2})$ denote the largest load of second-echelon vehicle k_2 at satellite s on route r_{k_2}. To calculate $Z^{k_2}(s, \Gamma_s^{k_2})$, we first sort the customers in set $\mathcal{C}_s^{k_2}$ in descending order based on their maximum demand deviations $\hat{q}_i$. Then, we select the first $\Gamma_s^{k_2}$ customers to form a new set $\mathcal{C}_s^{k_2}(\Gamma_s^{k_2})$. Based on sets $\mathcal{C}_s^{k_2}$ and $\mathcal{C}_s^{k_2}(\Gamma_s^{k_2})$, we can calculate $Z^{k_2}(s, \Gamma_s^{k_2})$ using equation (5.41). To examine the vehicle capacity constraints with second-echelon route r_{k_2}, we only need to compare the largest load $Z^{k_2}(s, \Gamma_s^{k_2})$ of vehicle k_2 at every satellite s on route r_{k_2} with its capacity Q_2.

$$Z^{k_2}(s, \Gamma_s^{k_2}) = \sum_{i \in \mathcal{C}_s^{k_2}} \overline{q}_i + \sum_{i \in \mathcal{C}_s^{k_2}(\Gamma_s^{k_2})} \hat{q}_i. \tag{5.41}$$

However, calculating the largest vehicle load $Z^{k_2}(s, \Gamma_s^{k_2})$ at satellite s on route r_{k_2} based on equation (5.41) can be computational expensive due to the sorting process for the customers in set $\mathcal{C}_s^{k_2}$. To reduce the computational efforts, we overcalculate $Z^{k_2}(s, \Gamma_s^{k_2})$ if the vehicle capacity constraint associated with satellite s on route r_{k_2} is violated even without considering demand uncertainty. Specifically, if the total load of second-echelon vehicle k_2 at satellite s is larger than its capacity Q_2 considering only the nominal values $\overline{q}_i$

of uncertain customer demands $\tilde{q}_i$ for all $i \in \mathcal{C}_s^{k_2}$, we ignore the restriction of the uncertainty budget $\Gamma_s^{k_2}$ in polytope $\mathcal{U}_q^{k_2}$ and overcalculate $Z^{k_2}(s, \Gamma_s^{k_2})$ under the assumption $\tilde{q}_i = \overline{q}_i + \hat{q}_i$ for all $i \in \mathcal{C}_s^{k_2}$.

Algorithm 5.3 Constructing a set of second-echelon routes.

```
Function CSER($(\omega_d, \omega_u, \omega_w)$)
 1: Initialize $\mathcal{CL}$;
 2: $\mathcal{R}_2 \leftarrow \emptyset$; $k_2 \leftarrow 1$;
 3: while $\mathcal{CL} \neq \emptyset$ do
 4:     $r_{k_2} \leftarrow \{o_2\}$;
 5:     Choose seed customer $i \in \mathcal{CL}$ with the smallest lower bound of its time window;
 6:     $r_{k_2} \leftarrow r_{k_2} \cup \{i\}$; $\mathcal{CL} \leftarrow \mathcal{CL} - \{i\}$;
 7:     while $\mathcal{CL} \neq \emptyset$ do
 8:         for Customer $i \in \mathcal{CL}$ do
 9:             if Customer $i$ can be added to route $r_{k_2}$ without casuing infeasibility then
10:                 Evaluate the fitness value $fitness(i)$ of customer $i$;
11:             end if
12:         end for
13:         if No customer can be added to route $r_{k_2}$ due to temporal constraints then
14:             break;
15:         end if
16:         if No customer can be added to route $r_{k_2}$ due to capacity constraints then
17:             Find the nearest feasible satellite $s^* \in \mathcal{S}$
18:             $r_{k_2} \leftarrow r_{k_2} \cup \{s^*\}$;
19:             continue;
20:         end if
21:         Find customer $i \in \mathcal{CL}$ who has the smallest fitness value;
22:         $r_{k_2} \leftarrow r_{k_2} \cup \{i\}$; $\mathcal{CL} \leftarrow \mathcal{CL} - \{i\}$;
23:     end while
24:     if The last node on route $r_{k_2}$ is not a satellite then
25:         Find the best feasible satellite $s^* \in \mathcal{S}$;
26:         $r_{k_2} \leftarrow r_{k_2} \cup \{s^*\}$;
27:     end if
28:     $r_{k_2} \leftarrow r_{k_2} \cup \{o_2'\}$; $\mathcal{R}_2 \leftarrow \mathcal{R}_2 \cup \{r_{k_2}\}$;
29:     $k_2 \leftarrow k_2 + 1$;
30: end while
31: return $\mathcal{R}_2$;
```

Next, we show function CSER() in pseudocode in Algorithm 5.3. In the beginning, we initialize a candidate list ($\mathcal{CL}$) with all customers in set $\mathcal{C}$ (line 1) and set the second-echelon vehicle counter k_2 to 1 (line 2). A seed route r_{k_2} is created by assigning the second-echelon vehicle depot o_2 to it (line 4). A seed customer $i \in \mathcal{CL}$ with the smallest lower bound of its time window (earliest start-of-service time) is also inserted into the partial route (lines 5-6). Next, customers in $\mathcal{CL}$ who will not cause route infeasibility are sequentially added to the end of route r_{k_2} (lines 7-23) based on an insertion criterion $fitness$. This criterion was first proposed in Pang (2011) and the fitness value of an unrouted customer i can be expressed as $fitness(i) = \omega_d d_{ji} + \omega_u u_{ji} + \omega_w w_{ji}$. The criterion considers three factors d_{ji}, u_{ji}, and w_{ji}. Specifically, d_{ji} denotes the distance between the last visited node j on the partial route r_{k_2} and the unrouted customer i. u_{ji} denotes the time urgency of vehicle k_2 to serve the unrouted customer i after the last visited node j. w_{ji} denotes the waiting time of vehicle k_2 to serve the unrouted customer i after the last visited node j. The corresponding weights $(\omega_d, \omega_u, \omega_w)$ for the three factors are nonnegative and $\omega_d + \omega_u + \omega_w = 1$. As introduced in Algorithm 5.2, set Ω contains a number of weight combinations $(\omega_d, \omega_u, \omega_w)$. Specifically, we consider a total of 66 weight combinations in set Ω, which includes $\{(1,0,0),(0.9,0.1,0),(0.8,0.2,0),\cdots,(0,0.2,0.8),(0,0.1,0.9),(0,0,1)\}$. If no customer in $\mathcal{CL}$ can be added to the end of route r_{k_2} due to the temporal (time window or maximum planning horizon) constraints (lines 13-15), we use a new second-echelon vehicle. If no customer in $\mathcal{CL}$ can be added to the end of route r_{k_2} due to the vehicle capacity constraints, the nearest feasible satellite $s^* \in \mathcal{S}$ to the last customer on route r_{k_2} is added to the end of the route (lines 16-20). $\mathcal{S}$ denotes the set of different satellites in the problem. Herein, a satellite $s \in \mathcal{S}$ is called feasible if the second-echelon vehicle k_2 can successfully synchronize with a first-echelon vehicle which just departs from its depot at satellite s and both vehicles can return to their depots within the maximum planning horizon after synchronization. Since each second-echelon vehicle must visit a satellite before returning to its depot, we add the best feasible satellite s^* to the end of route r_{k_2} if the last node on route r_{k_2} is not a satellite (lines 24-27). s^* is the satellite which incurs the smallest additional travel distance among all feasible satellites. Note that a customer $i \in \mathcal{CL}$ also can not be added to route r_{k_2} if no feasible satellites can be added to the end of the new route after inserting the customer. The route construction procedure terminates when all customers are routed.

Based on the above discussion, it can be found that inserting a feasible satellite into a second-echelon route is not an easy task. When inserting a satellite into a second-echelon route, we need to guarantee that the corresponding second-echelon vehicle can synchronize with a first-echelon vehicle at the satellite and both vehicles can return to their depots before the maximum planning horizon after synchronization. Therefore, if a second-echelon route is called feasible, every satellite on the route has to meet two extra conditions. Firstly, there must exist a first-echelon vehicle which can successfully synchronize with the second-echelon vehicle at the satellite without causing any temporal constraint violations associated with the second-echelon route. Secondly, the first-echelon vehicle can return to its depot before the maximum planning horizon after synchronizing with the second-echelon vehicle at the satellite. Note that these two extra conditions are considered when checking the feasibility of a given second-echelon route in other phases of the VNS-based metaheuristic.

5.3.2.2 Constructing a Set of First-Echelon Vehicle Routes

As shown in Algorithm 5.2, function CFER() generates a set of first-echelon vehicle routes. Before describing its details, we first introduce the idea of how to construct a set of first-echelon routes based on a set of feasible second-echelon routes without considering customer demand uncertainty. Figure 5.2 depicts the routes of two second-echelon vehicles. The squares represent the depot of the second-echelon vehicles. The circles and triangles respectively denote the customers and the satellites on these two routes. The number on each arc represents the time of a second-echelon vehicle travelling across that arc. The numbers in the bracket above a circle denote the time window of a customer and the number below the circle denotes the demand of the customer. The capacity of each second-echelon vehicle is set to a value of 50 and the maximum planning horizon (route duration) is set to a value of 32. For simplicity, we assume that the service times at all satellites and customers are zero. Since a second-echelon vehicle needs to synchronize with a first-echelon vehicle at each satellite on its route, we assume that travel times of a first-echelon vehicle from its depot to satellites 1, 2, and 3 are 10, 5, and 8, respectively. Moreover, the travel times of a first-echelon vehicle from these satellites to its depot are assumed to be the same as those from its depot to the corresponding satellites.

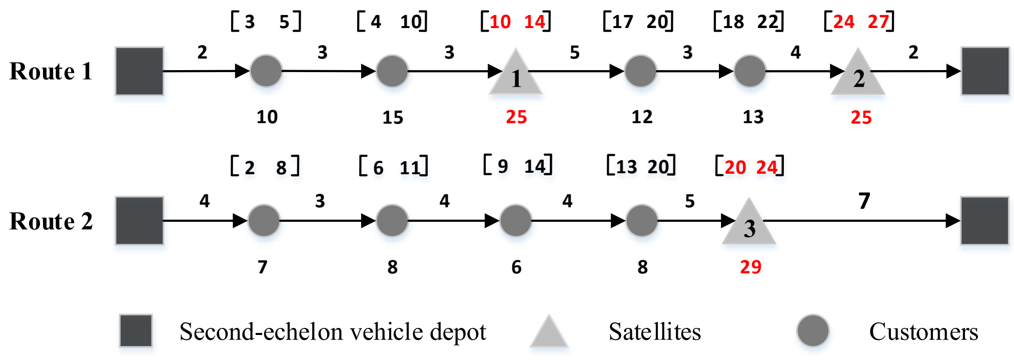


Figure 5.2 An example of two second-echelon vehicle routes.

Look at the second-echelon route 1 in Figure 5.2. If the corresponding second-echelon vehicle leaves its depot at time 0 and synchronizes with two first-echelon vehicles which just depart from their depot at satellites 1 and 2, its earliest synchronization times at satellites 1 and 2 can be calculated as 10 and 24, respectively. Since vehicles in both echelons have to return to their depots before the maximum planning horizon, the latest synchronization times of the second-echelon vehicle at satellites 1 and 2 also can be respectively calculated as 14 and 27. Moreover, the loads of the vehicle at satellites 1 and 2 are both 25. Similarly, we can calculate the load and the earliest and latest synchronization times of another second-echelon vehicle at satellite 3 on route 2, as shown in Figure 5.2. If we find a set of first-echelon routes such that the corresponding first-echelon vehicles can synchronize with the two second-echelon vehicles at satellites 1, 2, and 3 within the time windows [10 14], [24 27], and [20, 24] without violating the vehicle capacity and duration constraints, this set of first-echelon vehicle routes are considered as feasible. Thus, generating a set of first-echelon routes based on a set of feasible second-echelon routes without considering demand uncertainty can be seen as the VRPTW. Specifically, the calculated load and the calculated earliest and latest synchronization times of a second-echelon vehicle at a satellite can be respectively considered as the demand and the time window of the satellite which needs to be visited by a first-echelon vehicle. As discussed in Section 2.2, the VRPTW is an NP-hard problem. However, the number of synchronization operations at satellites is relatively small compared to the number of customers in the 2E-MTVRPTWSS even for large-sized instances. Thus, we can adopt the fast heuristics developed for the VRPTW in the literature to efficiently generate a set of first-echelon routes based on a set of feasible second-echelon routes without considering uncertainty.

However, uncertainty in customer demands makes it difficult to implement the above introduced idea to generate a set of first-echelon routes. The most challenging part is how to check the vehicle capacity constraint with a given first-echelon route based on the corresponding set of second-echelon routes and the defined uncertainty set and polytopes. To address this issue, we can also calculate the largest possible load of a first-echelon vehicle on its route. The detailed procedure to calculate this value is described as follows. Suppose there is a set of feasible second-echelon routes $\mathcal{R}_2$. We first generate a set $\mathcal{S}_{\mathcal{R}_2}$ which contains the satellites on all second-echelon routes $r_{k_2} \in \mathcal{R}_2$. Note that a satellite $s \in \mathcal{S}$ is replicated $n-1$ times if it appears $n > 1$ times on all second-echelon routes. The replications of satellite s are treated as different satellites but have the same location of satellite s and they are included in set $\mathcal{S}_{\mathcal{R}_2}$. As discussed in Section 5.3.2.1, we can obtain a set of customers $\mathcal{C}_s^{k_2}$ who are served by second-echelon vehicle k_2 via every satellite s on route $r_{k_2} \in \mathcal{R}_2$. In addition, we can obtain a set of customers $\mathcal{C}_s^{k_2}(\Gamma_s^{k_2})$ who have the $\Gamma_s^{k_2}$ largest demand deviations $\hat{q}_i$ in set $\mathcal{C}_s^{k_2}$. Thus, we generate a set $\mathcal{C}_s^{k_2}$ and a set $\mathcal{C}_s^{k_2}(\Gamma_s^{k_2})$ for each satellite $s \in \mathcal{S}_{\mathcal{R}_2}$ based on the given set of second-echelon routes $\mathcal{R}_2$. Next, let r_{k_1} be the route of first-echelon vehicle k_1 and r_{k_1} consists of several satellites in set $S_{\mathcal{R}_2}$.

Obviously, we can obtain a set $\mathcal{C}^{k_1}$ of customers who are served by vehicle k_1 based on sets $\mathcal{C}_s^{k_2}$ and $\mathcal{C}^{k_1} = \cup_{s \in r_{k_1}} \mathcal{C}_s^{k_2}$. We also can obtain a set $\mathcal{C}^{k_1\prime}$ which contains all customers in sets $\mathcal{C}_s^{k_2}(\Gamma_s^{k_2})$ for all $s \in r_{k_1}$ and $\mathcal{C}^{k_1\prime} = \cup_{s \in r_{k_1}} \mathcal{C}_s^{k_2}(\Gamma_s^{k_2})$. However, due to the restriction of the uncertainty budget Γ^{k_1} in polytope $\mathcal{U}_q^{k_1}$, at most Γ^{k_1} customers can simultaneously have the largest demands $\overline{q}_i + \hat{q}_i$ in set $\mathcal{C}^{k_1}$. Thus, we again sort the customers in set $\mathcal{C}^{k_1\prime}$ in decreasing order based on their maximum demand deviations $\hat{q}_i$ and select the first Γ^{k_1} customers to form a new set $\mathcal{C}^{k_1}(\Gamma^{k_1})$. Let $Z^{k_1}(\Gamma^{k_1})$ denote the largest load of first-echelon vehicle k_1 on its route r_{k_1}. Based on sets $\mathcal{C}^{k_1}$ and $\mathcal{C}^{k_1}(\Gamma^{k_1})$, $Z^{k_1}(\Gamma^{k_1})$ can be calculated using equation (5.42). To check the vehicle capacity constraint with first-echelon route r_{k_1}, we only need to compare the largest load $Z^{k_1}(\Gamma^{k_1})$ of vehicle k_1 with its capacity Q_1. Note that using equation (5.42) to calculate $Z^{k_1}(\Gamma^{k_1})$ can be computational expensive due to the complex sorting process. To reduce the computational efforts, we overcalculate $Z^{k_1}(\Gamma^{k_1})$ if route r_{k_1} is infeasible with respect to vehicle capacity constraint even without considering demand uncertainty. Specifically, if the load of first-echelon vehicle k_1 on its route r_{k_1} is larger than its capacity Q_1 considering only the nominal values $\overline{q}_i$ of uncertain customer demands $\tilde{q}_i$ for all $i \in \mathcal{C}^{k_1}$, we ignore the restriction of the uncertainty budget Γ^{k_1} and overcalculate $Z^{k_1}(\Gamma^{k_1})$ under the assumption $\tilde{q}_i = \overline{q}_i + \hat{q}_i$ for all $i \in \mathcal{C}^{k_1}$.

$$Z^{k_1}(\Gamma^{k_1}) = \sum_{i \in \mathcal{C}^{k_1}} \overline{q}_i + \sum_{i \in \mathcal{C}^{k_1}(\Gamma^{k_1})} \hat{q}_i. \tag{5.42}$$

Next, we present function CFER() in pseudocode in Algorithm 5.4. Given a set of feasible second-echelon routes $\mathcal{R}_2$, we first generate a set $\mathcal{S}_{\mathcal{R}_2}$ and an information matrix $\mathcal{I}_{\mathcal{R}_2}$ associated with $\mathcal{S}_{\mathcal{R}_2}$ using function IFER() (line 1). Set $\mathcal{S}_{\mathcal{R}_2}$ contains the satellites on all second-echelon routes $r_{k_2} \in \mathcal{R}_2$. Note that a satellite $s \in \mathcal{S}$ is replicated $n-1$ times if it appears $n > 1$ times on all second-echelon routes. The replications of satellite s are treated as different satellites but have the same location of satellite s and they are included in set $\mathcal{S}_{\mathcal{R}_2}$. To obtain the information matrix $\mathcal{I}_{\mathcal{R}_2}$, we first generate a set $\mathcal{C}_s^{k_2}$ for each

Algorithm 5.4 Constructing a set of first-echelon routes.

```
Function CFER($\mathcal{R}_2$)
1:  ($\mathcal{S}_{\mathcal{R}_2}$,$\mathcal{I}_{\mathcal{R}_2}$) $\leftarrow$ IFER($\mathcal{R}_2$);
2:  $\mathcal{R}_1 \leftarrow$ NNHFER($\mathcal{S}_{\mathcal{R}_2}$,$\mathcal{I}_{\mathcal{R}_2}$);
3:  $\mathcal{R}_1 \leftarrow$ LSMTDFER($\mathcal{R}_1$,$\mathcal{S}_{\mathcal{R}_2}$,$\mathcal{I}_{\mathcal{R}_2}$);
4:  while $|\mathcal{R}_1| > V_1^{min}$ and $\mathcal{R}_1$ is feasible do
5:      $\mathcal{R}_1' \leftarrow$ CFERFNV($\mathcal{R}_2$, $|\mathcal{R}_1| - 1$);
6:      if $\mathcal{R}_1'$ is feasible then
7:          $\mathcal{R}_1 \leftarrow \mathcal{R}_1'$;
8:      else
9:          break;
10:     end if
11: end while
12: return $\mathcal{R}_1$ and ($\mathcal{S}_{\mathcal{R}_2}$,$\mathcal{I}_{\mathcal{R}_2}$);
```

$s \in \mathcal{S}_{\mathcal{R}_2}$ in function IFER(). $\mathcal{C}_s^{k_2}$ consists of customers who are served by a second-echelon vehicle k_2 via satellite s. Then, we calculate the earliest and latest synchronization times of a second-echelon vehicle at satellite s for each satellite $s \in \mathcal{S}_{\mathcal{R}_2}$. Finally, we store all these information associated with satellite s in the information matrix $\mathcal{I}_{\mathcal{R}_2}$ for all $s \in \mathcal{S}_{\mathcal{R}_2}$. Next, we generate a set of first-echelon routes $\mathcal{R}_1$ based on $\mathcal{S}_{\mathcal{R}_2}$ and $\mathcal{I}_{\mathcal{R}_2}$ using function NNHFER() (line 2). In function NNHFER(), the nearest neighbour heuristic is employed to route all satellites in $\mathcal{S}_{\mathcal{R}_2}$. The set of first-echelon routes $\mathcal{R}_1$ is then improved by function LSMTDFER() (line 3). Function LSMTDFER() is a local search procedure which adopts the first six neighbourhood structures introduced in Section 5.3.1. In the local search procedure, the first improvement strategy is considered and a neighbourhood structure is reused until the current solution cannot be further improved. In addition, all neighbourhood structures are selected in random order. Moreover, a change tracking method proposed in Benjamin and Beasley (2013) is adopted and slightly extended to reduce the computational complexity of the procedure. When $\mathcal{R}_1$ is feasible and $|\mathcal{R}_1|$ is larger than the minimum required number of first-echelon vehicles $V_1^{min} = \lceil \sum_{i \in \mathcal{C}} \overline{q}_i / Q_1 \rceil$, a new set of first-echelon routes $\mathcal{R}_1'$ is generated using function CFERFNV() (line 5). Function CFERFNV() aims to find a set of feasible first-echelon routes with $|\mathcal{R}_1| - 1$ vehicles given the second-echelon routes $\mathcal{R}_2$. We show function CFERFNV() in pseudocode in Algorithm 5.5. If the resulting first-echelon routes $\mathcal{R}_1'$ from function CFERFNV() are feasible, we update $\mathcal{R}_1$ with $\mathcal{R}_1'$ and the route reduction process (lines 5-10) is repeated. Otherwise, $\mathcal{R}_1$ is returned. Note that set $\mathcal{S}_{\mathcal{R}_2}$ and matrix $\mathcal{I}_{\mathcal{R}_2}$ are also returned because they link the vehicle routes in both echelons. Moreover, we keep and update $\mathcal{S}_{\mathcal{R}_2}$ and $\mathcal{I}_{\mathcal{R}_2}$ in the VNS-based metaheuristic to reduce its computational complexity.

Algorithm 5.5 Constructing a set of first-echelon routes with a fixed number of vehicles.

```
 **Function** CFERFNV($\mathcal{R}_2, V_1$)
 1: ($\mathcal{S}_{\mathcal{R}_2}$,$\mathcal{I}_{\mathcal{R}_2}$) $\leftarrow$ IFER($\mathcal{R}_2$);
 2: $\mathcal{R}_1 \leftarrow$ NNHFERFNV($\mathcal{S}_{\mathcal{R}_2}$,$\mathcal{I}_{\mathcal{R}_2}$,$V_1$);
 3: $\mathcal{R}_1 \leftarrow$ LSMTDFER($\mathcal{R}_1$,$\mathcal{S}_{\mathcal{R}_2}$,$\mathcal{I}_{\mathcal{R}_2}$);
 4: $IterF \leftarrow 0$; $i \leftarrow 1$;
 5: **while** $IterF < MaxIterF$ **do**
 6:     $\mathcal{R}_1' \leftarrow$ SPFER($\mathcal{R}_1$,$\mathcal{N}_{shake}^{(i)}$);
 7:     $\mathcal{R}_1' \leftarrow$ LSMTDFER($\mathcal{R}_1'$,$\mathcal{S}_{\mathcal{R}_2}$,$\mathcal{I}_{\mathcal{R}_2}$);
 8:     **if** $Cost(\mathcal{R}_1') < Cost(\mathcal{R}_1)$ **then**
 9:         $\mathcal{R}_1 \leftarrow \mathcal{R}_1'$;
10:         $IterF \leftarrow 0$; $i \leftarrow 1$;
11:     **else**
12:         $IterF \leftarrow IterF + 1$; $i \leftarrow (i \bmod |\mathcal{N}_{shake}|) + 1$;
13:     **end if**
14: **end while**
15: **return** $\mathcal{R}_1$ and ($\mathcal{S}_{\mathcal{R}_2}$,$\mathcal{I}_{\mathcal{R}_2}$);
```

As shown in Algorithm 5.5, function CFERFNV() aims to generate a set of feasible first-echelon routes with a fixed number of vehicles based on the VNS framework. Initially, we generate a set of satellites $\mathcal{S}_{\mathcal{R}_2}$ and the corresponding information matrix $\mathcal{I}_{\mathcal{R}_2}$ based on the given second-echelon routes $\mathcal{R}_2$ by using function IFER() (line 1). Next, function NNHFERFNV() generates a set of first-echelon routes $\mathcal{R}_1$ with V_1 vehicles (line 2). In function NNHFERFNV(), we first adopt the nearest neighbour heuristic to generate a set of feasible partial first-echelon routes. However, since there are only V_1 first-echelon vehicles, some satellites in set $\mathcal{S}_{\mathcal{R}_2}$ may not be routed without violating the vehicle capacity and time window constraints. For these satellites, we sequentially inset them at the best position in the partial routes, which yields the smallest additional insertion costs. Since the resulting first-echelon routes $\mathcal{R}_1$ can be infeasible, we use a cost function $Cost(\mathcal{R}_1) = Distance(\mathcal{R}_1) + Penalty(\mathcal{R}_1)$ to evaluate a given set of first-echelon routes $\mathcal{R}_1$. $Distance(\mathcal{R}_1)$ denotes the total travel distance of the first-echelon routes $\mathcal{R}_1$. $Penalty(\mathcal{R}_1)$ denotes the penalty cost for the possible constraint violations associated with the first-echelon routes $\mathcal{R}_1$ and $Penalty(\mathcal{R}_1) = \rho_1(V_1^{cap}(\mathcal{R}_1) + V_1^{tw}(\mathcal{R}_1))$. $V_1^{cap}(\mathcal{R}_1)$ and $V_1^{tw}(\mathcal{R}_1)$ respectively denote the vehicle capacity violations and the time window violations with the first-echelon routes $\mathcal{R}_1$. ρ_1 is the penalty factor. Note that the vehicle capacity violations $V_1^{cap}(\mathcal{R}_1)$ can be determined by calculating the largest load of a first-echelon vehicle k_1 on its route r_{k_1} for each $r_{k_1} \in \mathcal{R}_1$ considering demand uncertainty. In function CFERFNV(), functions LSMTDFER() and SPFER() are employed in the VNS framework to help finding a set of feasible first-echelon routes (lines 3-14). Function SPFER() is a perturbation (shaking) procedure. It adopts the 25 neighbourhood structures in set $\mathcal{N}_{shake}$ which are shown in Table 5.1. Note that $\mathcal{N}_{shake}^{(i)}$ denotes the ith neighbourhood structure in set $\mathcal{N}_{shake}$.

5.3.3 Minimizing the Number of Vehicle Routes in Two Echelons

After obtaining an initial solution in the first phase, we minimize the number of first- and second-echelon routes in the initial solution using function RouteMinimization() in the second phase. Function RouteMinimization() is shown in pseudocode in Algorithm 5.6.

It can be observed that function RouteMinimization() is an iterative procedure which contains two main steps in each iteration. In the first main step, we reduce the number of first-echelon routes (lines 3-15). When the number of the current first-echelon routes $|\mathcal{R}_1'|$ is larger than the minimum required number of vehicles V_1^{min}, we generate a new set of first-echelon routes $\mathcal{R}_1'$ with $|\mathcal{R}_1|$ or $|\mathcal{R}_1| - 1$ vehicles using function CFERFNV() based on the corresponding second-echelon routes $\mathcal{R}_2'$ (lines 4-8). As $\mathcal{R}_1'$ is generally infeasible, we use function RFFER() to recover its feasibility (line 9). The details of function RFFER()

Algorithm 5.6 Minimizing the number of vehicle routes in two echelons.

```
Function RouteMinimization($\mathcal{R}_1, \mathcal{R}_2$)
1:  $(\mathcal{R}'_1, \mathcal{R}'_2) \leftarrow (\mathcal{R}_1, \mathcal{R}_2)$;
2:  while $\mathcal{R}'_2$ is feasible do
3:      while $|\mathcal{R}'_1| > V_1^{min}$ do
4:          if $|\mathcal{R}'_1| > |\mathcal{R}_1|$ then
5:              $\mathcal{R}'_1$ and $(\mathcal{S}_{\mathcal{R}'_2}, \mathcal{I}_{\mathcal{R}'_2}) \leftarrow$ CFERFNV($\mathcal{R}'_2, |\mathcal{R}_1|$);
6:          else
7:              $\mathcal{R}'_1$ and $(\mathcal{S}_{\mathcal{R}'_2}, \mathcal{I}_{\mathcal{R}'_2}) \leftarrow$ CFERFNV($\mathcal{R}'_2, |\mathcal{R}_1| - 1$);
8:          end if
9:          $(\mathcal{R}'_1, \mathcal{R}'_2) \leftarrow$ RFFER($\mathcal{R}'_1, \mathcal{R}'_2, \mathcal{S}_{\mathcal{R}'_2}, \mathcal{I}_{\mathcal{R}'_2}$);
10:         if $\mathcal{R}'_1$ is feasible then
11:             $(\mathcal{R}_1, \mathcal{R}_2) \leftarrow (\mathcal{R}'_1, \mathcal{R}'_2)$;
12:         else
13:             break;
14:         end if
15:     end while
16:     $\mathcal{R}'_2 \leftarrow$ ROR($\mathcal{R}'_2$);
17:     $\mathcal{R}'_2 \leftarrow$ RFSER($\mathcal{R}'_2$);
18:     if $\mathcal{R}'_2$ is feasible then
19:         $\mathcal{R}'_1 \leftarrow$ CFER($\mathcal{R}'_2$);
20:         if $|\mathcal{R}'_1| = V_1^{min}$ then
21:             $(\mathcal{R}_1, \mathcal{R}_2) \leftarrow (\mathcal{R}'_1, \mathcal{R}'_2)$;
22:         end if
23:     end if
24: end while
25: return $(\mathcal{R}_1, \mathcal{R}_2)$;
```

are shown in Algorithm 5.7. If $\mathcal{R}'_1$ becomes feasible after the feasibility recovery process, we update the best found solution $(\mathcal{R}_1, \mathcal{R}_2)$ (line 11). Otherwise, we go to the second main step (lines 16-17). In the second main step, we reduce the number of second-echelon routes. Specifically, we first generate a new set of second-echelon routes $\mathcal{R}'_2$ with one less vehicle using function ROR() (line 16). In function ROR(), the second-echelon route with the least number of nodes is removed and the customers on that route are randomly inserted into the remaining routes. In function ROR(), we also try to insert satellites in a second-echelon route if the route becomes infeasible owing to violating the vehicle capacity constraints after inserting a removed customer. Specifically, we insert the satellite $s^* \in \mathcal{S}$

which incurs the least additional travel distance at the position where vehicle capacity violations are detected on the route. Generally, the resulting second-echelon routes $\mathcal{R}'_2$ from function ROR() are infeasible due to the random insertion process of the removed customers. Thus, we try to recover the feasibility of $\mathcal{R}'_2$ using function RFSER() (line 17). The details of function RFSER() are shown in Algorithm 5.9. If $\mathcal{R}'_2$ becomes feasible after the feasibility recovery process, we again generate a new set of first-echelon routes $\mathcal{R}'_1$ using function CFER() (line 19) and the above two main steps are repeated. Otherwise, the route minimization procedure terminates and returns the best found solution $(\mathcal{R}_1, \mathcal{R}_2)$.

Algorithm 5.7 Recovering the feasibility of a set of first-echelon routes.

```
Function RFFER($\mathcal{R}_1$,$\mathcal{R}_2$,$\mathcal{S}_{\mathcal{R}_2}$,$\mathcal{I}_{\mathcal{R}_2}$)
1:  $(\mathcal{R}_1, \mathcal{R}_2) \leftarrow$ LSRFFER($\mathcal{R}_1, \mathcal{R}_2, \mathcal{S}_{\mathcal{R}_2}, \mathcal{I}_{\mathcal{R}_2}$);
2:  if $\mathcal{R}_1$ is infeasible then
3:      $IterR \leftarrow 0$; $i \leftarrow 1$;
4:      while $Penalty(\mathcal{R}_1) > 0$ and $IterR < MaxIterR$ do
5:          do
6:              $\mathcal{R}'_2 \leftarrow$ SPSER($\mathcal{R}_2, \mathcal{N}^{(i)}_{shake}$);
7:              $\mathcal{R}'_2 \leftarrow$ RFSER($\mathcal{R}'_2$);
8:          while $\mathcal{R}'_2$ is infeasible;
9:          $\mathcal{R}'_1$ and $(\mathcal{S}_{\mathcal{R}'_2}, \mathcal{I}_{\mathcal{R}'_2}) \leftarrow$ CFERFNV($\mathcal{R}'_2, |\mathcal{R}_1|$);
10:         $(\mathcal{R}'_1, \mathcal{R}'_2) \leftarrow$ LSRFFER($\mathcal{R}'_1, \mathcal{R}'_2, \mathcal{S}_{\mathcal{R}'_2}, \mathcal{I}_{\mathcal{R}'_2}$);
11:         if $Penalty(\mathcal{R}'_1) < Penalty(\mathcal{R}_1)$ then
12:             $(\mathcal{R}_1, \mathcal{R}_2) \leftarrow (\mathcal{R}'_1, \mathcal{R}'_2)$;
13:             $i \leftarrow 1$;
14:         else
15:             $i \leftarrow (i \bmod |\mathcal{N}_{shake}|) + 1$;
16:         end if
17:         $IterR \leftarrow IterR + 1$;
18:     end while
19: end if
20: return $(\mathcal{R}_1, \mathcal{R}_2)$;
```

In Algorithm 5.7, Function RFFER() aims to recover the feasibility of a set of first-echelon routes $\mathcal{R}_1$ given a set of feasible second-echelon routes $\mathcal{R}_2$, the corresponding satellite set $\mathcal{S}_{\mathcal{R}_2}$, and the information matrix $\mathcal{I}_{\mathcal{R}_2}$. In the beginning, the input solution $(\mathcal{R}_1, \mathcal{R}_2)$ is improved by function LSRFFER() (line 1). Function LSRFFER() is a local search procedure which is used to reduce the penalty costs of a set of first-echelon routes. The detailed procedure of function LSRFFER() is shown in pseudocode in Algorithm 5.8. If $\mathcal{R}_1$ is still infeasible after the local search procedure, the VNS framework is utilized (lines 3-18). In the VNS main loop, a new set of feasible second-echelon routes $\mathcal{R}_2'$ is first generated (lines 5-8). Function SPSER() is a shaking procedure which adopts the neighbourhood structures in set $\mathcal{N}_{shake}$ to perturb a set of second-echelon routes (line 6). Function RFSER() aims to recover the feasibility of the newly-generated second-echelon routes $\mathcal{R}_2'$ (line 7) and it is shown in pseudocode in Algorithm 5.9. Next, we generate a new set of first-echelon routes $\mathcal{R}_1'$ with $|\mathcal{R}_1|$ vehicles based on the new second-echelon routes $\mathcal{R}_2'$ using function CFERFNV() (line 9). The new complete solution $(\mathcal{R}_1', \mathcal{R}_2')$ is immediately improved by function LSRFFER() (line 10). The feasibility recovery procedure terminates when $\mathcal{R}_1$ becomes feasible ($Penalty(\mathcal{R}_1) = 0$) or a maximum number of iterations $MaxIterR$ is reached.

Function LSRFFER() aims to recover the feasibility of a set of first-echelon routes using a local search procedure. As shown in Algorithm 5.8, we first initialize a neighbourhood list ($\mathcal{NL}$) which contains all neighbourhood structures in set $\mathcal{N}_{search}$ introduced in Section 5.3.1 (line 1). Next, we randomly select a neighbourhood structure $\mathcal{N}_{search}^{(j)}$ and perform local search (lines 4-21). In the search process, we examine the neighbours in neighbourhood $\mathcal{N}_{search}^{(j)}(\mathcal{R}_2')$ in a fixed order. If the ith neighbour $\mathcal{R}_2^i$ of the current second-echelon routes $\mathcal{R}_2'$ is feasible, we generate a new set of first-echelon routes $\mathcal{R}_1^i$, a set of satellites $\mathcal{S}_{\mathcal{R}_2^i}$, and an information matrix $\mathcal{I}_{\mathcal{R}_2^i}$ (line 12). In set $\mathcal{N}_{search}$, most neighbourhood structures are very simple and the corresponding neighbourhood moves are only associated with one or two customers (satellites). Thus, we can generate $\mathcal{R}_1^i$ without or slightly modifying $\mathcal{R}_1'$ although $\mathcal{R}_1^i$ may be infeasible. Moreover, we only need to update the information associated with a few satellites to obtain $(\mathcal{S}_{\mathcal{R}_2^i}, \mathcal{I}_{\mathcal{R}_2^i})$ based on $(\mathcal{S}_{\mathcal{R}_2'}, \mathcal{I}_{\mathcal{R}_2'})$. Next, the set of first-echelon routes $\mathcal{R}_1^i$ is immediately improved by function LSMTDFER() (line 13). We update the current complete solution $(\mathcal{R}_1', \mathcal{R}_2')$ if the penalty

Algorithm 5.8 Local search for recovering the feasibility of a set of first-echelon routes.

```
Function LSRFFER($\mathcal{R}_1$,$\mathcal{R}_2$,$\mathcal{S}_{\mathcal{R}_2}$,$\mathcal{I}_{\mathcal{R}_2}$)
1: Initialize neighbourhood list $\mathcal{NL}$;
2: $(\mathcal{R}'_1, \mathcal{R}'_2, \mathcal{S}_{\mathcal{R}'_2}, \mathcal{I}_{\mathcal{R}'_2}) \leftarrow (\mathcal{R}_1, \mathcal{R}_2, \mathcal{S}_{\mathcal{R}_2}, \mathcal{I}_{\mathcal{R}_2})$;
3: while $\mathcal{NL} \neq \emptyset$ and $Penalty(\mathcal{R}_1) > 0$ do
4:     Select a neighbourhood structure $\mathcal{N}^{(j)}_{search} \in \mathcal{NL}$ at random;
5:     $Improve \leftarrow 1$;
6:     while $Penalty(\mathcal{R}'_1) > 0$ and $Improve = 1$ do
7:         $i \leftarrow 0$; $Improve \leftarrow 0$;
8:         while $i < |\mathcal{N}^{(j)}_{search}(\mathcal{R}'_2)|$ do
9:             $i \leftarrow i + 1$;
10:            Find the $i$th neighbour $\mathcal{R}^i_2 \in \mathcal{N}^{(j)}_{search}(\mathcal{R}'_2)$;
11:            if $\mathcal{R}^i_2$ is feasible then
12:                Generate $\mathcal{R}^i_1$ and $(\mathcal{S}_{\mathcal{R}^i_2}, \mathcal{I}_{\mathcal{R}^i_2})$ based on $\mathcal{R}'_1$ and $(\mathcal{S}_{\mathcal{R}'_2}, \mathcal{I}_{\mathcal{R}'_2})$ given $\mathcal{R}^i_2$;
13:                $\mathcal{R}^i_1 \leftarrow$ LSMTDFER($\mathcal{R}^i_1, \mathcal{S}_{\mathcal{R}^i_2}, \mathcal{I}_{\mathcal{R}^i_2}$);
14:                if $Penalty(\mathcal{R}^i_1) < Penalty(\mathcal{R}'_1)$ then
15:                    $(\mathcal{R}'_1, \mathcal{R}'_2, \mathcal{S}_{\mathcal{R}'_2}, \mathcal{I}_{\mathcal{R}'_2}) \leftarrow (\mathcal{R}^i_1, \mathcal{R}^i_2, \mathcal{S}_{\mathcal{R}^i_2}, \mathcal{I}_{\mathcal{R}^i_2})$;
16:                    $Improve \leftarrow 1$;
17:                    break;
18:                end if
19:            end if
20:        end while
21:    end while
22:    if $Penalty(\mathcal{R}'_1) < Penalty(\mathcal{R}_1)$ then
23:        $(\mathcal{R}_1, \mathcal{R}_2) \leftarrow (\mathcal{R}'_1, \mathcal{R}'_2)$;
24:        Initialize $\mathcal{NL}$;
25:    end if
26:    Remove $\mathcal{N}^{(j)}_{search}$ from $\mathcal{NL}$;
27: end while
28: return $(\mathcal{R}_1, \mathcal{R}_2)$;
```

cost of $\mathcal{R}^i_1$ is less than that of $\mathcal{R}'_1$ (line 15). It can be observed that the first-improvement strategy is adopted in the neighbourhood search and a neighbourhood structure is reused until the penalty cost of the current first-echelon routes $\mathcal{R}'_1$ cannot be further reduced or becomes zero (lines 6-21). If the penalty cost of $\mathcal{R}'_1$ is less than that of $\mathcal{R}_1$, the best complete solution $(\mathcal{R}_1, \mathcal{R}_2)$ is updated and the neighbourhood list $\mathcal{NL}$ is reinitialized (lines 22-25). The same procedure (lines 4-26) is repeated until the neighbourhood list $\mathcal{NL}$ becomes empty or the set of first-echelon routes $\mathcal{R}_1$ becomes feasible.

Algorithm 5.9 Recovering the feasibility of a set of second-echelon routes.

```
Function RFSER($\mathcal{R}_2$)
1:  $\mathcal{R}_2 \leftarrow$ LSMTDSER($\mathcal{R}_2$);
2:  $IterS \leftarrow 0$; $i \leftarrow 1$;
3:  while $Penalty(\mathcal{R}_2) > 0$ and $IterS < MaxIterS$ do
4:      $\mathcal{R}_2' \leftarrow$ SPSER($\mathcal{R}_2$,$\mathcal{N}_{shake}^{(i)}$);
5:      $\mathcal{R}_2' \leftarrow$ LSMTDSER($\mathcal{R}_2'$);
6:      if $Cost(\mathcal{R}_2') < Cost(\mathcal{R}_2)$ then
7:          $\mathcal{R}_2 \leftarrow \mathcal{R}_2'$;
8:          $IterS \leftarrow 0$; $i \leftarrow 1$;
9:      else
10:         $IterS \leftarrow IterS + 1$; $i \leftarrow (i \bmod |\mathcal{N}_{shake}|) + 1$;
11:     end if
12:     if $Penalty(\mathcal{R}_2) \geq MaxPenalty1$ and $IterS \geq IterS1$ then
13:         break;
14:     end if
15:     if $Penalty(\mathcal{R}_2) \geq MaxPenalty2$ and $IterS \geq IterS2$ then
16:         break;
17:     end if
18: end while
19: return $\mathcal{R}_2$;
```

In Algorithm 5.9, function RFSER() adopts the VNS framework to recover the feasibility of a set of second-echelon routes $\mathcal{R}_2$. Since infeasible second-echelon routes are allowed in function RFSER(), we use a cost function $Cost(\mathcal{R}_2) = Distance(\mathcal{R}_2) + Penalty(\mathcal{R}_2)$ to evaluate a given set of second-echelon routes $\mathcal{R}_2$. $Distance(\mathcal{R}_2)$ denotes the total travel distance of the second-echelon routes $\mathcal{R}_2$. $Penalty(\mathcal{R}_2)$ denotes the penalty cost for the possible constraint violations associated with the second-echelon routes $\mathcal{R}_2$ and $Penalty(\mathcal{R}_2) = \rho_2(V_2^{cap}(\mathcal{R}_2) + V_2^{tw}(\mathcal{R}_2))$. $V_2^{cap}(\mathcal{R}_2)$ and $V_2^{tw}(\mathcal{R}_2)$ respectively denote the vehicle capacity violations and the time window violations with the second-echelon routes $\mathcal{R}_2$. ρ_2 is the corresponding penalty factor. Note that the vehicle capacity violations $V_2^{cap}(\mathcal{R}_2)$ can be determined by calculating the largest load of a second-echelon vehicle k_2 at every satellite s on its route r_{k_2} for each $r_{k_2} \in \mathcal{R}_2$ considering demand uncertainty. In function RFSER(), $\mathcal{R}_2$ is first improved by function LSMTDSER() (line 1). Function LSMTDSER() is a local search procedure which adopts all nine neighbourhood structures in set $\mathcal{N}_{search}$ introduced in Section 5.3.1. Note that we only consider neighbourhood movements associated with customer nodes in the first six neighbourhood structures. Because those associated with satellite nodes are considered in the remaining three neighbourhood structures. In the local search procedure, the first improvement strategy is employed and all neighbourhood structures are selected in random order. In addition, a change tracking method proposed in Benjamin and Beasley (2013) is adopted and slightly extended to reduce the computational complexity of the procedure. Moreover, the local search procedure immediately stops when a set of feasible second-echelon routes is found. In the VNS main loop (lines 3-18), the second-echelon routes $\mathcal{R}_2$ is improved by functions SPSER() and LSMTDSER(). Function SPSER() is a shaking procedure which adopts the neighbourhood structures in set $\mathcal{N}_{shake}$. In function SPSER(), we only use the neighbourhood structures in $\mathcal{N}_{shake}$ that exchange more than five nodes among selected routes for instances with relatively few second-echelon routes (e.g., 2EC2, 2ER2, and 2ERC2 instances in Section 5.4). Because exchanging more nodes among fewer routes may be effective. In addition, the neighbourhood structures in $\mathcal{N}_{shake}$ that simultaneously perturb four routes are only used for instances with relatively more second-echelon routes (e.g., 2EC1, 2ER1, and 2ERC1 instances). Because exchanging fewer nodes among more routes may be better suited as discussed in Stenger et al. (2013). Moreover, we also try to insert satellites in a perturbed second-echelon route if it is infeasible due to the existence of vehicle capacity violations after the shaking process in function SPSER(). Specifically, we sequentially insert the satellite $s^* \in \mathcal{S}$ which incurs the least additional travel distance at the positions where vehicle capacity violations are detected on the route. The feasibility recovery process terminates when $\mathcal{R}_2$ becomes feasible ($Penalty(\mathcal{R}_2) = 0$) or a maximum number of iterations without an improvement $MaxIterS$ is reached. Two extra conditions are added in function RFSER() to reduce the computational time of the feasibility recovery process (lines 12-17).

5.3.4 Minimizing the Total Travel Distance of Vehicle Routes in Two Echelons

The total travel distance of the resulting solution from function RouteMinimization() is minimized by function DistanceMinimization() in the third phase of the metaheuristic. Function DistanceMinimization() also adopts the VNS framework and it is shown in pseudocode in Algorithm 5.10.

Algorithm 5.10 Minimizing the total travel distance of vehicle routes in two echelons.

```
Function DistanceMinimization($\mathcal{R}_1$,$\mathcal{R}_2$)
 1: $IterD \leftarrow 0$; $i \leftarrow 1$;
 2: while $IterD < MaxIterD$ do
 3:     do
 4:         $\mathcal{R}'_2 \leftarrow$ SPSER($\mathcal{R}_2$,$\mathcal{N}^{(i)}_{shake}$);
 5:         $\mathcal{R}'_2 \leftarrow$ RFSER($\mathcal{R}'_2$);
 6:     while $\mathcal{R}'_2$ is infeasible;
 7:     $\mathcal{R}'_1$ and $(\mathcal{S}_{\mathcal{R}'_2},\mathcal{I}_{\mathcal{R}'_2}) \leftarrow$ CFERFNV($\mathcal{R}'_2$, $|\mathcal{R}_1|$);
 8:     $(\mathcal{R}'_1,\mathcal{R}'_2) \leftarrow$ LSMTDTER($\mathcal{R}'_1,\mathcal{R}'_2,\mathcal{S}_{\mathcal{R}'_2},\mathcal{I}_{\mathcal{R}'_2}$);
 9:     if $Cost(\mathcal{R}'_1,\mathcal{R}'_2) < Cost(\mathcal{R}_1,\mathcal{R}_2)$ and $Penalty(\mathcal{R}'_1) = 0$ then
10:         $(\mathcal{R}_1,\mathcal{R}_2) \leftarrow (\mathcal{R}'_1,\mathcal{R}'_2)$;
11:         $i \leftarrow 1$;
12:     else
13:         $i \leftarrow (i \bmod |\mathcal{N}_{shake}|) + 1$
14:     end if
15:     $IterD \leftarrow IterD + 1$;
16: end while
17: return $(\mathcal{R}_1,\mathcal{R}_2)$;
```

In the VNS main loop of function DistanceMinimization(), we first generate a new set of feasible second-echelon routes $\mathcal{R}'_2$ using functions SPSER() and RFSER() (lines 3-6). Then, we construct a new set of first-echelon routes $\mathcal{R}'_1$ with $|\mathcal{R}_1|$ vehicles using function CFERFNV() based on the new second-echelon routes $\mathcal{R}'_2$ (line 7). Next, the total travel distance of the new complete solution $(\mathcal{R}'_1,\mathcal{R}'_2)$ is minimized by function LSMTDTER() (line 8). Function LSMTDTER() is a local search procedure which is shown in pseudocode in Algorithm 5.11. Note that function LSMTDTER() is very similar to function LSRFFER() in Algorithm 5.8. However, function LSMTDTER() aims

Algorithm 5.11 Local search for minimizing the travel distance of the vehicle routes in two echelons.

```
Function LSMTDTER($\mathcal{R}_1$,$\mathcal{R}_2$,$\mathcal{S}_{\mathcal{R}_2}$,$\mathcal{I}_{\mathcal{R}_2}$)
1: Initialize neighbourhood list $\mathcal{NL}$;
2: $(\mathcal{R}'_1,\mathcal{R}'_2,\mathcal{S}_{\mathcal{R}'_2},\mathcal{I}_{\mathcal{R}'_2}) \leftarrow (\mathcal{R}_1,\mathcal{R}_2,\mathcal{S}_{\mathcal{R}_2},\mathcal{I}_{\mathcal{R}_2})$;
3: while $\mathcal{NL} \neq \emptyset$ do
4:     Select a neighbourhood structure $\mathcal{N}^{(j)}_{search} \in \mathcal{NL}$ at random;
5:     $Improve \leftarrow 1$;
6:     while $Improve = 1$ do
7:         $i \leftarrow 0$; $Improve \leftarrow 0$;
8:         while $i < |\mathcal{N}^{(j)}_{search}(\mathcal{R}'_2)|$ do
9:             $i \leftarrow i+1$;
10:            Find the $i$th neighbour $\mathcal{R}^i_2 \in \mathcal{N}^{(j)}_{search}(\mathcal{R}'_2)$;
11:            if $\mathcal{R}^i_2$ is feasible then
12:                Generate $\mathcal{R}^i_1$ and $(\mathcal{S}_{\mathcal{R}^i_2},\mathcal{I}_{\mathcal{R}^i_2})$ based on $\mathcal{R}'_1$ and $(\mathcal{S}_{\mathcal{R}'_2},\mathcal{I}_{\mathcal{R}'_2})$ given $\mathcal{R}^i_2$;
13:                $\mathcal{R}^i_1 \leftarrow$ LSMTDFER($\mathcal{R}^i_1,\mathcal{S}_{\mathcal{R}^i_2},\mathcal{I}_{\mathcal{R}^i_2}$);
14:                if $Cost(\mathcal{R}^i_1,\mathcal{R}^i_2) < Cost(\mathcal{R}'_1,\mathcal{R}'_2)$ then
15:                    $(\mathcal{R}'_1,\mathcal{R}'_2,\mathcal{S}_{\mathcal{R}'_2},\mathcal{I}_{\mathcal{R}'_2}) \leftarrow (\mathcal{R}^i_1,\mathcal{R}^i_2,\mathcal{S}_{\mathcal{R}^i_2},\mathcal{I}_{\mathcal{R}^i_2})$;
16:                    $Improve \leftarrow 1$;
17:                    break;
18:                end if
19:            end if
20:        end while
21:    end while
22:    if $Cost(\mathcal{R}'_1,\mathcal{R}'_2) < Cost(\mathcal{R}_1,\mathcal{R}_2)$ then
23:        $(\mathcal{R}_1,\mathcal{R}_2) \leftarrow (\mathcal{R}'_1,\mathcal{R}'_2)$;
24:        Initialize $\mathcal{NL}$;
25:    end if
26:    Remove $\mathcal{N}^{(j)}_{search}$ from $\mathcal{NL}$;
27: end while
28: return $(\mathcal{R}_1,\mathcal{R}_2)$;
```

to minimize the total travel distance of the vehicle routes in two echelons. In function LSMTDTER(), infeasible second-echelon vehicle routes are not allowed while infeasible first-echelon vehicle routes are allowed. Because it employs function LSMTDFER() to improve a new set of first-echelon routes $\mathcal{R}^i_1$ (line 13) and $\mathcal{R}^i_1$ can be infeasible. We use a cost function $Cost(\mathcal{R}_1,\mathcal{R}_2) = Distance(\mathcal{R}_1,\mathcal{R}_2) + Penalty(\mathcal{R}_1)$ to evaluate a complete solution $(\mathcal{R}_1,\mathcal{R}_2)$ in functions LSMTDTER() and DistanceMinimization(). $Penalty(\mathcal{R}_1)$ can be calculated using the same expression introduced in the last paragraph in Section

5.3.2.2. In function DistanceMinimization(), we update the best solution $(\mathcal{R}_1, \mathcal{R}_2)$ with the new improved complete solution $(\mathcal{R}'_1, \mathcal{R}'_2)$ if $(\mathcal{R}'_1, \mathcal{R}'_2)$ is feasible and its cost function value $Cost(\mathcal{R}'_1, \mathcal{R}'_2)$ is less than $Cost(\mathcal{R}_1, \mathcal{R}_2)$ (lines 9-10). The distance minimization procedure stops after a number of $MaxIterD$ main VNS iterations and function DistanceMinimization() returns the best found solution $(\mathcal{R}_1, \mathcal{R}_2)$.

5.4 Computational Experiments

In this section, we perform an extensive computational study on the adapted benchmark instances, which aims to highlight the important features of the proposed methods. In Section 5.4.1, we give a detailed description about the adapted test instances for the 2E-MTVRPTWSS under demand uncertainty and the parameter settings for the proposed VNS-based metaheuristic. Computational results and analyses follow in Section 5.4.2, alongside a comparison between the VNS-based metaheuristic and the ALNS algorithm proposed for the 2E-MTVRPTWSS in Grangier et al. (2016).

5.4.1 Experiment Description and Parameter Setting

To the best of our knowledge, there are no available instances for the 2E-MTVRPTWSS based on a two-echelon collection system under customer demand uncertainty. However, Grangier et al. (2016) studied a 2E-MTVRPTWSS based on a two-echelon distribution system without considering uncertainty and the Solomon's benchmark instances (Solomon, 1987) were adapted as the test instances. Thus, we also adapt the Solomon's instances for the 2E-MTVRPTWSS under demand uncertainty in the computational experiments. The adapted test instances carry the following assumptions: (1) the geographical locations, service times, and time windows of all customers in each adapted instance stay the same as those in the corresponding Solomon's instance; (2) the depot node (node 0) in each Solomon's instance serves as the depot for second-echelon vehicles; (3) one logistics centre and eight satellites are considered in each adapted instance and their geographical locations are shown in Figure 5.3; (4) service times at all satellites are assumed to be zero; (5) the latest allowable arrival time of the depot node in each Solomon's instance is considered as the maximum planning horizon T_{max} and a constant $d_{o_1o_2}$ is also added to T_{max} to enable the existence of feasible solutions for all adapted instances. Similar to the Solomon's instances, the adapted test instances can be divided into six classes, referred to as 2EC1, 2EC2, 2ER1, 2ER2, 2ERC1, and 2ERC2. The capacities of first- and second-echelon vehicles in the 2EC1, 2ER1, and 2ERC1 instances are set to 4 and 0.5 times the vehicle capacity considered in Solomon's C1, R1, and RC1 instances, respectively. The capacities of first- and second-echelon vehicles in the 2EC2, 2ER2, and 2ERC2 instances are set to 2 and 0.25 times the vehicle capacity considered in Solomon's C2, R2, and RC2 instances, respectively. The vehicles in both echelons are assumed to travel at the same speed. Note that the instances of classes 2EC1, 2ER1, and 2ERC1 have narrow time windows and relatively small-capacity vehicles in both echelons whereas those of classes 2EC2, 2ER2, and 2ERC2 exhibit the converse. Our experiments focus on medium-sized instances with 50 customers and large-sized instances with 100 customers. Each medium-sized instance contains the first 50 customers in the corresponding large-sized instance.

In the 2E-MTVRPTWSS under uncertainty introduced in Section 5.3.2, customer demands are uncertain parameters. Moreover, uncertain customer demands $\tilde{q}_i \, (\forall i \in \mathcal{C})$ take values in the demand uncertainty set $\mathcal{U}_q$ (5.30) defined with uncertainty polytopes $\mathcal{U}_q^{k_1}$ (5.31) and $\mathcal{U}_q^{k_2}$ (5.32). In polytopes $\mathcal{U}_q^{k_1}$ and $\mathcal{U}_q^{k_2}$, the nominal values $\overline{q}_i$ of uncertain parameters $\tilde{q}_i \, (\forall i \in \mathcal{C})$ are critical and they need to be set by decision-makers (route planners). Thus, we set the nominal values $\overline{q}_i$ of uncertain customer demands $\tilde{q}_i$ in each medium- or large-sized instance to the original values of deterministic customer demands in the corresponding Solomon's instance. In addition, there are two important types of parameters in uncertainty polytopes $\mathcal{U}_q^{k_1}$ and $\mathcal{U}_q^{k_2}$, which are the maximum deviations $\hat{q}_i$ of the uncertain demands $\tilde{q}_i$ and the uncertainty budget coefficients (θ_1, θ_2). The maximum

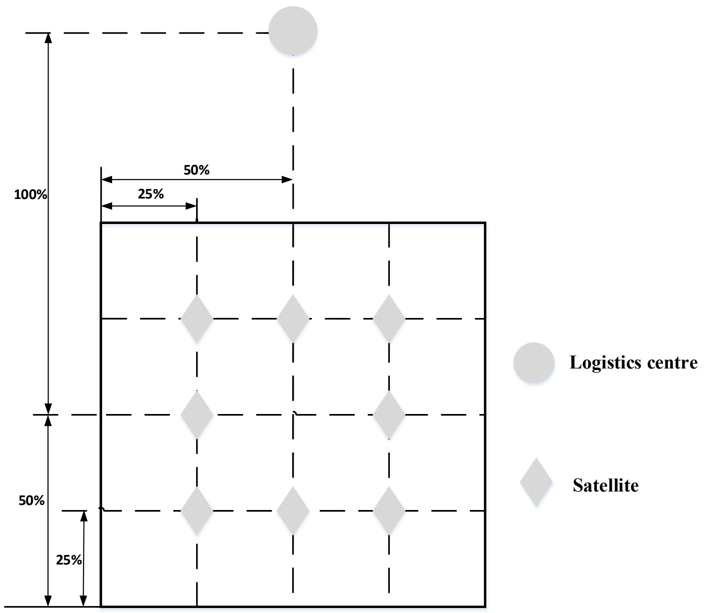


Figure 5.3 The geographical locations of the logistics centre and satellites.

deviations $\hat{q}_i$ restrict the maximum variation ranges of the uncertain parameters $\tilde{q}_i$ for all $i \in \mathcal{C}$ and the uncertainty budget coefficients (θ_1, θ_2) help to determine the uncertainty budgets $(\Gamma^{k_1}, \Gamma^{k_2}_s)$ in the corresponding uncertainty polytopes. To investigate the impacts of these two types of parameters on deriving robust solutions, we first solve the medium-sized instances with 50 customers and 8 satellites considering different parameter settings in uncertainty polytopes $\mathcal{U}^{k_1}_q$ and $\mathcal{U}^{k_2}_q$. To investigate the impacts of the maximum deviations $\hat{q}_i$ of uncertain customer demands $\tilde{q}_i$, we consider two cases. Specifically, we consider a low uncertainty case in which $\hat{q}_i = 0.2\overline{q}_i$ for all $i \in \mathcal{C}$ and a high uncertainty case in which $\hat{q}_i = 0.4\overline{q}_i$ for all $i \in \mathcal{C}$. To investigate the impacts of the uncertainty budget coefficients (θ_1, θ_2), we consider a total of five combinations of coefficient values including $(0,0)$, $(0.1,0.1)$, $(0.1,0.2)$, $(0.2,0.2)$, and $(0.2,0.4)$. As discussed in Section 5.2.2, no demand uncertainty is considered in all polytopes when the uncertainty budgets $(\theta_1, \theta_2) = (0,0)$. Thus, the solution derived with $(\theta_1, \theta_2) = (0,0)$ is referred to as the deterministic solution for each medium-sized instance. To illustrate the effectiveness of the VNS-based metaheuristic, we also derive the robust solutions for the large-sized instances with 100 customers and 8 satellites considering the following parameter settings in uncertainty polytopes $\mathcal{U}_q^{k_1}$ and $\mathcal{U}_q^{k_2}$. We set the uncertainty budget coefficients $(\theta_1, \theta_2) = (0.2, 0.4)$ for the 2EC1, 2ER1, and 2ERC1 instances and $(\theta_1, \theta_2) = (0.1, 0.2)$ for the 2EC2, 2ER2, and 2ERC2 instances. In addition, we only consider a low uncertainty case in which $\hat{q}_i = 0.2\overline{q}_i$ for all $i \in \mathcal{C}$. The corresponding deterministic solutions are also derived with $(\theta_1, \theta_2) = (0, 0)$ for the large-sized instances. Thus, we generate solutions with different parameter settings in the uncertainty polytopes for both medium- and large-sized instances in the experiments.

All experiments are conducted on a 3.4 GHz Inter core i7 machine with 8G of RAM. The VNS-based metaheuristic is coded in Matlab 2018b and its detailed parameter settings are described as follows. The parameter values which need to be set in the VNS-based metaheuristic are those in the key functions of the metaheuristic. Specifically, we set $MaxIterF = 20$ and $\rho_1 = 1000$ in function CFERFNV(). We set $MaxIterS = 50$ and $\rho_2 = 1000$ in function RFSER(). As two extra conditions are added in function RFSER() to speed up the feasibility recovery process for a set of infeasible second-echelon routes, we set $MaxPenalty1 = 100\rho_2$, $IterS1 = 5$, $MaxPenalty2 = 10\rho_2$, and $IterS2 = 25$. To guarantee a reasonable running time for all adapted instances, we set $MaxIterR = 20$ in function RFFER() and $MaxIterD = 500$ in function DistanceMinimization() for the 2EC2, 2ER2, and 2ERC2 instances. However, we set $MaxIterR = 10$ in function RFFER() and $MaxIterD = 250$ in function DistanceMinimization() for the 2EC1, 2ER1, and 2ERC1 instances. Because the solutions for these instances with large sizes generally contain more first-echelon routes. Thus, more computational efforts are needed in the key function LSMTDFER() which uses a local search procedure to improve first-echelon routes. We also consider a time limit of 900 seconds for the first two phases of the VNS-metaheuristic and a time limit of 7200 seconds for implementing the whole algorithm.

We design Monte Carlo simulation tests to evaluate the robustness of the obtained solutions for all test instances. In the simulation tests, 1000 random and independent scenarios for uncertain customer demands $\tilde{q}_i \, (\forall i \in \mathcal{C})$ are generated. Since the exact distributions of uncertain customer demands are unknown in the 2E-MTVRPTWSS under uncertainty, uniform distributions are assumed for uncertain parameters $\tilde{q}_i\,(\forall i \in \mathcal{C})$ to generate these scenarios following the ideas used in Lu and Gzara (2019) and Munari et al. (2019). Specifically, we generate a customer demand vector in each scenario. The vector consists of component values which are uniformly drawn from the interval $[\overline{q}_i - \hat{q}_i, \overline{q}_i + \hat{q}_i]$ for all $i \in \mathcal{C}$. Given the 1000 generated scenarios, we assess the robustness of a given solution based on the following indicators.

Feasibility ratio (FR). This indicator reflects the robustness of a given solution in terms of its feasibility ratio. It is calculated as the percentage of the scenarios in which the solution remains feasible over the 1000 randomly generated ones. Given the generated demand vector in each scenario and the vehicle routes in the evaluated solution, the feasibility of the solution can be determined by simply checking the capacity constraints of the vehicles in two echelons.

Average unfulfilled demand (AUD). This indicator reflects the solution robustness of a given solution in terms of the unfulfilled demands of all customers. Given the generated demand vector in each scenario and the vehicle routes in the evaluated solution, the total unfulfilled customer demands can be calculated by summing the unfulfilled demand at each customer's location caused by the inefficient loading capacity of a second-echelon vehicle and the unfulfilled demand at each visited satellite caused by the inefficient loading capacity of a first-echelon vehicle. Given the 1000 generated scenarios, the average unfulfilled customer demand associated with the evaluated solution can accordingly be computed.

5.4.2 Computational Results

In this section, we first present the computational results for the adapted medium-sized instances with 50 customers and 8 satellites. The impact of customer demand uncertainty on deriving robust solutions for the medium-sized instances of different classes is investigated. Next, we present the computational results for the adapted large-sized instances with 100 customers and 8 satellites. A detailed analysis of the computational results is performed and managerial insights for practical 2E-MTVRPTWSS applications under demand uncertainty are derived. Finally, we show the effectiveness of the VNS-based metaheuristic by comparing it with the effective ALNS algorithm developed for the 2E-MTVRPTWSS in Grangier et al. (2016).

5.4.2.1 Results for Medium-Sized Instances

As introduced in Section 5.4.1, the adapted test instances are grouped into six classes, referred to as 2EC1, 2EC2, 2ER1, 2ER2, 2ERC1, and 2ERC2. We present the average results of the solutions generated with different parameter settings in the uncertainty polytopes for the medium-sized instances of different classes in Tables 5.2-5.7. In these tables, average results of the obtained solutions are tabulated in terms of the number of first-echelon vehicles used (NVFE), the number of second-echelon vehicles used (NVSE), the travel distance (TD), and two robustness indicators (FR and AUD) from the Monte Carlo simulation tests. As mentioned in Section 5.4.1, we consider five value combinations for the uncertainty budget coefficients (θ_1, θ_2) and two cases for the maximum deviations $\hat{q}_i$ of uncertain customer demands $\tilde{q}_i\,(\forall i \in \mathcal{C})$ in the uncertainty polytopes. Given every setting for the uncertainty budget coefficients (θ_1, θ_2) and the maximum demand deviations $\hat{q}_i$, we run each instance (e.g., instance 2EC101) in its class (e.g., class 2EC1) ten times and select the best solution from ten runs for the instance. We then calculate the average results for all instances of a class based on the selected best solutions of all instances in that class under each considered setting for the uncertainty budget coefficients (θ_1, θ_2) and the maximum demand deviations $\hat{q}_i\,(\forall i \in \mathcal{C})$.

We perform a detailed analysis of the computational results presented in Tables 5.2-5.7. Novel findings are observed and useful routing suggestions for practical 2E-MTVRPTWSS applications under customer demand uncertainty are provided.

Table 5.2. Average results of the solutions derived with different parameter settings in the uncertainty polytopes for the 2EC1 instances with 50 customers and 8 satellites.

(θ_1,θ_2)	Low uncertainty ($\hat{q}_i = 0.2\overline{q}_i$)					High uncertainty ($\hat{q}_i = 0.4\overline{q}_i$)				
	NV FE	NV SE	TD	FR (%)	AUD	NV FE	NV SE	TD	FR (%)	AUD
(0,0)	2.00	5.11	905.39	7.48	9.41	2.00	5.11	905.39	4.67	20.73
(0.1,0.1)	2.00	5.11	935.87	86.78	0.24	2.00	5.22	958.20	76.41	0.91
(0.1,0.2)	2.00	5.11	936.38	86.41	0.25	2.00	5.22	962.44	88.69	0.33
(0.2,0.2)	2.00	5.11	936.91	86.27	0.25	2.00	5.22	962.72	88.98	0.32
(0.2,0.4)	2.00	5.22	962.72	88.98	0.32	2.00	5.22	965.31	93.30	0.18

Table 5.3. Average results of the solutions derived with different parameter settings in the uncertainty polytopes for the 2ER1 instances with 50 customers and 8 satellites.

(θ_1,θ_2)	Low uncertainty ($\hat{q}_i = 0.2\overline{q}_i$)					High uncertainty ($\hat{q}_i = 0.4\overline{q}_i$)				
	NV FE	NV SE	TD	FR (%)	AUD	NV FE	NV SE	TD	FR (%)	AUD
(0,0)	1.00	7.58	1177.20	32.68	3.31	1.00	7.58	1177.20	14.97	11.34
(0.1,0.1)	1.00	7.58	1202.57	79.93	0.44	1.00	7.67	1212.97	65.15	1.63
(0.1,0.2)	1.00	7.67	1206.46	92.42	0.14	1.00	7.75	1226.57	81.44	0.75
(0.2,0.2)	1.00	7.67	1207.66	93.81	0.10	2.00	7.75	1388.68	84.03	0.59
(0.2,0.4)	1.00	7.75	1210.73	99.30	0.01	2.00	8.00	1410.19	98.38	0.04

Table 5.4. Average results of the solutions derived with different parameter settings in the uncertainty polytopes for the 2ERC1 instances with 50 customers and 8 satellites.

(θ_1,θ_2)	Low uncertainty ($\hat{q}_i = 0.2\overline{q}_i$)					High uncertainty ($\hat{q}_i = 0.4\overline{q}_i$)				
	NV FE	NV SE	TD	FR (%)	AUD	NV FE	NV SE	TD	FR (%)	AUD
(0,0)	2.00	7.50	1426.89	1.26	14.81	2.00	7.50	1426.89	0.94	31.25
(0.1,0.1)	2.00	7.63	1543.84	81.94	0.35	2.00	8.13	1591.79	71.28	1.27
(0.1,0.2)	2.00	7.63	1547.75	83.69	0.31	2.00	8.38	1576.49	81.35	0.65
(0.2,0.2)	2.00	7.63	1550.67	81.81	0.35	2.00	8.38	1581.19	82.56	0.58
(0.2,0.4)	2.00	8.13	1588.39	96.55	0.05	2.00	8.50	1592.49	93.08	0.20

1) The deterministic solutions generated with $(\theta_1,\theta_2) = (0,0)$ for the medium-sized instances of classes 2EC1, 2ER1, and 2ERC1 are sensitive to customer demand uncertainty. As shown in Tables 5.2 and 5.4, the average FRs of the deterministic solutions for the 2EC1 and 2ERC1 instances are, respectively, 7.48% and 1.26% in

Table 5.5. Average results of the solutions derived with different parameter settings in the uncertainty polytopes for the 2EC2 instances with 50 customers and 8 satellites.

(θ_1,θ_2)	Low uncertainty ($\hat{q}_i = 0.2\overline{q}_i$)					High uncertainty ($\hat{q}_i = 0.4\overline{q}_i$)				
	NV FE	NV SE	TD	FR (%)	AUD	NV FE	NV SE	TD	FR (%)	AUD
(0,0)	1.00	2.00	734.73	50.49	2.34	1.00	2.00	734.73	27.24	9.71
(0.1,0.1)	1.00	2.00	748.23	96.46	0.07	1.00	2.00	754.60	85.89	0.84
(0.1,0.2)	1.00	2.00	751.01	97.93	0.04	1.00	2.00	758.86	97.09	0.10
(0.2,0.2)	1.00	2.00	753.08	97.86	0.04	1.00	2.00	759.50	97.88	0.07
(0.2,0.4)	1.00	2.00	756.83	100.0	0.00	1.00	2.00	778.04	99.43	0.02

Table 5.6. Average results of the solutions derived with different parameter settings in the uncertainty polytopes for the 2ER2 instances with 50 customers and 8 satellites.

(θ_1,θ_2)	Low uncertainty ($\hat{q}_i = 0.2\overline{q}_i$)					High uncertainty ($\hat{q}_i = 0.4\overline{q}_i$)				
	NV FE	NV SE	TD	FR (%)	AUD	NV FE	NV SE	TD	FR (%)	AUD
(0,0)	1.00	2.00	930.46	61.50	2.34	1.00	2.00	930.46	47.85	7.12
(0.1,0.1)	1.00	2.00	934.73	97.46	0.07	1.00	2.00	935.71	94.26	0.34
(0.1,0.2)	1.00	2.00	935.84	99.94	0.00	1.00	2.00	939.75	99.45	0.02
(0.2,0.2)	1.00	2.00	935.87	99.97	0.00	1.00	2.00	940.79	99.55	0.02
(0.2,0.4)	1.00	2.00	938.56	100.0	0.00	1.00	2.00	946.92	100.0	0.00

Table 5.7. Average results of the solutions derived with different parameter settings in the uncertainty polytopes for the 2ERC2 instances with 50 customers and 8 satellites.

(θ_1,θ_2)	Low uncertainty ($\hat{q}_i = 0.2\overline{q}_i$)					High uncertainty ($\hat{q}_i = 0.4\overline{q}_i$)				
	NV FE	NV SE	TD	FR (%)	AUD	NV FE	NV SE	TD	FR (%)	AUD
(0,0)	1.00	2.13	1006.95	69.94	2.22	1.00	2.13	1006.95	57.43	6.08
(0.1,0.1)	1.00	2.13	1012.30	97.99	0.06	1.00	2.13	1015.55	92.98	0.51
(0.1,0.2)	1.00	2.13	1015.45	99.13	0.03	1.00	2.13	1027.89	98.50	0.07
(0.2,0.2)	1.00	2.13	1015.09	99.26	0.02	1.00	2.13	1030.16	98.60	0.07
(0.2,0.4)	1.00	2.13	1016.37	99.99	0.00	1.00	2.13	1047.30	99.91	0.00

the Monte Carlo simulation tests even considering the low uncertainty case. However, such solutions for the 2EC2, 2ER2, and 2ERC2 instances exhibit relatively high robustness in terms of FR and AUD. As shown in Tables 5.5-5.7, the average FRs of the deterministic solutions for the 2EC2, 2ER2, and 2ERC2 instances are all above 50% in the simulation tests considering the low uncertainty case. Moreover, even considering the high uncertainty case in the simulation tests, their average FRs for the 2ER2 and 2ERC2 instances are around 50% and their average AUDs are less than 10. Thus, deterministic routing strategies for practical 2E-MTVRPTWSS applications with narrow time windows and small-capacity vehicles are very likely to become infeasible under customer demand uncertainty. However, such strategies may not be very sensitive to uncertainty in customer demands for 2E-MTVRPTWSS applications with wide time windows and large-capacity vehicles.

2) The robust solutions generated for the medium-sized instances of all classes considering customer demand uncertainty ($(\theta_1, \theta_2) \neq (0,0)$) attain substantial robustness in terms of FR and AUD. Consider the robust solutions generated with $(\theta_1, \theta_2) = (0.2, 0.4)$ for the 2EC1 instances in the high uncertainty case (Table 5.2). Their average FR is 93.30% and their average AUD is only 0.18. Similar observations are made for the robust solutions derived for the medium-sized instances of other classes in both low and high uncertainty cases. Thus, robust routing strategies can be reliable for real-life 2E-MTVRPTWSS applications under customer demand uncertainty.

3) Despite their substantial robustness for the 2EC1, 2ER1, and 2ERC1 instances, the robust solutions may incur large additional costs in terms of the travel distance and the number of vehicles used in two echelons, especially in the high uncertainty case. Consider the robust solutions generated with $(\theta_1, \theta_2) = (0.2, 0.4)$ for the 2ER1 instances in the high uncertainty case (Table 5.3). Compared to their deterministic counterparts, the average number of first-echelon vehicles used increases from 1.00 to 2.00, the average number of second-echelon vehicles used increases from 7.58 to 8.00, and the average travel distance significantly increases from 1177.20 to 1410.19 in the robust solutions. Similar observations are made for the robust solutions derived for the 2ERC1 instances in the high uncertainty case. Thus, it can be

very expensive to generate robust routing strategies for practical 2E-MTVRPTWSS applications with narrow time windows and small-capacity vehicles under high demand uncertainty. However, the robust solutions derived for the 2EC2, 2ER2, and 2ERC2 instances can reach a high robustness level at almost no additional cost. Consider the robust solutions generated with $(\theta_1, \theta_2) = (0.1, 0.2)$ for the 2EC2 instances in the high uncertainty case (Table 5.5). Their average FR is 97.09% and their average AUD is close to 0. Compared to their deterministic counterparts, the average number of first- and second-echelon vehicles used stays the same and the average travel distance only increases from 734.73 to 758.86 in the robust solutions. Similar observations are made for the robust solutions derived for the 2ER2 and 2ERC2 instances in both low and high uncertainty cases. Thus, highly robust routing strategies can be derived for 2E-MTVRPTWSS applications with wide time windows and large-capacity vehicles under demand uncertainty at little additional cost.

4) Increasing the values of uncertainty budget coefficients (θ_1, θ_2) generally can improve the robustness of the robust solutions derived for the medium-sized instances of all classes, as shown in Tables 5.2-5.7. However, it may significantly increase their routing costs in terms of the travel distance and the number of vehicles used in both echelons. Consider the robust solutions generated for the 2ER1 instances in the high uncertainty case (Table 5.3). When the uncertainty budget coefficients (θ_1, θ_2) increase from $(0.1, 0.2)$ to $(0.2, 0.4)$, their average FR increases from 81.44% to 98.38% and their average AUD decreases from 0.75 to 0.04. However, the average number of first-echelon vehicles used increases from 1.00 to 2.00, the average number of second-echelon vehicles used increases from 7.75 to 8.00, and the average travel distance drastically increases from 1226.57 to 1410.19 in the robust solutions. Thus, decision-makers may need to generate robust routing strategies with different uncertainty budget coefficient values for a practical 2E-MTVRPTWSS application under high demand uncertainty. Then, they can choose ideal robust routing strategies

which have a good balance between routing cost and robustness and determine the appropriate values for the uncertainty budget coefficients (θ_1, θ_2).

5.4.2.2 Results for Large-Sized Instances

To investigate the effectiveness of the VNS-based metaheuristic, we generate both the deterministic and robust solutions for the large-sized instances with 100 customers and 8 satellites. The robust solutions are generated with the following parameter settings in the uncertainty polytopes $\mathcal{U}_q^{k_1}$ and $\mathcal{U}_q^{k_2}$. Specifically, we set the uncertainty budget coefficients $(\theta_1, \theta_2) = (0.2, 0.4)$ for the 2EC1, 2ER1, and 2ERC1 instances and $(\theta_1, \theta_2) = (0.1, 0.2)$ for the 2EC2, 2ER2, and 2ERC2 instances. In addition, we only consider the low uncertainty case, in which $\hat{q}_i = 0.2\overline{q}_i$ for all $i \in \mathcal{C}$. The corresponding deterministic solutions are generated with $(\theta_1, \theta_2) = (0, 0)$ for all large-sized instances by using the VNS-based metaheuristic with slight modifications. Note that each test instance is run ten times given the robust or deterministic setting. The best deterministic and robust solutions among ten runs are selected for each instance. Next, we first present the selected best deterministic and robust solutions for the large-sized instances of classes 2ER1, 2EC1, and 2ERC1 in Table 5.8. The columns under "Deterministic" and "Robust" show the detailed results of the deterministic and robust solutions derived for each test instance, respectively. The headings NVFE, NVSE, TD, FR, and AUD have the same meaning as those in Tables 5.2-5.7. The columns under "Gap" show the differences between these two types of solutions.

As shown in Table 5.8, the deterministic solutions derived for the large-sized instances of classes 2ER1, 2EC1, and 2ERC1 are fragile. The FRs of most deterministic solutions are less than 5% and their AUDs are more than 10 in the Monte Carlo simulation tests. Thus, deterministic routing strategies may be unreliable for large-scale practical 2E-MTVRPTWSS applications with narrow time windows and small-capacity vehicles under customer demand uncertainty. However, the robust solutions derived for the 2ER1, 2EC1, and 2ERC1 instances are almost immune against customer demand uncertainty. The

Table 5.8. Detailed results of the deterministic and robust solutions derived for the 2EC1, 2ER1, and 2ERC1 instances with 100 customers and 8 satellites.

Instance	Deterministic					Robust					Gap		
	NV FE	NV SE	TD	FR (%)	AUD	NV FE	NV SE	TD	FR (%)	AUD	NV FE	NV SE	TD (%)
2EC101	3	11	2046.42	2.00	14.11	3	11	2278.85	83.70	0.35	0	0	11.36
2EC102	3	9	2131.29	0.40	20.04	3	10	2164.37	94.20	0.07	0	1	1.55
2EC103	3	9	1946.39	0.30	17.75	3	9	2098.42	90.10	0.13	0	0	7.81
2EC104	3	9	1852.77	0.10	23.91	3	9	2015.45	91.60	0.14	0	0	8.78
2EC105	3	10	1928.49	0.80	16.35	3	10	2140.94	95.50	0.05	0	0	11.02
2EC106	3	10	1957.17	2.30	11.65	3	11	2126.00	96.90	0.04	0	1	8.63
2EC107	3	10	1932.53	0.60	16.60	3	10	2080.63	96.90	0.04	0	0	7.66
2EC108	3	10	1918.47	0.00	23.45	3	10	2051.51	92.00	0.12	0	0	6.93
2EC109	3	9	1907.56	0.00	30.69	3	9	2059.09	95.50	0.07	0	0	7.94
2ER101	2	19	2333.75	21.70	4.43	2	19	2358.57	99.60	0.01	0	0	1.06
2ER102	2	17	2176.30	12.30	9.76	2	17	2230.39	99.60	0.01	0	0	2.49
2ER103	2	14	1954.28	6.90	8.38	2	14	2014.89	98.20	0.02	0	0	3.10
2ER104	2	11	1777.65	9.00	6.89	2	11	1867.89	99.80	0.00	0	0	5.08
2ER105	2	14	2107.59	2.20	11.94	2	15	2201.65	99.30	0.01	0	1	4.46
2ER106	2	12	2032.14	1.70	14.58	2	12	2221.84	99.90	0.00	0	0	9.33
2ER107	2	11	1881.99	9.60	7.36	2	11	2103.83	98.10	0.02	0	0	11.79
2ER108	2	10	1709.58	9.00	7.51	2	10	1892.93	97.90	0.02	0	0	10.72
2ER109	2	12	2060.47	2.10	11.15	2	13	2129.47	97.00	0.05	0	1	3.35
2ER110	2	12	1875.02	2.70	9.08	2	12	2053.38	96.10	0.06	0	0	9.51
2ER111	2	11	1847.44	3.20	10.37	2	11	1977.49	99.20	0.01	0	0	7.04
2ER112	2	11	1760.13	5.20	8.26	2	11	1917.41	99.70	0.00	0	0	8.94
2ERC101	3	16	2647.37	0.70	16.32	3	16	2818.37	97.80	0.02	0	0	6.46
2ERC102	3	14	2596.56	0.60	14.54	3	15	2701.34	97.40	0.03	0	1	4.04
2ERC103	3	12	2377.25	1.00	14.77	3	13	2511.13	96.60	0.04	0	1	5.63
2ERC104	3	11	2310.15	4.10	10.60	3	12	2338.65	96.20	0.05	0	1	1.23
2ERC105	3	15	2598.79	0.20	19.72	3	16	2679.96	93.70	0.08	0	1	3.12
2ERC106	3	13	2501.47	5.70	8.43	3	14	2647.51	98.00	0.03	0	1	5.84
2ERC107	3	13	2309.69	2.10	11.60	3	13	2582.51	90.40	0.13	0	0	11.81
2ERC108	3	12	2302.79	0.70	16.18	3	13	2422.69	96.00	0.06	0	1	5.21

FRs of most robust solutions are more than 95% and their AUDs are around 0.1 in the simulation tests. Compared to their deterministic counterparts, the robust solutions do not use additional first-echelon vehicles for all instances and they only adopt one more second-echelon vehicle for several instances (e.g., 2EC102, 2ER105, and 2ERC103). Moreover, the travel distances of most robust solutions only have a relatively small increase (less than 10%) compared to their deterministic counterparts. However, the travel distances of some robust solutions do increase by a relatively large amount. Consider the robust solution derived for instance 2ER107. Despite its FR reaches 98.10%, its travel distance increases by 11.79% compared to that of its deterministic counterpart. Thus, decision-makers may first need to generate robust routing strategies with different uncertainty budget coefficient values for practical 2E-MTVRPTWSS applications with narrow time windows and small-capacity vehicles under customer demand uncertainty. Then, they can select ideal robust routing strategies which have a good trade-off between routing cost and robustness.

Table 5.9. Detailed results of the deterministic and robust solutions derived for the 2EC2, 2ER2, and 2ERC2 instances with 100 customers and 8 satellites.

Instance	Deterministic					Robust					Gap		
	NV FE	NV SE	TD	FR (%)	AUD	NV FE	NV SE	TD	FR (%)	AUD	NV FE	NV SE	TD (%)
2EC201	2	3	1351.52	56.00	2.05	2	3	1378.85	93.60	0.12	0	0	2.02
2EC202	2	3	1335.88	41.10	3.29	2	3	1378.09	93.60	0.12	0	0	3.16
2EC203	2	3	1313.15	21.90	5.72	2	3	1347.21	93.30	0.13	0	0	2.59
2EC204	2	3	1270.46	32.00	4.19	2	3	1308.47	93.70	0.12	0	0	2.99
2EC205	2	3	1327.33	31.50	3.95	2	3	1364.51	95.80	0.08	0	0	2.80
2EC206	2	3	1324.73	31.50	3.95	2	3	1358.74	93.80	0.12	0	0	2.57
2EC207	2	3	1325.19	23.90	4.98	2	3	1347.89	94.80	0.10	0	0	1.71
2EC208	2	3	1317.74	18.00	6.62	2	3	1355.26	93.80	0.12	0	0	2.85
2ER201	1	4	1585.64	72.30	1.47	1	4	1607.86	100.0	0.00	0	0	1.40
2ER202	1	3	1525.83	34.70	4.50	1	3	1540.36	100.0	0.00	0	0	0.95
2ER203	1	3	1254.86	21.70	6.65	1	3	1268.89	99.30	0.02	0	0	1.12
2ER204	1	2	1183.16	61.00	2.21	1	2	1209.16	99.40	0.01	0	0	2.20
2ER205	1	3	1337.26	35.00	4.70	1	3	1357.19	99.30	0.02	0	0	1.49
2ER206	1	3	1246.32	52.90	2.90	1	3	1249.22	99.10	0.02	0	0	0.23
2ER207	1	2	1260.74	50.50	2.74	1	2	1262.81	100.0	0.00	0	0	0.16
2ER208	1	2	1068.54	65.90	1.81	1	2	1070.32	99.90	0.00	0	0	0.17
2ER209	1	3	1257.81	33.00	4.80	1	3	1261.15	100.0	0.00	0	0	0.27
2ER210	1	3	1292.21	26.80	5.69	1	3	1306.20	99.30	0.01	0	0	1.08
2ER211	1	2	1276.30	26.20	6.10	1	2	1351.37	99.90	0.01	0	0	5.88
2ERC201	1	4	1798.49	64.90	1.63	1	4	1816.45	99.60	0.01	0	0	1.00
2ERC202	1	3	1805.76	29.00	6.13	1	3	1808.93	100.0	0.00	0	0	0.18
2ERC203	1	3	1454.61	60.00	2.14	1	3	1463.53	96.40	0.10	0	0	0.61
2ERC204	1	3	1205.41	25.00	6.75	1	3	1222.57	99.20	0.02	0	0	1.42
2ERC205	1	4	1650.38	69.70	1.38	1	4	1667.08	99.60	0.01	0	0	1.01
2ERC206	1	3	1589.54	53.90	3.00	1	3	1621.93	100.0	0.00	0	0	2.04
2ERC207	1	3	1513.72	57.90	2.68	1	3	1552.03	96.80	0.08	0	0	2.53
2ERC208	1	3	1311.08	46.40	3.44	1	3	1319.77	98.50	0.04	0	0	0.66

The detailed results of the deterministic and robust solutions derived for the large-sized instances of classes 2EC2, 2ER2, and 2ERC2 are shown in Table 5.9. Unlike the deterministic solutions for the 2EC1, 2ER1, and 2ERC1 instances, such solutions for the 2EC2, 2ER2, and 2ERC2 instances are not very fragile. The FRs of the deterministic solutions for some instances (e.g., 2ER201, 2ER204, and 2ERC201) are even more than 60% in the Monte Carlo simulation tests. The robust solutions for the 2EC2, 2ER2, and 2ERC2 instances attain substantial robustness in terms of FR and AUD. For example, the FRs of most robust solutions are more than 95% and their AUDs are close to 0. Moreover, the robust solutions do not use additional first- and second-echelon vehicles and their travel distances only increase by a small amount compared to their deterministic counterparts. Thus, robust routing strategies can be cost-effective and reliable for real-life 2E-MTVRPTWSS applications with wide time windows and large-capacity vehicles under customer demand uncertainty.

To demonstrate the performance of the VNS-based metaheuristic, we compare it with the ALNS algorithm proposed in Grangier et al. (2016). Grangier et al. (2016) studied a 2E-MTVRPTWSS based on a two-echelon distribution system without considering uncertainty. The problem considers a hierarchical objective which minimizes the number of first-echelon vehicles used (primary criterion), the number of second-echelon vehicles used (secondary criterion), and the total travel distance (tertiary criterion). Thus, we slightly modify the VNS-based metaheuristic to solve all instances in Grangier et al. (2016). Note that each test instance is tested ten times with the same parameter settings introduced in Section 5.4.1 for the VNS-based metaheuristic. The best solution among ten runs is then selected for each instance and it is compared with the corresponding best solution generated by the ALNS algorithm among ten runs. The best results of the VNS-based metaheuristic in comparison to the ALNS algorithm on the 2E-MTVRPTWSS instances with 100 customers and 8 satellites in Grangier et al. (2016) are shown in Table 5.10.

In Table 5.10, the columns under "ALNS" and "VNS-based metaheuristic" respectively summarize the detailed results of the best solutions generated by the ALNS algorithm and the VNS-based metaheuristic. The columns under "Gap" show the differences between the best solutions generated by the two methods for each instance in terms of the number of first-echelon vehicles used (NVFE), the number of second-echelon vehicles used (NVSE), and the travel distance (TD) in percentage. As shown in 5.10, the VNS-based metaheuristic finds better solutions for 24 out of 56 test instances and they are highlighted in bold in the columns under "Gap". Specifically, the metaheuristic finds the new best solutions with the same number of first-echelon vehicles and one less second-echelon vehicle for 8 instances. In addition, it identifies the new best solutions with the same number of first- and second-echelon vehicles and shorter travel distance for 16 instances. Thus, the performance of the VNS-based metaheuristic is comparable to the effective ALNS algorithm proposed for the 2E-MTVRPTWSS in Grangier et al. (2016).

Table 5.10. Best results of the VNS-based metaheuristic in comparison to the ALNS on the 2E-MTVRPTWSS instances in Grangier et al. (2016).

Instance	ALNS				VNS-based metaheuristic				Gap		
	NV FE	NV SE	TD	Time (min)	NV FE	NV SE	TD	Time (min)	NV FE	NV SE	TD (%)
C101	3	11	2022.4	35.8	3	11	2033.9	77.6	0	0	0.57
C102	3	10	1947.6	99.6	3	9	2178.9	91.8	0	**-1**	11.88
C103	3	9	1880.7	64.6	3	9	1962.3	40.4	0	0	4.34
C104	3	9	1811.1	58.4	3	9	1851.7	88.3	0	0	2.24
C105	3	10	1934.0	59.9	3	10	1926.2	93.9	0	0	**-0.40**
C106	3	10	1945.0	51.0	3	10	1957.0	111.9	0	0	0.62
C107	3	10	1888.9	41.3	3	10	1898.2	61.8	0	0	0.49
C108	3	10	1875.3	32.2	3	10	1921.3	35.1	0	0	2.45
C109	3	9	1863.1	52.6	3	9	1922.9	37.8	0	0	3.21
C201	2	3	1389.3	53.3	2	3	1344.0	120.0	0	0	**-3.26**
C202	2	3	1305.0	50.2	2	3	1291.9	71.2	0	0	**-1.00**
C203	2	3	1272.7	66.1	2	3	1262.4	67.1	0	0	**-0.81**
C204	2	3	1237.9	64.1	2	3	1258.0	95.5	0	0	1.62
C205	2	3	1312.1	35.9	2	3	1305.1	70.4	0	0	**-0.53**
C206	2	3	1312.6	36.4	2	3	1297.6	69.1	0	0	**-1.14**
C207	2	3	1280.4	42.9	2	3	1278.4	64.8	0	0	**-0.16**
C208	2	3	1278.3	40.4	2	3	1303.8	67.3	0	0	1.99

Table 5.10 Continued.

Instance	ALNS				VNS-based metaheuristic				Gap		
	NV FE	NV SE	TD	Time (min)	NV FE	NV SE	TD	Time (min)	NV FE	NV SE	TD (%)
R101	2	19	2333.5	48.3	2	19	2321.9	35.2	0	0	**-0.50**
R102	2	18	2136.8	52.6	2	17	2122.3	56.5	0	**-1**	**-0.68**
R103	2	13	1942.7	62.4	2	13	1943.7	111.9	0	0	0.05
R104	2	10	1777.2	86.4	2	10	1761.5	42.0	0	0	**-0.88**
R105	2	14	2096.8	40.8	2	14	2094.0	111.4	0	0	**-0.13**
R106	2	12	1992.4	45.1	2	12	2030.7	120.0	0	0	1.92
R107	2	11	1779.2	37.0	2	11	1837.1	41.4	0	0	3.25
R108	2	10	1654.3	49.4	2	10	1718.0	30.3	0	0	3.85
R109	2	12	1925.9	36.5	2	12	1945.4	67.9	0	0	1.01
R110	2	12	1833.6	69.3	2	11	1960.1	120.0	0	**-1**	6.90
R111	2	12	1770.8	67.9	2	12	1822.4	28.7	0	0	2.91
R112	2	11	1746.0	93.8	2	11	1777.9	26.6	0	0	1.83
R201	1	4	1587.8	33.4	1	4	1581.0	24.4	0	0	**-0.43**
R202	1	3	1530.8	79.4	1	3	1542.6	117.7	0	0	0.77
R203	1	3	1255.1	62.7	1	3	1269.8	58.8	0	0	1.17
R204	1	2	1191.7	57.4	1	2	1184.4	75.0	0	0	**-0.61**
R205	1	3	1319.1	39.6	1	3	1320.1	29.2	0	0	0.08
R206	1	3	1228.3	58.9	1	3	1233.8	59.1	0	0	0.45
R207	1	3	1140.2	54.4	1	2	1256.8	120.0	0	**-1**	10.23
R208	1	2	1050.2	55.5	1	2	1055.7	41.1	0	0	0.52
R209	1	3	1258.7	54.6	1	3	1255.8	42.1	0	0	**-0.23**
R210	1	3	1279.8	54.9	1	3	1308.1	46.8	0	0	2.21
R211	1	3	1118.2	51.8	1	2	1400.5	120.0	0	**-1**	25.25
RC101	3	16	2577.0	45.4	3	16	2579.3	60.8	0	0	0.09
RC102	3	14	2407.1	63.0	3	14	2474.7	93.9	0	0	2.81
RC103	3	11	2476.9	121.3	3	12	2290.0	49.1	0	1	-7.55
RC104	3	11	2125.9	81.1	3	10	2228.1	118.7	0	**-1**	4.81
RC105	3	15	2542.6	58.9	3	15	2575.9	120.0	0	0	1.31
RC106	3	13	2494.9	83.8	3	13	2462.2	120.0	0	0	**-1.31**
RC107	3	13	2271.1	78.4	3	12	2360.2	109.1	0	**-1**	3.92
RC108	3	12	2202.9	94.1	3	12	2219.5	38.5	0	0	0.75
RC201	1	4	1787.6	33.6	1	4	1797.1	27.1	0	0	0.53
RC202	1	4	1513.8	105.0	1	3	1769.0	120.0	0	**-1**	16.86
RC203	1	3	1416.2	53.4	1	3	1427.1	53.5	0	0	0.77
RC204	1	3	1188.2	51.4	1	3	1225.1	109.9	0	0	3.11
RC205	1	4	1693.7	36.0	1	4	1672.4	31.4	0	0	**-1.26**
RC206	1	3	1583.1	39.6	1	3	1579.6	24.9	0	0	**-0.22**
RC207	1	3	1449.8	61.3	1	3	1481.3	32.7	0	0	2.17
RC208	1	3	1257.3	61.9	1	3	1271.9	45.8	0	0	1.16

5.5 Summary

In this chapter, the 2E-MTVRPTWSS under customer demand uncertainty is studied. A number of practical features are considered in the problem including a two-echelon transportation system, time windows, multiple trips, vehicle synchronization, and customer demand uncertainty. The demand uncertainty is captured by a novel uncertainty set with route-dependent uncertainty polytopes which are defined based on vehicle routes in two echelons. We present a robust mathematical formulation with the defined uncertainty set to model the problem. We also design a three-phase metaheuristic solution approach based on the classical VNS framework. Extensive numerical experiments are conducted on the adapted benchmark instances. Both deterministic and robust solutions are derived for the adapted medium- and large-sized instances. The obtained solutions are further evaluated using Monte Carlo simulation tests. The computational results show that the robust solutions are very reliable and generally incur little unfulfilled customer demand under uncertainty. Moreover, a comprehensive analysis of the computational results is performed to generate useful routing suggestions for practical 2E-MTVRPTWSS applications under customer demand uncertainty. The performance of the VNS-based metaheuristic is demonstrated by comparing it to the effective ALNS algorithm proposed for the 2E-MTVRPTWSS in Grangier et al. (2016). The comparison results show that the VNS-based metaheuristic is comparable to the ALNS algorithm in terms of solution quality and computational time.

Chapter 6

Conclusion and Further Study

This final chapter summarizes the whole research, discusses the main findings, and identifies directions for future research.

6.1 Conclusions

Given the substantial cost savings by addressing the CVRP and its various extensions in the academic field, many logistics and transportation companies have become interested in optimizing real-life routing problems through the modelling and solution techniques developed in the VRP literature. In the CVRP and most of its extensions, a common assumption is that the values of all problem parameters are readily available and can be precisely known in advance. However, this assumption does not always hold in many real-life applications due to uncertainty. Uncertainty can be caused by different factors, such as imprecise information on customer demands, unfixed service times for customers, and varying travel times for vehicles. In the existing literature, research on various extensions of the CVRP under uncertainty is still limited due to the difficulty of properly modelling uncertainty and the complexity of solving these extensions with large-sized instances. In this thesis, we thus have studied three important extensions of the CVRP

- the VRPTW, the VRPSPDTW, and the 2E-MTVRPTWSS - with the consideration of uncertainty. Specifically, we have focused on defining novel uncertainty sets to capture different types of uncertainty and developing mathematical formulations to model the studied problems based on the robust optimization paradigm. Although researchers have adopted robust optimization to address the CVRP and some of its extensions considering uncertainty over the last decade, most of the solution methods developed for these problems are exact algorithms which can handle only small- or medium-sized instances. Thus, we have designed effective and efficient metaheuristics to solve large-sized instances in this thesis. The proposed mathematical formulations and metaheuristics are of both theoretical and practical significance. They can not only enrich the current literature on modelling and solving different VRPs under uncertainty, but also help logistics practitioners generate reliable and cost-effective routing strategies for real-life routing problems. The main conclusions of this research are summarized hereafter.

In Chapter 3, we have studied the VRPTW with the consideration of uncertainty in customer demands, service times, and travel times. Unlike most of the previous studies on the VRPTW under uncertainty, we have simultaneously considered three different types of uncertainty. To capture these three types of uncertainty, we have defined novel route-dependent uncertainty sets. Based on such sets, we have modelled the problem with a robust mathematical formulation and solved it by using an AVNS-based metaheuristic. We have conducted extensive numerical experiments which adopt the well-known Solomon's benchmark instances (Solomon, 1987). Both deterministic and robust solutions have been generated for the medium-sized instances with 50 customers and the large-sized instances with 100 customers. The computational results show that robust solutions can be derived for both medium- and large-sized instances within a reasonable running time. The performance of the AVNS-based metaheuristic has been further demonstrated by comparing the best deterministic solutions found by the metaheuristic with the current best-known solutions for the Solomon's instances. The comparison results show that the

AVNS-based metaheuristic is comparable to the state-of-the-art heuristic solution methods proposed for the standard VRPTW in the literature. In addition, Monte Carlo simulation tests have been designed to assess the robustness of all obtained solutions. A detailed analysis of the computational results has been performed and useful managerial insights have been derived for real-life VRPTW applications under uncertainty.

In Chapter 4, we have studied the VRPSPDTW under uncertainty. To the best of our knowledge, little literature has focused on the VRPSPDTW under multiple types of uncertainty. Thus, we have extended the VRPSPDTW by simultaneously considering uncertainty in pickup demands and travel times. Two route-dependent uncertainty sets have been defined to capture these two types of uncertainty. Based on such sets, we have modelled the problem with a robust mathematical formulation. Due to the complexity of solving the formulation with exact algorithms, an ALNS-based metaheuristic has been proposed, which can efficiently solve the problem with large-sized instances. We have employed the benchmark instances of Wang and Chen (2012) for the standard VRPSPDTW and conducted extensive numerical experiments. The computational results show that both deterministic and robust solutions can be generated for the medium-sized instances with 50 customers and the large-sized instances with 100 customers within a reasonable running time. In addition, the performance of the ALNS-based metaheuristic has been demonstrated by comparing the best deterministic solutions generated by the metaheuristic with the current best-known solutions for the benchmark instances of Wang and Chen (2012) with 100 customers. The ALNS-based metaheuristic has identified the current best-known solutions for more than half of the benchmark instances and even generated new best solutions for one fourth of them. We have designed Monte Carlo simulation tests to evaluate the robustness of the obtained solutions in the computational experiments. In addition, we have performed a comprehensive analysis of the computational results to generate useful routing tips for practical VRPSPDTW applications under pickup demand and travel time uncertainty.

In Chapter 5, we have studied the 2E-MTVRPTWSS under uncertainty. A number of practical features have been considered in the problem including a two-echelon transportation system, time windows, multiple trips, vehicle synchronization, and uncertainty in customer demands. To the best of our knowledge, few studies have extended the basic 2E-CVRP by simultaneously considering all these features. To capture customer demand uncertainty, a novel uncertainty set has been defined based on vehicle routes in two echelons. The problem has been modelled by a robust mathematical formulation with the defined uncertainty set and solved via a VNS-based metaheuristic. We have conducted extensive numerical experiments on the adapted benchmark instances to illustrate the effectiveness of the robust mathematical formulation and the VNS-based metaheuristic. Both deterministic and robust solutions have been derived for the medium-sized instances with 50 customers and 8 satellites and the large-sized instances with 100 customers and 8 satellites. In addition, Monte Carlo simulation tests have been designed to further assess the robustness of the obtained solutions. The computational results have been comprehensively analysed to highlight the features of proposed methods and to derive routing suggestions for real-life 2E-MTVRPTWSS under customer demand uncertainty. The performance of the VNS-based metaheuristic has been demonstrated through comparison with the effective ALNS algorithm proposed for the 2E-MTVRPTWSS in Grangier et al. (2016). The comparison results show that the VNS-based metaheuristic is comparable to the ALNS algorithm in terms of solution quality and computational time.

6.2 Further Study

Notwithstanding the demonstrable effectiveness of our proposed methodologies, several directions for research are worth further pursuit.

Firstly, uncertainty sets are critical components of the robust optimization framework. In using such a framework to address VRPs under uncertainty, the construction of practical and effective uncertainty sets to capture different types of uncertainty will continue to be an interesting research direction. In this era of data exploration, historical data for important parameters such as customer demands and service times may be available in some practical logistics activities. Thus, it will be interesting to construct uncertainty sets based on promising data-driven approaches for reducing the level of conservatism in some robust routing strategies derived for real-life VRP applications.

Secondly, the existing literature is still limited on modelling and solving different extensions of the CVRP under uncertainty. Thus, there is a need to study more complex extensions of the CVRP with the consideration of multiple types of uncertainty. For example, it will be interesting to extend the 2E-MTVRPTWSS under demand uncertainty in Chapter 5 by incorporating uncertainty in service times and in travel times. However, since solving complex extensions of the CVRP without considering uncertainty is itself challenging, addressing these problems under multiple types of uncertainty will be even more difficult. Thus, efficient exact and metaheuristic solution approaches for complex VRPs under uncertainty are worth studying.

Finally, it will be worthwhile to adapt the proposed mathematical formulations and the designed metaheuristics in addressing real-life routing problems in daily logistics and transportation activities such as parcel delivery, press and grocery distribution, municipal solid waste collection, and home healthcare services. Adapting the proposed methodologies to address real-world applications can further verify their effectiveness and generate more practical managerial insights for the logistics and transportation industry.

Bibliography

Agra, A., Christiansen, M., Figueiredo, R., Hvattum, L. M., Poss, M., and Requejo, C. (2012). Layered formulation for the robust vehicle routing problem with time windows. In *International Symposium on Combinatorial Optimization*, pages 249–260. Springer.

Agra, A., Christiansen, M., Figueiredo, R., Hvattum, L. M., Poss, M., and Requejo, C. (2013). The robust vehicle routing problem with time windows. *Computers & Operations Research*, 40(3):856–866.

Ai, T. J. and Kachitvichyanukul, V. (2009a). Particle swarm optimization and two solution representations for solving the capacitated vehicle routing problem. *Computers & Industrial Engineering*, 56(1):380–387.

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

Anderluh, A., Hemmelmayr, V. C., and Nolz, P. C. (2016). Synchronizing vans and cargo bikes in a city distribution network. *Central European Journal of Operations Research*, pages 1–32.

Anderluh, A., Larsen, R., Hemmelmayr, V. C., and Nolz, P. C. (2019). Impact of travel time uncertainties on the solution cost of a two-echelon vehicle routing problem with synchronization. *Flexible Services and Manufacturing Journal*, pages 1–23.

Angelelli, E. and Mansini, R. (2002). The vehicle routing problem with time windows and simultaneous pick-up and delivery. In *Quantitative Approaches to Distribution Logistics and Supply Chain Management*, pages 249–267. Springer.

Augerat, P. (1995). *Approche polyèdrale du problème de tournées de véhicules*. PhD thesis, Institut National Polytechnique de Grenoble-INPG.

Australian Logistics Council (2014). The economic significance of the australian logistics industry. http://austlogistics.com.au/wp-content/uploads/2014/07/Economic-Significance-of-the-Australian-Logistics-Indsutry-FINAL.pdf.

Avci, M. and Topaloglu, S. (2015). An adaptive local search algorithm for vehicle routing problem with simultaneous and mixed pickups and deliveries. *Computers & Industrial Engineering*, 83:15–29.

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

Baldacci, R., Bartolini, E., Mingozzi, A., and Roberti, R. (2010). An exact solution framework for a broad class of vehicle routing problems. *Computational Management Science*, 7(3):229–268.

Baldacci, R., Christofides, N., and Mingozzi, A. (2008). An exact algorithm for the vehicle routing problem based on the set partitioning formulation with additional cuts. *Mathematical Programming*, 115(2):351–385.

Baldacci, R., Hadjiconstantinou, E., and Mingozzi, A. (2004). An exact algorithm for the capacitated vehicle routing problem based on a two-commodity network flow formulation. *Operations Research*, 52(5):723–738.

Baldacci, R., Mingozzi, A., and Roberti, R. (2011). New route relaxation and pricing strategies for the vehicle routing problem. *Operations Research*, 59(5):1269–1283.

Baldacci, R., Mingozzi, A., and Roberti, R. (2012). Recent exact algorithms for solving the vehicle routing problem under capacity and time window constraints. *European Journal of Operational Research*, 218(1):1–6.

Baldacci, R., Mingozzi, A., Roberti, R., and Calvo, R. W. (2013). An exact algorithm for the two-echelon capacitated vehicle routing problem. *Operations Research*, 61(2):298–314.

Bard, J. F., Kontoravdis, G., and Yu, G. (2002). A branch-and-cut procedure for the vehicle routing problem with time windows. *Transportation Science*, 36(2):250–269.

Bartolini, E., Goeke, D., Schneider, M., and Ye, M. (2021). The robust traveling salesman problem with time windows under knapsack-constrained travel time uncertainty. *Transportation Science*, 55(2):371–394.

Ben-Tal, A., El Ghaoui, L., and Nemirovski, A. (2009). *Robust optimization*. Princeton University Press.

Ben-Tal, A., Goryashko, A., Guslitzer, E., and Nemirovski, A. (2004). Adjustable robust solutions of uncertain linear programs. *Mathematical Programming*, 99(2):351–376.

Benjamin, A. M. and Beasley, J. (2013). Metaheuristics with disposal facility positioning for the waste collection VRP with time windows. *Optimization Letters*, 7(7):1433–1449.

Bent, R. and Van Hentenryck, P. (2004). A two-stage hybrid local search for the vehicle routing problem with time windows. *Transportation Science*, 38(4):515–530.

Berger, J. and Barkaoui, M. (2004). A parallel hybrid genetic algorithm for the vehicle routing problem with time windows. *Computers & Operations Research*, 31(12):2037–2053.

Bertsimas, D., Brown, D. B., and Caramanis, C. (2011). Theory and applications of robust optimization. *SIAM Review*, 53(3):464–501.

Bertsimas, D. and Sim, M. (2003). Robust discrete optimization and network flows. *Mathematical Programming*, 98(1-3):49–71.

Bertsimas, D. and Sim, M. (2004). The price of robustness. *Operations Research*, 52(1):35–53.

Bianchessi, N. and Righini, G. (2007). Heuristic algorithms for the vehicle routing problem with simultaneous pick-up and delivery. *Computers & Operations Research*, 34(2):578–594.

Birge, J. R. and Louveaux, F. (2011). *Introduction to stochastic programming*. Springer Science & Business Media.

Braaten, S., Gjønnes, O., Hvattum, L. M., and Tirado, G. (2017). Heuristics for the robust vehicle routing problem with time windows. *Expert Systems with Applications*, 77:136–147.

Braekers, K., Ramaekers, K., and Van Nieuwenhuyse, I. (2016). The vehicle routing problem: State of the art classification and review. *Computers & Industrial Engineering*, 99:300–313.

Bräysy, O. (2003). A reactive variable neighborhood search for the vehicle-routing problem with time windows. *INFORMS Journal on Computing*, 15(4):347–368.

Bräysy, O. and Gendreau, M. (2005a). Vehicle routing problem with time windows, Part I: Route construction and local search algorithms. *Transportation Science*, 39(1):104–118.

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

Breunig, U., Schmid, V., Hartl, R. F., and Vidal, T. (2016). A large neighbourhood based heuristic for two-echelon routing problems. *Computers & Operations Research*, 76:208–225.

Bullnheimer, B., Hartl, R. F., and Strauss, C. (1999). An improved ant system algorithm for the vehicle routing problem. *Annals of Operations Research*, 89:319–328.

Çatay, B. (2010). A new saving-based ant algorithm for the vehicle routing problem with simultaneous pickup and delivery. *Expert Systems with Applications*, 37(10):6809–6817.

Chabrier, A. (2006). Vehicle routing problem with elementary shortest path based column generation. *Computers & Operations Research*, 33(10):2972–2990.

Chen, J.-F. and Wu, T.-H. (2006). Vehicle routing problem with simultaneous deliveries and pickups. *Journal of the Operational Research Society*, 57(5):579–587.

Chen, P., Huang, H.-k., and Dong, X.-Y. (2010). Iterated variable neighborhood descent algorithm for the capacitated vehicle routing problem. *Expert Systems with Applications*, 37(2):1620–1627.

Chiang, W.-C. and Russell, R. A. (1996). Simulated annealing metaheuristics for the vehicle routing problem with time windows. *Annals of Operations Research*, 63(1):3–27.

Chiang, W.-C. and Russell, R. A. (1997). A reactive tabu search metaheuristic for the vehicle routing problem with time windows. *INFORMS Journal on Computing*, 9(4):417–430.

Christofides, N. and Eilon, S. (1969). An algorithm for the vehicle-dispatching problem. *Journal of the Operational Research Society*, 20(3):309–318.

Christofides, N., Mingozzi, A., and Toth, P. (1981). Exact algorithms for the vehicle routing problem, based on spanning tree and shortest path relaxations. *Mathematical Programming*, 20(1):255–282.

Clarke, G. and Wright, J. W. (1964). Scheduling of vehicles from a central depot to a number of delivery points. *Operations Research*, 12(4):568–581.

Cordeau, J.-F., Laporte, G., and Mercier, A. (2001). A unified tabu search heuristic for vehicle routing problems with time windows. *Journal of the Operational Research Society*, 52(8):928–936.

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

Crainic, T., Mancini, S., Perboli, G., and Tadei, R. (2013). GRASP with path relinking for the two-echelon vehicle routing problem. In *Advances in Metaheuristics*, pages 113–125. Springer.

Crainic, T. G., Errico, F., Rei, W., and Ricciardi, N. (2015). Modeling demand uncertainty in two-tier city logistics tactical planning. *Transportation Science*, 50(2):559–578.

Crainic, T. G., Mancini, S., Perboli, G., and Tadei, R. (2008). Clustering-based heuristics for the two-echelon vehicle routing problem. *Montreal, Canada, Interuniversity Research Centre on Enterprise Networks, Logistics and Transportation*, 17:26.

Crainic, T. G., Mancini, S., Perboli, G., and Tadei, R. (2011). Multi-start heuristics for the two-echelon vehicle routing problem. In *European Conference on Evolutionary Computation in Combinatorial Optimization*, pages 179–190. Springer.

Crainic, T. G., Ricciardi, N., and Storchi, G. (2004). Advanced freight transportation systems for congested urban areas. *Transportation Research Part C: Emerging Technologies*, 12(2):119–137.

Crainic, T. G., Ricciardi, N., and Storchi, G. (2009). Models for evaluating and planning city logistics systems. *Transportation Science*, 43(4):432–454.

Cuda, R., Guastaroba, G., and Speranza, M. G. (2015). A survey on two-echelon routing problems. *Computers & Operations Research*, 55:185–199.

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

De La Vega, J., Munari, P., and Morabito, R. (2020). Exact approaches to the robust vehicle routing problem with time windows and multiple deliverymen. *Computers & Operations Research*, 124:105062.

Dellaert, N., Dashty Saridarq, F., Van Woensel, T., and Crainic, T. G. (2019). Branch-and-price–based algorithms for the two-echelon vehicle routing problem with time windows. *Transportation Science*, 53(2):463–479.

Dell'Amico, M., Righini, G., and Salani, M. (2006). A branch-and-price approach to the vehicle routing problem with simultaneous distribution and collection. *Transportation Science*, 40(2):235–247.

Desaulniers, G., Desrosiers, J., and Spoorendonk, S. (2010). The vehicle routing problem with time windows: State-of-the-art exact solution methods. *Wiley Encyclopedia of Operations Research and Management Science*.

Desaulniers, G., Madsen, O. B., and Ropke, S. (2014). Chapter 5: The vehicle routing problem with time windows. In *Vehicle Routing: Problems, Methods, and Applications, Second Edition*, pages 119–159. SIAM.

Desrochers, M., Desrosiers, J., and Solomon, M. (1992). A new optimization algorithm for the vehicle routing problem with time windows. *Operations Research*, 40(2):342–354.

Dethloff, J. (2001). Vehicle routing and reverse logistics: the vehicle routing problem with simultaneous delivery and pick-up. *OR-Spektrum*, 23(1):79–96.

Ehmke, J. F., Campbell, A. M., and Urban, T. L. (2015). Ensuring service levels in routing problems with time windows and stochastic travel times. *European Journal of Operational Research*, 240(2):539–550.

Errico, F., Desaulniers, G., Gendreau, M., Rei, W., and Rousseau, L.-M. (2016). A priori optimization with recourse for the vehicle routing problem with hard time windows and stochastic service times. *European Journal of Operational Research*, 249(1):55–66.

Errico, F., Desaulniers, G., Gendreau, M., Rei, W., and Rousseau, L.-M. (2018). The vehicle routing problem with hard time windows and stochastic service times. *EURO Journal on Transportation and Logistics*, 7(3):223–251.

Feillet, D., Dejax, P., Gendreau, M., and Gueguen, C. (2004). An exact algorithm for the elementary shortest path problem with resource constraints: Application to some vehicle routing problems. *Networks: An International Journal*, 44(3):216–229.

Feillet, D., Gendreau, M., and Rousseau, L.-M. (2007). New refinements for the solution of vehicle routing problems with branch and price. *INFOR: Information Systems and Operational Research*, 45(4):239–256.

Ferrier Hodgson (2014). Transport and logistics insight 2014. https://www.ferrierhodgson.com/au/-/media/ferrier/files/documents/publications/transport-and-logistics/transport-and-logistics-insights--january-2014.pdf.

Ferrier Hodgson (2017). Transport 2050: Lookout! Here comes the future. https://www.ferrierhodgson.com/au/-/media/ferrier/files/documents/publications/transport-and-logistics/2017/lookout-here-comes-the-future.pdf.

Fisher, M. L. (1994). Optimal solution of vehicle routing problems using minimum k-trees. *Operations Research*, 42(4):626–642.

Fukasawa, R., Longo, H., Lysgaard, J., De Aragão, M. P., Reis, M., Uchoa, E., and Werneck, R. F. (2006). Robust branch-and-cut-and-price for the capacitated vehicle routing problem. *Mathematical Programming*, 106(3):491–511.

Gabrel, V., Murat, C., and Thiele, A. (2014). Recent advances in robust optimization: An overview. *European Journal of Operational Research*, 235(3):471–483.

Gajpal, Y. and Abad, P. (2009). An ant colony system (ACS) for vehicle routing problem with simultaneous delivery and pickup. *Computers & Operations Research*, 36(12):3215–3223.

Gambardella, L. M., Taillard, É., and Agazzi, G. (1999). MACS-VRPTW: A multiple colony system for vehicle routing problems with time windows. In *New Ideas in Optimization*.

Gan, X., Wang, Y., Li, S., and Niu, B. (2012). Vehicle routing problem with time windows and simultaneous delivery and pick-up service based on MCPSO. *Mathematical Problems in Engineering*, 2012.

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

Gendreau, M., Hertz, A., and Laporte, G. (1992). New insertion and postoptimization procedures for the traveling salesman problem. *Operations Research*, 40(6):1086–1094.

Gendreau, M., Hertz, A., and Laporte, G. (1994). A tabu search heuristic for the vehicle routing problem. *Management Science*, 40(10):1276–1290.

Gendreau, M., Jabali, O., and Rei, W. (2014). Chapter 8: Stochastic vehicle routing problems. In *Vehicle Routing: Problems, Methods, and Applications, Second Edition*, pages 213–239. SIAM.

Gendreau, M., Jabali, O., and Rei, W. (2016). 50th anniversary invited article—future research directions in stochastic vehicle routing. *Transportation Science*, 50(4):1163–1173.

Gendreau, M. and Tarantilis, C. D. (2010). Solving large-scale vehicle routing problems with time windows: The state-of-the-art. Technical report, Cirrelt Montréal, Canada.

Gillett, B. E. and Miller, L. R. (1974). A heuristic algorithm for the vehicle-dispatch problem. *Operations Research*, 22(2):340–349.

Glover, F. (1992). New ejection chain and alternating path methods for traveling salesman problems. In *Computer Science and Operations Research*, pages 491–509. Elsevier.

Gonzalez-Feliu, J. (2008). *Models and methods for the city logistics: The two-echelon capacitated vehicle routing problem*. PhD thesis.

Gounaris, C. E., Wiesemann, W., and Floudas, C. A. (2013). The robust capacitated vehicle routing problem under demand uncertainty. *Operations Research*, 61(3):677–693.

Grangier, P., Gendreau, M., Lehuédé, F., and Rousseau, L.-M. (2016). An adaptive large neighborhood search for the two-echelon multiple-trip vehicle routing problem with satellite synchronization. *European Journal of Operational Research*, 254(1):80–91.

Groër, C., Golden, B., and Wasil, E. (2011). A parallel algorithm for the vehicle routing problem. *INFORMS Journal on Computing*, 23(2):315–330.

Hashimoto, H. and Yagiura, M. (2008). A path relinking approach with an adaptive mechanism to control parameters for the vehicle routing problem with time windows.

In *European Conference on Evolutionary Computation in Combinatorial Optimization*, pages 254–265. Springer.

Hashimoto, H., Yagiura, M., and Ibaraki, T. (2008). An iterated local search algorithm for the time-dependent vehicle routing problem with time windows. *Discrete Optimization*, 5(2):434–456.

Hemmelmayr, V. C., Cordeau, J.-F., and Crainic, T. G. (2012). An adaptive large neighborhood search heuristic for two-echelon vehicle routing problems arising in city logistics. *Computers & Operations Research*, 39(12):3215–3228.

Hiermann, G., Puchinger, J., Ropke, S., and Hartl, R. F. (2016). The electric fleet size and mix vehicle routing problem with time windows and recharging stations. *European Journal of Operational Research*, 252(3):995–1018.

Ho, S. C. and Gendreau, M. (2006). Path relinking for the vehicle routing problem. *Journal of Heuristics*, 12(1-2):55–72.

Ho, W.-K., Ang, J. C., and Lim, A. (2001). A hybrid search alogrithm for the vehicle routing problem with time windows. *International Journal on Artificial Intelligence Tools*, 10(03):431–449.

Hof, J. and Schneider, M. (2019). An adaptive large neighborhood search with path relinking for a class of vehicle-routing problems with simultaneous pickup and delivery. *Networks*, 74(3):207–250.

Hof, J., Schneider, M., and Goeke, D. (2017). Solving the battery swap station location-routing problem with capacitated electric vehicles using an AVNS algorithm for vehicle-routing problems with intermediate stops. *Transportation Research Part B: Methodological*, 97:102–112.

Homberger, J. and Gehring, H. (1999). Two evolutionary metaheuristics for the vehicle routing problem with time windows. *INFOR: Information Systems and Operational Research*, 37(3):297–318.

Hou, L. and Zhou, H. (2010). Stochastic vehicle routing problem with uncertain demand and travel time and simultaneous pickups and deliveries. In *2010 Third International Joint Conference on Computational Science and Optimization*, volume 1, pages 32–35. IEEE.

Hu, C., Lu, J., Liu, X., and Zhang, G. (2018). Robust vehicle routing problem with hard time windows under demand and travel time uncertainty. *Computers & Operations Research*, 94:139–153.

Ibaraki, T., Imahori, S., Kubo, M., Masuda, T., Uno, T., and Yagiura, M. (2005). Effective local search algorithms for routing and scheduling problems with general time-window constraints. *Transportation Science*, 39(2):206–232.

Ioannou, G., Kritikos, M., and Prastacos, G. (2001). A greedy look-ahead heuristic for the vehicle routing problem with time windows. *Journal of the Operational Research Society*, 52(5):523–537.

Jacobsen, S. K. and Madsen, O. B. (1980). A comparative study of heuristics for a two-level routing-location problem. *European Journal of Operational Research*, 5(6):378–387.

Jepsen, M., Petersen, B., Spoorendonk, S., and Pisinger, D. (2008). Subset-row inequalities applied to the vehicle-routing problem with time windows. *Operations Research*, 56(2):497–511.

Jepsen, M., Spoorendonk, S., and Ropke, S. (2013). A branch-and-cut algorithm for the symmetric two-echelon capacitated vehicle routing problem. *Transportation Science*, 47(1):23–37.

Kalayci, C. B. and Kaya, C. (2016). An ant colony system empowered variable neighborhood search algorithm for the vehicle routing problem with simultaneous pickup and delivery. *Expert Systems with Applications*, 66:163–175.

Kassem, S. and Chen, M. (2013). Solving reverse logistics vehicle routing problems with time windows. *The International Journal of Advanced Manufacturing Technology*, 68(1-4):57–68.

Knight, K. and Hofer, J. (1968). Vehicle scheduling with timed and connected calls: A case study. *Journal of the Operational Research Society*, 19(3):299–310.

Koç, Ç., Laporte, G., and Tükenmez, İ. (2020). A review of vehicle routing with simultaneous pickup and delivery. *Computers & Operations Research*, 122:104987.

Kohl, N., Desrosiers, J., Madsen, O. B., Solomon, M. M., and Soumis, F. (1999). 2-path cuts for the vehicle routing problem with time windows. *Transportation Science*, 33(1):101–116.

Kytöjoki, J., Nuortio, T., Bräysy, O., and Gendreau, M. (2007). An efficient variable neighborhood search heuristic for very large scale vehicle routing problems. *Computers & Operations Research*, 34(9):2743–2757.

Labadie, N., Prins, C., Prodhon, C., Monmarché, N., and Siarry, P. (2016). *Metaheuristics for vehicle routing problems*. Wiley Online Library.

Laporte, G., Mercure, H., and Nobert, Y. (1986). An exact algorithm for the asymmetrical capacitated vehicle routing problem. *Networks*, 16(1):33–46.

Laporte, G., Nobert, Y., and Desrochers, M. (1985). Optimal routing under capacity and distance restrictions. *Operations Research*, 33(5):1050–1073.

Laporte, G., Ropke, S., and Vidal, T. (2014). Chapter 4: Heuristics for the vehicle routing problem. In *Vehicle Routing: Problems, Methods, and Applications, Second Edition*, pages 87–116. SIAM.

Laporte, G. and Semet, F. (2002). Classical heuristics for the capacitated VRP. In *The Vehicle Routing Problem*, pages 109–128. SIAM.

Lee, C., Lee, K., and Park, S. (2012). Robust vehicle routing problem with deadlines and travel time/demand uncertainty. *Journal of the Operational Research Society*, 63(9):1294–1306.

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

Li, H., Liu, Y., Jian, X., and Lu, Y. (2018). The two-echelon distribution system considering the real-time transshipment capacity varying. *Transportation Research Part B: Methodological*, 110:239–260.

Li, H., Wang, H., Chen, J., and Bai, M. (2020). Two-echelon vehicle routing problem with time windows and mobile satellites. *Transportation Research Part B: Methodological*, 138:179–201.

Li, H., Yuan, J., Lv, T., and Chang, X. (2016a). The two-echelon time-constrained vehicle routing problem in linehaul-delivery systems considering carbon dioxide emissions. *Transportation Research Part D: Transport and Environment*, 49:231–245.

Li, H., Zhang, L., Lv, T., and Chang, X. (2016b). The two-echelon time-constrained vehicle routing problem in linehaul-delivery systems. *Transportation Research Part B: Methodological*, 94:169–188.

Li, X., Tian, P., and Leung, S. C. (2010). Vehicle routing problems with time windows and stochastic travel and service times: Models and algorithm. *International Journal of Production Economics*, 125(1):137–145.

Lin, S. (1965). Computer solutions of the traveling salesman problem. *The Bell System Technical Journal*, 44(10):2245–2269.

Lin, S.-W., Lee, Z.-J., Ying, K.-C., and Lee, C.-Y. (2009). Applying hybrid meta-heuristics for capacitated vehicle routing problem. *Expert Systems with Applications*, 36(2):1505–1512.

Liu, R., Tao, Y., Hu, Q., and Xie, X. (2017). Simulation-based optimisation approach for the stochastic two-echelon logistics problem. *International Journal of Production Research*, 55(1):187–201.

Liu, R., Xie, X., Augusto, V., and Rodriguez, C. (2013). Heuristic algorithms for a vehicle routing problem with simultaneous delivery and pickup and time windows in home health care. *European Journal of Operational Research*, 230(3):475–486.

Lu, D. and Gzara, F. (2019). The robust vehicle routing problem with time windows: Solution by branch and price and cut. *European Journal of Operational Research*, 275(3):925–938.

Lysgaard, J. (2006). Reachability cuts for the vehicle routing problem with time windows. *European Journal of Operational Research*, 175(1):210–223.

Lysgaard, J., Letchford, A. N., and Eglese, R. W. (2004). A new branch-and-cut algorithm for the capacitated vehicle routing problem. *Mathematical Programming*, 100(2):423–445.

Marinakis, Y. and Marinaki, M. (2010). A hybrid genetic–particle swarm optimization algorithm for the vehicle routing problem. *Expert Systems with Applications*, 37(2):1446–1455.

Marinakis, Y., Marinaki, M., and Migdalas, A. (2019). A multi-adaptive particle swarm optimization for the vehicle routing problem with time windows. *Information Sciences*, 481:311–329.

Masson, R., Trentini, A., Lehuédé, F., Malhéné, N., Péton, O., and Tlahig, H. (2017). Optimization of a city logistics transportation system with mixed passengers and goods. *EURO Journal on Transportation and Logistics*, 6(1):81–109.

Mester, D. and Bräysy, O. (2005). Active guided evolution strategies for large-scale vehicle routing problems with time windows. *Computers & Operations Research*, 32(6):1593–1614.

Mester, D. and Bräysy, O. (2007). Active-guided evolution strategies for large-scale capacitated vehicle routing problems. *Computers & Operations Research*, 34(10):2964–2975.

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

Mingyong, L. and Erbao, C. (2010). An improved differential evolution algorithm for vehicle routing problem with simultaneous pickups and deliveries and time windows. *Engineering Applications of Artificial Intelligence*, 23(2):188–195.

Miranda, D. M. and Conceição, S. V. (2016). The vehicle routing problem with hard time windows and stochastic travel and service time. *Expert Systems with Applications*, 64:104–116.

Mladenović, N. and Hansen, P. (1997). Variable neighborhood search. *Computers & Operations Research*, 24(11):1097–1100.

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

Munari, P., Moreno, A., De La Vega, J., Alem, D., Gondzio, J., and Morabito, R. (2019). The robust vehicle routing problem with time windows: compact formulation and branch-price-and-cut method. *Transportation Science*, 53(4):1043–1066.

Nagata, Y. and Bräysy, O. (2009). Edge assembly-based memetic algorithm for the capacitated vehicle routing problem. *Networks: An International Journal*, 54(4):205–215.

Nagata, Y., Bräysy, O., and Dullaert, W. (2010). A penalty-based edge assembly memetic algorithm for the vehicle routing problem with time windows. *Computers & Operations Research*, 37(4):724–737.

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

Or, I. (1976). *Traveling salesman-type combinatorial problems and their relation to the logistics of regional blood banking*. PhD thesis, Northwestern University.

Osman, I. H. (1993). Metastrategy simulated annealing and tabu search algorithms for the vehicle routing problem. *Annals of Operations Research*, 41(4):421–451.

Oyola, J., Arntzen, H., and Woodruff, D. L. (2017). The stochastic vehicle routing problem, a literature review, part II: solution methods. *EURO Journal on Transportation and Logistics*, 6(4):349–388.

Oyola, J., Arntzen, H., and Woodruff, D. L. (2018). The stochastic vehicle routing problem, a literature review, part I: models. *EURO Journal on Transportation and Logistics*, 7(3):193–221.

Pang, K.-W. (2011). An adaptive parallel route construction heuristic for the vehicle routing problem with time windows constraints. *Expert Systems with Applications*, 38(9):11939–11946.

Pecin, D., Contardo, C., Desaulniers, G., and Uchoa, E. (2017a). New enhancements for the exact solution of the vehicle routing problem with time windows. *INFORMS Journal on Computing*, 29(3):489–502.

Pecin, D., Pessoa, A., Poggi, M., and Uchoa, E. (2017b). Improved branch-cut-and-price for capacitated vehicle routing. *Mathematical Programming Computation*, 9(1):61–100.

Pelletier, S., Jabali, O., and Laporte, G. (2019). The electric vehicle routing problem with energy consumption uncertainty. *Transportation Research Part B: Methodological*, 126:225–255.

Perboli, G., Pezzella, F., and Tadei, R. (2008). EVE-OPT: a hybrid algorithm for the capacitated vehicle routing problem. *Mathematical Methods of Operations Research*, 68(2):361.

Perboli, G., Tadei, R., and Fadda, E. (2018). New valid inequalities for the two-echelon capacitated vehicle routing problem. *Electronic Notes in Discrete Mathematics*, 64:75–84.

Perboli, G., Tadei, R., and Vigo, D. (2011). The two-echelon capacitated vehicle routing problem: models and math-based heuristics. *Transportation Science*, 45(3):364–380.

Pessoa, A., De Aragao, M. P., and Uchoa, E. (2008). Robust branch-cut-and-price algorithms for vehicle routing problems. In *The vehicle routing problem: Latest advances and new challenges*, pages 297–325. Springer.

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

Poggi, M. and Uchoa, E. (2014). Chapter 3: New exact algorithms for the capacitated vehicle routing problem. In *Vehicle Routing: Problems, Methods, and Applications, Second Edition*, pages 59–86. SIAM.

Potvin, J.-Y. and Bengio, S. (1996). The vehicle routing problem with time windows part II: genetic search. *INFORMS Journal on Computing*, 8(2):165–172.

Potvin, J.-Y., Kervahut, T., Garcia, B.-L., and Rousseau, J.-M. (1996). The vehicle routing problem with time windows part I: tabu search. *INFORMS Journal on Computing*, 8(2):158–164.

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

Potvin, J.-Y. and Rousseau, J.-M. (1995). An exchange heuristic for routeing problems with time windows. *Journal of the Operational Research Society*, 46(12):1433–1446.

Prescott-Gagnon, E., Desaulniers, G., and Rousseau, L.-M. (2009). A branch-and-price-based large neighborhood search algorithm for the vehicle routing problem with time windows. *Networks: An International Journal*, 54(4):190–204.

Prins, C. (2004). A simple and effective evolutionary algorithm for the vehicle routing problem. *Computers & Operations Research*, 31(12):1985–2002.

Prins, C. (2009). A GRASP $\times$ evolutionary local search hybrid for the vehicle routing problem. In *Bio-inspired Algorithms for the Vehicle Routing Problem*, pages 35–53. Springer.

Pullen, H. and Webb, M. (1967). A computer application to a transport scheduling problem. *The Computer Journal*, 10(1):10–13.

Reimann, M., Doerner, K., and Hartl, R. F. (2004). D-Ants: Savings Based Ants divide and conquer the vehicle routing problem. *Computers & Operations Research*, 31(4):563–591.

Renaud, J., Boctor, F. F., and Laporte, G. (1996). An improved petal heuristic for the vehicle routeing problem. *Journal of the Operational Research Society*, 47(2):329–336.

Rieck, J. and Zimmermann, J. (2013). Exact solutions to the symmetric and asymmetric vehicle routing problem with simultaneous delivery and pick-up. *Business Research*, 6(1):77–92.

Røpke, S. (2012). Branching decisions in branch-and-cut-and-price algorithms for vehicle routing problems. *Presentation in Column Generation*, 2012.

Ropke, S. and Pisinger, D. (2006). An adaptive large neighborhood search heuristic for the pickup and delivery problem with time windows. *Transportation Science*, 40(4):455–472.

Russell, R. and Urban, T. (2008). Vehicle routing with soft time windows and Erlang travel times. *Journal of the Operational Research Society*, 59(9):1220–1228.

Russell, R. A. and Chiang, W.-C. (2006). Scatter search for the vehicle routing problem with time windows. *European Journal of Operational Research*, 169(2):606–622.

Salhi, S. and Nagy, G. (1999). A cluster insertion heuristic for single and multiple depot vehicle routing problems with backhauling. *Journal of the Operational Research Society*, 50(10):1034–1042.

Santos, F. A., Mateus, G. R., and da Cunha, A. S. (2014). A branch-and-cut-and-price algorithm for the two-echelon capacitated vehicle routing problem. *Transportation Science*, 49(2):355–368.

Savelsbergh, M. W. (1985). Local search in routing problems with time windows. *Annals of Operations research*, 4(1):285–305.

Savelsbergh, M. W. (1992). The vehicle routing problem with time windows: Minimizing route duration. *ORSA Journal on Computing*, 4(2):146–154.

Schneider, M., Stenger, A., and Hof, J. (2015). An adaptive VNS algorithm for vehicle routing problems with intermediate stops. *Or Spectrum*, 37(2):353–387.

Semet, F., Toth, P., and Vigo, D. (2014). Chapter 2: Classical exact algorithms for the capacitated vehicle routing problem. In *Vehicle Routing: Problems, Methods, and Applications, Second Edition*, pages 37–57. SIAM.

Shaw, P. (1998). Using constraint programming and local search methods to solve vehicle routing problems. In *International Conference on Principles and Practice of Constraint Programming*, pages 417–431. Springer.

Shi, Y., Boudouh, T., and Grunder, O. (2019). A robust optimization for a home health care routing and scheduling problem with consideration of uncertain travel and service times. *Transportation Research Part E: Logistics and Transportation Review*, 128:52–95.

Shi, Y., Boudouh, T., Grunder, O., and Wang, D. (2018). Modeling and solving simultaneous delivery and pick-up problem with stochastic travel and service times in home health care. *Expert Systems with Applications*, 102:218–233.

Shi, Y., Zhou, Y., Boudouh, T., and Grunder, O. (2020). A lexicographic-based two-stage algorithm for vehicle routing problem with simultaneous pickup–delivery and time window. *Engineering Applications of Artificial Intelligence*, 95:103901.

Solomon, M. M. (1987). Algorithms for the vehicle routing and scheduling problems with time window constraints. *Operations Research*, 35(2):254–265.

Stenger, A., Vigo, D., Enz, S., and Schwind, M. (2013). An adaptive variable neighborhood search algorithm for a vehicle routing problem arising in small package shipping. *Transportation Science*, 47(1):64–80.

Subramanian, A., Drummond, L. M. d. A., Bentes, C., Ochi, L. S., and Farias, R. (2010a). A parallel heuristic for the vehicle routing problem with simultaneous pickup and delivery. *Computers & Operations Research*, 37(11):1899–1911.

Subramanian, A., Uchoa, E., and Ochi, L. S. (2010b). New lower bounds for the vehicle routing problem with simultaneous pickup and delivery. In *International Symposium on Experimental Algorithms*, pages 276–287. Springer.

Subramanian, A., Uchoa, E., and Ochi, L. S. (2013a). A hybrid algorithm for a class of vehicle routing problems. *Computers & Operations Research*, 40(10):2519–2531.

Subramanian, A., Uchoa, E., Pessoa, A. A., and Ochi, L. S. (2011). Branch-and-cut with lazy separation for the vehicle routing problem with simultaneous pickup and delivery. *Operations Research Letters*, 39(5):338–341.

Subramanian, A., Uchoa, E., Pessoa, A. A., and Ochi, L. S. (2013b). Branch-cut-and-price for the vehicle routing problem with simultaneous pickup and delivery. *Optimization Letters*, 7(7):1569–1581.

Sungur, I., Ordónez, F., and Dessouky, M. (2008). A robust optimization approach for the capacitated vehicle routing problem with demand uncertainty. *IIE Transactions*, 40(5):509–523.

Taillard, É., Badeau, P., Gendreau, M., Guertin, F., and Potvin, J.-Y. (1997). A tabu search heuristic for the vehicle routing problem with soft time windows. *Transportation Science*, 31(2):170–186.

Tarantilis, C. D. (2005). Solving the vehicle routing problem with adaptive memory programming methodology. *Computers & Operations Research*, 32(9):2309–2327.

Taş, D., Dellaert, N., Van Woensel, T., and De Kok, T. (2013). Vehicle routing problem with stochastic travel times including soft time windows and service costs. *Computers & Operations Research*, 40(1):214–224.

Taş, D., Gendreau, M., Dellaert, N., Van Woensel, T., and De Kok, A. (2014). Vehicle routing with soft time windows and stochastic travel times: A column generation and branch-and-price solution approach. *European Journal of Operational Research*, 236(3):789–799.

Tasan, A. S. and Gen, M. (2012). A genetic algorithm based approach to vehicle routing problem with simultaneous pick-up and deliveries. *Computers & Industrial Engineering*, 62(3):755–761.

Thompson, P. M. and Psaraftis, H. N. (1993). Cyclic transfer algorithm for multivehicle routing and scheduling problems. *Operations Research*, 41(5):935–946.

Toth, P. and Vigo, D. (2002). Models, relaxations and exact approaches for the capacitated vehicle routing problem. *Discrete Applied Mathematics*, 123(1-3):487–512.

Toth, P. and Vigo, D. (2003). The granular tabu search and its application to the vehicle-routing problem. *Informs Journal on Computing*, 15(4):333–346.

Toth, P. and Vigo, D. (2014). *Vehicle routing: problems, methods, and applications*. SIAM.

Vidal, T., Crainic, T. G., Gendreau, M., and Prins, C. (2013). 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.

Vidal, T., Crainic, T. G., Gendreau, M., and Prins, C. (2014). A unified solution framework for multi-attribute vehicle routing problems. *European Journal of Operational Research*, 234(3):658–673.

Wang, C., Mu, D., Zhao, F., and Sutherland, J. W. (2015). A parallel simulated annealing method for the vehicle routing problem with simultaneous pickup–delivery and time windows. *Computers & Industrial Engineering*, 83:111–122.

Wang, C. and Qiu, Y. (2011). Vehicle routing problem with stochastic demands and simultaneous delivery and pickup based on the cross-entropy method. In *Advances in Automation and Robotics, Vol. 2*, pages 55–60. Springer.

Wang, C., Zhao, F., Mu, D., and Sutherland, J. W. (2013). Simulated annealing for a vehicle routing problem with simultaneous pickup-delivery and time windows. In *IFIP International Conference on Advances in Production Management Systems*, pages 170–177. Springer.

Wang, H.-F. and Chen, Y.-Y. (2012). A genetic algorithm for the simultaneous delivery and pickup problems with time window. *Computers & Industrial Engineering*, 62(1):84–95.

Wang, K., Lan, S., and Zhao, Y. (2017). A genetic-algorithm-based approach to the two-echelon capacitated vehicle routing problem with stochastic demands in logistics service. *Journal of the Operational Research Society*, 68(11):1409–1421.

Wassan, N. A., Wassan, A. H., and Nagy, G. (2008). A reactive tabu search algorithm for the vehicle routing problem with simultaneous pickups and deliveries. *Journal of Combinatorial Optimization*, 15(4):368–386.

Zachariadis, E. E. and Kiranoudis, C. T. (2011). A local search metaheuristic algorithm for the vehicle routing problem with simultaneous pick-ups and deliveries. *Expert Systems with Applications*, 38(3):2717–2726.

Zachariadis, E. E., Tarantilis, C. D., and Kiranoudis, C. T. (2009). A hybrid metaheuristic algorithm for the vehicle routing problem with simultaneous delivery and pick-up service. *Expert Systems with Applications*, 36(2):1070–1081.

Zachariadis, E. E., Tarantilis, C. D., and Kiranoudis, C. T. (2010). An adaptive memory methodology for the vehicle routing problem with simultaneous pick-ups and deliveries. *European Journal of Operational Research*, 202(2):401–411.

Zeng, Z.-y., Xu, W.-s., Xu, Z.-y., and Shao, W.-h. (2014). A hybrid GRASP+VND heuristic for the two-echelon vehicle routing problem arising in city logistics. *Mathematical Problems in Engineering*, pages 1–11.

Zhang, T., Chaovalitwongse, W. A., and Zhang, Y. (2012). Scatter search for the stochastic travel-time vehicle routing problem with simultaneous pick-ups and deliveries. *Computers & Operations Research*, 39(10):2277–2290.

Zhang, Y., Zhang, Z., Lim, A., and Sim, M. (2021). Robust data-driven vehicle routing with time windows. *Operations Research*, 69(2):469–485.

Abbreviations

ACO	Ant Colony Optimization
ACDR	Average Customer Dissatisfaction Rate
ALNS	Adaptive Large Neighbourhood Search
AUD	Average Unfulfilled Demand
AUDD	Average Unfulfilled Delivery Demand
AUPD	Average Unfulfilled Pickup Demand
AVNS	Adaptive Variable Neighbourhood Search
BB	Branch and Bound
BC	Branch and Cut
BCP	Branch and Cut and Price
BP	Branch and Price
CCP	Chance-Constrained Programming
CVRP	Capacitated Vehicle Routing Problem
FR	Feasibility Ratio
GA	Genetic Algorithm
GRASP	Greedy Randomized Adaptive Search Procedure
ILS	Iterated Local Search
LNS	Large Neighbourhood Search

PR	Path Relinking
PSO	Particle Swarm Optimization
SA	Simulated Annealing
SPR	Stochastic Programming with Recourse
SS	Scatter Search
SVRP	Stochastic Vehicle Routing Problem
2E-CVRP	Two-Echelon Capacitated Vehicle Routing Problem
2E-MTVRPTWSS	Two-Echelon Multiple-Trip Vehicle Routing Problem with Time Windows and Satellite Synchronization
TS	Tabu Search
VNS	Variable Neighbourhood Search
VRP	Vehicle Routing Problem
VRPSPD	Vehicle Routing Problem with Simultaneous Pickup and Delivery
VRPSPDTW	Vehicle Routing Problem with Simultaneous Pickup and Delivery and Time windows
VRPTW	Vehicle Routing Problem with Time Windows