

University of Nottingham
UK | CHINA | MALAYSIA

Optimisation Models and Algorithms for Real-life Transportation Routing and Scheduling Problems

Binhui Chen

Thesis submitted to the University of Nottingham
for the degree of *Doctor of Philosophy*

October 2018

Abstract

This thesis investigates the transportation routing and scheduling problems derived from the container transportation in Ningbo Port, which is the second largest port in China. In the container transportation problem, a fleet of trucks are scheduled to accommodate a sequence of container transshipment requests among multiple container terminals. This problem closely relates to the classic vehicle routing problem with time windows (VRPTW) in which the customers are routed to be serviced with minimal cost, satisfying capacity and time constraints. Facing the fast growing throughput in the port, an efficient automatic routing and scheduling system is urgently needed. In the Ningbo Port problem, there are mainly two types of transshipment requests (long-distance and short-distance) need to be scheduled to optimise fleet usage, while the transportation cost for the two types of tasks are different. As a result, to promote the vehicle utility and reduce operational cost at the same time, not only producing solutions of high quality are pursued in this research but also advanced problem modelling methods.

The research starts from studying the VRPTW to understand the characteristics of vehicle routing problems and the features of solution methodologies in detail. An improved variable neighbourhood search algorithm is developed. With the proposed new compounded neighbourhoods, the algorithm combines variable neighbourhood search and variable-depth neighbourhood search. In this single solution-based metaheuristic method, two objectives are optimised simultaneously, generating a number of new non-dominated solutions on benchmark instances. This research explores the ways of balancing diversification and intensification in metaheuristic search, providing experience and guidance for addressing the real-life Ningbo Port container transportation problem.

After the preliminary research on the VRPTW, the Ningbo Port problem then is studied in two steps. At the first stage, the short-haul

problem in the port districts is considered. The mathematical model of Vehicle Routing Problem with Time Windows and Open routes (VRPTW-O) is established in this study, providing a more efficient scheduling scheme for Ningbo Port company. Two metaheuristic algorithms are implemented for this new problem, while both of them outperform the state-of-the-art methods for similar problems. Different construction heuristics are compared on diverse instances, providing recommendation heuristics for real-world scenarios. At the second stage of the Ningbo Port problem research, the routing and scheduling of long-haul requests related to inland dry ports are combined with the short-haul requests. By introducing artificial depot to the problem, both types of transshipment tasks are integrated into one vehicle routing problem model, called Mixed-Shift Vehicle Routing Problem with Time Windows (MS-VRPTW). The driver salary cost is taken into consideration in this model to optimise the operational cost of the company, leading to a bi-objective VRP variant. To provide a generic solution methodology for multi-objective VRPs instead of a problem-specific algorithm, a selection perturbation hyper-heuristic is implemented in this study. The experimental results show that the proposed method performs well in both the real-life Ningbo Port problem and the classic VRPTW.

Acknowledgements

Foremost, I would like to express my sincere gratitude to my supervisor Dr. Rong Qu for her continuous support in my Ph.D study and research, for her patience, motivation, enthusiasm, and immense knowledge. Her guidance not only helped me in research but also encouraged me to bravely face up to the challenging time in life and move on. I would like to thank my second supervisor Dr. Dario Landa-Silva for his suggestions and comments which improved my work. I could not have imagined having better supervisors for my Ph.D study. My gratitude also goes to Prof. Ruibin Bai who gave me considerate help and guidances when I was in the University of Nottingham Ningbo China and back in the UK. I am profoundly grateful to Prof. Hisao Ishibuchi for his comments and advices which greatly improved my research. The rigorous scholastic style I learned from these experts stimulates my future career and life.

This Ph.D research would not have been possible without the financial support of Vice-Chancellor's Scholarship provided by School of Computer Science, University of Nottingham.

My sincere thanks to many researchers of ASAP research group. Among them: Wasakorn, Rodrigo, Arturo, Haneen, Imo, Peng, Wenwen, Raras, Jonata and Tim. Thanks to Dr. Jianjun Chen and Dr. Ning Xue for helping me take the first step of my research. Equally important, I appreciate the support provided by the administrative staff at the school of computer science, in particular Deborah and Christine.

Finally, I would like to thank my family: my parents Mr. Shian Chen and Ms. Daolian Li, my wife Wansha Cai and my two lovely sons, for the constant support and motivation throughout my life. I am especially thankful to my friends Isaac, Julian, Kadambari and Jaya, who gave me considerate help and friendship making me go through the difficulties in the UK.

Contents

1 Introduction **1**

1.1 Motivations and Challenges 1

1.1.1 Background and Motivation 1

1.1.2 The Real-life Full-truckload Container Transportation Problem: Challenges 9

1.2 Summary of Contributions 12

1.3 Structure of Thesis 13

1.4 List of Publications 15

2 Literature Review **16**

2.1 The Vehicle Routing Problem 17

2.1.1 Vehicle Routing Problem with Time Windows 21

2.1.2 Some Other Vehicle Routing Problem Variants 23

2.1.3 Multi-Objective Vehicle Routing Problem 29

2.2 A Review of Solution Methodologies for Vehicle Routing Problems . 33

2.2.1 Exact Methods 33

2.2.2 Classical Heuristics 37

2.2.3 Metaheuristics 43

2.2.3.1 Population-Based Metaheuristics 43

2.2.3.2 Single Solution-Based Metaheuristics 46

2.2.3.3 Learning Mechanisms 51

2.2.3.4 Discussion 53

2.2.4 Hyper-Heuristics 54

2.3 The Full-truckload Transportation Problem 55

2.4 Service Network Design Problem for Freight Transportation 57

2.5 Summary 58

3 Models of Real-life Transportation Scheduling and Routing Problems and Benchmarks 60
3.1 Periodic Vehicle Routing Problem with Time Windows and Open Routes . . . 61
3.1.1 Description of Problem . . . 62
3.1.2 Mathematical Model . . . 66
3.1.3 Benchmark Instances . . . 68
3.2 Mixed-Shift Vehicle Routing Problem with Time Windows . . . 69
3.2.1 Description of Problem . . . 70
3.2.2 Mathematical Model . . . 74
3.2.3 Benchmark Instances . . . 76
3.3 Summary . . . 78

4 A Variable Neighbourhood Search Algorithm with Compounded Neighbourhoods for VRPTW 80
4.1 Variable Neighbourhood Search with Compounded Neighbourhoods 81
4.1.1 Compounded Operators and Neighbourhoods . . . 81
4.1.2 Shaking and Shaking Operators . . . 83
4.1.3 Local Search . . . 83
4.1.4 The Framework of VNS-C . . . 84
4.2 Experiments and Results Analysis . . . 85
4.2.1 Compounded Neighbourhoods and Shaking . . . 85
4.2.2 Neighbourhood Order . . . 86
4.2.3 Experiment Results and Analysis . . . 87
4.3 Summary . . . 90

5 Two Metaheuristic Approaches for Periodic Vehicle Routing Problem with Time Windows and Open Routes 92
5.1 Exact Search . . . 93
5.2 Solution Construction Heuristic . . . 95
5.2.1 A Urgency Level-Based Insertion Heuristic . . . 95
5.2.2 Construction Scheme Selection . . . 97
5.3 A Variable Neighbourhood Search Algorithm with Reinforcement Learning . . . 99
5.3.1 Algorithmic Details . . . 100
5.3.1.1 Algorithm Framework . . . 100
5.3.1.2 Shaking . . . 101
5.3.1.3 Sampling of Neighbourhood . . . 101

5.3.1.4 Local Search with Reinforcement Learning 103
5.3.2 Experiment Results and Analysis 104
5.3.2.1 Impact of the Sampling 104
5.3.2.2 Feasibility Indicators & Feasible Solution Space Exploration 107
5.3.2.3 Comparison with the general VNS and State-of-the-art Algorithms 112
5.3.2.4 Comparison with the Previous Scheduling Scheme . 113
5.4 A Variable-Depth Adaptive Large Neighbourhood Search Algorithm 113
5.4.1 Algorithmic Details 114
5.4.1.1 Algorithm Framework 114
5.4.1.2 Variable-Depth Setting 116
5.4.1.3 Operators of Destroy and Repair 116
5.4.1.4 Acceptance Criterion 118
5.4.1.5 Weight Adjustment 118
5.4.2 Experiment Results and Analysis 119
5.4.2.1 Parameter Tuning and Analysis 119
5.4.2.2 Comparison of Solution Algorithms 120
5.4.2.3 Contributions of Operators 123
5.4.2.4 Analysis of Runtime 123
5.5 Summary 124

6 A Hyper-Heuristic with Two Guidance Indicators for MS-VRPTW 127
6.1 Exact Search for MS-VRPTW 128
6.2 Initial Solution Construction Heuristic 128
6.3 A Selective Perturbation Hyper-Heuristic with Two Guidance Indicators 130
6.3.1 High-Level Heuristic 130
6.3.2 Guidance Indicators and Weight Adjustment Scheme 132
6.3.3 Low-Level Heuristics 133
6.4 Experiments and Results Analysis 134
6.4.1 Parameter Setting and Complexity Discussion 134
6.4.2 Impacts of the Guidance Indicators 135
6.4.3 Impacts of Solution Selection and Acceptance Criterion . . . 138
6.4.4 Comparison with State-of-the-art Algorithms 140
6.4.5 Results on VRPTW Benchmarks 141
6.5 Summary 143

7 Conclusions and Future Research **145**
7.1 Research Summary and Conclusions 145
7.1.1 Mathematical Models of the Ningbo Port Problem and Benchmark Instances 146
7.1.2 Vehicle Routing Problem with Time Windows and Variable Compounded Neighbourhood Search 147
7.1.3 Inter-Dock Container Transshipment Problem and Metaheuristic Solution Methods 147
7.1.4 Mixed-Shift Container Transportation Problem and a Hyper-Heuristic Method 148
7.2 Future Research Directions 149
7.2.1 Path Planning 149
7.2.2 Dynamic and Stochastic Factors 150
7.2.3 Other Real-Life Objectives 150
7.2.4 Vessel Routing Problem and Maritime Location Based Services 151

A A Tool of Statistical Analysis and Visualisation for Big Data Maritime Location-Based Services **152**
A.1 Geographic Information View 152
A.2 Statistics by Port 153
A.3 Statistics by Vessel 156

Bibliography **159**

List of Figures

1.1 Distribution of the port districts and dry ports of the Ningbo Port company, obtained from Google Maps [207]. The balloon icons represent dry ports while the ball stick icons represent port districts. . 3
1.2 Map of the Ningbo Port Districts, taken from Google Maps [208]. . 4
1.3 A screenshot of the declaration information interface in the EDI platform. 5
1.4 A screenshot of the freight scheduling system interface of the EDI platform, monitoring the implementation of transshipment service. . 7
1.5 A screenshot of the freight positioning system interface in the EDI platform. 8

2.1 Comparison of the neighbourhood structures of VNS and VDNS. . . 50

3.1 A scheduling example of two consequent shifts with five trucks. The solid line circles represent the transportation tasks to be completed. The right sub-figure indicates the first route in the schedule. 65
3.2 A schedule example of one *long shift* (two *short shifts*). Three *one-driver* trucks and five *double-driver* trucks are used in this schedule. The right subgraph presents the first and sixth routes in the schedule. 73

4.1 Examples of neighbourhoods in VNS-C. (a). Link (1,2,3) exchanged with link (5,6) by the intra-route *Or-opt-i* compound operator (i=3). (b). Link (2,3,4) exchanged with link (5,6) by the inter-route compound operator *CROSS-i* (i=3). (c). Link (4,5) is removed by *LinkMove-i* from its original route and inserted to a random position in the target route, which brings a route reduction. 82
4.2 Two best found solutions with 2 and 3 vehicles on R204. 89
4.3 Comparison on average NV with different Max_i values of *LinkMove-i*. 89
4.4 Comparison on the best found NV with different Max_i values of *LinkMove-i*. 90

5.1 Comparison of infeasible rates on 25% NP4-3 (a) and 100% LB4-1 (b). 111

6.1 The iteration times and computation time of the three algorithms on eight sample instances. 138
6.2 Comparison of the iteration time reduction. The bars indicate the number of reduction of iterations. Longer bars represent greater reduction, while the negative values indicate more iterations than RHH. 139

A.1 Main interface of Visual_GlobaVista. 153
A.2 Port geographic distribution and the related routes. 154
A.3 Zoom view of the geographic information. 154
A.4 The interface for retrieving information by port. 155
A.5 The interface for retrieving information by vessel. 157

Chapter 1

Introduction

The 21st century is the century of extensively exploring and utilising the ocean to humankind, and this development greatly relies on modern, automated ports and logistics. Every day, huge amounts of commodities are sent from/to ports, and the ports are connected to the hinterland via railway/shippers, building a logistics network covering the whole world. The fast growing goods and container flow in ports have caused congestion, crowded terminals and long waiting time for commodities. Nowadays, intermodal transportation and dry port construction greatly improve the connection between seaports and inland cities, which urgently calls for more intelligent logistic management system to efficiently handle the routing and scheduling of the huge amounts of freight transportation. Ningbo Port, located on the coast of East China Sea, is one of the world's largest and busiest seaports. Its throughput has exceeded 20 million twenty-feet equivalent units (TEU) since 2015. In the first quarter of 2018, it dealt with 6.837 million TEU, that is, 191.96 million tons cargoes, which puts it the top in terms of all cargo ports around the world [218]. This thesis studies the container transportation routing and scheduling problem in Ningbo Port.

1.1 Motivations and Challenges

1.1.1 Background and Motivation

Recently, the change from the third generation to the fourth generation intelligent port is happening globally. The third generation ports play the role of logistics centre linking with cities and users. They are characterised by integrating transportation, trade, information services and goods distribution, supporting international trade. However, a certain weakness of these ports is that the functioning of port supply centre separates from the distribution chain. In contrast, the fourth generation ports play a super-regional role creating a hub from where cargo is

transported to smaller outlying ports. In this case, the management of ports pays more attention to IT usage, port community development, presence of port cluster and logistics centre, connections between hinterland on the land side and the foreland on the sea side. Port authorities in this case usually are connected by a common administration or a common convenor of container terminals. [235]

With the promotion and popularisation of location and tracking technologies, e.g. Global Positioning System (GPS), Internet of Things and wireless network, more and more intelligent and efficient port management systems have been developed. The Hamburg Port, which is the largest port in Europe, has explored intelligent port for a long time and is ahead of the world in this field. In Hamburg Port, with the help of integrated information retrieval platform and pre-scheduling in central system, the transportation tasks can be assigned to trucks in advance without waiting. The truck drivers receive the traffic information before entering the port area via in-vehicle terminal device or mobile phone, and then they can independently choose the best path to enter the port.

Information management technologies have made a great deal of achievements in port management in China as well. For example, Shanghai Port has employed information technologies in port management relatively early. By adopting advanced Radio Frequency Identification (RFID) technology in the integrated customs clearance system, the efficiency of customs clearance is significantly improved. Shanghai Port has been ranked the first dealing with the most TEU in all ports of the world for many years. In Qingdao Port, the loaded ratio of trailers increases to 54% from 47% with the help of intelligent container management systems. It completes 6.9% throughput of cargo with only 1.3% berths of China [218].

With the development of logistics industry and intermodal transportation, more and more dry ports are built in hinterland regions to reduce the logistical cost and extend the functionality of seaports. "*Dry Port is an inland intermodal terminal directly connected to seaport(s) with high capacity transport mean(s), where customers can leave/pick up their standardized units as if directly to a seaport*" [263]. Without physically expanding the seaport, dry ports can promote the service of shippers and transport managers, and also the market security in hinterland. In addition, as dry ports reduce the road congestion and infrastructure investment in seaport cities, the development of dry port also has ecological benefits.

Container transshipment is an essential part of the freight transport system of ports. When the location of a commodity is not its pickup point appointed by the shippers (truck/train/vessel), the commodity will be transshipped. The containers are transferred from their unload points (origin port/terminal) to the

destination hubs or pickup points appointed (destination port/terminal). This transshipment procedure is sometimes defined as the *nodes* in the freight transport network [61]. Except transporting containers between terminals, other operations such as container sorting, storage, consolidation, container loading and unloading on vehicle can also be integrated into the *node*. All of these activities have to be scheduled in the container transportation management system. In practice, the managers of ports may outsource parts of these jobs to third party logistics providers.

The Ningbo Port company is in charge of the operation of Ningbo Port, which has 11 dry ports and six port districts (seaports). Each port district has a number of docks. Every single dry port or dock is associated with a container terminal, which is connected by either highway or railway. The geographic locations of the port districts and dry ports are shown in Figure 1.1.

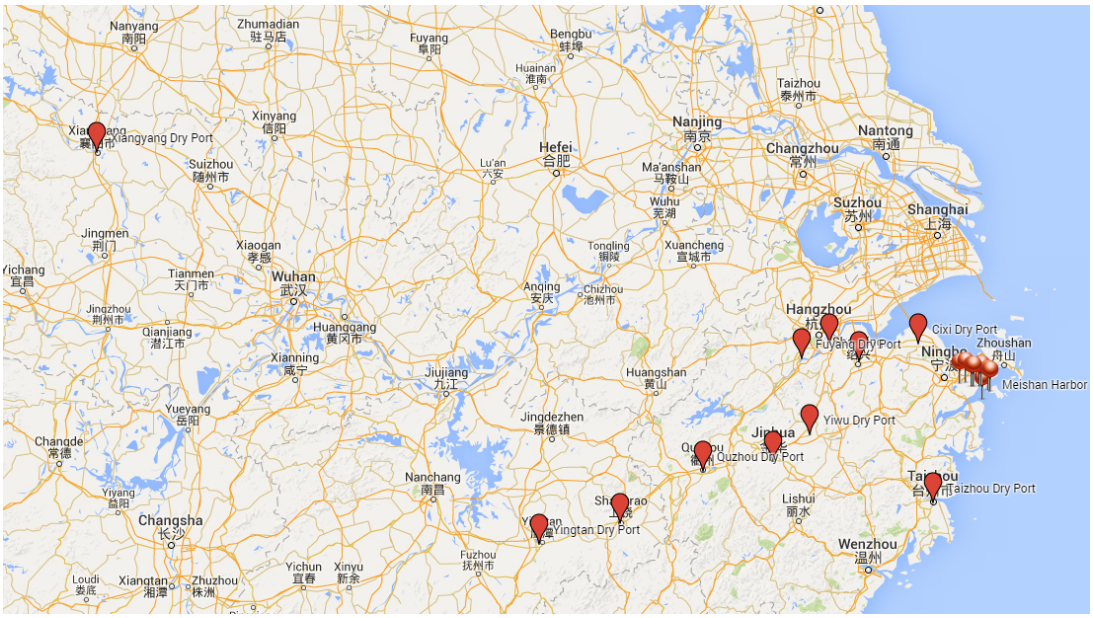


Figure 1.1: Distribution of the port districts and dry ports of the Ningbo Port company, obtained from Google Maps [207]. The balloon icons represent dry ports while the ball stick icons represent port districts.

The Ningbo Port company also employs intelligent information technologies in the vessel management, customs clearance, storage management and freight transport system. The first version management platform of the company, which is called Electronic Data Interchange (EDI), was developed in 1995. As the throughput growing sharply and many subsystems being developed independently, data synchronism is becoming an obstacle for the application of EDI. For example, because of the information asymmetry between terminals and fleets, the scheduling of

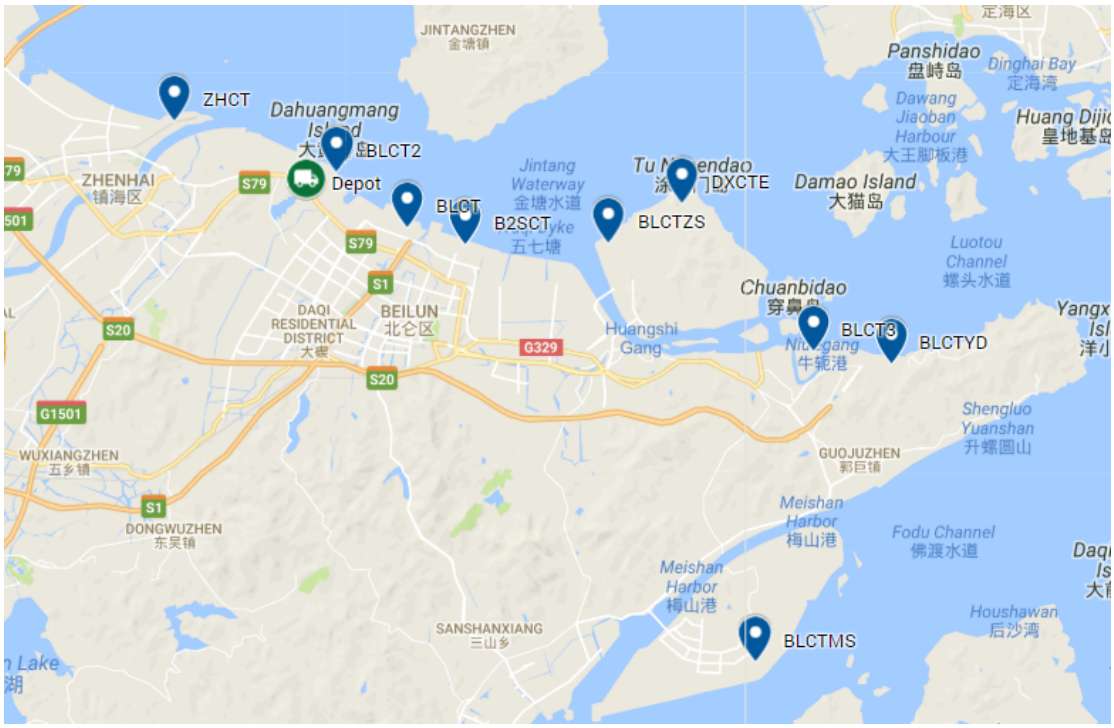


Figure 1.2: Map of the Ningbo Port Districts, taken from Google Maps [208].

vehicle still relies on manual operation in a sense, causing long queuing and serious congestion at the gates, cranes and other devices. This section later introduces the automatic scheduling procedure of EDI.

The container transshipment requests confronted by the company can be categorised into three classes: *dock-to-dock*, *dry port-related* and *railway-related*. The first two classes of requests are transshipment between container terminals, while the *railway-related* transshipment is between train station and container terminal. To be precise, containers are transported between docks in *dock-to-dock* transshipment. This type of transshipment are short-haul which can be completed in no more than seven hours. Differently, the *dry port-related* transshipment are long-haul tasks relating to the inland dry ports. This type of tasks usually take more than 13 hours to complete. *Railway-related* transshipment happens before commodities being loaded to or after being unloaded from the train. Completing a task request of this type usually takes no more than half an hour as the travel is very short-distance. The small travel distance results into limited improvement in this case.

Transshipment requests are associated with the declaration forms issued by the customs office. Once the declaration forms are received, the company schedules the fleet to service the corresponding transshipment requests. The declaration forms are either digital or printed. At present, a digital declaration form system

has been deployed at all the dry ports and nine docks (see Figure 1.2). Because the Xiangyang Dry Port is too far away from the other ports, all transshipment tasks related to it are completed by trains. Among the other 10 dry ports and the nine docks, those dock-to-dock and dry port-related transshipment requests are responded to by container trucks. Following railway-related transshipment requests, truck drivers have to collect the printed declaration forms from customs office before servicing them. In this thesis, only the transshipment tasks using digital declaration forms are considered, aiming to build an efficient automatic freight transportation system.

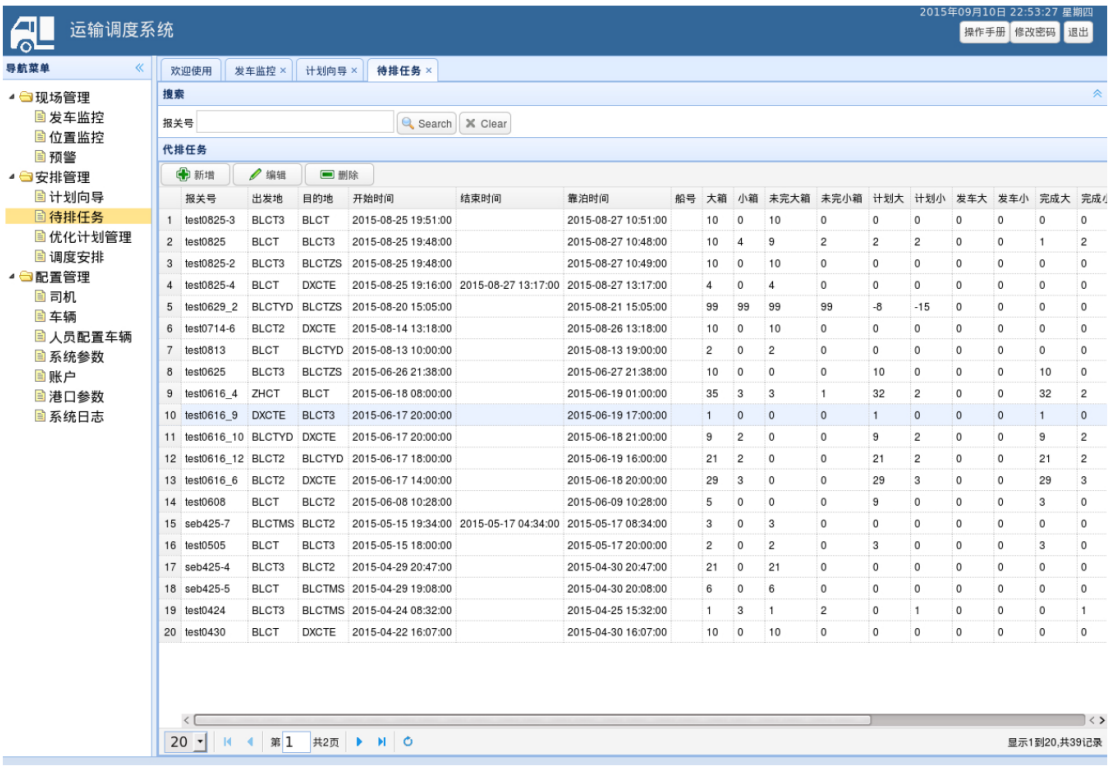


Figure 1.3: A screenshot of the declaration information interface in the EDI platform.

The digital declaration form in the EDI system composes of a unique identifier (ID) and the detailed description of the transshipment order for a commodity (see Figure 1.3). Each row in the figure represents a commodity, while the columns are commodity ID, transshipment origin and destination, available time, docking time, the number of containers to be transported, and other details about the transshipment request. Here, the available time is the starting time the containers can be picked up. The containers should arrive their destinations before the docking time, which is the vessel arriving time, defining the deadlines of the transshipment tasks.

Before starting assigning trucks in EDI, the company's coordinators would sieve through the received requests first. Note that not all the requests imported will be serviced. This is because there are a few requests that are imported when they are very close to their deadlines, and it is impossible to complete them within their time windows. The coordinators will reject those requests that cannot be completed in time, however, all the requests accepted must be completed. In addition, apart from the digital declaration forms, there still are some paper declaration forms that are sent to the coordinators by express mail. These associated transshipment requests will be input to an Excel spreadsheet and then be imported to the request pool of EDI as well. These requests might also be rejected by the coordinators according to the closeness to their deadlines.

In EDI, the *short-haul* (dock-to-dock transshipment) and *long-haul* (dry port-related transshipment) are separately scheduled and serviced by two independent fleets. The trucks of the two fleets are identical, which can carry one TEU each time. The two fleets are located at the same depot which is close to the BLCT2 dock (see Figure 1.2). Every working day, trucks depart the depot with a list of transshipment tasks to be completed, and return to the depot when completing all tasks assigned. The transport path between any two terminals are specified by the company and fixed.

Because the working time for completing short-haul and long-haul tasks are significantly different, the shift lengths for the drivers of the two fleets are different as well. In the fleet for dock-to-dock transshipment, the shift length is 12 hours. More precisely, the day-shift starts from 8am and the evening shift starts from 8pm. On the other hand, in the fleet for dry port-related transshipment, one shift is a whole day (24 hours) as the vast majority of this type of transshipment cannot be completed within 12 hours. In this case, two drivers are assigned to one truck at the same time. This arrangement allows the two drivers take turns to drive and rest during the shift, thus avoiding fatigue while driving and violating the associated regulations in the Labour Law.

Figure 1.4 shows the interface of the transshipment scheduling system of EDI, which presents the details of scheduled transshipment on each truck. In the figure, each row represents the state of the transshipment of one container, notated as a *task* for brevity. From the interface, we can see the plate numbers of trucks used, origin and destination terminals of tasks, commodity IDs, container types and number, scheduled service starting time, latest starting time and scheduled finish time etc. Two rows with the same commodity ID means the two containers belong to the same commodity order. Note that, the vessel docking time may change in

practice, and missing the scheduled vessel will cause late container shipping with serious consequences. Thus, when setting the latest starting time, the coordinators always do their best to complete the transshipment six hours ahead of the docking time to accommodate potential emergency situations.

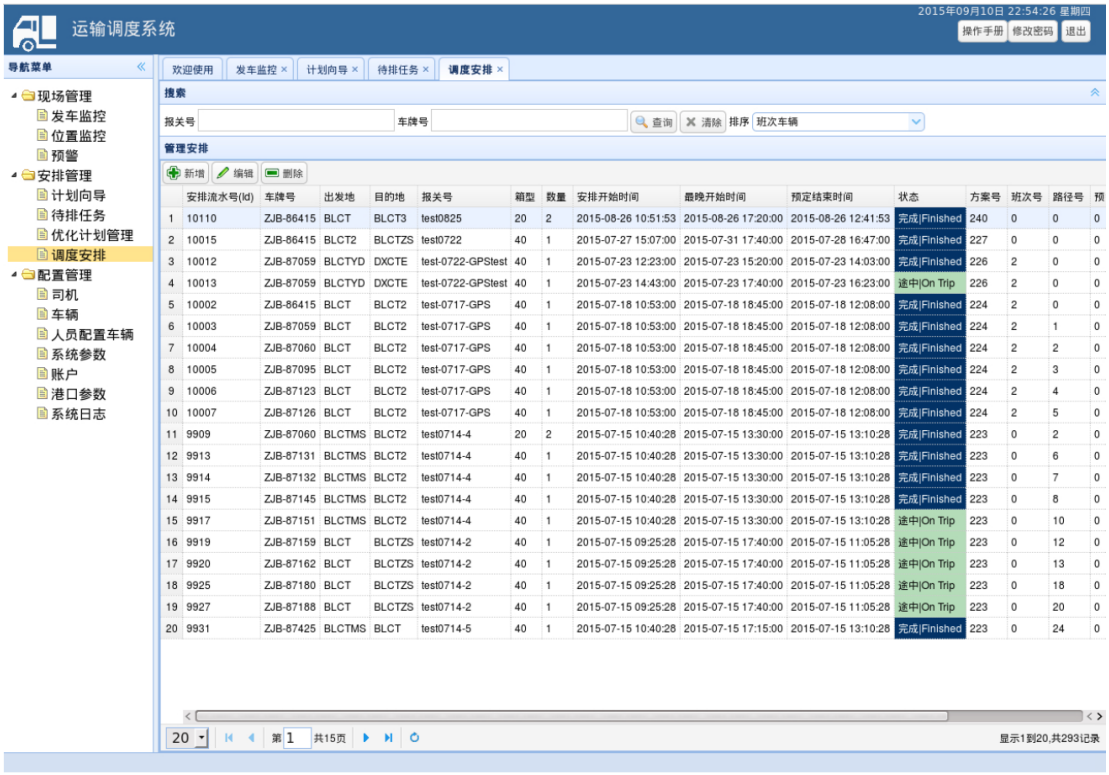


Figure 1.4: A screenshot of the freight scheduling system interface of the EDI platform, monitoring the implementation of transshipment service.

Two online heuristics are adopted in EDI to schedule the services of the accepted requests. The first heuristic matches the destination and origin between two consecutive tasks to reduce the deadheading (empty-load travel) mileage. The second heuristic is a task prioritising rule based on the closeness to deadline. In this heuristic, the tasks which are closer to their deadlines will be assigned first. Both heuristics work in real-time, where once a task is finished, a new task selected by the two heuristics will be assigned to the corresponding free truck.

In the EDI scheduling system, a truck driver confirms his completion of the current tasks and requests a next task by sending a message to the scheduling center. Once the message is received, the two above-mentioned heuristics will determine a new task for the truck according to its location. The trucks are tracked by built-in GPS devices. With the help of the GPS service, the positions of the trucks can be monitored by the scheduling centre, as shown in Figure 1.5.

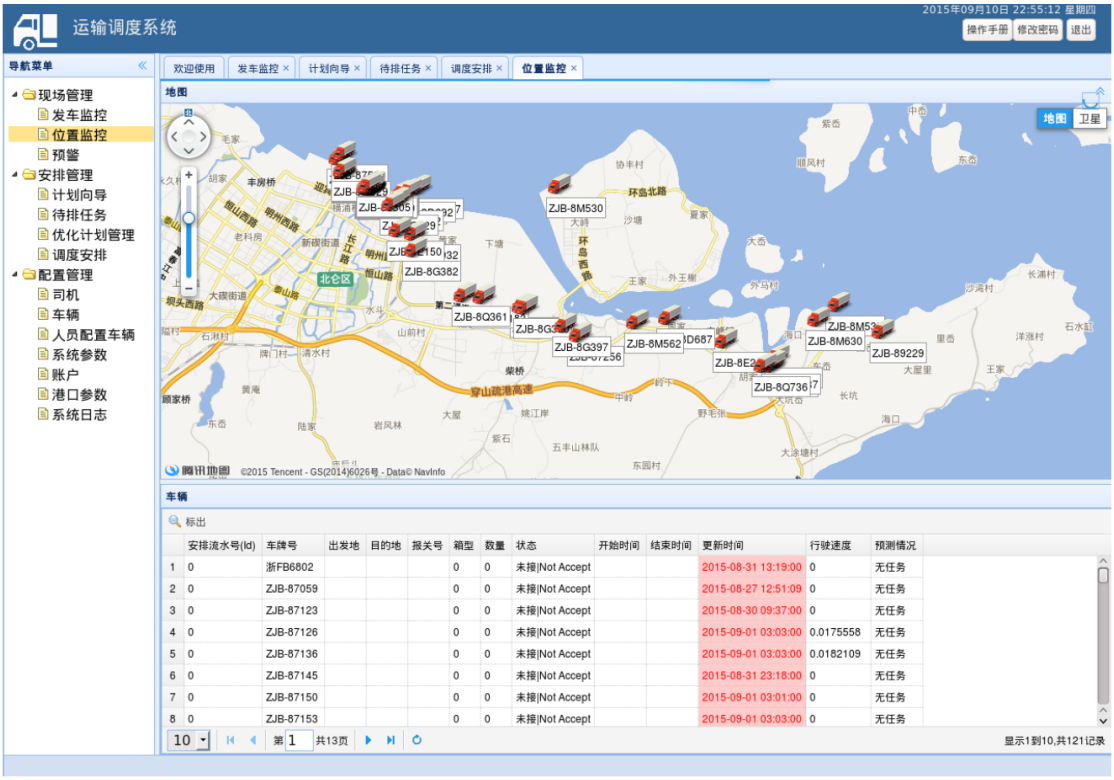


Figure 1.5: A screenshot of the freight positioning system interface in the EDI platform.

Real-time interference by coordinators is also applied in the current version EDI. In busy seasons, the company may confront hundreds of containers to be transshipped every day, and the current scheduling system often fails to assign some critical and urgent tasks. Therefore, the coordinators have to manually choose the specific tasks to be serviced first and postpone those non-critical tasks, aiming for a trade-off between system efficiency and effectiveness. Otherwise, there would be many delays and high consequent penalties to the company. However, this manual interference decreases the stability of the performance of the scheduling system.

Another deficiency of the current system is that, in practice, the GPS information often mismatches the scheduling center data, leading to poor scheduling. In real-life, the drivers usually send the "task completed" message earlier to reduce the waiting time for the next task, which causes inaccurate positioning and travel time prediction. When a truck is not at the source terminal of its next task, it has to go there with an empty load.

The company is deploying the digitalised system at more and more ports to improve the container transshipment scheduling and routing efficiency. However, the growth of declaration forms also increases the scheduling pressure to the system.

An automatic scheduling and routing system with higher efficiency is urgently needed. The main reason causing the low efficiency of the current system is that neither the two adopted greedy heuristics nor the manual scheduling work beyond the current step, lacking forward planning. In fact, 90% of the transshipment requests are known 24 hours ahead of their available time by the company. So, a higher level planning system, which takes all requests and resources in a wider planning horizon into account, is expected to improve the efficiency of scheduling and routing, reducing the wait time of trucks and promoting the utility of fleets.

1.1.2 The Real-life Full-truckload Container Transportation Problem: Challenges

In the Ningbo Port problem, the specific real-life constraints make it a unique problem different from the problems studied in previous research. This problem shares several common features with the classic Vehicle Routing Problem (VRP) [139], Service Network Design Problem (SNDP) [314] and their variants. Both VRP and SNDP have been extensively studied in the community, obtaining massive achievements. However, the Ningbo Port problem has a number of distinct characteristics, which needs further investigation.

- In VRP and SNDP, each customer is usually serviced once. However, in the Ningbo Port Problem, each container terminal would be visited multiple times within the planning horizon.

- In the classic Vehicle Routing Problem with Pickup and Delivery (VRPPD) [79], one customer is either a pickup point or a delivery destination. But in the Ningbo Port problem, the terminals are both pickup and delivery points for containers. In fact, the activities of containers being loaded to and unloaded from trucks are conducted at the same time in practice. The container flow at terminals is bi-directional in this problem.

- In the VRPPD and SNDP, the service time for customers is usually fixed and uniform. Because of the limitation of plant in Ningbo Port, e.g. the gates and cranes, the time of uploading and unloading a container at terminals are comparative in the total time of completing a container transshipment. The service time at different terminals are different. Thus, in the Ningbo Port problem, the service time at container terminals cannot be simplified or combined into the travel time, which is normal in VRPPD and SNDP for freight transportation.

- In both the Multi-Period Vehicle Routing Problem (MPVRP) [116] and the Ningbo Port problem, the long planning horizon is divided into multiple shifts. However, the difference between them is that the lengths of shifts in MPVRP are identical, while there are two types of shifts in the latter.

- To increase the utility of vehicles, consolidation is often adopted in SNDP and VRPPD. Differently, in the Ningbo Port Problem, each truck can carry only one container at a time, where consolidation is unavailable. In this case, a pickup at the source terminal is always followed by the delivery to the destination terminal, and no other activities such as consolidation and load split are allowed during the service.

- Another difference between the Ningbo Port problem and SNDP is the value type of customer demands. In SNDP for freight transportation, the customer demands are usually continuous, e.g. the size and weight of goods. However, the commodities in the Ningbo Port are measured by the number of containers which is an integer.

Although the existing models and methodologies cannot be directly applied to the Ningbo Port problem because of the above mentioned differences from the previous research, there are still some similar studies that can be found in the literature. Wang and Regan [311] investigate the local truckload pickup and delivery problem in a model of multiple traveling salesman problem with time window constraints (m-TSPTW). The loaded trips are treated as nodes in this model. The proposed solution method works in a real-time manner, searching better solutions until the gap between the best-found solution and the approximate optimality is acceptable. The optimality gap is obtained by solving the over-time-constraint and under-time-constraint versions of the problem. Jula et al. [164] use the same model to solve a container transportation problem in local area. In this problem, a fleet initially located at a depot is used to move the containers among the depot, end customers and service stations. A two-phase exact method based on Dynamic Programming (DP) and a hybrid method combining DP and genetic algorithm are developed for normal and large sizes of instances respectively.

Zhang et al. [322] study a container transportation problem with multiple depots and terminals. In this study, both full and empty containers transportation are taken into consideration. Furthermore, according to the different types of transport origin and destination, the transportation tasks are categorised into four types before scheduling. By using the time window partition based method proposed in [311], the computation time of scheduling is reduced remarkably. Chen

et al. [60] propose a task based Variable Neighbourhood Search approach for a short-haul container transportation problem, improving the Heavy-Loaded Distance Rate (HLDR) of trucks significantly. Bai et al. [16] later model the problem as a Set Covering Problem. By combining close terminals and relaxing the time constraint, the simplified problem is solved exactly, obtaining new best results and lower bounds on the benchmark.

This thesis has reviewed previous research which investigates various application scenarios, but there are still some specific real-life difficulties that have to be solved in the Ningbo Port problem. For example, when completing the transportation tasks among the docks in the Ningbo Port problem, the truck travelling distance and time are relatively short, while container loading and unloading time at terminals are comparable. This type of short-haul transportation is greatly different from the long-haul transportation tasks related to the dry ports. The scheduling systems or mechanisms for these two types of tasks cannot be simply the same. In accordance with the above description and the information collected from the company, the main challenges of solving the Ningbo Port problem can be summarised as follows.

- The size of the container truck fleet is fixed. No doubt the company will gradually increase the physical facilities according to the development of port throughput. However, the cost of expanding a port physically is very high, e.g. container truck, crane, gate and so on. Completing as more container transshipment tasks as possible with the current limited resources is vital to the transportation management system.

- The efficiency of the scheduling and routing system needs to be improved. Firstly, the routing plans generated should be improved to increase the utility of trucks and reduce company operating costs. Reducing the unloaded travel distance of trucks is a straightforward objective. In addition, the computation time of generating a high quality schedule should be reduced.

- Both the short-haul and long-haul transportation problems should be solved. Tight constraint problems cannot be handled in the same manner as the loose problems [183]. Apart from the trip distances and time of short-haul tasks and long-haul tasks are greatly different, the constraints such as time constraint vary obviously in these two problems. Sophisticatedly designed high-performance mechanism is needed to tackle both types of problems.

- There are two types of shifts for the truck drivers. Drivers can choose their shift type according to their availability. Proper scheduling of driver can reduce the operational cost for the company. The scheduling scheme striking a balance between flexibility and stability of the system is expected.

- The number of transshipment requests is large. In busy seasons, the company faces hundreds of transportation tasks to be assigned every day. As the vast majority of requests are issued one day ahead of their available time, there can be more than one thousand requests in the request pool of the system. High-performance scheduling algorithms are expected to handle this big size problem.

1.2 Summary of Contributions

This thesis investigates the real-life freight transportation problems from Ningbo Port and the associated solution methodologies. The study starts from investigating the classical Vehicle Routing Problem with Time Windows (VRPTW), exploring the basic features of vehicle routing and scheduling problems. The preliminary achievements and findings are then applied to the real-life Ningbo Port problem. A number of heuristic algorithms (metaheuristics and hyper-heuristics) are developed for different application scenarios, producing promising results. The contributions of this thesis include:

- A review of VRPs. The definitions of related problem variants and their solution methodologies, including exact methods and approximate approaches, are reviewed.

- Research and implementation of a variable neighbourhood search with compounded neighbourhood for VRPTW. The research proposes and explores the contributions of a new mechanism of compounding independent neighbourhood operators. The proposed mechanism combines different neighbourhood structures and the variable-depth neighbourhood scheme, and a number of new non-dominated solutions of the benchmark are generated by it.

- Introduction of two new VRP variants arising from real-life freight transportation problems to the research community. The two variants are defined as the Periodic Vehicle Routing Problem with Time windows and Open Routes (PVRPTW-O) and the Mixed-Shift Vehicle Routing Problem with

Time windows (MS-VRPTW), respectively. The former focuses on the short-haul problem in Ningbo Port districts, and the latter combines the routing and scheduling work of short-haul and long-haul into one framework. These two problems are mathematically modelled and introduced in Chapter 3. The benchmarks of the problems are generated, which can be used in following study and extended for other MPVRP research.

- Investigation and implementation of two metaheuristics for PVRPTW-O, which are a variable neighbourhood search algorithm with reinforcement learning and sampling and a variable-depth adaptive large neighbourhood search algorithm, respectively. The proposed algorithms explore the trade-off between diversification and intensification of search in metaheuristics. Both of them improve the quality of solutions at some level.

- Investigation and implementation of a hyper-heuristic with two guidance indicators for MS-VRPTW. The proposed algorithm is a generic method which can be applied to various multi-objective routing problems. In the MS-VRPTW and the classic VRPTW, the proposed method produces promising results.

- Implementation of a statistical analysis and visualisation system for big data maritime location based services. The system provides the port managers and coordinators a supporting tool to observe and analyse the operational situation of the ports. This tool can be integrated to the routing and scheduling system into EDI, improving the prediction capability and scheduling performance.

1.3 Structure of Thesis

This thesis is structured as follows:

- A literature review is presented in Chapter 2, which covers the research on various aspects related to the problems studied in this thesis, including problem definitions, models and solution methodologies. Starting from the introduction of the general vehicle routing problem, the mathematical models, side constraints, objective functions and other features in various VRP variants and related applications are discussed. Both exact methods and heuristic algorithms for VRPs are reviewed while more attention is spent on the latter.

- Two mathematical problem models are extracted from the Ningbo Port container transportation problems, presented in Chapter 3. The first model formulates the short-haul container transshipment problem between docks of Ningbo Port, while the other model describes the routing and scheduling problem concerning both short-haul and long-haul transportation. By introducing the artificial depot into the model, the scheduling of two types of shifts with different lengths are integrated into one system. The benchmarks of the two problems are introduced in this chapter as well, which are composed of the instances with diverse features.

- Chapter 4 introduces a variable neighbourhood search algorithm with compounded neighbourhood for VRPTW. With the sophisticatedly designed shaking and neighbourhood operators, the proposed algorithm explores solution space at variable neighbourhood depth, while the neighbourhood structure is also shifted during the search. Some new best solutions of the benchmark are found in this study.

- For the short-haul container transportation problem in Ningbo Port, two metaheuristic methods are proposed and introduced in Chapter 5. The first method is a VNS-based algorithm with an adaptive scheme guiding the search. Different initial solution construction strategies and the impact of sampling scheme on single solution-based methods are investigated. In addition, an improved adaptive large neighbourhood search is also developed, which generates some new best solutions on the benchmark. Exact search and several state-of-the-art metaheuristic algorithms are compared in this study.

- A hyper-heuristic for mixed-shift VRPTW is presented in Chapter 6. In this selective perturbation hyper-heuristic, the performance of the proposed guidance indicators are deeply investigated. The proposed algorithm is tested on the benchmark introduced in Chapter 3 and the classic VRPTW dataset. The results of computation experiment show the powerful search performance of the proposed method in different problems.

- Chapter 7 summarises the contributions of this thesis. Some potential future research directions are also concluded in this chapter.

- In addition, Appendix A introduces a tool of statistical analysis and visualisation for big data maritime location based services, which is a collaborative project sponsored by INNOVATEUK and University of Nottingham. The

implemented tool deals with the big data of global ports and vessels information, which can be helpful for optimising the routing and scheduling in ports in future research.

1.4 List of Publications

1. Binhui Chen, Rong Qu, Ruibin Bai and Hisao Ishibuchi. **A Variable Neighbourhood Search Algorithm with Compound Neighbourhoods for VRPTW**. In *Proceedings of the 5th International Conference on Operations Research and Enterprise Systems (ICORES 2016)*, pp. 25-35, Scitepress, Rome, Italy, February 2016. (Best student paper nominee)

2. Binhui Chen, Rong Qu, Ruibin Bai and Hisao Ishibuchi. **An Investigation on Compound Neighborhoods for VRPTW**. In *International Conference on Operations Research and Enterprise Systems*, Series Communications in Computer and Information Science, Vol. 695, pp. 3-19. Springer, 2016.

3. Binhui Chen, Rong Qu and Hisao Ishibuchi. **Variable-Depth Adaptive Large Neighbourhood Search Algorithm for Open Periodic Vehicle Routing Problem with Time Windows**. In *Proceedings of the International Conference on Harbor, Maritime and Multimodal Logistic Modelling and Simulation (HMS 2017)*, pp. 25-34, Barcelona, Spain, September 2017. (Best paper nominee)

4. Binhui Chen, Rong Qu, Ruibin Bai and Wasakorn Laesanklang. **A Hyper-Heuristic with Two Guidance Indicators for Bi-Objective Mixed-Shift Vehicle Routing Problem with Time Windows**. *Applied Intelligence*, 48(7): 1-23, 2018.

5. Binhui Chen, Rong Qu, Ruibin Bai and Wasakorn Laesanklang. **A Variable Neighborhood Search Algorithm with Reinforcement Learning for a Real-life Periodic Vehicle Routing Problem with Time Windows and Open Routes**. Submitted to *RAIRO - Operations Research*. Under Review.

Chapter 2

Literature Review

Transportation is an essential human activity and a determining factor to many social and economic issues, e.g. the container transportation in ports. The effective management of transportation involves three levels of planning: strategic planning, tactical planning, and operational planning [314]. Strategic level management focuses on the decisions about the location of facilities (e.g. the location of depots and fleets), while the key tactical issues are terminal operation specification, service selection and other mixed decision making. Strategic planning and tactical planning are the preconditions to transportation problems, and they are long-term and medium-term planning. Last but not least, the operational planning concentrates on the Vehicle Routing and Scheduling Problem (VRSP), which is the major issue of the Ningbo Port problem.

This chapter provides the background knowledge for vehicle routing and scheduling problems. By adding or relaxing the constraints in problem models according to various real-world scenarios, a large number of VRSP models have been proposed in the literature. Section 2.1 reviews the definitions and models of the classic Vehicle Routing Problem (VRP) and a number of variants relating to this study. Diverse objective functions are also discussed. Section 2.2 introduces the methodologies for VRPs, while more attention is spent on approximate methods for their powerful performance on solving real-life large-scale problems. The research on the full-truckload transportation problem and service network design problem are reviewed in Sections 2.3 and 2.4. Section 2.5 provides a summary and conclusions of the previous studies discussed in this chapter, introducing the research direction of this thesis. To facilitate easier referencing and reading, a number of common abbreviations of classic VRP variants and relevant research domains are listed in Table 2.1.

Table 2.1: Abbreviations of the classic vehicle routing problems and relevant research domains.

Abbreviation	**Full Name**
CVRP	Capacitated Vehicle Routing Problem
DARP	Dial-A-Ride Problem
DVRP	Dynamic Vehicle Routing Problem
FTL	Full Truckload Transportation Problem
MOVRP	Multi-Objective Vehicle Routing Problem
MPVRP	Multi-Period Vehicle Routing Problem
m-TSPTW	Multiple Travelling Salesman Problem with Time Windows
OVRP	Open Vehicle Routing Problem
PDP	Pickup and Delivery Problem
PVRP	Periodic Vehicle Routing Problem
PVRPTW	Periodic Vehicle Problem with Time Windows
SNDP	Service Network Design Problem
SP	Swapping Problem
TSP	Travelling Salesman Problem
VRP	Vehicle Routing Problem
VRPB	Vehicle Routing Problem with Backhauls
VRPPD	Vehicle Routing Problem with Pickups and Deliveries
VRPPDTW	Vehicle Routing Problem with Pickups and Deliveries with Time windows
VRPTW	Vehicle Routing Problem with Time Windows
VRSP	Vehicle Routing and Scheduling Problem

2.1 The Vehicle Routing Problem

Vehicle Routing Problem (VRP) is one of the mostly investigated combinatorial optimisation problems in operational research. The early research on VRP can be traced back to [88], which provides a planning method for dispatching trucks to deliver gasoline from a terminal to a number of geographically scattered service stations with the minimal total mileage. Since then, more and more research is conducted exploring high performance planning algorithms and extending the problem definition by considering the real-life associated constraints. The first use of the name "vehicle routing" is found in [65]. Nowadays, the terminology of VRP may have two concepts in narrow and broad senses respectively. The former refers in particular to the Capacitated Vehicle Routing Problem (CVRP) while the latter includes all the extended problem variants and relevant applications.

CVRP is often seen as a generalisation of the classic Travelling Salesman Prob-

lem (TSP) by considering multiple routes. TSP can be described like this: given a number of scattered cities and the distances between any two cities, schedule a route connecting all the cities with the minimal travel distance for the salesman. The scheduled route starts from a city and terminates at it, while each of the other cities is visited exactly once, called a Hamiltonian Cycle. More details of TSP can be seen in [10].

In CVRP, similar to the TSP, there are a number of customers with demands to be serviced/picked up. However, in CVRP, the customers are serviced by a fleet of K vehicles instead of only one salesman, leading to multiple routes. At the beginning of a planning horizon, the fleet parks at a central facility (*depot*), which is the starting and terminating point of the scheduled routes. All customer demands collected will be unloaded to the depot eventually. These vehicles are identical with a homogeneous capacity Q. The CVRP can be formulated as follows: let $G = (V, A)$ be a directed graph where $V = \{v_i,\ i = 0, \ldots, n\}$ denotes the depot (v_0) and n customers ($v_i,\ i = 1, \ldots, n$). A non-negative service demand q_i is associated with v_i, while $q_0 = 0$. A is a set of arcs with non-negative weights d_{ij} (which represents distance) between v_i and v_j ($v_i, v_j \in V$). A decision variable $X_{ij}^k = 1$ if the arc from v_i to v_j is assigned in route k ($k \in K$); Otherwise $X_{ij}^k = 0$. The objective function is defined as in (2.1),

$$Minimise \qquad \sum_{k\in K}\sum_{v_i\in V}\sum_{v_j\in V} X_{ij}^k \cdot d_{ij} \tag{2.1}$$

Subject to:

$$\sum_{k\in K}\sum_{v_j\in V} X_{ij}^k = 1 \qquad \forall\, v_i \in V\backslash\{v_0\} \tag{2.2}$$

$$\sum_{k\in K}\sum_{v_i\in V} X_{ij}^k = 1 \qquad \forall\, v_j \in V\backslash\{v_0\} \tag{2.3}$$

$$\sum_{k\in K}\sum_{v_i\in V}\sum_{v_j\in V\backslash\{v_0\}} X_{ij}^k = n \tag{2.4}$$

$$\sum_{v_j\in V} X_{0j}^k = 1 \qquad \forall k \in K \tag{2.5}$$

$$\sum_{v_i\in V} X_{ij}^k - \sum_{v_l\in V} X_{jl}^k = 0 \qquad \forall\, k \in K, v_j \in V\backslash\{v_0\} \tag{2.6}$$

$$\sum_{v_i\in V} X_{i0}^k = 1 \qquad \forall\, k \in K \tag{2.7}$$

$$\sum_{v_i\in V}\sum_{v_j\in V} X_{ij}^k \cdot q_i \le Q \qquad \forall k \in K \tag{2.8}$$

$$X_{ij}^k \in \{0, 1\} \qquad \forall v_i, v_j \in V, k \in K \tag{2.9}$$

Objective (2.1) minimises the total travel distance of the fleet. Constraints (2.2)-(2.4) limit every customer to be served exactly once and all customers are visited. Constraints (2.5)-(2.7) define the route by vehicle k. Constraint (2.8) guarantees the feasibility with respect to the service demands and capacity constraint (Q) on vehicles. Constraint (2.9) defines the domain of the decision variable X_{ij}^k. Other formulations of the CVRP can be found in [19, 254], while the objective might be different as well, e.g. minimising the number of vehicles used, or a hierarchical one including more than one objective.

Both TSP and VRP have been proved to be NP-hard [191]. "*However, from a practical point of view, VRPs are in general much more difficult to solve than TSPs of the same size.* [183] " Previous studies have shown that, the TSP instances with thousands of nodes can be solved to optimality [143], while the VRP instances with more than 100 nodes are unsolvable in general [125]. Johnson [160] thoroughly analyses the hardness of TSP and VRP, and a further discussion on the algorithm running time can be found in [1].

With in-depth research, based on the basic CVRP model, a large number of VRP variants are proposed to reflect the real-world scenarios, for example, by considering more practical constraints (called side constraints) or objectives in the problem definitions. Different classification schemes have been proposed for describing and categorising the VRP variants, aiming to improve the management of models and algorithms on this domain. Eksioglu et al. [101] make a taxonomic review of the existing articles on VRP at that time, extensively discussing the various aspects of the VRP research. Recently, greater attention has been devoted to the so-called Rich-VRP considering the planning at both strategic and tactical levels, such as multiple constraints related to managerial and physical characteristics [181]. Some common features and constraints in VRP research are listed as follows:

- **Capacity consideration**. This feature reflects the relation between the vehicle capacity and customer demands. To uncapacitated vehicles, the number of customers/demands on each route is unlimited, e.g. the postman problem. On the other hand, in capacitated problems, apart from the capacity limit, load splitting [11] and consolidation [61] may also be considered to increase the utility of vehicles in logistics.

- **Time constraints and time horizon**. In real-life scenarios, customer demand usually associates with a time window for the time of service being conducted. According to the tightness of the constraints, the time windows

can be divided into soft time windows [174], strict/hard time windows [282], and mix of both [271]. In addition, the complexity of the problem can be further increased if the planning horizon is composed of multiple periods instead of a single one.

- **Maximum travelling constraint**. This is a common constraint in VRP research [187], which can be embodied in the forms of maximum travelling distance, travelling time, and the number of customers serviced, etc. Vehicles have to return to the depot or park at the last serviced customer before reaching the largest travelling length allowed. This constraint could be caused by the fuel capacity of vehicle, the associated regulations about the continuous working hour in law, or the workload impartiality.

- **Vehicle homogeneity**. The vehicles usually are homogeneous in VRPs. However, due to the diversity of customer demands in Rich-VRP, recently, the problems with heterogeneous vehicles are seen more frequently in the literature. For example, the frozen food must be transported by refrigerated truck instead of normal vehicles. In heterogeneous fleet vehicle routing problems, the vehicles used can be different on capacity, size, functionality and other aspects, such as fuel consumption, driver salary and other operational cost [18].

- **Number of points of loading/unloading facilities (depot)**. In some studies, the customer requests are supported by multiple fleets scattering at different depots. This type of Multi-Depot Vehicle Routing Problems (MDVRP) can be divided into *fixed* or *non-fixed*, according to the vehicles that are required to return to their origin depots or any depot at the end of the scheduled routes [168]. Most research about MDVRP focuses on the fixed problems.

- **Route structure**. The constraint of returning to the starting depot can be relaxed in practice. When the vehicles used are rented from third party logistic providers, the vehicles pick up goods from the central depot and deliver them to customers. Since the rented vehicles do not need to return to the depot, the scheduled routes terminate at the last serviced customers [103]. This type of routes are Hamiltonian paths (called *open* routes/trips) instead of Hamiltonian Cycles (closed routes/trips).

- **Representation of the service targets**. Different from the above models in which the service and resource constraints are on the nodes, in some other

problems, service and constraints are associated to the arcs between nodes, e.g. the postman problem, refuse collection, meter reading [34, 152]. This type of problems are classified as the Arc Routing Problems (ARP) while the previous CVRP model represents a Node Routing Problem as the counterpart. When solving Capacitated ARP, the summation of the service on arcs of a route should not be larger than the capacity of a vehicle [316].

- **Information Characteristics**. In *Static* VRPs, all the customer demands, vehicle and traffic information are known in advance. Differently, those information is input during the execution of planning in *Dynamic* VRPs (D-VRPs), which are also referred to as real-time or online VRPs in the literature [243]. In addition, the problems can also be classified into *Deterministic* VRP and *Stochastic* VRP depending on the input information is deterministic or stochastic. In Stochastic VRP, some random variables are realised with a known probability distribution during the planning procedure [128]. Considering that almost all the customer requests in the Ningbo Port problem can be obtained 24 hours in advance, the study in this thesis focuses on the static deterministic VRP. For recent reviews on dynamic and stochastic VRP, see [31, 259].

- **Extra Considerations**. In real-life situations, because of the limitation of resource or management policies, some extra conditions have to be taken into account when making routing decision. These conditions may have no relation with the fleet. For example, the inventory routing problem integrates inventory management with routing problem. This problem is found in the environment where vendor managed inventory resupply policies are applied [78]. While in the case of workforce scheduling and routing problem, the skills and qualifications of the service providers and the teaming constraints have to be considered when scheduling [180]. These extra considerations drastically extend the VRP research.

2.1.1 Vehicle Routing Problem with Time Windows

Time window constraint is first introduced to VRP by Solomon [283], indicating the fact that the customers are used to impose the available time windows for servicing their demands. Based on the above CVRP model, the Vehicle Routing Problem with Time windows (VRPTW) can be modeled as follows: to customer v_i, the service starting time b_i must be in a time window $[e_i, f_i]$, where e_i and f_i are the earliest and latest time to serve q_i. If a vehicle arrives at v_i at time $a_i < e_i$, a

waiting time $w_i = max\{0, e_i - a_i\}$ is required. Consequently, the service start time $b_i = max\{e_i, a_i\}$. Each vehicle travels on a route connecting a subset of customers starting from v_0 and ending within schedule horizon $[e_0, f_0]$. The decision variable $X_{ij}^k = 1$ if the arc from v_i to v_j is assigned in route k ($k \in K$); Otherwise $X_{ij}^k = 0$. The problem can be formulated as follows [59]:

$$Minimise \qquad NV = K \tag{2.10}$$

$$Minimise \qquad TD = \sum_{k \in K} \sum_{v_i \in V} \sum_{v_j \in V} X_{ij}^k \cdot d_{ij} \tag{2.11}$$

Subject to:

$$\sum_{k \in K} \sum_{v_j \in V} X_{ij}^k = 1 \qquad \forall\, v_i \in V \backslash \{v_0\} \tag{2.12}$$

$$\sum_{k \in K} \sum_{v_i \in V} X_{ij}^k = 1 \qquad \forall\, v_j \in V \backslash \{v_0\} \tag{2.13}$$

$$\sum_{k \in K} \sum_{v_i \in V} \sum_{v_j \in V \backslash \{v_0\}} X_{ij}^k = n \tag{2.14}$$

$$\sum_{v_j \in V} X_{0j}^k = 1 \qquad \forall k \in K \tag{2.15}$$

$$\sum_{v_i \in V} X_{ij}^k - \sum_{v_i \in V} X_{ji}^k = 0 \qquad \forall\, k \in K, v_j \in V \backslash \{v_0\} \tag{2.16}$$

$$\sum_{v_i \in V} X_{i0}^k = 1 \qquad \forall\, k \in K \tag{2.17}$$

$$e_i \leq b_i \leq f_i \qquad \forall\, v_i \in V \tag{2.18}$$

$$\sum_{v_i \in V} \sum_{v_j \in V} X_{ij}^k \cdot q_i \leq Q \qquad \forall k \in K \tag{2.19}$$

$$X_{ij}^k \in \{0, 1\} \qquad \forall v_i, v_j \in V, k \in K \tag{2.20}$$

There are two commonly used objectives in VRPTW. Objective (2.10) aims to minimise the requested number of vehicles (NV), and the second objective (2.11) minimises the total travel distance of the fleet (TD). The two objectives are usually considered in a hierarchical structure, where NV has a higher priority than TD. Other objectives in VRPTW include the minimisation of the total waiting time and so on, which are less studied [163, 282]. Constraint (2.18) guarantees the feasibility with respect to the time constraints on service demands, and all the other constraints place the same requirements as the CVRP model associated in Section 2.1.

The well-known VRPTW benchmark "Solomon Instances" are introduced in [282], which includes six problem sets (C1, C2, R1, R2, RC1 and RC2). The

customers in C1 and C2 are clustered geographically, randomly distributed in R1 and R2, while mix of both in sets RC1 and RC2. The scheduling horizons in C1, R1 and RC1 are short with the smaller capacity of vehicle, leading to fewer customers in each route (approximately 5 to 10). In contrast, the scheduling horizons and vehicle capacities in C2, R2 and RC2 are relatively larger, which means permitting more customers per route (they could be more than 30). The number of customers in each Solomon instance is 100. The scaled down instances with 25, 50 and 75 customers are also used by researchers. Based on the same scheme, Gehring and Homberger [124] generate five groups of instances with larger sizes, i.e. 200, 400, 600, 800, 1000 customers per instance. The Solomon benchmark and Gehring & Homberger benchmark have been extended and applied to many other VRP variants research as well.

VPRTW is a basic model to many other more complex VRP variants, thus great attention has been paid to it in the operational research community. Because of its generality, VRPTW is even used as a benchmark problem to test the performance and general applicability of the methodologies proposed in the research of other VRP variants [50]. In a scheduling problem, the main concern is that *when* the scheduled tasks should be executed or candidate resources should be assigned to specific objects. By considering the time constraints, the VRPTW can be seen as both a routing problem and a scheduling problem. Because the time windows constraints are common in VRPs and the routing problem and the scheduling problem can be reformulated to each other easily, Vehicle Routing and Scheduling Problem (VRSP) is sometimes used as an alternative term of the generalized VRP in the literature [26, 99].

2.1.2 Some Other Vehicle Routing Problem Variants

By considering various side constraints and specific features deriving from real-life scenarios, a large number of extended VRP variants are proposed based on the CVRP and VRPTW. This section introduces three classes of VRP variants sharing the common characteristics with the Ningbo Port problem. The history of the relevant literature is reviewed, presenting the specific examples for each class.

Pickup and Delivery Problem (PDP)

The Pickup and Delivery Problem (PDP) is a broad definition representing the problems which schedule the routes to satisfy customer demands of pickup and delivery of commodities or people from origins to destinations. In [240, 241], the authors divide the PDPs into two classes: the transportation between customers

and depot, and the transportation between pickup and delivery locations. The first class refers to the situation where the depot plays both the roles of the commodities supplier and commodities storehouse. In this case, commodities are either delivered to linehaul customers (with delivery demands) from the depot or delivered to the depot from backhaul customers (with pickup demands). This class is denoted as Vehicle Routing Problem with Backhauls (VRPB). The second class represents the problems in which the commodities are transported only between the customers, denoted as Vehicle Routing Problem with Pickup and Delivery (VRPPD). The classic Dial-A-Ride Problem (DARP) belongs to this class in this classification.

The terms in PDP are often confusing in the literature, because "the same problem types are denoted by various names and different problem classes are referred to by the same denotations" [240]. For example, in [29], the models of VRPPD and DARP are the same and they are only different on the application domain. The authors claim that the VRPPD deals with the transportation of objects while the DARP copes with the transportation of people. However, the VRPPD is a larger concept involving the DARP in the study of [240]. To better describe the features of the large number of PDP variants, great effort has been made to sum up the variants from different aspects. Among various descriptive approaches, Berbeglia et al. [29] propose a three-filed classification scheme for PDPs, which is [Structure | Visits | Vehicles]. This scheme is adopted in this thesis for its brief but comprehensive classification as a descriptive language.

In the scheme defined in [29], the first field *Structure* specifies the constraints on the origin and destination of demands. If all the commodities delivered to customers are picked up from the depot, and all commodities picked up from customers are delivered to the depot, the problem is classified as a One-to-Many-to-One problem (1-M-1). This type of problems can refer to the VRPB in [240]. Whilst the commodities collected from customers can be split or combined to satisfy other customers' delivery demands, it is a Many-to-Many problem (M-M). In addition, in One-to-One problems (1-1), one customer's pickup demand is the delivery demand of another specific customer. Both M-M and 1-1 problems are categorised to the VRPPD in [240], while the VRPPD exactly represents the 1-1 problems in [29]. The second field *Visits* describes the operations performed at customer vertices, which can be Pickup & Delivery pair (PD), one of them but not both (P/D), or P-D which means independent and dependent executions are both permitted. Besides, one more operation Transshipment (T) is also introduced representing the load splitting and consolidation on truck routes. The last field

Vehicles indicates the number of vehicles allowed in the solution, which can be multiple vehicles (m), single vehicle (1) or undefined (-).

In 1-M-1 problems, the demand of each customer can be divided into two types of commodities, namely, pickup commodities and delivery commodities. Multiple delivery commodities can be viewed as a single delivery commodity as long as they share the same origin, while multiple pickup commodities can be seen as a single pickup commodity whenever they share a common destination. Beullens et al. [35] discuss several issues related to this type of problems. Transshipment is not allowed in 1-M-1 problems.

The research of M-M problems mainly focuses on single vehicle scenarios, which indicates the fact that M-M problem is not frequently encountered in practice. The Swapping Problem (SP) discussed in [9] is an instance of M-M PDP, where a single vehicle is used to exchange the commodities between paired customers carrying one commodity each time. In the preemptive version SP, transshipment is allowed which means the commodity can be left at any customer temporarily. It is worthy to note that those single-vehicle PDPs are often treated as TSPs since there is only one route in the scheduling.

The 1-1 PDP is more common than the other two classes in the literature for its generality in real-life. The VRPPD and DARP are the two major problems in this category. In the VRPPD, commodities are picked up and delivered between the specified locations while transshipment (e.g. commodity being left at a location temporarily) is not allowed [52]. The VRPPD with Time Windows (VRPPDTW) has also been studied in which service should be rendered within time windows at both pickup and delivery locations. The VRPPDTW gives rise to the dynamic scheduling problems [48, 220]. In the DARP, the point-to-point transportation service is provided to customers, e.g. the elderly and handicapped people [38, 94]. The customer inconvenience and preference are often taken into consideration in DARP, e.g. time window, vehicle type, and maximum trip duration [74, 76, 77].

In terms of the transshipment, when consolidation is allowed it is a problem of Less-than Truckload Transportation; otherwise, it is a Full Truckload Transportation problem (FTL) [314]. Container transportation problem is a specific variant of FTL, where one truck can carry only one demand item (container). More discussion on FTL is presented in Section 2.3. The container transportation problem of the Ningbo Port can be categorised as a [1-1 | PD | m] FTL problem, which means a multi-vehicle One-to-One PDP where the pickup and delivery operations can be executed simultaneously at each container terminal.

Multi-Period Vehicle Routing Problem (MPVRP) and Periodic Vehicle Routing Problem (PVRP)

In some cases of VRP, the mid-term or long-term planning is considered, e.g. in soft drink industry, grocery distribution and waste collection. Their scheduling is usually performed over a week or a month, and the associated problems are notated as Multi-Period Vehicle Routing Problem (MPVRP) [224]. Especially, when there is a specific service frequency to each customer over the scheduling horizon, the problem is called a Periodic Vehicle Routing Problem/Period Vehicle Routing Problem (PVRP) [89, 116]. In this case, each customer may be visited more than once. In other words, the other MPVRP variants are PVRPs with the customer service frequency of one in the planning horizon. The solution of PVRP is usually represented as a combination of service shifts of customers, instead of the scheduled routes of shifts.

The MPVRP is common in practice. Kim and Kim [173] study a multi-period retailer replenishment problem, where a central depot supplies multiple retailers with products of a single type. One vehicle can service only one retailer in one period in this problem. Bostel et al. [39] solve a problem of the planning and routing of technician visits, where each customer is visited exactly once by a technician. A multi-period vehicle routing and crew scheduling problem is introduced in [320]. During the horizon of multiple days, all shipments are collected to the depot from the pickup customers and then dispatched to the delivery customers. Before dispatching the shipments, the type of vehicle and the associated driver should be scheduled satisfying a number of side constraints. The feature of multi-shift adds one more dimension to the problem solution, which drastically increases the complexity of problem. Thus, the MPVRP is harder than the other VRP variants in general, and most existing solution methods for it are heuristic-based or hybrid [39].

Dynamic Multi-Period Vehicle Routing Problem (DMPVRP) is an important domain in the research, where the demands of customers are revealed over time incrementally. For example, in [7], the customer pickup request made in real-time would be either served on the same day or postponed to the next day, and one uncapacitated vehicle is scheduled to satisfy those requests. In [8], two short term strategies and a re-optimisation framework are proposed for the long-term optimality in DMPVRP. A three-phase approach for DMPVRP can be found in [313], which solves the sub-problems of partial periods in the first two phases, and then the third phase is applied as a post-optimisation procedure for the optimality over the whole planning horizon.

The earliest research on PVRP is in [28], which studies a waste collection problem. The garbage sites in this problem are visited with different frequencies. Comparing with the standard CVRP, the authors make a good exposition of the complexity of PVRP and propose the heuristic solution methods. A formal definition of PVRP is given by Russell and Igo [265] notated as the assignment routing problem. The method in [224] combines the Column Generation and a rounding heuristic, solving the instances with 50-80 customers over five days which is the size of most instances in the PVRP research.

The side constraints, e.g. the time windows, are considered in PVRP research as well. The first research on Periodic Vehicle Problem with Time Windows (PVRPTW) is in [80]. Considering travel time, capacity, duration and time windows, a construction heuristic followed by a Tabu Search (TS) is proposed. An improved TS method adopting the *Forward Time Slack* [272] is later proposed to further reduce the route [81]. An Ant Colony Optimisation (ACO) model for PVRPTW can be found in [318] where a multiple pheromone information matrix is designed. The flexibility of customer visiting date in PVRPTW further increases the number of parameters and the complexity of this algorithm. More solutions for PVRPTW and other PVRP variants can be found in [116, 244, 253].

Open Vehicle Routing Problem (OVRP)

When a company does not own a fleet or its vehicles are inadequate or inappropriate to satisfy its customers, its delivery job might be outsourced to third party logistics (3PL) providers. In practice, more and more companies use hired vehicles (e.g. private cars, common carrier vehicles) and allocate their distribution function to 3PL providers for operational cost reduction. Although the use of hired vehicles is expensive, the cost per unit distance travelled still decreases since without paying vehicle maintenance fee [290]. Different from the classic VRP with closed trip, those hired vehicles do not need to return to the depot when finishing all the deliveries. Accordingly, the companies hiring the vehicles do not pay the mileage back to the depot. The scheduled routes terminate at the last customers serviced leading to *open* trips, and this type of VRPs are denoted as Open Vehicle Routing Problems (OVRPs) [194].

The earliest description of OVRP is presented by Schrage [274], who introduces the notation of "open trip" and provides an air express courier instance. Eppen and Schrage [103] propose the formal OVRP definition introducing a distribution problem between a central depot and multiple warehouses. From the early 1980s to the late 1990s, OVRP attracted little attention from the VRP researchers. However,

with the prevalence and success of metaheuristics, a large number of methodologies for OVRP have been developed since 2000, giving rise to the research on it [269].

Tarantilis et al. [291, 292] propose two local search metaheuristics which use annealing based and list based threshold acceptance criteria. Both algorithms outperform previous approaches for OVRP. Two TS-based metaheuristic algorithms for OVRP can be found in [47, 117]. The Record-to-Record Travel algorithm is later adopted as the acceptance criterion by Li et al. [194], and the associated algorithm generates better solutions than 11 previous algorithms for OVRP. A static move descriptor is proposed in [319] to speed up the evaluation in best improvement search. The associated algorithm produces the best results on benchmarks of OVRP. However, it is not applicable to OVRP with Time Windows (OVRPTW) where time constraints are considered. The first exact method for OVRP is proposed by Letchford et al. [192], which is only tested on small and medium size instances (less than 151 customers and 14 vehicles).

The first introduction of the OVRPTW is in [257]. In the proposed solution methodology, an insertion-based construction heuristic employs an improved *IMPACT* criterion [157] to select customers to be inserted. Since then, many metaheuristic methods are developed for OVRPTW. A Variable Neighbourhood Search algorithm can be found in [242], which outperforms the *IMPACT* approach. An ACO algorithm and a Genetic Algorithm are also applied to the problems of relatively small size [141, 142].

In a specific case of OVRPTW, the routes start from geographically scattered customers with goods to be picked up. After visiting the customers assigned, vehicles return to the central depot to unload the goods collected. The routes in this case are open at the starting point, which is opposite to the standard OVRP. An adaptive large neighbourhood search algorithm is proposed for this Reverse-OVRPTW in [273]. School Bus Routing Problem is another special case of OVRP, where its morning and afternoon routing problems can be addressed as the same problem. Every morning, school buses send students to the school from pickup stops, while in the afternoon students are sent back to the stops in the reverse order of the morning routes [238]. In addition, Liu and Jiang [203] and Brito et al. [51] study the VRP with both open and close routes.

An Open Periodic Vehicle Routing Problem (OPVRP) model is introduced in [86]. In this case, vehicles are not obliged to return to the depot at the end of each day in a planning horizon of multiple days. In the proposed solution construction heuristic, a k-means clustering algorithm is used to assign customers to routes, without considering the capacity limit. A feasibility procedure would then fix the

infeasible assignments. This approach can quickly generate a feasible solution for small size problems. However, it would be inefficient for larger problems or when the depot is not in the geographic center. Time constraints are not considered in this model.

After over half century research, many VRP variants are formulated reflecting the real-world demands in the industry. It is undoubted that more variants will be proposed with the popularisation of automatic routing scheduling technologies and driverless technologies. More existing VRP variants in the literature can be found in [139, 181, 298].

2.1.3 Multi-Objective Vehicle Routing Problem

Most early VRP models are single-objective and focus on optimising the trip cost (e.g. distance, time), reflecting the financial targets of logistics companies. However, many real industrial problems deal with multiple objectives which are not limited to costs. In fact, when considering the financial costs to a logistics company, there are multiple factors that are involved, e.g. fuel, driver salary. Apart from costs, other aspects like fairness, lateness, environmental factors can be taken into account in VRP research as well. These types of VRPs fall into the Multi-Objective Optimisation Problem (MOOP), which deals with more than one objective function [100]. The research on MOOP is rooted in the works of [98] and [237] while the context is on economic research. It is denoted as a Multi-Objective Vehicle Routing Problem (MOVRP) when the problem background of the MOOP is a vehicle routing problem [163].

From Single-Objective to Multi-Objective

The study of MOVRP mainly brings three aspects of changes to classic single-objective VRP research: a) without losing the insight of the original objectives, it can improve the practical application of VRP study by extending the single-objective problem definitions and models. b) generalise the classic problems. c) take the objectives clearly identified by the decision-maker in real-life cases into consideration, expanding the research scope.

By reconsidering and/or reconstructing the initial objective, the classic problems can be extended to multi-objective. For example, the driver workload is an important factor relating to the driver welfare which may affect the operation of companies, and drivers would complain when they find disparity in their schedules. The balance of route length, which may reflect the parity of working hour,

is thus considered in [190]. Lee and Ueng [190] use a balance objective, associating the fairness into the task allocation. To improve the customer satisfaction with regard to delivery dates, extra constraints on the time windows are added in [276]. In addition, in commercial distribution industry, marketing issues such as maintaining the relation between the customer and the driver can also be involved in objective function, since they are important for improving company reputation and sales [258]. Some revised classic objectives are adopted in MOVRPs. Chitty and Hernandez [63] define a Dynamic VRP which seeks to minimise the mean travelling time and the variance of travelling time over all scheduled routes. The two objectives in [225] are minimisation of the number of vehicles used and maximisation of the travelling time over all routes. Two similar objectives of minimising the maximum/longest route of a single vehicle and the average route length are studied in [284].

Generalising the constraints and/or parameters in classic models to objectives is another manner of MOVRP study. For example, in most CVRP studies, the number of vehicles used (NV) is treated as an input parameter or constraint while the only one objective is to minimise the travel distance (TD) [254]. However, in some bi-objective CVRP research, an hierarchical objective is adopted considering the NV first and the TD secondly [49]. Obviously, this type of problems can be easily solved with single-objective methods by considering the objectives sequentially. In contrast, in the study of [230, 131, 300], the NV and TD are equally treated. A list of implicitly multi-objective routing problems can be found in [37].

The objectives defined by decision-makers in real-life cases is an essential source to MOVRP research. In the delivery routing problem of [102], the goods are delivered to customers from a location which is not the depot, and eight objectives are considered during the scheduling. In the urban schoolbus route planning problem of [40], four objectives are defined based on realistic operational consideration: the minimisation of the total route length and student walking distance, and the fairness of distribution of the load and workload to buses (i.e. the number of students transported and travel distance of a single bus). In a real-life multi-period vehicle routing problem [223] with up to 6000 customers over a planning horizon of 20 days, the solution with balanced workload and compact geographical clusters of customers is pursued. Doerner et al. [95] study a stop location and routing problem for mobile healthcare facilities in developing countries. The associated objective functions are defined from three aspects: the efficiency of workforce deployment, average accessibility and coverage. The first objective maximises the ratio of time spent on medical procedures over the total time spent, including the time of

travelling and facility setting up. The second one minimises the average walking distance from the habitants to their nearest stops. The last one is to maximise the number of habitants in the area within a specified walking distance. It can be expected that more distinct objectives will be proposed according to the practical considerations in specific problems.

Multi-objective Optimisation Algorithms

Generally, the solution methodologies for MOOPs can be divided into three classes. In the first class (*Pareto Methods*), a set of *non-dominated solutions* based on Pareto Dominance [100] are generated, providing the decision makers a pool of candidate solutions as a reference. In the literature, the *Pareto Methods* are mainly used in Evolutionary Algorithms [70]. Differently, in the other two classes of algorithms, one single optimal solution is pursued. In this case, either the problem studied is converted into single-objective and then is solved as a single-objective problem (*Scalar Techniques*), or those different objectives are considered separately (*Non-Pareto and Non-Scalar Algorithms*).

Pareto Methods use the notation of Pareto Dominance when comparing solutions. This type of methods do not allow a solution compromise to be favoured over another, but all non-dominated solutions are kept which is composed of the so-called Pareto Optimal Set. The Pareto Dominance was introduced to Genetic Algorithm by Goldberg and Holland [138], and then is widely used in evolutionary computation and MOVRPs [179, 230]. Apart from evolutionary algorithms, Pareto Dominance is also employed by other methods. For example, Ulungu et al. [299] propose a Simulated Annealing method for MOOP, and a similar idea is applied to VRPTW by Baños et al. [22].

Among various *Scalar Techniques*, the most popular one is the weight linear aggregation (also called Weight-Sum Approach), which linearly combines multiple values of objectives into one scalar value. Specific weights are assigned to different objectives. With this method, the studied problems are converted into single-objective problems pursuing the optimal scalar value. This method is widely used in heuristic methods [40], local search approaches [258] and genetic algorithms [230]. However, the weight linear aggregation still has some disadvantages in solving MOOPs. Firstly, it is hard work to determine the weights of objectives according to the importance of the objectives. In addition, this method does not save all non-dominated solutions, i.e. only the solutions on the convex hull of Pareto optimal set are stored.

Another type of prevailing Scalar Techniques use goal programming. In this type of methods, a point in the objective space (*goal*) is selected first, then a search is conducted to minimise the distance between the current solution and the *goal*. A difficulty here is the definition of the *goal*. In [239], the goal is defined according to decision-maker preferences on the objectives, while the goals in [131] are defined by pre-defined formulas. Jozefowiez et al. [161] propose a dynamic goal point defining technique, where multiple Tabu Searches are conducted. In the method, the goal for each search is iteratively computed based on the starting solution's neighbour solutions.

One other well-known scalar method is the ϵ-Constraint method, in which only one objective is kept while the other objectives are considered as constraints: $f_i(x) \leq \epsilon_i$ [144]. The MOOP is transformed to a single-objective problem with additional constraints. Here, the ϵ_i is a user-specified restricting value to the objective $f_i(x)$. Changing the value of ϵ_i yields different solutions. Corberán et al. [73] and Pacheco and Martí [234] use local search to optimise the route length (first objective) for the different numbers of vehicles used (second objective). Within the same framework, Jozefowiez et al. [162] use a branch-and-cut algorithm to solve the single-objective problem obtained.

Some researchers use *Non-Pareto and Non-Scalar Algorithms* to address MOVRP, which neither convert the problem studied into a single-objective problem nor maintain a Pareto optimal set during the search. Doerner et al. [95] propose the vector evaluated genetic algorithm in which the solution population is divided into n subpopulations, while n is the number of objectives. Each subpopulation is associated with a specific objective. In [169], the objectives are considered lexicographically, and the optimised objective would be fixed as a constraint when considering the remaining objectives. This lexicographic method is also used by Current and Schilling [84]. In the ant colony system of [63], two types of pheromones are defined according to the two objectives in the associated DVRP, while Doerner et al. [95] consider the objectives simultaneously proposing the heuristic of Pareto ant colony optimization.

Some basic principles can be used when studying a MOVRP. First of all, whether Pareto Dominance is considered and whether Pareto optimal solution set is maintained during the search should be confirmed. This is an essential issue to the problem definition, which greatly determines the type of the associated solution methodologies used. Secondly, in terms of the solution methodologies, the Pareto methods with population strategy have shown powerful performance in solving MOVRP, which are capable to deal with any type of problems with

any number of objectives, e.g. multi-objective evolutionary algorithms. However, their high requirements on computation resource have to be dealt with properly. To those MOVRPs arising from adding new objectives to the classic VRP models, scalar methods such as the goal programming and ϵ-Constraint method can easily produce the corresponding single-objective problem models. These models then can be addressed with a huge number of existing algorithms. Last but not least, integration between different methods is an important element for successful multi-objective methods. For instance, using local search such as tabu search as a post-optimisation phase in genetic algorithm [234] and using exact methods such as branch-and-bound to solve the sub-problem in evolutionary algorithm [162] can significantly improve the quality of solutions.

2.2 A Review of Solution Methodologies for Vehicle Routing Problems

After decades of study in VRP, both *exact methods* and *approximate algorithms* have been extensively investigated. Exact methods (also called mathematical methods) explore the solution space of a problem completely. Usually, they solve the problems to optimality, but some exact methods terminate when the ending conditions are met and return the best solution found without guarantee of optimality. A critical issue of exact methods is the unrealistic computational time needed for searching the enormous size of the solution space in large scale problems. On the other hand, approximate algorithms approach toward the optimal solution with the guidance of "rule of thumb" or domain knowledge, which mostly are heuristic methods in VRPs. They do not guarantee the optimality of solutions produced, but generate an approximation of the optimal solution in an acceptable computation time [287]. In approximate algorithms for VRPs, the initial and classic heuristics are route construction and improvement algorithms. Nowadays, more and more Metaheuristics and Hyper-Heuristics methods are developed showing powerful performance in solving diverse large scale and complex VRPs since 1990s [50]. This subsection reviews the well-known exact methods and different types of heuristic algorithms for VRPs.

2.2.1 Exact Methods

Branch-and-Bound

The branch-and-bound method is first proposed in [182] and introduced to TSP by Little et al. [202]. It is a state space search method which explores solutions in a rooted tree. Starting from the root, the algorithm explores the tree branches which represent the solution subsets, searching the optimal solution. To reduce the computation time spent on low quality solutions in the algorithm process, before enumerating the solutions on a branch, the branch is checked against the *lower bound* and *upper bound*. Those branches which cannot provide better solutions are excluded from the searching tree with the solutions on them unexplored (called pruning). For example, in a minimisation problem, the upper bound is the bottom line of the optimal solution's objective value, which is updated during the search. The lower bound in this case is the estimation of the optimal objective in the current branch. When the lower bound is larger than the upper bound, the branch is pruned.

The name of "branch-and-bound" is proposed by Balas and Toth [17] for TSP, while Laporte et al. [186] present the first branch-and-bound method for asymmetric CVRP. The efficient estimation of the lower and upper bounds is an essential element to a successful branch-and-bound algorithm. Based on the bounding procedure from [107], Fischetti et al. [108] introduce an improved branch-and-bound algorithm for the asymmetrical CVRP, which solves the instances with 300 customers and four vehicles. More recently, Almoustafa et al. [4] develop a branch-and-bound method for solving the asymmetric ditance-constrained vehicle routing problem on the instances with up to 1000 customers. However, there is much less progress made on VRPTW and other tight-constrained problems.

Branch-and-Cut

The branch-and-cut method combines the branch-and-bound and cutting plane techniques. Given a Combinatorial Optimisation Problem (COP) formulated as an Integer Linear Program (ILP), a *cutting plane* is a linear inequality that is satisfied by all feasible solutions to the ILP, but violated by a feasible solution to the Linear Programming (LP) relaxation that is obtained by removing the integral constraint from the ILP. Cutting planes are added to the LP relaxation until it finds an integral optimal solution, which means the associated ILP is solved. The branch-and-cut method applies this cutting plane method at every node of the branch-and-bound tree. When applying the branch-and-cut method, the LP relaxation of a COP is solved with a LP solver first, which is an easier job than solving the ILP. If the solution of the LP model is integral, then the ILP is solved optimally; If not, either a *branching* or a *cutting* can be used. In *branching*, two complementary

constraints are added to a picked variable respectively, splitting the original ILP into two sub-problems. The two sub-problems are then solved with branch-and-bound methods. On the other hand, *cutting* is executed when the LP relaxation is hard to solve with LP solver. For example, when the ILP contains a large number of linear constraints, or adding valid inequalities to the LP relaxation significantly increases the complexity of model, the cutting planes methods can be used for the ILP model. Because the branch-and-cut method performs well on addressing COPs, it is used by almost all ILP packages.

Branch-and-cut has been widely used in solving routing problems. It is able to solve the instances with hundreds or thousands of nodes for TSP [10], while the progress in VRP is relatively less. The early branch-and-cut method introduced by Laporte et al. [185] is capable of solving instances with 60 customers for the distance-constrained CVRP. Taking advantage of the polyhedral study of CVRP from [82], the branch-and-cut in [12] solves some CVRP instances with up to 134 customers. A branch-and-cut algorithm for VPRTW can be found in [23]. On the Solomon instances with 50 and 100 customers, the proposed algorithm produces optimal solutions with up to 27 minutes of computation time. The main reason for the relatively poor performance in VRP is that branch-and-cut does not work well on highly-constrained COPs, while VRP is constrained more tightly than TSP. It works the best when the solution region polyhedral (i.e. the shape of promising searching region in solution space) is easy to analyse [293].

Lagrangian Relaxation

Lagrangian Relaxation methods relax the hard constraints in ILP by converting the violation of the constraints to the penalty on objective value. In terms of a minimisation problem, the relaxed constraints multiplying non-negative Lagrangian multipliers are added to the objective function, leading to the relaxed Lagrangian problem. The main motivation of Lagrangian Relaxation method is that, the Lagrangian problem is easy to solve and the solution of the Lagrangian problem provides a lower bound of the original ILP problem. Held and Karp [148] and Held et al. [149] determine the values of Lagrange multipliers with the *subgradient optimisation*, obtaining good results in TSP. Miller [217] introduces a method of generating better lower bound with Lagrangian Relaxation in CVRP. Two recent comprehensive introduction of Lagrangian Relaxation methods for ILP problems can be found in [1] and [110]

In the relaxation-based methods for VRP, diverse constraints relaxations are adopted. For example, in VRPTW, the time window and capacity constraints

can be relaxed, which transforms the problem to a network flow problem. This relaxation relies on high quality lower bound and balance of complexity over subproblems. This method produces the lower bound with a similar quality as that obtained by the linear programming relaxation [78, 298]. In general, better lower bounds are obtained when fewer constraints are involved in the Lagrangian relaxation. In the VRPTW, for better Lagrangian relaxation, some complicating constraints are suggested to be kept by Toth and Vigo [298].

In [112], the authors formulate the VPRTW as a minimum K-tree problem, which is solved with the Lagrangian method from [109]. This method solves a few Solomon C1 instances including four 100-customer instances. By using subgradient and bundle methods, Kohl and Madsen [175] relax the demand constraints solving 27 C1 instances with 25 - 100 customers to optimality. Kallehauge et al. [167] solve the Lagrangian dual problem by embedding the stabilised cutting-plane algorithm in a branch-and-bound method. The proposed algorithm shows competitive performance on solving C1 and C2 problems while optimally solving two Gehring & Homberger C1 instances with 400 and 1000 customers respectively, but it fails to tackle R and RC 100-customer problems within one hour generally.

Column Generation

Column Generation was first designed to solve the LP problems in which the number of variables is huge while the constraints are few [119]. It is described as "*a special way of updating the multipliers associated with the relaxed constraints*" by Cordeau et al. [78]. With this method, a restricted master problem is created which only considers a subset of constraints of the original problem. The other constrained variables (columns) are added one-by-one to the restricted master problem until no negative reduced cost can be generated. The first Column Generation for VRPTW is proposed by Desrochers et al. [93] in which the added columns are obtained by solving the shortest path problem with time windows. 29 Solomon instances including five 100-customer C1 instances are solved with the algorithm. Furthermore, Kohl et al. [176] use 2-path cut to produce better lower bounds, solving 70 of 87 Solomon instances with short horizons. However, both Desrochers et al. and Kohl et al. algorithms fail to produce optimal solutions on the instances with random customer and long horizons. Irnich and Villeneuve [158] improve the lower bound with k-cycle elimination solving 15 previously unsolved instances. Both exact and heuristic dominance rules are considered in [58], where the exact one is used when the heuristic approach cannot find the negative cost.

This method solves 17 long-horizon Solomon instances, but still fails on most 100-customer instances. The application of Column Generation to other VRP variants can be found in [224, 64].

Branch-and-Price

The "branch-and-price" method was first introduced by Savelsbergh [270]. It represents the approaches using Column Generation to solve the LP relaxation of the master problem in the branch-and-bound framework. A separation problem (pricing problem) is solved to identify the columns to be added to the restricted master problem [24]. Fukasawa et al. [118] introduce Column Generation into branch-and-cut, proposing the branch-and-cut-and-price for CVRP. The proposed algorithm solves the instance with up to 135 customers to optimality. In [87], a hybrid method combining Column Generation and local search is devised for VRPTW, where the branch-and-price is employed to improve the search diversification. An extensive tutorial on column generation and branch-and-price for routing problems is presented by Feillet [106].

2.2.2 Classical Heuristics

Classical heuristics usually produce good quality solutions in moderate running time, and they can be adjusted to address real-life problems with easy extension. However, the quality gaps between the solutions produced and the optimal solutions are often large on hard instances, since the classic heuristic methods explore a relative limited solution region in general. In this thesis, the classification scheme in [78] and [184] is adopted, which classifies the classical heuristics for VRP into three categories: construction heuristics, two-phase methods and improvement heuristics.

Construction Heuristics

Construction heuristics typically start from an empty solution with no route or customer in it, then the complete solution is constructed iteratively. At each iteration, one or more customers are added to the routes constructed in previous iterations until all customers are routed. A construction heuristic is specified by three components, namely initial criterion, selection criterion and insertion criterion. An initial criterion initialises the routes in a solution, a selection criterion specifies the customers to be added at the current iteration, while an insertion criterion determines where and how to locate the customers specified. Furthermore,

construction heuristics can be categorised as sequential or parallel according to the number of candidate routes considered at each iteration. Sequential methods expand only one route at every iteration, while multiple routes are simultaneously considered in parallel methods.

The first well-known construction heuristic is the *Saving Algorithm* proposed by Clarke and Wright [68]. This method constructs solution based on the concept *saving*, which is the cost reduction obtained by serving two customers sequentially in one route instead of in two separate ones, calculated in eq. (2.21)

$$s_{ij} = c_{i0} + c_{0j} - c_{ij} \tag{2.21}$$

In eq. (2.21), i and j are the last and first customers of two different routes respectively, while 0 is the depot. c_{ij} represents the cost of travelling from i to j. Initially, each customer composes a route. The customer pairs of two routes are then examined and sorted by the saving caused. At each iteration, the two routes bringing the largest saving are merged, generating an improved solution. These steps are repeated until no feasible merge can be found. Obviously, Saving Algorithm is a parallel method as more than one route are considered when computing the saving value. However, Laporte and Semet [184] propose a sequential saving method, which is faster but produces poor quality solutions. This method can be easily extended to various VRP variants by taking the specific constraints into account in feasibility checks, e.g. time window and vehicle capacity. Desrochers and Verhoog [92], Wark and Holt [312] merge routes by using matching algorithm and a sophisticatedly defined saving function. The algorithms produce better results than the previous saving algorithms in general, but longer computing time caused by the matching algorithm is required. Details of saving algorithms for VRPTW can be found in [283, 302].

Another classical construction heuristic for VRP is the *Insertion Heuristic*, first proposed by Mole and Jameson [222] and later improved by Christofides et al. [66] for CVRP. In this algorithm, unrouted customers are inserted sequentially or in parallel into the solution until all customers are routed. When the existing routes are too full to accept new customers, a new route would be created. Route optimisation can be done during the construction procedure. Solomon [283] proposes three insertion heuristics (i.e, I1, I2 and I3 heuristics) for VRPTW. Comparing with the saving algorithm, nearest neighbour algorithm and sweep heuristic, the I1 heuristic produces the best results but longer running time is required. The nearest neighbour algorithm is a greedy construction heuristic that always visits the unrouted customer closest to the last routed customer.

Because of its adaptability, insertion heuristic has been widely applied to various VRP variants. Since the routes generated by the insertion heuristic increase gradually, the number of vehicles needed could be impacted by employing specific selection criteria and insertion criteria during the solution construction. In the algorithm for VRPTW proposed by Potvin and Rousseau [248], a regret criterion is used in the insertion criterion. By examining the difference between the best and second best insertion positions, the algorithm shows a good compromise between efficiency and effectiveness. The method of [157] takes the impact on both routed and unrouted customers into account when choosing insertion positions, outperforming the existing algorithms at that time. In OVRP, since the vehicles are not required to return to the depot, visiting the farthest customer away from the depot at the end of a route may reduce the travel back to visited positions. Based on this hypothesis, Fu et al. [117] propose the Farthest First Heuristic (FFH) for constructing the initial solution of OVRP. In FFH, the farthest unrouted customer i is added to the new created route first, and then the unrouted customers are inserted between i and the depot using the shortest route rule until the route is full. FFH performs well not only in node routing problems, but also in arc routing problems. The farthest unvisited arc is treated first when constructing the routes in [197]. More recently, Chen et al. [60] introduce an insertion-based heuristic for short-haul commodity transportation problem, satisfying the limited fleet size and time constraints on commodities.

Two-phase Methods

Two-phase methods decompose the VRP solution process into two subproblems: clustering and routing. Clustering procedure divides customers into subsets, while each one represents a route. Routing determines the sequence of customers in a route. This type of methods can be classified into *cluster-first-route-second* methods and *route-first-cluster-second* methods.

The route-first-cluster-second methods solve the problem as a TSP first, generating a giant tour including all the customers. Then the giant tour is split into multiple feasible routes in the second phase. The look-ahead and demand splitting features can be used to further reduce the vehicles used, but this type of methods perform poorly in general [33].

In cluster-first-route-second methods, after determining the partition of customers, each subset of customers is routed in the second phase. *Sweep* algorithm [317, 132] is one of the earliest and successful applications of this type of algorithms. The algorithm starts from a randomly selected customer as the first customer

(*seed*) to a vehicle, then the unrouted customers are added to the vehicle by means of their polar angle with respect to the depot and the seed. When no customer can be added to the route, e.g. the demand of customers reaches the capacity of a vehicle, a new route is created. The customers to each vehicle are then sorted by solving a TSP. The original sweep algorithm does not control the number of vehicles used.

An extension of sweep algorithm is called *petal* algorithm. A set of feasible routes, called petals, are generated first in this algorithm. Then, by solving a set partitioning problem, the final solution is generated consisting of a number of petals. To speed up the selection of petals, Foster and Ryan [115] and Ryan et al. [266] propose the heuristics for determining the set of candidate routes. An extended algorithm allowing embedded/intersecting routes, called 2-petals, is later proposed [256]. Generally, these methods perform better than sweep algorithms.

Fisher and Jaikumar [111] solve the clustering step as a Generalized Assignment Problem (GAP) for the minimal cost. After determining a number of seed customers, the cost in this GAP is defined as the distances between unrouted customers and the seeds. In the algorithm of [46], m seed customers are selected to initialize the routes while m equals to the number of vehicles available. The remaining unrouted customers are then inserted to the routes considering the insertion cost. This algorithm provides a direct control of the number of routes.

Improvement Heuristics

Improvement Heuristics start with an initial solution generated by other heuristics and apply specific methods to improve the solution quality pursuing better solutions. *Local search* algorithms are often used in improvement heuristics, which modify the current solution by applying neighbourhood operators. Here *neighbourhood* of a solution s is a mapping $N : S$ in solution space S. A neighbourhood structure assigns each solution $s \in S$ a set of solutions (neighbours) $N(s) \subseteq S$, and the number of neighbours $N(s)$ is called the size of the neighbourhood. The choice of a solution $s' \in N(s)$ is called a *move* from s to s' [120]. If the quality of neighbour solutions is better than the current solution, the current solution will be updated to the neighbour solutions and the process iterates; otherwise a local optimum is identified. The modification operation of a neighbourhood operator (neighbourhood structure) defines the local searching area, which greatly determines the success of a local search algorithm. Looking from the perspective of operating object, neighbourhood operations in VRP can be classified into two categories: *edge-exchange* and *node-exchange* (some authors use the term *interchange*

instead of *exchange*). The former is used when the number of direct operating nodes is larger than the number of direct operating edges, and the latter term is used when the opposite is true [120]. Besides, neighbourhood operators can also be divided into two classes, namely *inter-route* and *intra-route*, in terms of operating scale. When the involved objects in a neighbourhood operation are located in more than one route, the neighbourhood is inter-route; otherwise it is intra-route [78].

A large number of neighbourhood operators have been devised for routing problems. Lin [201] proposes λ-*optimality* mechanism, which is an edge-exchange operator widely applied in VRPs while its first application was in TSP. It deletes λ edges from the tour, and then relinks the segments caused in feasible ways. To a tour with n nodes, n^{λ} neighbours should be evaluated. Thus only *2-opt* and *3-opt* are employed in practice for the reason of computing time. λ-opt neighbourhood can bring tour orientation reverse, which opts to cause time constraint violation. *2-opt** is a subset of 2-opt neighbourhood for inter-route operation [249]. It removes two edges from two independent routes and reconnects the routes with no inversion of the orientation of route segments. This characteristic greatly improves the efficiency of local search in the problems with time window constraints. Similar operators 3-opt* and 4-opt* can be found in [150, 256]. Or [231] presents a node-exchange operator called Or-opt, which relocates a sequence of nodes (consecutive and with not more than three nodes) to a new location in the original route. Or-opt does not allow inversion as well, and it is actually a 3-opt exchange without reverse.

Osman [233] introduces $\lambda-$*interchange* mechanism, which is a node-exchange operator. By replacing no more than λ nodes of a route with the nodes from another route (not more than λ as well), a new solution is generated. Here the nodes from a route can be either consecutive or non-consecutive. In the *CROSS-exchange* [286], two strings of nodes from different routes are swapped. Its difference from $\lambda-$interchange is that the nodes swapped must be consecutive, and this preserves the sequence (direction) of customers in their original routes. These two neighbourhood operators have been applied to local search algorithms (e.g. tabu search and simulated annealing), producing promising solutions in VRPTW. 2-opt* and Or-opt are two special cases of CROSS-exchange.

Thompson et al. [297] propose the *c-cyclic k-transfer* method, where c routes (called *cyclics*) are selected first. Iteratively, k nodes are randomly selected from each cyclic and transferred to its next route in sequence until the stop condition is met, producing updated solutions [296]. In *GENI-exchange* [126], one customer node v_i is repositioned to another route (destination). In the destination route, v_i

will be positioned between the two customers which are closest to v_i. This insertion breaks the original links and may bring infeasibility in the destination route, thus rescheduling might be executed on the route.

Glover [134] proposes an *ejection chain* approach which provides a different way to construct compound neighbourhoods generating powerful moves. In an ejection chain for VRP, one node is removed from a route and reinserted into another route (R), which may cause constraint violations in R. To eliminate the violations caused, one or more customer nodes are removed (ejected) from R and inserted into another route. These steps are iterated until a node can be inserted without constraint violation caused on its destination route. Four ejection chain strategies are developed in [135]. With these strategies, different compound neighbourhoods with attractive properties are generated for TSP.

High-performance improvement methods usually adopt more than one operator in the algorithm frameworks. Based on the Cyclic-Transfer method, three perturbing operators are developed in [301]. The first operator is called *String Relocation*, which relocates a customer string from a route to another. The second one called *String Exchange* exchanges customer strings between two routes. Finally, *String Mixed* employs both of the previous operators and accepts the better one from the two solutions produced. Bräysy [49] proposes a two-phase approach, where Or-opt is invoked to construct the initial solution first, then ejection chain is adopted to reduce the number of routes. Similarly, Rousseau et al. [264] reduce the number of routes with ejection chain as well in the first phase, while a Large Neighbourhood Search method is adopted to further improve the solution in the second phase. In the end, an exact post-optimisation procedure is applied to further improve routes.

Compound neighbourhoods have performed well in VRPs. Ergun et al. [105] combine independent moves such as 2-opts, swaps and insertions in a Very Large-scale Neighbourhood Search framework. These independent moves are executed in different positions of a route simultaneously, while their operator settings are the same. This method is applied to TSP and VRP with side constraints. The experiment results show that searching with compounded neighbourhoods is a competitive approach to routing problems. This compounding way with sequential addition and deletion of edges has also been used in the ejection chain approach [255]. An investigation of compound neighbourhood for VRPTW is conducted in [59], which is introduced in Chapter 4 with detailed analysis.

2.2.3 Metaheuristics

Various definitions of *Metaheuristic* have been proposed in the literature, which describe and conclude diverse metaheuristic methods from different aspects. Generally speaking, metaheuristics are strategies for guiding the exploration of solution space (search space) [232, 36]. Of great importance in metaheuristic algorithms is the balance between *diversification* and *intensification* of search. In state space search theory, the term of diversification usually refers to the exploration of solution space i.e. search breadth, while intensification represents the exploitation of solution space i.e. search depth. The metaheuristics for VRP can be categorised into three classes namely: population-based metaheuristics, single solution-based metaheuristics and learning mechanisms.

2.2.3.1 Population-Based Metaheuristics

Population-based metaheuristics (also called population search) iteratively improve a population of solutions. With the help of the high diversification from the diverse solutions, population-based methods show powerful exploration performance and produce high quality solutions in multi-objective and highly constrained problems [287]. Four widely used population-based metaheuristics for VRP are concluded as follows:

Adaptive Memory Procedure (AMP)

The Adaptive Memory Procedure (AMP) for VRP was first proposed by Rochat and Taillard [260]. The adaptive memory is a pool of solutions updated by replacing poor solutions with higher quality solutions during the procedure. To generate a new solution, routes are extracted from the solution pool as the components of the new solution. The routes containing routed customers would be excluded from the extraction candidate list. When there is no available route in the memory any more, a search process (e.g. tabu search) is employed to add the remaining unrouted customers into the current partial solution. This method produced a number of best solutions on the Solomon instances at the time of proposal. In the AMP method of [289], the route segments (called bones) are stored in the adaptive memory instead of complete routes. More recently, an AMP algorithm called solutions' elite parts search is introduced in [292], which produces some new best solutions on benchmark instances.

Genetic Algorithms (GA)

Inspired by the principle of natural evolution, various Evolutionary Algorithms (EAs) are developed, e.g. genetic algorithm, evolutionary strategy, evolutionary programming and genetic programming [57, 287]. Among these algorithms, Genetic Algorithms (GAs) are a popular class of methods widely used to solving VRPs, especially for MOVRPs. It was first introduced by Holland [155], which uses a sequential solution representation called *chromosome*. Traditionally, the elements in a chromosome (called genotype/gene) are binary, but nowadays many different types of solution encodings have been proposed. At each generation of GAs, two individuals called *parents* are selected from the population to generate two *offspring* solutions by executing a *crossover* operation. The offspring then are perturbed with a *mutation* operator. High fitness (solution quality) offspring will replace the individuals with the lowest fitness in the population before the next generation.

GAs show powerful performance in solving MOVRPs. A hybrid GA for VRPTW, which evolves two populations respectively pursuing distinct objectives, is proposed by Berger et al. [30]. Heuristic operators, e.g. large neighbourhood search, are employed at both the reproductive and mutation steps in this algorithm. Combining goal programming and genetic algorithm, Ghoseiri and Ghannadpour [131] produce promising results on the Solomon benchmark. More recently, Vidal et al. [304] propose a hybrid GA with adaptive diversity management which outperforms the state-of-the-art algorithms in a large class of VRPs with time windows.

Prins [251] proposes an approach which also uses crossover and mutation. In the approach, the solution is encoded as a giant tour without route delimiter. The mutation is implemented with a local search executing reinsertion and swap over vertex and arc. This combination of solution recombination and local search is also known as *Memetic Algorithm*. Another memetic algorithm from [215] combines guided local search with evolutionary strategy. In each generation of the algorithm, the evolutionary strategy produces one offspring based on a parent selected following a deterministic rule. The offspring replaces his parent in the population if higher fitness is obtained. In terms of mutation of this algorithm, offspring are improved by means of a search procedure including tabu search, 2-opt*, LNS, vertex relocation and swap. More comprehensive reviews of GAs and EAs for VRP can be found in [122, 300].

There are mainly two deficiencies in GAs for VRP. Firstly, the holistic search in GAs requires high computation cost. To overcome this, search localization

and problem decomposition methods are often employed, which needs dedicated design. On the other hand, the one-dimensional solution representation is hard to be directly applied to high-dimensional and complicated VRPs. These two deficiencies hinder the application of GAs in big size and complex structure VRPs. It should be noted that, there are many extended solution representations and encoding methods that have been developed for high-dimensional VRPs by the evolutionary computation community, e.g. hierarchical chromosome and random key [25]. However, the computation time spent on conversion from chromosome to separated routes is quite long, which is usually solved as a shortest path procedure. For example, in MPVRP, tackling the positioning problems of period delimiters and route delimiters is time-consuming.

Particle Swarm Optimisation (PSO)

Particle Swarm Optimisation (PSO) was first proposed by Eberhart and Kennedy [97], which simulates bird flocking and fish schooling. PSO algorithms maintain a swarm of particles (solution population), while each particle occupies a location in the multi-dimensional search space. The particle fitness at a location is calculated by an objective function. Starting at random locations, the particles move in the search space aiming to find the optimal location with the best fitness. The movement of a particle is determined by its *velocity*, which is calculated based on the current fitness, the best fitness position obtained by the particle so far (local best) and the best fitness position obtained by the swarm so far (global best). Shi and Eberhart [279] conduct an empirical study which shows that PSO converges quickly towards the optimal position but the convergence speed towards local optima can be slow. For a more comprehensive review on the performance and applications of PSO see [69, 170].

Wang et al. [310] propose the first PSO for the TSP, generating the optimal solutions with a high convergence speed on relatively small instances. A basic PSO for the CVRP is introduced in [165], which is later extended and produced satisfying results on instances with up to 200 customers. PSO is also combined with other heuristic methods in the literature. A hybrid PSO method for VRP proposed by Marinakis et al. [210] combining PSO and Greedy Randomized Adaptive Search Procedure. It produces the results superior to other genetic approaches on benchmark instances. More PSO approaches for other VRPs can be found in [166, 211].

Population-Based Incremental Learning (PBIL)

Population-Based Incremental Learning (PBIL) is a probability learning approach based on evolutionary strategies proposed by Baluja [20], which is an evolution of the solution population according to Probability Vectors (PVs). PBIL is an estimation distribution algorithm, where a PV indicates the probabilities of one customer appearing at different gene positions. The probabilities are equally set at the beginning and updated based on the solution evaluation during evolution. This algorithm combines some components of GA with competitive learning. The one-dimensional solution representation in GAs is adopted in the basic version of PBIL [21], called Solution Vector (SV). The mutate operator is also employed but the other essential operator of GA crossover is not. The reason for this is that the author believes that crossover opt to cause an infeasible implementation. PBIL is simpler than GA but can prevent local convergence effectively. A detailed research on the performance of PBIL is conducted by Höhfeld and Rudolph [154]. Lourens [205] proposes the PBIL method for VRP while the algorithm performance is comprehensively investigated. The work of [308] studies a real-life VRPTW, while the proposed PBIL method cooperates with PSO showing good adaptability, validity and stability performance. For more applications of PBIL in VRP, see [309].

2.2.3.2 Single Solution-Based Metaheuristics

Different from population-based metaheuristics, *Single Solution-Based* metaheuristics (or called trajectory metaheuristics/local search) approach the optimal solutions by searching the neighbourhood of the current solution, starting from one initial solution [287]. At each iteration, only one solution is explored, and the search processes are characterised by a search trajectory in the solution space. The search iterates until a solution deemed optimal is found or a stopping condition is reached.

Simulated Annealing (SA)

Simulated Annealing (SA) is inspired by annealing in metallurgy, which systematically adjusts the probability of accepting worse solutions during the search. In SA, a solution better than the current best solution s is accepted to be the new current solution immediately. However, an inferior solution s' may also be accepted with a decreasing probability $p = exp(-\frac{|f(s')-f(s)|}{T})$. Here $f(s)$ is the quality (objective value) of s, and T is a temperature. As the search progresses,

the temperature T cools down (decreases) gradually, and the probability of accepting inferior solution becomes correspondingly smaller. It can be concluded that, SA starts from exploring a wide space (with loose acceptance criterion) to find out the promising solution regions, then intensively exploits around the regions with a stricter acceptance criterion until convergence. The study of [206] shows that SA converges to global optimum asymptotically. More analysis on the convergence speed and performance of SA can be seen in [32].

An early SA algorithm for VRP can be found in [2] which improves solutions with 3-opt. In the SA of [301], Cyclic-Transfer and its extended neighbourhoods are employed. The use of initial solution with higher quality is recommended by this algorithm. One of the most successful SA for VRP is proposed by Osman [233], where a new temperature update scheme different from the conventional non-increasing function is applied. Here, T is decremented when improvement is obtained, otherwise is set to either $T/2$ or the temperature at which the current best solution was found. On the instances with up to 199 customers, 11 new best solutions at that time were found with the algorithm. Czech and Czarnas [85] propose a parallel SA heuristic, which invokes a non-homogeneous Markov Chain to implement the cooperation among different annealing processes at each temperature plateaus. To further improve solutions, the impact of the number of all customers and the number of customers in the shortest route are enhanced in the cost evaluation function. This algorithm produces the best results at the time.

Apart from the probabilistic acceptance scheme, some deterministic annealing methods are also developed based on SA. Dueck [96] proposes Great Deluge (GD) and Record-to-Record Travel (RRT) heuristics which also accept the worse solutions during search, as long as the solution quality deterioration is acceptable. The difference between GD and RRT is that, the acceptance gap of GD is fixed while that of RRT is proportional to the current best objective value. SA, GD and RRT have been widely employed as the acceptance heuristic in various algorithms. Their applications in VRP can be found in [189, 193, 194, 195].

Tabu Search (TS)

Glover [133] proposes Tabu Search (TS), which uses a *tabu list* to avoid search cycling. The tabu list acts as a short-term memory which stores the recent search history. The memory content could be solution features, objective values, neighbourhood moves and so on. At every iteration, search moves to the best non-tabu solution, preventing short-term search cycling. On the other hand, to avoid the tabu list overly restricting the search and skipping some parts of promising solution

space, a scheme called *Aspiration Criterion* is set up to allow the acceptance of some tabu solutions under the specially defined conditions. The stopping criterion can be a limited running time, iterations or the iterations without improvement. More essential aspects of TS and valuable strategies for applying TS are studied by Glover and Taillard [137], Glover and Laguna [136].

The first TS for VRP was proposed by Willard [315], in which the routes are connected as a giant tour sectioned by duplicated depots. 3-opt exchange is employed to improve the solution during the searching procedure. This method is further improved in the studies of [233, 285, 127]. Among them, Osman [233] proposes a TS with λ-opt neighbourhood and a variable size of tabu list storing the moves forbidden. Investigating different search strategies, the proposed TS produces better results than SA. The algorithm of [285] uses 1-interchange neighbourhood and periodic route re-optimisation. The tabu duration in this algorithm is randomly set, while the frequently performed moves would be penalised so as to provide a more comprehensive exploration. Besides, a decomposition scheme is employed to implement parallel computing. This algorithm remains one of the most successful TS to this day. Gendreau et al. [127] further extend the algorithm by forbidding vertex returning to its former route, called Taburoute algorithm. Nontabu moves are executed in a GENI-exchange framework in this algorithm.

Tabu search has been widely employed in various VRP variants. Taillard et al. [286] propose a TS for VRP with soft time windows, in which the decomposition and reconstruction procedure introduced by Taillard [285] works with the adaptive memory search from [260]. By penalising the exchanges frequently employed, Cordeau and Laporte [75] present a new TS for the VRPTW and two entended variants namely: the periodic VRPTW and the multi-depot VRPTW. A unified TS is proposed by Cordeau et al. [80] for solving VRPTW with heterogeneous fleet. A reactive variable neighbourhood TS algorithm is later presented for the same problem [236], which yields high quality solutions on benchmark data sets. A VRP with stochastic demands and its TS solution method are investigated in [129].

Diverse techniques have been combined with TS. Mechti et al. [214] introduce Compound Neighbourhoods into TS for a real-life mail collecting application. Greistorfer [140] proposes an algorithm called Tabu Scatter Search. It combines TS with a population-based solution combination method. The algorithm uses TS to build a candidate population (elite solution pool), then some heuristics are developed to select edges from the candidate population to construct a new solution. The population and Tabu list are updated when a better solution is found. More

analysis and discussion on performance and features of TS algorithms for VRP can be found in [250, 188, 178].

Guided Local Search (GLS)

The basic principle of Guided Local Search (GLS) is that, by adding a penalty term to an objective function, the search gradually moves away from local optima [36]. Starting from an initial solution, GLS applies a local search until a local optimum is obtained. Then the objective function is modified by updating the penalty term involved and the local search restarts. These steps are repeated until reaching the stopping condition. In the GLS for VRPTW proposed by Kilby et al. [172], the initial solution is a virtual vehicle where all the customers are not assigned. The penalty in objective function is defined to punish unassigned customers. To minimise the cost, the customers gradually move to actual vehicles by performing four different neighbourhood moves (i.e. 2-opt, customer switch, customers exchange, route ending swap) in a local search. De Backer et al. [90] later devise a constraint programming framework embedding the GLS and a tabu search. For a more comprehensive review on Guided Local Search, see [306].

Greedy Randomised Adaptive Search Procedure (GRASP)

As one of the most popular multi-start algorithms for COP, Greedy Randomised Adaptive Search Procedure (GRASP) uses multiple initial solutions to diversify the search so as to avoid getting stuck in a local optimum. A GRASP algorithm usually composes of two phases. Firstly, in the construction phase, a number of initial solutions are constructed with a greedy randomised strategy, where the candidate components of solutions are provided by diverse heuristics. Then, in the improvement phase, local search is applied when new components are inserted, optimising the routes generated. The procedure is repeated multiple times and the best solution obtained will be returned. Marinakis [209] proposes a GRASP algorithm for the CVRP, while Kontoravdis and Bard [177] introduce one for the VRPTW. Another multi-start algorithm for VRPTW can be found in [196], which employs a tabu-embedded simulated annealing with k-restart strategy. In the tabu search framework, a simulated annealing acceptance criterion is adopted in this algorithm. Local search is repeated with the initial solution of the current best solution until no improvement is found after k iterations. The algorithm produces several new best solutions of Solomon instances.

Variable Neighbourhood Search (VNS) and Variable-Depth Neighbourhood Search (VDNS)

Variable Neighbourhood Search (VNS) is a framework for building heuristics which shifts the search among different neighbourhood structures. The search regions defined by the neighbourhood structures are different. VNS is proposed based on the observation that, for many problems, the local optima in terms of different neighbourhoods are relatively close to each other, while the local optimum of a neighbourhood structure is not necessarily so for another. In addition, the pursued global optimum is a local optimum in terms of all possible neighbourhood structures. Different variants of VNS have been studied in the literature. In the Basic VNS, a *Local Search* finds local optimal solutions using different neighbourhood structures, and *Shaking* is used to perturb the search to enhance diversification. Variable Neighbourhood Descent (VND) algorithm changes the neighbourhoods in a deterministic way, while Reduced VNS selects neighbourhood moves randomly from a neighbourhood set. General VNS is an extension of Basic VNS, where the local search is a VND [146]. More comprehensive study on VNS variants and performance analysis can be found in [145, 147].

VNS and its extensions have been studied extensively in various VRP problems. Bräysy [49] proposes a four-phase approach based on VND for VRPTW, while Polacek et al. [247] develop a VNS method for Multi-Depot Vehicle Routing Problem with Time Windows where routes start and end at different depots. A VNS algorithm for the Open Vehicle Routing Problem without time constraint is presented in [113]. The VNS in [150] is for the Periodic Vehicle Routing Problem, where the schedule horizon is very large without time constraint.

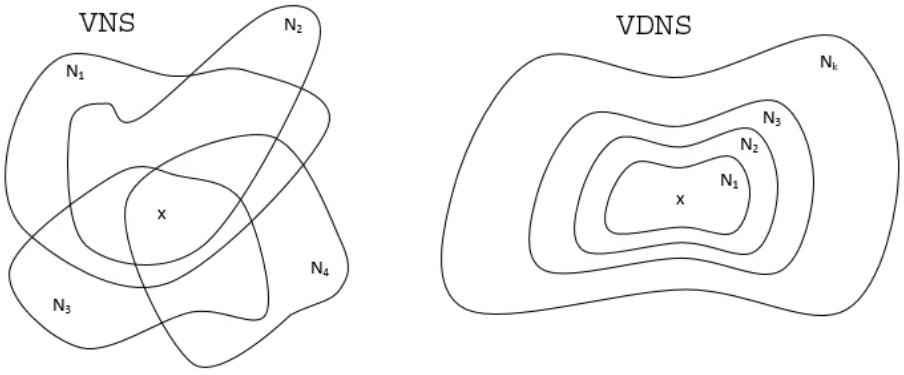


Figure 2.1: Comparison of the neighbourhood structures of VNS and VDNS.

In Variable-Depth Neighbourhood Search (VDNS), the neighbourhood structure is used at variable depths. VDNS is widely applied in Very Large Scale

Neighbourhood search [246]. The difference between VNS and VDNS can be described as in Figure 2.1. Chapter 4 introduces an algorithm combining VNS and VDNS with compounded neighbourhood operators for the VRPTW, which generates a number of new best solutions on Solomon benchmark. In addition to it, a VNS algorithm with reinforcement learning for the PVPRTW with open routes is presented in Chapter 5.

Large Neighbourhood Search (LNS)

Shaw [277, 278] proposes the Large Neighbourhood Search (LNS) heuristic which removes a number of nodes (e.g. demands/customers) from the current solution and then reinserts them with specific heuristics, generating an updated new solution (*Destroy & Repair*). This method brings greater changes (higher perturbation) to escape from local optima and avoids premature convergence. Pisinger and Ropke [245] further propose the Adaptive Large Neighbourhood Search (ALNS), which has been applied to many VRPs [13, 273]. ALNS obtains the best results in several VRP variants, but the larger computation time is required in each iteration. Poorly designed LNS can be beaten by traditional metaheuristics [246]. A similar strategy called *Ruin & Recreate* is proposed in [275].

Nagata and Bräysy [226] propose the Guided Ejection Search (GES) heuristic which combines the ideas of LNS and Ejection Pool methods [199]. In each iteration of GES, one route is removed and then the nodes of the removed route are reinserted into the destroyed solution. The infeasible partial solutions are accepted with penalties during the procedure of algorithm. GES outperforms the existing heuristics on minimising the number of routes, but longer computation time for each iteration is needed. For more details, see [228, 227].

2.2.3.3 Learning Mechanisms

Nowadays, various learning mechanisms have been applied on diverse combinatorial optimisation domains, making great achievements [216]. Generally, the heuristics based on learning mechanisms for VRP is relatively limited, and the proposed methods are not very competitive to the best available approaches. However, there still are some well-performing heuristics that are proposed recently, which embed learning schemes in the algorithm frameworks.

Ant Colony Optimization (ACO)

Ant Colony Optimization (ACO) algorithm is a swarm intelligence method which is inspired by the behaviour of ants seeking food [71]. In the natural world,

a swarm of ants wander randomly to search food and return to their colony upon finding food, and pheromones are laid down on the trails as the ants passed. The path with more pheromone will be used by other ants with higher probability, while the pheromone evaporates over time. Since the succeeding ants also reinforce the pheromone over the paths they passed, after sufficient generations, the shortest path from the colony to the food would have the highest pheromone level. This procedure can be mimicked by searching the optimal solution in optimisation problems. The search is guided by the information (pheromone) learned from historical search experience.

An ACO for VRPTW is proposed in [121] in which two colonies are maintained for the two objectives of minimising the number of vehicles used (NV) and the total travel distance (TD), respectively. Once the NV is updated, both search of colonies would restart with the new NV. This algorithm produces some improved solutions on Solomon instances. Yu and Yang [318] introduce the ACO to the route construction for PVRPTW, which uses Multiple Pheromone Information Matrix to update the pheromone for each day. Two operators are developed to perturb the exploration and increase the diversification. The first one, so-called one-point crossover, splits each operated route into two parts and then swaps the beginning parts between two routes. The second operator, called two-point crossover, reorders and connects four nodes from the two edges of two routes. 2-opt is used in both operations to improve the generated routes. A common deficiency of ACO algorithms is that many parameters are used in pheromone computation, which requires a great deal of parameter tuning work. Some other ACO methods for other VRP variants can be found in [141, 203].

Artificial Neural Networks (ANN)

Artificial Neural Networks (ANNs) simulate the learning mechanism in animal brains. This mechanism works based on the interactions between neurons. In an ANN, the neurons process the data received in specific ways and pass data via weighted connections. An ANN learns from the fed training data, where the weights of the connections are updated as the learning proceeds, obtaining a stronger model of solutions. The earliest investigation on ANN for combinatorial optimisation problems can be tracked to [156], while its applications to VRP can be found in [213, 130, 221]. However, the known neural networks based methods for VRPs are not competitive with the alternative metaheuristics in general [298, 78].

Reinforcement Learning (RL)

Reinforcement Learning (RL) is an adaptive learning and decision-making scheme, based on a probability distribution sampled over the candidate set. The probability distribution is continuously updated according to the reinforcement feedback from the learning environment [294]. RL scheme can be used in *Adaptive Operator Selection* (AOS), guiding the search process [303]. Credit Assignment mechanism and Selection Rule are the two essential components in AOS. According to the historical performance of operators, credit assignment mechanism updates the invoked probabilities (or weights) of candidate operators during the search, e.g. increasing or reducing the probability with elaborately designed reward function [245]. Then, the operators to be executed are chosen with a Selection Rule. Roulette Wheel scheme and Pursuit Algorithm [295] are two commonly used selection rules. The former chooses operators on the basis of their probability distribution and the latter always selects the operator with the largest probability/weight.

AOS has been widely used in metaheuristics. In population-based metaheuristics, credit assignment mechanism calculates the feedback values based on the fitness of solution population [281, 200]. To single solution-based approaches, RL and AOS have also shown outstanding performance in VRPPD and other VRP variants, e.g. in Adaptive Large Neighbourhood Search [151, 273]. Some associated methodologies produce the best results on the benchmarks of classic VRP variants [261, 262]. More applications of adaptive learning in single solution-based heuristics can be found in [55, 303].

2.2.3.4 Discussion

High-performance heuristics often combine the ideas from different methodologies. For example, in terms of accuracy and computation time, the methods of [215, 289, 251] obtain the best performance on benchmarks of [66] and [193]. The three methods all combine local search and population-based search, achieving the broad and deep exploration in search space. Comparing to the traditional construction heuristics and improvement heuristics, the quality of the solutions obtained by different metaheuristic methods usually is significantly better. The multiple solutions in population-based methods diversify the search bringing powerful search ability. But the price to pay is the high computational cost for maintaining the solution population, specially when the structure of individual is complex. By contrast, the exploration diversification in single solution-based metaheuristics is lower, but with a faster iteration speed.

The implementation and calibration of metaheuristic algorithms are often time-consuming in practice. The optimal parameter setting is an essential element to a successful metaheuristic method, and the setting needs sufficient comprehensive parameter tuning experiments to obtain. In addition, the adaptiveness of metaheuristics is another issue. A well-working metaheuristic to a problem may perform poorly in different problems when additional constraints are added. This issue leads to many state-of-the-art metaheuristic algorithms are problem-specific.

2.2.4 Hyper-Heuristics

In the literature, many of metaheuristic algorithms are knowledge-intensive, and these algorithms often are developed for specific problems. In contrast, *Hyper-Heuristics* is a type of high level algorithms which is aimed at developing generic approaches beyond the problem specific metaheuristics research [53, 14]. Hyper-heuristics work at a higher level to generate or select a set of Low-Level Heuristics (LLH) in a common framework, while the LLH execute the operations on problem solutions. Hyper-heuristics focus on designing the high level framework, called High-Level Heuristic (HLH), instead of seeking the specific solutions for the problem confronted. Hyper-heuristics can be described as the heuristics of creating heuristics for given problems. In a well-designed hyper-heuristics algorithm, the HLH would adaptively operate the LLH used, creating proper algorithms for various instance scenarios and problems.

Hyper-heuristics approaches can be categorised into two classes: *Heuristic Selection* and *Heuristic Generation* [56]. *Heuristic Selection* methodologies choose existing heuristics from the LLH pool to tackle the problem given, while the methodologies of *Heuristic Generation* generate new heuristics using existing heuristics as the components. What is more, each of the above classes can be further divided into two subcategories namely: *Construction Heuristic* and *Perturbation Heuristic*, according to the constructive or perturbative low level heuristics used. *Construction Heuristics* construct solutions using the given LLH, while *Perturbation Heuristics* produce new solutions by perturbing existing solutions, see Table 2.2. More comprehensive surveys of hyper-heuristics can be found in [252, 54].

As a classic combinatorial optimisation problem, VRP is an essential application of hyper-heuristics. Garrido and Riff [123] propose an evolutionary hyper-heuristic for DVRP. Each genotype in this evolutionary algorithm consists of a constructive heuristic, an improvement heuristic and an ordering heuristic. This

Table 2.2: A classification of Hyper-Heuristics

Heuristic Selection (Methodologies to select)	Construction Heuristics
	Perturbation Heuristics
Heuristic Generation (Methodologies to generate)	Construction Heuristics
	Perturbation Heuristics

generative construction hyper-heuristic adapts well to the dynamic scenario in D-VRP. Both hyper-heuristics of [267] and [307] obtain competitive results in CVRP. The former generates LLH by searching the space of heuristic component (i.e. neighbourhood structure, neighbourhood combination, local search configuration and acceptance criterion), while the latter adjusts the order of LLH to perturb the current solution, incorporating an adaptive ordering scheme in an iterated local search framework. In [268], besides the selection of LLH, a Gene Expression Programming framework is also proposed to automatically generate the acceptance criterion for different problem instances. The proposed method produces promising results in DVRP and CVRP.

Vidal et al. [305] propose a unified hybrid genetic search framework (UHGS). They replace the mutation with a unified local search (ULS). In ULS, the route-evaluation operators vary according to the change of problem *attributes*, aiming to provide a general-purpose solver for diverse VRP variants. UHGS produces results better than or close to the best-known results on benchmarks. However, the experiment results show that its computation time increases significantly in MPVRPs due to the period and route partition problem in genetic algorithms. This issue has been discussed in subsection 2.2.3.1. The long computation time impedes its application to large scale MPVRP. Many of existing hyper-heuristics are based on evolutionary computation, where the computational intensity is an issue being concerned.

2.3 The Full-truckload Transportation Problem

Neither consolidation nor intermediate transfer is allowed in full-truckload transportation problems (FTL). The Ningbo Port problem is a FTL pickup and delivery problem, where every vehicle is with a capacity which equals to one customer demand unit. In the FTL problems, each customer may have multiple pickup/delivery requests for different commodities which cannot be consolidated to one vehicle. This type of problems are common in the container transportation industry, since every terminal may have multiple containers to be transshipped.

Wang and Regan [311] study a multi-shift FTL problem in which the trucks scatter over the "service region" instead of a depot. Correspondingly, the routes start from the locations where the trucks completed the tasks of the last day, which may be impractical in real-life applications. In addition, in this problem, time windows are only applied to pickup customers but not to delivery customers, and this setting does not reflect the task urgency level in practice. The authors formulate this problem as an asymmetric m-TSPTW. The proposed solution method searches the optimal solution by iteratively narrowing the gap between the upper and lower bounds, which are obtained by solving the under-constrained and over-constrained problems, respectively. Mitrović-Minić and Krishnamurti [219] study the upper and lower bounds of the number of vehicles used in symmetric m-TSPTW, while only one shift is considered in the case.

Zhang et al. [321] introduce a container transportation problem which is formulated as an asymmetric m-TSPTW with multi-depot. In the proposed graph model, different types of vertices are defined involving specific activities. A cluster method and a reactive tabu search approach are introduced which find some optimal solutions on the instances with no more than 50 containers. A similar model is used by the study of another container transportation problem in [322]. With the time window partitioning scheme from [311], further improved results are achieved. A mixed integer programming model for the FTL container transportation problem with many-to-many dependency is studied in [229]. The authors implement a two-stage heuristic, which produces similar quality solutions but with only 16% of computation time comparing to the method of [322].

Based on Zhang et al.'s research, Braekers et al. [42] develop a deterministic annealing algorithm for the bi-objective version FTL transportation problem, in which the number of vehicles used is minimised first and the total travel distance is considered later. Only a single day planning is considered in this problem. The study of [91] introduces a single-shift FTL problem with heterogeneous fleet in timber transportation. After comparing various metaheuristics, the authors find that the LNS with improving random moves achieves the best results.

Braekers et al. [44] investigate a FTL problem with simultaneous pickup and delivery nodes. In a single shift, the locations of empty containers are either fixed or determined before scheduling. A two-phase deterministic annealing algorithm is proposed and tested on the instances with time windows from 60 to 240 minutes. The two objectives in this problem are considered in a priority order. Braekers et al. [43] further discuss the impact of the time-dependent traffic conditions to the routing and scheduling. Following that, the equal priority version bi-objective

problem is studied in [45]. In this study, three methods of an iterative method, a hybrid deterministic SA with tabu search (DA-TS) and the deterministic SA from [44] are tested on the instances with 100 to 200 customer requests. The research shows that DA-TS can generate better solutions than the other two methods within reasonable time of around 28.43 seconds in average.

Overall, the FTL problem shares some common features with VRPs, while it has been modelled as specific routing problems in many previous research. In the problems of big size, e.g. with a large number of containers in the container transportation problem, metaheuristic methods have shown their high performance.

2.4 Service Network Design Problem for Freight Transportation

The Service Network Design problem (SNDP) is often used in strategic (long-term) and tactical (medium-term) levels transportation planning, whose major issues mainly focus on the selection and scheduling of service, the specification of terminal operations and the routing of service flow. It makes use of network design formulations, such as the time and/or space mapping, to generate an effective and balance routing plan improving the logistics management level. The main difference between the SNDP and the fixed-charge multi-commodity routing problem is that, in the SNDP, the service assets are expected to be balanced in every period of the planning horizon. Therefore, consolidation is usually applied in the routes of SNDP for the balanced scheduling, which is not often considered in the VRP research.

Crainic [83] presents a comprehensive review which surveys the efforts of modelling and mathematical programming in SNDP. Andersen et al. [6] investigate the management and coordination of multiple fleets in the SNDP, which includes the border crossing necessitating changes of vehicles and intermodal operations. Different modelling approaches are compared in this study. A single solution-based metaheuristic for the SNDP with time-dependent stochastic demands is studied in [153], where both the balancing of empty vehicles and the cost of handling freight are considered in the model. Other recent research on stochastic SNDP can be found in [204, 15]. The service network design for freight transportation mainly focus on consolidation-based transportation problems, in which the goods are usually consolidated in the network of hubs and terminals. However, SNDP is also able to model the multi-shift problems neatly and provides a good tool to find

the bounds for FTL problems. For a more detailed review of SNDP for freight transportation, see [314].

2.5 Summary

This chapter reviews the literature related to the study of this thesis. The basic definitions and models of the vehicle routing problem and variants are introduced first. The VRPTW covers the basic aspects and some common features of many VRP variants, while its model and solutions can be easily extended for other routing problems. Based on the discussion on VRPTW, more complex variants including multi-objective VRPs are introduced as well. Secondly, the solution methodologies of VRPs are also reviewed including exact methods and approximate approaches. In addition, two related domains of FTL problem and SNDP for freight transportation are briefly reviewed in this chapter as well.

The time windows associated with the container transshipment requests, which span from several hours to a few days, is one of the most important features of the Ningbo Port problem. Since the fleet size is limited, it is critical to plan high quality routes over the long planning horizon of multiple shifts, instead of a single shift. In real-world scenarios, if the planning focuses on improving the efficiency of a specific shift, a large number of urgent tasks might be left to the following shifts, which often causes failed assignments. The dimension of shift in solution representation increases the complexity in routing and scheduling. In addition, the different shift lengths in the Ningbo Port problem also cause the growth of the scheduling difficulty, which should be delicately modelled. In spite of the rich literature in VRP and the extended fields, none of existing models and algorithms can be directly applied to the problems studied in this research. Two mathematical models formulating the container transportation problems faced by the Ningbo Port company are established and presented in Chapter 3, where one is for the short-haul container transshipment problem in the port districts and the other considers the routing and scheduling related to both port districts and dry ports.

In terms of the solution methodologies for VRPs, exact methods can find the optimal or near optimal solution, but the computation time required will drastically grow along as the problem size increases. Take the VRPTW as an example, exact algorithms can optimally solve a 50-node instance within a reasonable time, while more than one hour computation time would be needed for the instances with more than 100 nodes, which are still small-size in real-life scenarios. In the

Ningbo Port problem, there can be hundreds of containers transshipment requests placed in each shift of busy seasons. When taking the availability of requests in multiple shifts into consideration, the corresponding growth of solution diversity will significantly increase the search space and further increase the computation resource requirements from exact methods.

Approximate methods perform powerfully in addressing large scale optimisation problems with complex structure, therefore the methods might tackle the Ningbo Port problem efficiently as well. Considering the high computational cost of population-based heuristics in large scale and high-dimensional problems as summarised in Section 2.2.3, single solution-based methods would be more adaptive to the studied problem of this thesis. Thus, this thesis mainly focuses on meta-heuristics, hyper-heuristics and their applications on classic and real-life problems. The implementation strategies of the proposed algorithms and the computational comparison with state-of-the-art heuristic approaches are introduced in detail.

Chapter 3

Models of Real-life Transportation Scheduling and Routing Problems and Benchmarks

Two new mathematical problem models, which formulate the short-haul and long-haul container transshipments in the Ningbo Port problem, are established in this Chapter. The first problem model focuses on short-haul container transportation among docks, while the second one combines the routing and scheduling of both types of transportation tasks into one framework. After introducing the background to these problems, the mathematical models of both problems are presented. Both models formulate the problem of large scale with nonlinear constraints. The task node-based route representation is employed by both models. The benchmark instances for these two problems are generated and introduced in this chapter. The two benchmark datasets include real-life historical instances obtained from the company and the randomly generated instances. The historical instances can be used to estimate the task completion ability of the proposed solution methods, while the artificial instances provide more testing scenarios with various features of transshipment requests as benchmarks.

According to the information provided by the Ningbo Port company, it can be found that the problems studied have the following characteristics.

- The travelling distance and time between any two container terminals are symmetrical. However, because of traffic conditions in real life, the travelling time is not proportional to the travelling distance.
- The service time at each terminal is determined by the time of loading and unloading containers.

- Both long-haul and short-haul container transshipments are encountered in scheduling. The transshipment tasks associated with the dry ports are *long-distance* task (LDT), which may take more than 12 hours to complete. Differently, those transshipments between docks are *short-distance* tasks (SDT), where the service time of each task is up to seven hours.

- The lengths of shifts for LDT drivers and SDT drivers are different. The company sets up two types of working shifts: *short shift* (12 hours) and *long shift* (24 hours). An SDT can be completed in a *short shift*, while LDT must be completed in *long shifts* for the long service time. The unit payments for the two types of drivers are different for their different length of shift, correspondingly.

- Every single working day, the trucks have to return the depot for maintenance and preparation for the next day. In other words, the truck routes are closed each day. Working shifts are consecutive, which means the end time of one shift is the start time of the next shift.

- Commodities are described by declaration forms (requests), which specify the number of containers to be transported, the time window, origin and destination terminals and so on. The time windows for commodities may span from several hours to a number of shifts, and the fleets have to accommodate the requests during time windows with the limited number of trucks.

- As one truck can carry only one container at a time, one container represents one transportation task. One commodity transshipment request may consist of multiple transportation tasks.

3.1 Periodic Vehicle Routing Problem with Time Windows and Open Routes

This section investigates the short-haul container transportation within the port district of Ningbo Port. Everyday a fleet of 100 trucks transport commodities (containers) between nine dock container terminals (see Figure 1.2). Bai et al. [16] have proposed a scheduling scheme based on PVRPTW for this problem. Since the size and operational cost of the fleet are fixed at the Ningbo Port, the port manager expects to further decrease the empty-load travel distance of the fleet, optimising truck usage. To this end, an updated scheduling scheme is proposed

in this thesis, which reduces the empty-load travel from and to the depot in truck routes.

3.1.1 Description of Problem

Scheduling Scheme

In the proposed new scheduling scheme, the locations of starting and finishing work are different for drivers. Drivers take shift change in the middle of a working day at different terminals, instead of one specific depot. Each working day in the scheme is divided into two 12-hour shifts, to satisfy the related regulations on drivers' working hours specified in Labour Law. Every day, at the beginning of the first shift (day shift), trucks depart the depot with a list of tasks to complete. When all tasks assigned are completed, the trucks park at the last task destinations, or at the first task source terminals of the second shift (night shift) as long as the trucks can arrive there before the end of the day shift. Then, at the beginning of the night shift, a new group of drivers are assigned their trucks at specified terminals (shift change). In the night shift, after completing all assigned tasks trucks must go back to the depot for maintenance. The travel of trucks head to the next task source is empty-load, with no container transported.

Truck Route Characteristics

It can be found that, in this problem, the routes in each shift are open as that in the OVRP. More precisely, the scheduled routes of day shifts are open as they do not finish at the depot, while the routes of night shifts are reverse open routes. The difference between this problem and the School Bus Routing Problem is that, in our problem, the routes in night shifts are not the same as the routes in day shifts in a reversed order.

Driver Shift Characteristics

As mentioned before, the vast majority of transshipment requests are received 24 hours before their available time with different time windows up to several days, which brings a long feasible scheduling horizon. In practice, there might be a few urgent commodities that are submitted when their deadlines are close, and they are either manually inserted to the existing schedule or rejected, which is not discussed in this study. The typical scheduling horizon in Ningbo Port spans from 2 to 4 days (4-8 shifts). The first shift of every working day is denoted

as an *odd-indexed* shift, while the second shift is *even-indexed*. In such a multi-shift problem, the number of containers in each commodity can be seen as the visiting frequency of customers in PVRP. However, the PVRP model cannot be directly applied because of the distinct practical constraints in this problem. This short-haul container transportation problem is modelled as a PVRPTW with Open Routes (PVRPTW-O) in this study.

Problem Descriptive Technique

In terms of the problem descriptive approach, the well-known Planning Domain Description Language (PDDL) is a complex descriptive system providing a standard and flexible formalism for various AI planning domains including the VRPs [67]. It is supported by state-of-the-art planning methodologies, producing high quality solutions in various planning problems. However, those methods have not shown to perform effectively or efficiently in solving large size real-life problems [3]. To simplify the problem model and make the prevailing neighbourhood search heuristics applicable, the node-based method of [311], instead of the PDDL, is employed for formulating the Ningbo Port problem.

In previous VRPPD research, the shipment loading and unloading time is usually ignored or simplified as they are small and/or identical. However, because of the limitation of cranes at terminals, the time of loading and unloading accounts for a considerable proportion of the total service time in the Ningbo Port problem. Besides, the service time are different at different terminals, thus cannot be ignored or simplified in this problem. Therefore, in the proposed PVRPTW-O model, the time of loading and unloading activities are combined with the travelling time, defining the *task nodes* with different service time. In the proposed model, a *task node* is composed of the loading at the source terminal, unloading at the destination terminal, loaded travel from the source to the destination, and the associated time window of the task.

Artificial depots (W) are introduced to connect a odd-indexed shift (day shift, S_{odd}) to the following even-indexed shift (night shift, S_{even}). In S_{odd}, artificial nodes are termination depots, while they become the starting depots in the following S_{even}. When a truck travels from S_{odd} to S_{even}, the first source terminal of a route in S_{even} serves as an artificial depot, if it can be reached before the end of S_{odd}. Otherwise, the last destination terminal of the route in S_{odd} will be the artificial depot. These artificial depots hold the time constraints on driver working time.

A Descriptive Example

An example of one day schedule (two consecutive shifts of an odd-indexed and an even-indexed) is presented in Figure 3.1. The fleet size is five (i.e. $K = 5$), corresponding to five routes. Take the first route as an example, two and three tasks are completed in the odd-indexed and the even-indexed shifts, respectively. The lines directly connecting starting depots and termination depots are empty routes, which means no task is completed on them. In this example, each shift has two empty routes. The artificial depots in W are either the last destinations on Shift 1 routes or the first sources on Shift 2 routes. The number of artificial depots is decided by the number of terminals at where shift-changes happen. In this example, two trucks change shift at the fourth artificial depot.

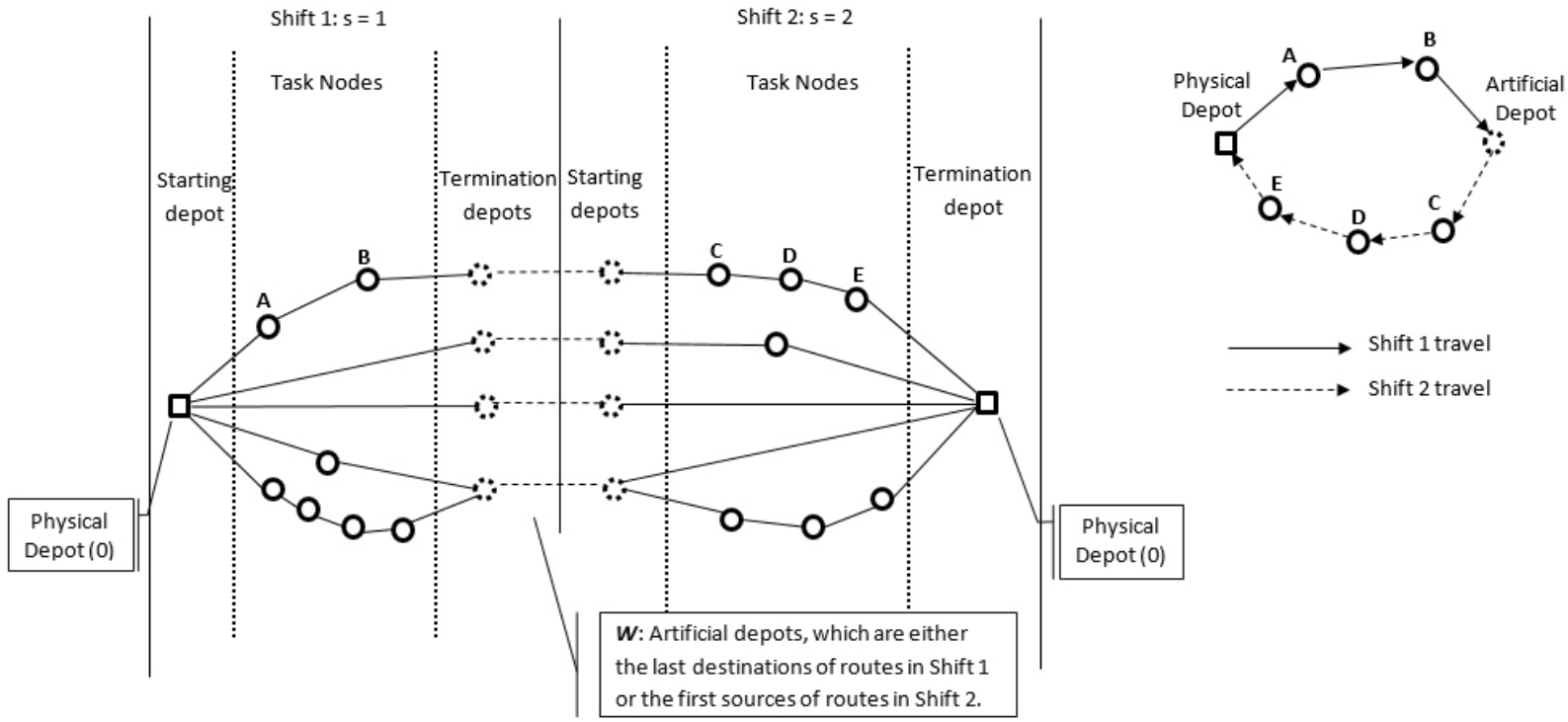


Figure 3.1: A scheduling example of two consequent shifts with five trucks. The solid line circles represent the transportation tasks to be completed. The right sub-figure indicates the first route in the schedule.

3.1.2 Mathematical Model

The notation used is introduced in Table 3.1. Every scheduled route starts at a *starting depot*, connects a number of task nodes (tasks to be completed) and ends at a *termination depot*. It is notable that, some of these routes are empty, which means no task is completed on them.

Table 3.1: The list of notations

Input Parameters:	
K	Fleet size, i.e. number of trucks.
S	The set of time-continuous working shifts, which can be divided into odd-indexed shifts (S_{odd}) and even-indexed shifts (S_{even}).
$[Y_s,Z_s]$	Time window of shift s
$N = \{0, 1, 2, ..., n\}$	Set of $n+1$ nodes. Each node represents a task, except node 0 is the physical depot.
$[a_i,b_i]$	Time window for node i. The time window for node 0 is zero at the boundary of a shift. If a truck arrives at the source of i early, it has to wait until a_i.
A	Set of arcs. Each arc(i,j) represents that node j is immediately serviced/visited after servicing/visiting node i.
c_{ij}	The cost of empty-load travel from node i to node j. When both i and j are task nodes, c_{ij} is the distance from the destination of i to the source of j. Otherwise, it is the distance between a terminal and the depot.
t_{ij}	The travel time from node i to node j. When both i and j are task nodes, t_{ij} is the travel time from the destination of i to the source of j. Otherwise, it is the travel time between a terminal and the depot.
l_i	The time for servicing node i, which includes the loading time, transportation time (from pick-up source to delivery destination) and unloading time. The service time of a depot is zero.
W	Set of *Artificial Depots*. This set of nodes are introduced to represent the termination depots in S_{odd} or starting depots in S_{even}. Which physical terminal would be included as an artificial depot is decided by if the associated truck in S_{odd} can arrive at that terminal before the end of shift. W varies with solution, i.e. a physical terminal may not be included or may appear more than once in W.
Variables:	
T_i	The arrival time at node i.
B_i	The time to begin the service of node i.
x_{ij}^s	A binary decision variable for nodes $i, j \in N \cup W$. Its value is 1 if arc(i,j) is included in the solution in shift s, otherwise is 0. $i, j \in W$ is not allowed

The problem can be formally defined as follows:

$$Minimise \qquad TD = \sum_{s\in S} \sum_{i\in N\cup W} \sum_{j\in N\cup W} c_{ij} \cdot x_{ij}^s \tag{3.1}$$

Subject to:

$$\sum_{s\in S} \sum_{i\in W\cup N\setminus\{j\}} x_{ij}^s = 1, \qquad \forall j \in N\setminus\{0\} \tag{3.2}$$

$$\sum_{s\in S} \sum_{j\in W\cup N\setminus\{i\}} x_{ij}^s = 1, \qquad \forall i \in N\setminus\{0\} \tag{3.3}$$

$$\sum_{i\in W\cup N\setminus\{j\}} x_{ij}^s = \sum_{f\in W\cup N\setminus\{j\}} x_{jf}^s, \qquad \forall j \in N\setminus\{0\}, s \in S \tag{3.4}$$

$$T_j = \sum_{i\in N\setminus\{0\}} (B_i + l_i + t_{ij}) \cdot x_{ij}^s + \sum_{i=\{0\}\cup W} (Y_s + t_{ij}) \cdot x_{ij}^s, \forall j \in N\setminus\{0\}, s \in S \tag{3.5}$$

$$B_j = T_j + max\{a_j - T_j, 0\}, \qquad \forall j \in N\setminus\{0\} \tag{3.6}$$

$$x_{ij}^s \cdot Y_s \quad \leq \quad x_{ij}^s \cdot T_j, \qquad \forall i \in \{0\} \cup W, j \in N \cup W, s \in S \tag{3.7}$$

$$x_{ij}^s \cdot (B_i + l_i) \quad \leq \quad x_{ij}^s \cdot Z_s, \quad \forall i \in N \cup W, j \in \{0\} \cup W, s \in S \tag{3.8}$$

$$a_i \leq B_i \leq b_i - l_i, \qquad \forall i \in N \backslash \{0\} \tag{3.9}$$

$$x_{ij}^s \in \{0, 1\}, \qquad \forall i, j \in N \cup W, s \in S \tag{3.10}$$

$$x_{vw}^s = 0, \qquad \forall v \in W, w \in W, s \in S \tag{3.11}$$

In odd-indexed shifts ($\forall s \in S_{odd}$):

$$\sum_{j \in N \backslash \{0\} \cup W} x_{0j}^s = K, \qquad \forall s \in S_{odd} \tag{3.12}$$

$$x_{i0}^s = 0, \qquad \forall i \in N \backslash \{0\} \cup W, s \in S_{odd} \tag{3.13}$$

$$\sum_{i \in N} \sum_{w \in W} x_{iw}^s = K, \qquad \forall s \in S_{odd} \tag{3.14}$$

In even-indexed shifts ($\forall s \in S_{even}$):

$$\sum_{j \in N} x_{jw}^{s-1} = \sum_{e \in N} x_{we}^s \qquad \forall w \in W, s \in S_{even} \tag{3.15}$$

$$x_{0j}^s = 0, \qquad \forall j \in N \backslash \{0\} \cup W, s \in S_{even} \tag{3.16}$$

$$\sum_{w \in W} \sum_{j \in N} x_{wj}^s = K, \qquad \forall s \in S_{even} \tag{3.17}$$

$$\sum_{i \in N \backslash \{0\} \cup W} x_{i0}^s = K, \qquad \forall s \in S_{even} \tag{3.18}$$

The objective of this problem is to minimize the total travel distance (TD), eq. (3.1). In classic VRPs, this objective is the secondary objective. However, the operational cost of the fleet in this real-life problem is fixed, thus, minimizing the travel distance become the only one objective. Since the loaded travel distance is fixed in each instance, the objective actually is equivalent to minimizing the empty-load travel distance.

Constraints (3.2) and (3.3) denote that every task node can be visited exactly once and all the tasks must be visited. Constraint (3.4) means the in-degree of a node equals its out-degree, specifying that a task may only be serviced after the previous task is completed. Constraints (3.2)-(3.4) together make sure arcs of over more than one shift are unacceptable. Constraint (3.5) is the arrival time at a task node. Constraint (3.6) defines the beginning time of servicing a task node. This time is calculated by the arrival time plus the waiting time at the source of a task. Constraints (3.5) and (3.6) enforce the correct temporal successive relationship between consecutive nodes.

Constraints (3.7) and (3.8) are the time window constraints of each shift, while constraint (3.9) represents the time constraint on each task. The domain of the respective decision variable is defined by constraints (3.10) and (3.11). Especially, constraint (3.11) prohibits the travel between two artificial depots.

The starting and termination depots in odd-indexed shifts and even-indexed shifts are different. Constraints (3.12) and (3.14) represent that K trucks leave the physical depot (0) at the beginning of an odd-indexed shift, and they would stop at artificial depots at the end of the shift. Constraint (3.13) represents that no truck returns to the physical depot in odd-indexed shifts. Constraints (3.16) - (3.18) place the reverse restraints in even-indexed shifts. Constraint (3.15) defines the shift change from an odd-indexed shift to the following even-indexed shift on artificial depots, where the in-degree of each artificial terminal in S_{odd} equals its out-degree in the following S_{even}.

From this integer programming model, we can find that this problem is nonlinearly constrained and with a huge solution space. It is no doubt that the nonlinear constraints can be converted to linear ones, but leads to a more complex model, which is discussed in *Section 5.1.* The size of the solution space of PVRPTW-O is decided by the length of the scheduling horizon ($|S|$), the fleet size (K) and the number of tasks (n). Since the number of total routes in a solution could be up to $|S| \cdot K$, while the number of permutations of tasks is $n!$, the size of the search space is $|S| \cdot K \cdot n!$. In real-life scenarios, n in the whole planning horizon could be larger than 1,000. This model is a basic framework for PVRPTW-O, in future extended research more specific constraints (e.g. subtour elimination constraints) can be added to further improve the problem formulation or expressing it in different forms. It is worth noting that, empty routes in S_{even} do not mean their travel distances are zero. The cost of empty route can be zero, only when the connected artificial node represents the physical depot.

3.1.3 Benchmark Instances

Bai et al. [16] extract 15 instances from the real-life Ningbo Port dataset. In different instances, the planning horizons are 4, 6 or 8 shifts, respectively, of 384 up to 1073 tasks. An artificial instance set including 17 instances is created as well, with different feature combinations on time window tightness (Tight/Loose), workload balance at terminals (Balanced/Unbalanced) and planning horizons of 4 and 8 shifts. Especially, a very large instance with more than 2000 tasks is created as well. The detailed features of instances are presented in Table 3.2 and Table 3.3. The time window of a task, defined by the time it becomes available and the

deadline it must be delivered to its destination terminal, varies from 1-2 hours, up to 6 shifts.

Table 3.2: The list of real-life instances of the Ningbo Port.

Instance	No. of shifts	No. of tasks
NP4-1	4	465
NP4-2	4	405
NP4-3	4	526
NP4-4	4	565
NP4-5	4	765
NP6-1	6	1073
NP6-2	6	920
NP6-3	6	384
NP6-4	6	746
NP6-5	6	557
NP8-1	8	913
NP8-2	8	827
NP8-3	8	786
NP8-4	8	1008
NP8-5	8	798

Table 3.3: The list of artificial instances.

Instance	Configuration	No. of Shifts	No. of tasks
LB4-1	Loose, Balanced	4	484
LB4-2	Loose, Balanced	4	396
TB4-3	Tight, Balanced	4	282
TB4-4	Tight, Balanced	4	368
LU4-5	Tight, Unbalanced	4	448
LU4-6	Tight, Unbalanced	4	479
TU4-7	Loose, Unbalanced	4	217
TU4-8	Loose, Unbalanced	4	354
LB8-1	Loose, Balanced	8	592
LB8-2	Loose, Balanced	8	657
TB8-3	Tight, Balanced	8	497
TB8-4	Tight, Balanced	8	621
LU8-5	Tight, Unbalanced	8	551
LU8-6	Tight, Unbalanced	8	559
TU8-7	Loose, Unbalanced	8	607
TU8-8	Loose, Unbalanced	8	525
Large	Mixed, Unbalanced	8	2614

Comparing to classic VRP benchmarks [282, 124], it is easy to notice that the number of tasks in the Ningbo Port benchmark is quite big. To systematically test the performance of VNS-RLS on diverse instances, we have generated two more datasets scaled down to 25% and 50% respectively by extracting tasks from the complete dataset, keeping the original features (available at http://www.cs.nott.ac.uk/˜pszrq/ benchmarks.htm).

3.2 Mixed-Shift Vehicle Routing Problem with Time Windows

This section introduces a problem considering both short-haul and long-haul transportation. In the problem, the routing and scheduling of the long-distance dry port-related tasks and the short-distance dock-to-dock tasks are simultaneously performed.

3.2.1 Description of Problem

Problem Background

In the Ningbo Port company, trucks carrying out short-haul and long-haul tasks are identical. Thus, integrating the routing and scheduling of both types of tasks into one system may easily increase the utilisation of vehicles and reduce operational costs, while efficient automatic routing and scheduling algorithms are required. In this case, commodities are transited between 19 container terminals including port district docks and inland dry ports of Ningbo Port (see Figure 1.1). In total, 250 trucks compose the servicing fleet locating at the Ningbo coast. The task node-based modelling method [311] introduced in the last section can also be used in this problem.

The distances between container terminals are greatly different. The service time of a short-distance task (SDT), which is dock-to-dock, is less than seven hours and all the harbour docks can be reached in less than 2 hours from the depot. On the other hand, because the long service time of dry port-related long-distance tasks (LDT) and the long travelling time between dry ports and the depot, the average time of completing a LDT is more than 13 hours. The objective is to determine the optimal scheduling of the fleet and truck routes, completing both types of tasks with a limited number of trucks and minimal cost.

Driver Shift Characteristics

The Ningbo Port company sets up two types of working shifts: *short shift* and *long shift*. A *short shift* is 12 hours, meaning one day is divided into two short shifts (day shift and night shift). As introduced in Section 3.1, this shift setting is designed for SDT. During the day shift, drivers drive the trucks away from the depot, and drivers of the night shift return the trucks to the depot after completing their tasks. The two drivers using the same truck (called *one-driver truck*) have a shift-change in the middle of a day at a terminal. Shift-change cannot happen within a task node, so the shift-change terminal is either the last destination terminal of the day shift or the first source terminal of the night shift. Differently, a *long shift* is 24 hours. In this case, two drivers will be assigned to a single truck (*double-driver truck*) at the same time so they can drive the truck by turns, satisfying the associated regulations on continuous working hours in Labour Law.

The two types of shifts are associated with two different driver salary schemes, which lead to different overall operational cost to the company. In a working day, two drivers are required for one truck of either type. The difference between the two types of trucks is that the two drivers of a *one-driver truck* route work separately within their own *short shifts*, while both drivers of a *double-driver truck* route have to stay in the truck during the whole *long shift*. Correspondingly, the unit payment to the drivers of *double-driver trucks* is higher for their longer shift length. SDT can be completed in a *short shift* using *one-driver trucks*, while LDT must be completed with *double-driver trucks* in *long shifts* due to the long service time. When optimizing the assignment of LDT and SDT, considering both types of trucks simultaneously can reduce the overall number of trucks used, consequently minimizing the overall total operational cost of driver payment. Previous research has noticed that, in the transportation industry, the labour cost (driver salary) usually is the dominated component in the overall cost [27].

Mixed-Shift Model

The truck scheduling for both types of shifts are combined in this study. Currently, the company handles LDT and SDT with two separate scheduling systems, resulting to inefficient use of trucks and lots of task lateness in busy seasons. This low efficiency of schedule is mainly caused by the two separate scheduling systems which do not share the limited truck resource. In my study, the two scheduling systems are integrated to increase the efficiency of the scheduling and the utility of trucks. *Artificial task* which represents the driver shift-change between two *short shifts* is thus proposed. The routes of a truck in two consecutive *short shifts* are thus converted to one route in a *long shift*. To the best of my knowledge, this is the first time the Mixed-Shift Vehicle Routing Problem with Time Windows (MS-VRPTW) is proposed in the literature. In the Ningbo Port, the trucks in the fleet are identical and can be appointed to be either *one-driver* or *double-driver* according to the commodity situation.

A Descriptive Example

An example schedule of a working day (with one *long shift* or two *short shifts*) is presented in Figure. 3.2 to illustrate the proposed model. There are in total eight routes, three for *one-driver* trucks and five for *double-driver* trucks. We can see that, LDT (represented by rectangles) only appear in *double-driver* truck routes, while SDT (solid circles) exist in both *one-driver* truck routes and *double-driver* truck routes. The hollow circles in the top three routes are *artificial tasks*.

The fourth route in Figure. 3.2 explains why LDT require *double-driver* trucks. Considering the travel time leaving and returning to the depot, completing a LDT takes more than 12 hours (maximum length of a *short shift*). In addition, if the distance between two LDT is small, more than one LDT might be serviced in one *double-driver* route. For instance, in the last route, as the destination of the first LDT is the source of the second LDT, the travel distance and time between the two tasks is zero. In this case, the two LDTs can be completed by one *double-driver* truck, leading to a more efficient use of vehicles.

Another special case of LDT is the rectangle in the seventh route. It represents a type of tasks which have short service time but can only be finished in *double-driver* routes. Because their time windows are narrow (i.e. 3 hours in this example) and across the middle of a working day, the shift-change between *short shifts* cannot be done when completing this type of tasks. Therefore, this type of tasks can only be assigned to *double-driver* trucks.

In different real-life scenarios, the shift lengths, the number of task types and the number of shift types might be different from that of the Ningbo Port problem. However, the method of using artificial task is still applicable, which integrates the scheduling and routing with different shift settings into one model. Therefore, the model of MS-VRPTW can cover various practical cases from real scenarios.

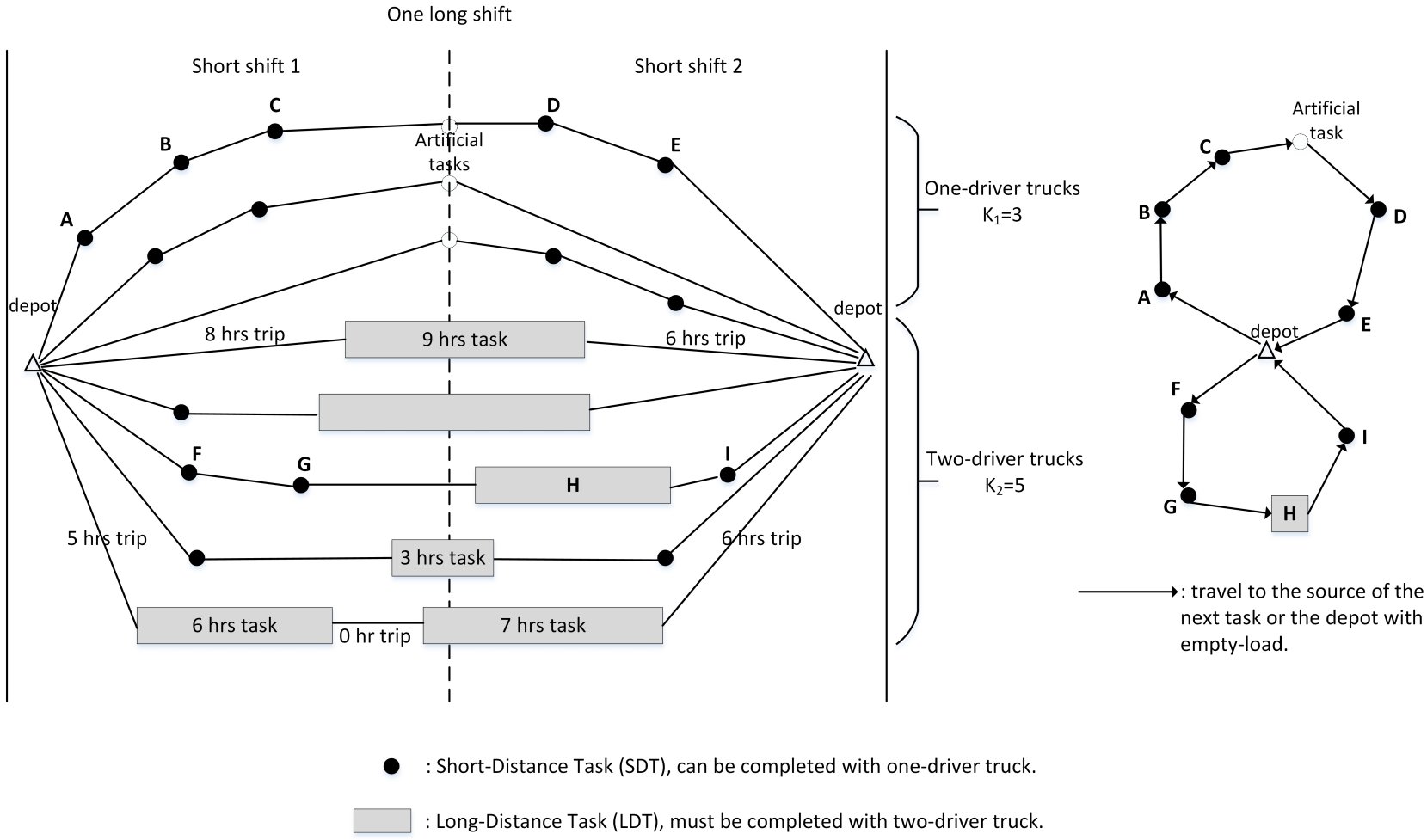


Figure 3.2: A schedule example of one *long shift* (two *short shifts*). Three *one-driver* trucks and five *double-driver* trucks are used in this schedule. The right subgraph presents the first and sixth routes in the schedule.

3.2.2 Mathematical Model

To model the MS-VRPTW, a number of notations are defined, see Table 3.4.

Table 3.4: Definition of Notations

Input Parameters:	
S	The set of time-continuous working shifts. Here one shift is 24 hours (*long shift*).
$P = \{O, D\}$	The set of truck types. O represents that the truck used is a *one-driver* truck, while D means it is a *double-driver* truck.
P_o, P_d	The operating cost of using a *one-driver* truck (P_o) and that of a *double-driver* truck (P_d). They are mainly determined by the payments to the drivers.
K	The fleet size, which is the number of available trucks.
$[Y_s, Z_s]$	Time window of shift $s \in S$.
$N = \{0, 1, 2, ..., n\}$	Set of $n+1$ nodes. Each node represents a task except node 0, which is the depot.
$[a_i, b_i]$	The time window for node $i \in N$. The time window for the depot is zero at the boundary of a shift. If a truck arrives at the source of i early, it has to wait until a_i.
A	Set of arcs. Each arc(i,j) represents node j being immediately serviced/visited after node i.
c_{ij}	The cost of traveling from node i to node j. If both nodes are tasks, it is the travel distance from the destination of i to the source of j. Otherwise, it is the distance from the depot to the first source or from the last destination to the depot. These travels are empty-load with no container carried.
t_{ij}	The travel time from node i to node j. When both nodes are tasks, t_{ij} is the travel time from the destination of i to the source of j. Otherwise, it is the travel time from the depot to the first source or from the last destination to the depot.
l_i	The time for servicing node i, which includes the loading time, transportation time (from pick-up source to delivery destination) and unloading time. The service time of the depot is zero.
W	Artificial task set. Artificial tasks ($w \in W$) can only be found in *one-driver* routes, representing the shift-change (e.g. the hollow circles in Figure. 3.2). Artificial task's service time (l_w) and loaded travel distance are zero (i.e. its source and destination are the same terminal). The source and destination of an artificial task must be either the last destination of the day shift or the first source of the night shift in that route. The time window of w is the mid-line of workday, i.e. $[a_w, b_w]$=[8pm,8pm].
Variables:	
T_i	The time of arrival at node i.
B_i	The time to begin the service of node i.
x_{ij}^{sp}	A binary decision variable for nodes $i, j \in N \cup W$, $s \in S$, $p \in P$. Its value is 1 when arc(i,j) is included in the solution in shift s by a truck type p, otherwise its value is 0.
$K_O^s \in \{0, 1, ..., K\}$	An integer variable of the number of *one-driver* trucks used in shift $s \in S$.
$K_D^s \in \{0, 1, ..., K\}$	An integer variable of the number of *double-driver* trucks used in shift $s \in S$.

The MS-OPVRPTW can be formally defined as follows:

Objective:

$$Minimise \qquad DP = \sum_{s \in S} (P_o \cdot K_O^s + P_d \cdot K_D^s) \tag{3.19}$$

$$Minimise \qquad TD = \sum_{p \in P} \sum_{s \in S} \sum_{i \in N \cup W} \sum_{j \in N \cup W} c_{ij} \cdot x_{ij}^{sp} \tag{3.20}$$

Subject to:

$$\sum_{p \in P} \sum_{s \in S} \sum_{i \in W \cup N \setminus \{0\}} x_{ij}^{sp} = 1, \qquad \forall j \in N \setminus \{0\} \tag{3.21}$$

$$\sum_{p\in P}\sum_{s\in S}\sum_{j\in W\cup N\setminus\{0\}} x_{ij}^{sp} = 1, \qquad \forall i \in N\setminus\{0\} \quad (3.22)$$

$$\sum_{i\in W\cup N\setminus\{0\}} x_{ij}^{sp} = \sum_{f\in W\cup N\setminus\{0\}} x_{jf}^{sp}, \qquad \forall j \in W\cup N\setminus\{0\}, s\in S, p\in P \quad (3.23)$$

$$K_O^s + K_D^s \le K, \qquad \forall s \in S \quad (3.24)$$

$$\sum_{j\in N\cup W} x_{0j}^{sp} = \begin{cases} K_O^s & \forall s\in S, p = O \\ K_D^s & \forall s\in S, p = D \end{cases} \quad (3.25)$$

$$\sum_{i\in N\cup W} x_{i0}^{sp} = \begin{cases} K_O^s & \forall s\in S, p = O \\ K_D^s & \forall s\in S, p = D \end{cases} \quad (3.26)$$

$$\sum_{p\in P}\sum_{j\in N} x_{wj}^{sp} = K_O^s \qquad \forall w \in W, s\in S \quad (3.27)$$

$$\sum_{p\in P}\sum_{i\in N} x_{iw}^{sp} = K_O^s \qquad \forall w \in W, s\in S \quad (3.28)$$

$$\sum_{j\in N} x_{wj}^{sO} = K_O^s \qquad \forall w \in W, s\in S \quad (3.29)$$

$$\sum_{i\in N} x_{iw}^{sO} = K_O^s \qquad \forall w \in W, s\in S \quad (3.30)$$

$$x_{wv}^{sp} = 0, \qquad \forall w, v \in W, s\in S, p\in P \quad (3.31)$$

$$T_j = \sum_{p\in P}\sum_{s\in S}((B_i + l_i + t_{ij})\cdot x_{ij}^{sp} + (Y_s + t_{0j})\cdot x_{0j}^{sp}), \quad \forall i \in N\setminus\{0\} \quad (3.32)$$

$$B_j = T_j + max\{a_j - T_j, 0\}, \qquad \forall j \in N\setminus\{0\} \quad (3.33)$$

$$x_{ij}^{s0}\cdot(B_i + l_i + t_{i0}) \le x_{i0}^{sp}\cdot Z_s, \qquad \forall i \in N\cup W, s\in S, p\in P \quad (3.34)$$

$$a_i \le B_i \le b_i - l_i, \qquad \forall i \in N\setminus\{0\} \quad (3.35)$$

$$x_{ij}^{sp} \in \{0,1\} \qquad \forall i,j \in N\cup W, s\in S, p\in P \quad (3.36)$$

$$K_O^s \in \{0,1,...,K\} \qquad \forall s\in S \quad (3.37)$$

$$K_D^s \in \{0,1,...,K\} \qquad \forall s\in S \quad (3.38)$$

MS-VRPTW is a bi-objective problem. The first objective is minimizing the total driver payment (DP), see equation (3.19), which depends on the number and types of the trucks used. It is notable that, the cost of a driver for *double-driver* truck is 1.5 times of a driver of *one-driver* truck in our study (i.e. $P_o = 1, P_d = 1.5$). Minimizing the total travel distance (TD) (eq. (3.20)) is the other objective. Actually, the target of TD is to minimize the empty-load travel distance for the same reason explained in Section 3.1.2. DP focuses on the operational cost, and TD concentrates on the utility of trucks which actually pursues a higher heavy-loaded travel distance rate in total travel distance.

Constraints (3.21) and (3.22) denote that every task node can be visited exactly once and all the tasks must be visited. Constraint (3.23) specifies that a task may

only be serviced after the previous task is completed. Constraints (3.21) - (3.23) together make sure arcs over more than one shift are unacceptable. Constraint (3.24) guarantees the number of trucks used is not larger than the fleet size.

Constraints (3.25) and (3.26) place the limits on *one-driver* truck ($p = O$) and *double-driver* truck ($p = D$). Constraints (3.27) - (3.30) guarantee that there must be K_O^s artificial tasks completed on the routes of *one-driver* trucks, while there is no artificial task on the routes of *double-driver* trucks. In addition, constraint (3.31) guarantees each route of *one-driver* truck has only one artificial task.

Equation (3.32) defines the arrival time at a task node. Equation (3.33) defines the beginning time of servicing a task node. This time is calculated by the arrival time plus the waiting time at the source of a task. Equations (3.32) and (3.33) enforce the correct successive relationship between consecutive nodes. Constraints (3.32) - (3.34) together define the time windows of shifts. Constraint (3.35) represents the time constraint on each task. The domains of the decision variables are presented in equations (3.36) - (3.38).

From this mixed integer programming (MIP) model, we can find that the MS-VRPTW is a large-scale and tightly constrained nonlinear problem. In MS-VRPTW, the size of solution space is decided by the number of tasks (n), the number of shifts ($|S|$) and the size of the fleet (K). Since there are $|S| \cdot K$ possible routes in a solution, which are either *one-drive* or *double-driver*, and each route has $n!$ permutations of tasks, the size of the search space is $2^{|S| \cdot K} \cdot n!$. In real-life, a logistics company may face hundreds to thousands of containers to be transited every day, leading to a highly complex problem with huge solution space.

3.2.3 Benchmark Instances

A benchmark of 24 instances with various features are generated (available at http://www.cs.nott.ac.uk/˜pszrq/benchmarks.htm) to evaluate the performance of solution methodologies in different scenarios.

Because there is no existing dataset including both long-haul and short-haul data, the two types of data from the company's historical dataset are merged as candidate data when creating the benchmark. The benchmark instances are then generated by extracting items from the merged dataset. In these instances, each item represents a commodity, which consists of its commodity ID, source terminal, destination terminal, available time to transport, deadline of completing the tasks, and the number of containers in this commodity. The same as the PVRPTW-O benchmark, here the number of containers in a commodity can also be larger

than one, meaning finishing one commodity transportation may need to complete multiple transport tasks.

To guarantee the instance diversity and benchmark dataset generality, a categorisation scheme similar to [16] is adopted to define the features of the instances. Firstly, to a LDT, if its time window is smaller than 20 hours, it will be classified as an *emergent* task. The time window for SDT is smaller than 10 hours. These two values are suggested by the port company's coordinator. In addition, an index B (eq. (3.39)) is used to measure the total throughput balance at terminals in each instance.

$$B = \frac{1}{|V|} \sum_{i \in V} |I_i - O_i| \tag{3.39}$$

Here, V is the set of terminals is composed of the harbors and dry ports. I_i and O_i respectively represent the number of incoming and outgoing tasks at terminal i. A smaller B represents a more balanced throughput in the instance. Based on these, four types of features are used to create the benchmark instances.

- ***Tight*** instance: 70%-80% tasks in the instance are emergent.
- ***Loose*** instance: less than 30% tasks in the instance are emergent.
- ***Balanced*** instance: the value of B in the instance is smaller than 30.
- ***Unbalanced*** instance: the value of B in the instance is larger than or equal to 30.

With two types of scheduling horizons (two and four days), we created in total eight combinations of features. They represent a comprehensive dataset of instances with various commodity emergency and workload balance. For each combination, three instances are generated in sizes of small, medium and large, respectively. The details of instances are presented in Table 3.5. The last column provides the total *loaded travel* distances which are fixed in instances. These instances are generated based on the problem characteristics at Ningbo Port, e.g. the geographical distribution of the terminals and the lengths of shifts, and can be used as a set of benchmark instances with diverse features for testing the solution methods of other MS-VRPTWs.

Table 3.5: Features of the benchmark instances. The shifts adopted are *long shifts*.

Instance	Configuration	No. of Shifts	No. of Commodities	No. of Tasks	Loaded TD
LB2-1	Loose Balanced	2	50	145	27474
LB2-2	Loose Balanced	2	100	566	122878
LB2-3	Loose Balanced	2	200	697	179802
LU2-1	Loose Unbalanced	2	50	390	78891
LU2-2	Loose Unbalanced	2	100	551	132220.5
LU2-3	Loose Unbalanced	2	200	768	196460
TB2-1	Tight Balanced	2	50	245	44674
TB2-2	Tight Balanced	2	100	446	98062.5
TB2-3	Tight Balanced	2	200	779	163255
TU2-1	Tight Unbalanced	2	50	364	55854
TU2-2	Tight Unbalanced	2	100	529	97656.5
TU2-3	Tight Unbalanced	2	200	895	190897.5
LB4-1	Loose Balanced	4	50	156	39471
LB4-2	Loose Balanced	4	100	578	121575.5
LB4-3	Loose Balanced	4	200	976	175464
LU4-1	Loose Unbalanced	4	50	395	97047
LU4-2	Loose Unbalanced	4	100	670	150680.5
LU4-3	Loose Unbalanced	4	200	1077	283463
TB4-1	Tight Balanced	4	50	321	69536
TB4-2	Tight Balanced	4	100	536	118923
TB4-3	Tight Balanced	4	200	914	185164.5
TU4-1	Tight Unbalanced	4	50	389	92008
TU4-2	Tight Unbalanced	4	100	606	127203
TU4-3	Tight Unbalanced	4	200	886	185556.5

3.3 Summary

The investigation of the Ningbo Port problem is divided into two stages in this thesis. The short-haul container transshipment problem between docks is studied first, defining a periodic vehicle routing problem with time windows and open routes (PVRPTW-O). To further improve fleet utilisation, the empty-load travel within a day is reduced in the newly established problem model. The proposed scheme differs from the one in [16]. Artificial depots are introduced to allow the shift-change to be carried out at container terminals instead of the genuine depot. The analysis of the mathematical model in subsection 3.1.2 shows the large size of the search space in PVRPTW-O. The proposed scheduling scheme of new shift changing rule cooperates with the solution methodologies introduced in Chapter 5, promoting the usage efficiency of the fleet. This model is also adopted by the next stage research.

A larger scale problem considering overall tasks is later studied. At the second stage of the Ningbo Port problem research, the vehicle routing and scheduling of two types of shifts with different durations are integrated into one algorithm framework, handling both long-distance and short-distance transshipment requests. The proposed mathematical problem model is bi-objective, aiming to minimise driver wage costs and the total travel distance of the fleet. Considering the time window tightness and the throughput balance at terminals, a benchmark of 24 instances

with diverse features is generated based on the company's history data, which can also be a testing benchmark for other MPVRPTW variants.

This chapter explores the models of real-life transportation problems in different scenarios. The proposed models show that artificial nodes can be used to adjust shift change rules and combine different types of shifts. This is promising in modelling other more complex practical problems. To Ningbo Port company, without upgrading hardware facilities in the port, the proposed new scheduling scheme and mathematical models lay the foundation for improving the efficiency of the truck fleet and reducing operational costs for the company (for more details see Chapters 5, 6). This two-stage research explores the room for improvement in the current automatic planning system and provides a two-step upgrading programme for the EDI.

Chapter 4

A Variable Neighbourhood Search Algorithm with Compounded Neighbourhoods for VRPTW

The Vehicle Routing Problem with Time Windows (VRPTW) is a basic distribution management problem which is often used to model real-life routing and scheduling problems. Many complex practical transportation routing and scheduling problems are investigated based on the extended VRPTW models. To develop better solution methodologies for the Ningbo Port problem, the methods for the classic VRPTW are studied first in this thesis. In this chapter, the formal formulation of VRPTW introduced in Section 2.1.1 is used.

Intensive research has been done for the VRPTW which mainly focuses on heuristic and metaheuristic methods because exploring exclusively the solution space of real-life size problems with exact search is unrealistic. Among the various metaheuristics, the Variable Neighbourhood Search (VNS) method has shown powerful exploration performance as it takes the advantages of diverse neighbourhood structures. A VNS algorithm with compounded neighbourhood operators (VNS-C) for VRPTW is introduced in this chapter, providing the new ideas from the following aspects:

- Comparing with independent neighbourhood structures, this study investigates the effectiveness and efficiency of compounded neighbourhoods in local search.
- New ways of compounding independent neighbourhood structures are explored. Comparing with existing neighbourhood operators, the proposed neighbourhood operators not only operate on different positions within the routes, but also have different lengths of exchange segments. A random selection scheme is adopted to set the lengths of the exchange segments, while

a deterministic limit is imposed. By using this way of compounding operations upon both intra-route and inter-route neighbourhood solutions, two compounded neighbourhoods are produced.

- A third neighbourhood operator which compounds segment insertion operators with the same length limit is developed to reduce the number of vehicles used (NV). By adopting the three new proposed operators, VNS-C is able to optimise both objectives (NV and TD) simultaneously in each iteration by shifting the search over broader search regions.

- In the General VNS framework, the impact of algorithm components on the algorithm performance is analysed, e.g. the usage of Shaking, the order of the neighbourhood operators used, and so on.

4.1 Variable Neighbourhood Search with Compounded Neighbourhoods

This section introduces the key components of VNS-C, including the neighbourhood and shaking operators used, the local search strategy and its acceptance criterion. The algorithm framework is presented in subsection 4.1.4.

4.1.1 Compounded Operators and Neighbourhoods

Because of the time window constraint in VRPTW, the reverse operation in the standard λ-opt and λ-interchange operators tends to cause infeasible solutions, thus *Or-opt* and *CROSS-exchange* are adopted in the compound operators in VNS-C.

Based on the Or-opt exchange operator, an improved Or-opt intra-route operator *Or-opt-i* is devised, where i is the length limit of two randomly selected exchange links. In an *Or-opt-i* exchange, the length of one exchange link is fixed to i to avoid redundant exchanges while the length of the other exchange link is randomly set to up to i. This exchange link length setting cooperates with random operation position selection, composing the compounding manner of the proposed compound neighbourhoods. For example, *Or-opt-3* is a compound neighbourhood which assembles three independent neighbourhood moves (where one exchange link length is fixed to 3, while the other one's length could be 1, 2 or 3). An illustrative example is presented in Figure 4.1 (a), where directions of both operated links are kept.

A *CROSS-i* compound operator is also proposed in VNS-C. It includes all independent CROSS-exchanges to exchange a link of length i with another link

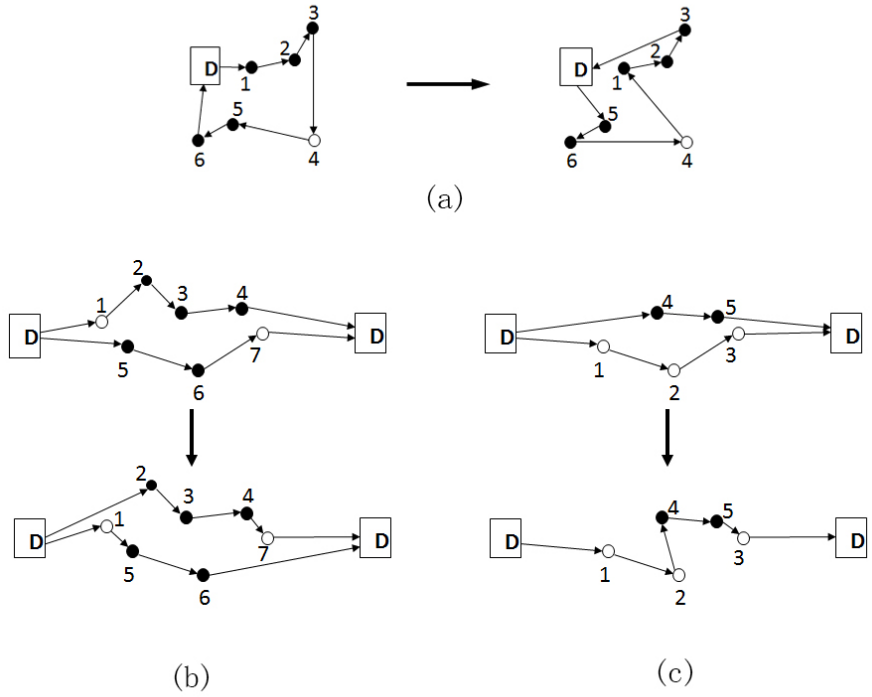


Figure 4.1: Examples of neighbourhoods in VNS-C. (a). Link (1,2,3) exchanged with link (5,6) by the intra-route *Or-opt-i* compound operator (i=3). (b). Link (2,3,4) exchanged with link (5,6) by the inter-route compound operator *CROSS-i* (i=3). (c). Link (4,5) is removed by *LinkMove-i* from its original route and inserted to a random position in the target route, which brings a route reduction.

of length up to i between routes. For instance, the compound neighbourhood of *CROSS-3* assembles three independent neighbourhood moves where length of one exchange link is fixed to 3 and the other one is randomly set to up to 3 (denoted as 3-1, 3-2 and 3-3 independent exchanges, respectively). In one move of *CROSS-3*, the best solution among all the three independent neighbourhoods is selected. While, in a standard independent neighbourhood search, the best solution based on only one of the 3-1, 3-2 or 3-3 exchanges will be selected. An example of *CROSS-3* is presented in Figure 4.1 (b), where the selected improvement solution is produced by a 3-2 exchange.

In the proposed VNS-C, an operator named *LinkMove-i* is developed to reduce both the vehicle number and total travel distance simultaneously (i is the max length of operated links), rather than in two separate phases. In *LinkMove-i*, a customer link of length α ($\alpha \leq i$) from route h is removed and reinserted into route t ($h \neq t$). When α is equal to the length of route h, route h would be removed thus leads to a solution with one less route. Figure 4.1 (c) presents an example of

LinkMove-i.

The proposed compound neighbourhoods explore larger search areas than standard independent neighbourhoods. Following the rule of invoking small neighbourhood moves first, the upper bound i of link length in compound operators is set to increase from 1 to 5 in VNS-C based on preliminary experiment results. The order to select the intra-route neighbourhood or inter-route neighbourhood first is shown to be an influence factor in VNS-C in our experimental results. This is studied in subsection 4.2.2.

4.1.2 Shaking and Shaking Operators

Shaking(S) is a phase of random perturbation in VNS-C, which randomly generates a neighbourhood solution S' of the current solution S using the three simple operators in Table 4.1, aiming to escape from local optima. *ExchangeInRoute-ml* and *Cross-ml* exchanges two segments within one route and between two randomly selected routes, respectively. *Move-ml* inserts a randomly selected route segment from a route to another route. In all three operators, the maximum length of the segment is *ml*. Different from the above-mentioned compound operators, operators in *Shaking* are more flexible, without the requirement that at least one segment's length must be *ml*.

Table 4.1: Set of neighbourhood operators in Shaking. z is a random variable for selecting an operator. L is the length of associated routes.

z	Operator	Min length	Max length(ml)
$0 \sim 3$	*ExchangeInRoute-ml*	1	Min($z+1$, L)
$4 \sim 8$	*Move-ml*	1	Min(z mod 3 , L)
$9 \sim 12$	*Cross-ml*	1	Min(z mod 8 , L)

The first feasible move will be accepted in *Shaking(S)*. To encourage farther moves, the segment of length ml is selected with a higher probability. If no feasible moves are found after a pre-specified number of evaluations, the original input solution S would be returned. The impact of Shaking is studied in subsection 4.2.1.

4.1.3 Local Search

In the local search of VNS-C (see Algorithm 1), NS_{max} evaluations are undertaken in each run of neighbourhoods. Literature [147] recommends that, when the initial solution is constructed by a heuristic, the Best-Improvement acceptance criterion should be used in VNS. The initial solution in VNS-C is constructed using the Nearest Neighbourhood heuristic from [282], and the best neighbourhood solutions

are chosen. To avoid being stuck to local optima, *Record-to-Record Travel* algorithm [96] is adopted as the acceptance criteria, where *Quality()* is defined by the total travel distance, and *DEVIATION* is set to 15. Here a solution with the lower *Quality()* value is better. The search stops at a time limit of $Time_{max}$ or when all three compound neighbourhoods are estimated but no improvement is found. In Algorithm 1, $N_r(S', i)$ represents the rth neighbourhood operator applied to the incumbent solution S' with operated link length limit of i.

Algorithm 1 Local Search(S', S)

Step 1: Input solution S' and S.
Step 2: Set $r \leftarrow 1$, $i \leftarrow 1$, $time \leftarrow 0$.
  while ($r < 4$ **And** $time < Time_{max}$) **do**
    Step 2.1: Neighbourhood Search
      $S'' \leftarrow$ Best Improvement of $N_r(S', i)$.
      $time \leftarrow time + NS_{max}$.
    Step 2.2: Move or Not
      if $Quality(S'') < Quality(S)$ **then**
        $S \leftarrow S'', S' \leftarrow S'', i \leftarrow 0, r \leftarrow 1$.
      else if $Quality(S'') - Quality(S) < DEVIATION$ **then**
        $S' \leftarrow S'', i \leftarrow 0, r \leftarrow 1$.
      end if
    Step 2.3: Shift Neighbourhood Structure
      $i \leftarrow i + 1$.
      if $i = 6$ **then** $r \leftarrow r + 1$, $i \leftarrow 1$.
  end while
Step 3: Output the best found solution S.

4.1.4 The Framework of VNS-C

Algorithm 2 The VNS-C framework

Step 1: Input the set of customer demands, locations and the associated time windows.
Step 2: Generate an initial feasible solution S by the Nearest Neighbourhood heuristic.
Step 3:
  Set $C \leftarrow 1$.
  while $C < C_{max}$ **do**
    Step 3.1: $S' \leftarrow$ **Shaking**(S).
    Step 3.2: $S \leftarrow$ **Local Search**(S', S).
    Step 3.3:
      if S is improved **then**
        $C \leftarrow 1$.
      else
        $C \leftarrow C + 1$.
      end if
  end while
Step 4: Output S.

The pseudo-code of VNS-C is presented in Algorithm 2, where the iteration time is set to C_{max}. In ***Step 2***, an initial solution is constructed using a heuristic,

which inserts the "closest" available customer into the incumbent partial route. Here the distance between two customers is defined by their Geographic distance d_{ij}, Temporal distance T_{ij} and the degree of Emergency v_{ij} which are used in [282], shown in eq. (4.1) as below:

$$Dis = \delta_1 \cdot d_{ij} + \delta_2 \cdot T_{ij} + \delta_3 \cdot v_{ij} \quad s.t.\ \delta_1 + \delta_2 + \delta_3 \ = \ 1 \tag{4.1}$$

The three coefficients δ_1, δ_2 and δ_3 define the importance of each component in the distance definition. They are set as $\delta_1 = 0.4$, $\delta_2 = 0.4$ and $\delta_3 = 0.2$ (empirically calculated by Solomon [282]).

4.2 Experiments and Results Analysis

The proposed VNS-C was evaluated on the Solomon benchmark [282], introduced in subsection 2.1.1. The tuning is conducted on only one parameter at a time, while fixing all the others on a small number of instances. Preliminary experiments show that most feasible solutions in Shaking are found in around 200 evaluations, thus 300 evaluations are conducted. For each incumbent solution in the local search, 400 neighbourhoods are evaluated, i.e. $NS_{max} = 400$. $Time_{max}$ is set to 1,000,000 evaluations while the max iteration time C_{max} of VNS-C is 300. All results are produced in 30 runs to conduct statistical analysis.

4.2.1 Compounded Neighbourhoods and Shaking

Table 4.2 presents the average results from VNS-C, VNS with independent operators (standard Or-opt and CROSS exchange) and VNS-C without Shaking on six instances from different groups. NV denotes the number of vehicles, TD represents the total travel distance, and Eval is the total number of evaluations. S.D is the standard deviation. It is shown that VNS-C produces significantly better and more stable results compared to the other variants. *Shaking* also improves VNS-C in terms of both quality and stability, thus is an essential and necessary component in VNS-C.

To verify whether the result of VNS-C is significantly different from the other three algorithms', T-test is executed between results of VNS-C and the other three algorithms. Here the confidence level is set as 95%. Table 4.3 presents the test result, where Y represents two populations are significantly different, and N the opposite. Notably, as the solutions have two dimensions of NV and TD, as long as the p-value (two-tail) is smaller than 5% in one dimension, the two populations

Table 4.2: Comparison of VNS-C and VNS of Independent Operator with and without Shaking. Best results are in bold.

Instance			C101	C201	R101	R201	RC101	RC201
VNS-C & Shaking	Best	NV	**10**	**3**	**19**	**4**	**15**	**4**
		TD	**828.94**	**591.56**	**1643.34**	**1190.52**	**1624.97**	**1310.44**
	Average	NV	**10**	**3**	**19.9**	4.83	**15.6**	4.93
		TD	**828.94**	**591.56**	**1647.9**	**1246.91**	**1652.38**	**1365.76**
		Eval	67,247,717	97,011,547	188,429,463	176,349,146	113,982,405	137,842,632
	S.D on NV		**0**	**0**	**0.31**	**0.38**	0.63	0.25
	S.D on TD		**0**	**0**	**5.59**	**45.12**	**12.88**	**41.57**
Independent Operators & Shaking	Best	NV	**10**	**3**	**19**	**4**	16	**4**
		TD	**828.94**	**591.56**	1700.42	1339.84	1753.49	1482.86
	Average	NV	**10**	**3**	20.13	4.73	16.3	4.97
		TD	**828.94**	**591.56**	1791.73	1538.66	1878.68	1569.51
		Eval	9,808,756	9,800,000	159,166,298	128,370,863	79,996,218	146,881,580
	S.D on NV		**0**	**0**	0.68	0.45	**0.47**	**0.18**
	S.D on TD		4.38	**0**	89.32	212.51	101.68	59.89
VNS-C without Shaking	Best	NV	**10**	**3**	**19**	**4**	**15**	**4**
		TD	**828.94**	**591.56**	1644.55	1294.36	1644.18	1340.79
	Average	NV	**10**	**3**	20.43	4.73	16.43	4.93
		TD	**828.94**	591.56	1823.72	1511.68	1856.99	1489.35
		Eval	10,026,352	97,010,797	77,128,988	124,141,441	104,983,387	113,826,219
	S.D on NV		**0**	**0**	0.63	0.45	0.68	0.25
	S.D on TD		**0**	**0**	178.68	394.21	213.67	270.99
Independent Operators without Shaking	Best	NV	**10**	**3**	**19**	**4**	**15**	**4**
		TD	**828.94**	**591.56**	1649.23	1226.43	**1624.97**	1332.74
	Average	NV	**10**	**3**	20.33	**4.6**	16.17	4.97
		TD	**828.94**	**591.56**	1828.49	1671.69	1859.35	1443.03
		Eval	10,198,465	97,066,537	24,214,377	101,344,273	60,738,070	128,573,526
	S.D on NV		**0**	**0**	0.76	0.5	0.91	**0.18**
	S.D on TD		**0**	**0**	173.87	401.52	211.84	199.56

Table 4.3: T-test between VNS-C and the other three algorithms.

Compared Algorithms		C101	C201	R101	R201	RC101	RC201
Independent Neighbourhoods with Shaking	P-value(NV)	1	1	0.094698	0.355754	1.49E-06	0.561629
	P-value(TD)	0.321464	1	1.1E-09	2.3E-08	1.33E-05	6.62E-21
	Different	N	N	Y	Y	Y	Y
Compound Neighbourhoods without Shaking	P-value(NV)	1	1	0.000138	0.355754	0.004581	1
	P-value(TD)	1	1	8.66E-06	0.000976	9.92E-06	0.019461
	Different	N	N	Y	Y	Y	Y
Independent Neighbourhoods without Shaking	P-value(NV)	1	1	0.006101	0.046114	5.73E-07	0.561629
	P-value(TD)	1	1	3.78E-06	2.74E-06	4.91E-13	0.04599
	Different	N	N	Y	Y	Y	Y

would be considered as significantly different. It can be seen that VNS-C produces significantly better solutions than the other three algorithms on complicated instances (R and RC). For the two C instances, there is no significant difference between VNS-C and the other three algorithms. Results against those in the literature (see Table 4.5) indicate that all these four algorithms obtained the best solution for these two instances, thus no significant difference has been found.

4.2.2 Neighbourhood Order

Table 4.4 compares different orders of neighbourhoods employed in VNS-C (M, C and I represent *LinkMove-i*, *CROSS-i* and *Or-opt-i*, respectively). It can be seen that, the Inter-Route move first group (MCI and CMI) achieves better results than the Intra-Route move first ones (ICM and IMC). In the former case, MCI performs better than CMI. It seems that optimising the route number first, and then assigning customers to a route and optimising the customer order in each route can bring better results. On R101 and R201, MCI obtains better NV while

CMI has better TD. As objective (1) is usually considered as primary, the order of MCI will be used in VNS-C.

Table 4.4: Results of VNS-C with four different operator orders. Best results are in bold.

Instance			C101	C201	R101	R201	RC101	RC201
MCI	Best	NV	**10**	**3**	19	**4**	15	**4**
		TD	**828.94**	**591.56**	1643.34	**1190.52**	1624.97	**1310.44**
	Average	NV	**10**	**3**	19.9	**4.83**	**15.6**	**4.93**
		TD	**828.94**	**591.56**	1647.9	**1246.91**	**1652.38**	**1365.76**
		Eval	67,247,717	97,011,54	188,429,463	176,349,146	113,982,405	137,842,632
CMI	Best	NV	**10**	**3**	**20**	**5**	**15**	5
		TD	**828.94**	**591.56**	**1642.88**	**1189.82**	**1623.58**	1317.97
	Average	NV	**10**	**3**	20.3	**5**	15.87	5
		TD	**828.94**	**591.56**	1650.42	**1214.66**	1654.65	1367.59
		Eval	69,224,764	97,011,573	181,014,272	166,688,866	184,803,844	158,724,815
ICM	Best	NV	**10**	**3**	**20**	4	15	5
		TD	**828.94**	**591.56**	**1642.88**	1237.76	1715.49	1354.72
	Average	NV	**10**	**3**	20.17	4.9	16.4	5
		TD	**828.94**	**591.56**	1648.9	1306.7	1762.34	1425.46
		Eval	51,225,270	97,015,461	223,733,841	307,952,033	692,934,814	516,816,576
IMC	Best	NV	**10**	**3**	**19**	4	16	5
		TD	**828.94**	**591.56**	**1643.18**	1234.09	1672.33	1376.17
	Average	NV	**10**	**3**	**19.9**	**4.7**	16.27	5
		TD	**828.94**	**591.56**	**1647.52**	**1334.32**	1765	1464.56
		Eval	67,357,647	97,015,347	199,078,925	311,452,951	735,044,303	492,990,103

4.2.3 Experiment Results and Analysis

Table 4.5 presents the results on all the 56 Benchmark Solomons instances. It illustrates that VNS-C is effective in improving both objectives simultaneously. In problems whose objectives are positively correlated (C and R), VNS-C can produce the current best known solutions in a reasonable time. In other instances, some better solutions with less TD are found comparing to the best known solutions with the same NV in the literature.

Table 4.5: VNS-C on Benchmark Solomon's instances. Best found solutions in the literature are shown in bold

Instance	Best Known			VNS-C			
				Best		Average	
	NV	TD	Ref.	NV	TD	NV	TD
C101	10	828.94	[280]	**10**	**828.94**	10	828.94
C102	10	828.94	[280]	**10**	**828.94**	10	876.79
C103	10	828.06	[280]	10	828.94	10	832.65
C104	10	824.78	[280]	10	825.65	10	831.79
C105	10	828.94	[280]	**10**	**828.94**	10	852.33
C106	10	828.94	[280]	**10**	**828.94**	10.07	836.25
C107	10	828.94	[280]	**10**	**828.94**	10	853.9
C108	10	828.94	[280]	**10**	**828.94**	10	840.48
C109	10	828.94	[280]	**10**	**828.94**	10	823.94
C201	3	591.56	[280]	**3**	**591.56**	3	591.56
C202	3	591.56	[280]	**3**	**591.56**	3.53	613.94
C203	3	591.17	[280]	**3**	**591.17**	3.07	599.16
C204	3	590.6	[280]	**3**	**590.6**	3.23	609.81
C205	3	588.88	[280]	**3**	**588.88**	3	588.88
C206	3	588.49	[280]	**3**	**588.49**	3	588.49
C207	3	588.29	[280]	**3**	**588.29**	3	588.29
C208	3	588.32	[280]	**3**	**588.32**	3	588.32
R101	19	1650.80	[280]	19	1652.47	19.9	1647.90
	20	1642.87	[5]	20	1643.34		
R102	17	1486.12	[280]	18	1476.06	18.9	1493.30
R103	13	1292.67	[280]	14	1219.89	14.17	1230.92
R104	9	1007.31	[280]				
	10	974.24	[288]	10	1007.27	11.1	1009.9
	11	971.5	[178]	11	994.85		

(continued)

Instance	Best Known			VNS-C			
				Best		Average	
	NV	TD	Ref.	NV	TD	NV	TD
R105	14	1377.11	[280]	14	1381.88	15.07	1377.24
	15	1346.12	[167]	15	1360.78		
R106	12	1252.03	[280]			13.57	1264.04
	13	1234.6	[72]	13	1243.72		
R107	10	1104.66	[280]			11.73	1097.07
	11	1051.84	[167]	11	1077.24		
R108	9	960.88	[280]			10.23	974.46
	10	932.1	[230]	10	956.22		
R109	11	1194.73	[280]				
	12	1013.2	[62]	12	1168.18	12.93	1181.99
	13	1151.84	[5]	13	1157.61		
R110	10	1118.84	[280]				
	11	1112.21	[230]			12.1	1106.02
	12	1068	[72]	12	1081.88		
R111	10	1096.72	[280]	**11**	**1087.5**	11.9	1080.1
	12	1048.7	[72]	12	1062.58		
R112	9	982.14	[280]			10.9	979.52
	10	953.63	[260]	10	958.7		
R201	4	1252.37	[280]	4	1282.75	4.83	1246.91
	5	1206.42	[288]	**5**	**1190.52**		
R202	3	1191.7	[280]			4	1146.34
	4	1091.21	[288]	4	1098.06		
R203	3	939.503	[280]	3	968.67	3.5	969.05
	4	935.04	[288]	**4**	**905.72**		
R204	2	825.52	[280]			3	809.88
	3	789.72	[288]	**3**	**766.91**		
R205	3	994.42	[280]	3	1059.91	3.83	1029.55
	5	954.16	[5]	**4**	**964.02**		
R206	3	906.142	[280]	3	931.762	3	994.92
R207	2	890.61	[280]			3	896.72
	3	814.78	[260]	3	855.37		
R208	2	726.82	[280]	**3**	**708.9**	3	740.94
	4	698.88	[300]				
R209	3	909.16	[280]	3	983.75	3.93	920.18
	5	860.11	[5]	4	871.63		
R210	3	939.37	[280]	3	978.11	3.63	992.18
				4	**935.01**		
R211	2	885.71	[280]			3	828.81
	4	761.1	[230]	**3**	**794.04**		
RC101	14	1696.94	[280]			15.6	1652.38
	15	1619.8	[300]	15	1624.97		
RC102	12	1554.75	[280]				
	13	1470.26	[288]	13	1497.43	13.97	1497.056
	14	1466.84	[5]	14	1467.25		
RC103	11	1261.67	[280]	11	1265.86	11.8	1284.24
RC104	10	1135.48	[280]	10	1136.49	10.7	1171.61
RC105	13	1629.44	[280]				
	14	1589.91	[288]	14	1642.81	15.6	1570.33
	15	1513.7	[5]	15	1524.14		
RC106	11	1424.73	[280]	**12**	**1396.59**	13.07	1408.7
	13	1371.69	[288]	13	1376.99		
RC107	11	1230.48	[280]	11	1254.68	11.93	1258.32
	12	1212.83	[5]	12	1233.58		
RC108	10	1139.82	[280]			11	1149.38
	11	1117.53	[5]	11	1131.23		
RC201	4	1406.94	[280]	4	1457.87	4.93	1365.76
	6	1134.91	[288]	**5**	**1310.44**		
RC202	3	1365.64	[280]			4	1278.96
	4	1181.99	[230]	4	1219.49		
RC203	3	1049.62	[280]			4	1020.716
	4	1026.61	[288]	**4**	**957.1**		
RC204	3	798.46	[280]	3	829.13	3	867.85
RC205	4	1297.65	[280]			5	1273.03
	5	1295.46	[288]	**5**	**1233.46**		
RC206	3	1146.32	[280]			4	1152.29
	4	1139.55	[288]	**4**	**1107.4**		
RC207	3	1061.14	[280]			4	1084.44
	4	1079.07	[260]	**4**	**1032.78**		
RC208	3	828.14	[280]	3	830.06	3	922.47

Table 4.5 also shows that VNS-C is effective in minimising TD by the results on the complicated datasets (R2, RC2) which use fewer vehicles to satisfy 100 demands. Thus these instances require a powerful neighbourhood operator to reduce NV. Figure 4.2 shows that the disparity between our best found solution (NV = 3 and TD = 766.91) and the best known solution with a lower NV (NV = 2 and TD = 825.52) on R204 is large, which indicates that the distance between them is large in the search space. It is difficult for VNS-C to find a lower NV in this case, as the link length limit in *LinkMove-i* (5) is too small compared to the

route length (33).

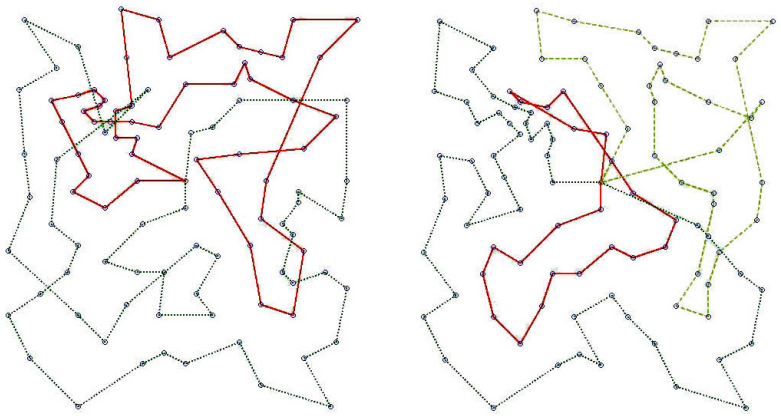


Figure 4.2: Two best found solutions with 2 and 3 vehicles on R204.

To further investigate the performance of *LinkMove-i* on reducing NV, six different upper bound values of i (Max_i) are set to this operator. Figures 4.3 and 4.4 demonstrate the performance of the *LinkMove-i* operator with diverse Max_i on minimizing NV. In Figure 4.3 it can be seen that, higher Max_i can produce a lower average NV on four complicated instances (R101, R201, RC101 and R201) while the best found NV is unchanged. Too small Max_i would insert only short routes to other routes, and the capability of reducing NV would become weaker consequently. This hypothesis is consistent to the observation on RC201 in Figure 4.4 that, when Max_i is small (1, 2 and 3) the best found NV (5) is greater than the one (4) found with larger Max_i values (5, 7 and 9). In addition, experiment results also show that some of the lowest TD can be found with small Max_i, while their average NVs are the largest. e.g. the lowest TDs are found on R101, RC101 and RC201 with $Max_i = 1$, as well as on R201 with $Max_i = 2$. This observation indicates the conflicting relation of both objectives on these instances.

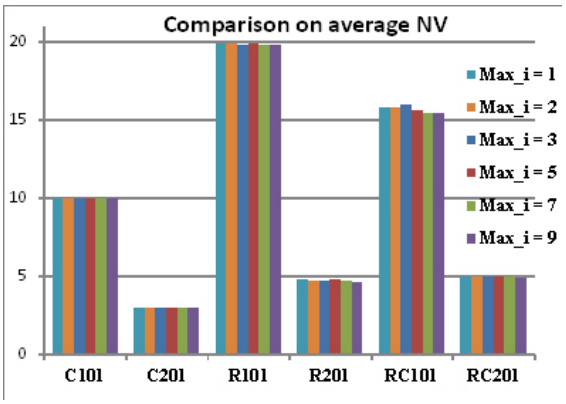


Figure 4.3: Comparison on average NV with different Max_i values of *LinkMove-i*.

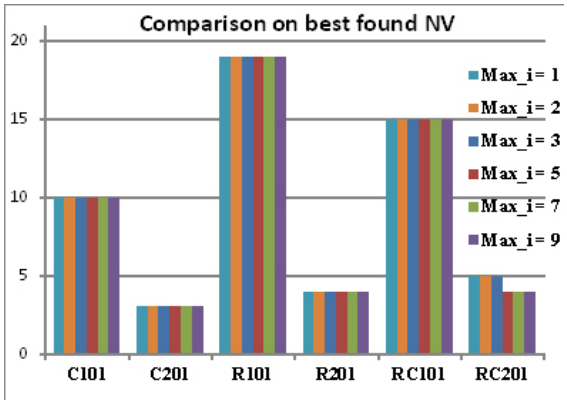


Figure 4.4: Comparison on the best found NV with different Max_i values of *LinkMove-i*.

Or-opt-i and *CROSS-i* reduce TD by intra-route moves and inter-route moves respectively. In theory, the neighbourhoods of *CROSS-i* cover the neighbourhoods of *Or-opt-i* in search space, while more move steps are required. To clarify whether *Or-opt-i* is redundant in our algorithm and which operator makes more contribution on reducing TD, we respectively remove one of these two operators from VNS-C and evaluate the two algorithm variants. The experiment results are presented in Table 4.6, which gives the gaps between the new variants' results and the original VNS-C's results on TD. It can be found that, both algorithm variants' performance decreases in varying degrees on sample instances. Therefore, both operators are essential to VNS-C and no one is redundant. In addition, it can also be found that *CROSS-i* plays a greater role than *Or-opt-i* on optimizing the TD objective, since the generated solutions will show more obvious setbacks (larger gap) when *CROSS-i* is removed.

Table 4.6: Gaps of two algorithm variants with the VNS-C

Instances		C101	C201	R101	R201	RC101	RC201
Gaps	Average TD	0.00%	0.00%	0.46%	6.23%	3.62%	3.36%
without CROSS-i	Best TD	0.00%	0.00%	0.02%	0.52%	2.00%	2.35%
Gaps	Average TD	0.00%	0.00%	0.02%	2.07%	1.14%	0.63%
without Or-opt-i	BestTD	0.00%	0.00%	0.00%	1.53%	0.79%	0.00%

4.3 Summary

Comparing to population-based algorithms, local search methods show relatively poorer performance in addressing problems with multiple objectives due to their lack of efficiency in addressing multiple objectives simultaneously. To simultaneously optimise both objectives in VRPTW, a variable neighbourhood search algorithm with compound neighbourhood operators (VNS-C) is developed. In the pro-

posed algorithm, a new way of compounding the independent operators concerning both operation position and exchange link length is adopted in three compound neighbourhood operators. Two proposed compound neighbourhood operators are developed based on the *Or-opt* and *CROSS* exchange operators by considering specific features in VRPTW. Another neighbourhood operator *LinkMove-i*, which can reduce the number of vehicles used simultaneously, is also proposed.

Comparing with the best known solutions in the current literature, experiment results on benchmark datasets show that VNS-C can produce promising results for VRPTW. Especially, on the instances where customers are clustered or route lengths are long, VNS-C generates higher quality solutions. The long-route instances, i.e. with wide scheduling horizon or large vehicle capacity, can be efficiently addressed with VNS-C. Besides, VNS-C shows stronger performance in minimising the total travel distance compared to reducing the number of vehicles/routes. In future work, heuristics with more powerful ability of reducing the number of routes can be integrated to improve the performance of VNS-C in diverse routing problems.

This chapter explores the solution methodology for the classic VRPTW. The proposed algorithm develops a new scheme to balance the diversification and intensification of search, combining the ideas of variable neighbourhood search and variable-depth neighbourhood search. This algorithm and the associated operators can be extended and applied to other real-life vehicle routing and scheduling problems, e.g. the Ningbo Port problem. Based on this research, three improved heuristic algorithms for the real-world container transportation problems are developed. These are introduced in Chapters 5 and 6.

Chapter 5

Two Metaheuristic Approaches for Periodic Vehicle Routing Problem with Time Windows and Open Routes

This chapter investigates the solution methods for the short-haul container transportation problem in the Ningbo Port. The problem is formulated as a periodic vehicle routing problem with time windows and open routes (PVRPTW-O) in Section 3.1. Two metaheuristic methods with different neighbourhood structures and search strategies are proposed for this problem. The study mainly focuses on the following aspects:

- Section 5.1 introduces the problem solutions generated by exact search. Given the formal formulations of an optimisation problem, optimisation solvers employ proper algorithms and strategies to explore the solution space exhaustively, returning the optimal solution or a close approximation of the optimum. There are many successful optimisation solvers in the market. In this section, a state-of-the-art solver is applied to PVRPTW-O, evaluating the complexity of the problem.

- Section 5.2 discusses the solution construction heuristic for PVRPTW-O. In real cases, the amount of customer demands may be huge with either tight or loose constraints, while the throughput at terminals may be balanced or imbalanced. Considering the limited trucks in the fleet, diverse constructive strategies are implemented and tested to develop a generic construction heuristic which can produce feasible solutions in various real cases.

- Learning and sampling mechanism in metaheuristics is studied in Section 5.3. Generally speaking, the heuristics of learning mechanisms do not show competitive performance in VRP research. However, the use of learning schemes and sampling has been widely accepted and shown strong performance in hybrid and adaptive methods. In this study, a reinforcement learning scheme is integrated to the VNS framework, guiding the neighbourhood operator selection adaptively. The impact of Sampling to the single solution-based methods is investigated as well.

- Section 5.4 studies the large neighbourhood search in tightly constrained problems. The PVRPTW-O is constrained by plenty of practical constrains, while some of them are nonlinear. Large Neighbourhood Search (LNS) methods have shown good performance in avoiding being trapped in local optima in the tightly constrained solution space. A new adaptive LNS algorithm with variable-depth neighbourhood is proposed in this section.

- Section 5.5 concludes the findings of this chapter. Some further analysis about this problem is also presented in this section, providing possible directions for further research.

5.1 Exact Search

In Section 3.1.2, the mathematical model of PVRPTW-O has been established based on the inter-dock container transshipment problem in Ningbo Port. CPLEX, which is one of most successful optimisation solvers in the industry, is first employed to solve the model with the default parameter setting in this study. Because CPLEX is a MILP/MIQP solver, some modifications have to be made to the original model of Section 3.1.2 to adapt it to the solver. To this end, the nonlinear constraints (3.5 - 3.8) in the original model are reformulated to the linear form as follows:

$$wait_j \geq a_j - T_j, \qquad \forall j \in N \quad (5.1)$$

$$wait_j \geq 0, \qquad \forall j \in N \quad (5.2)$$

$$B_j = T_j + wait_j, \qquad \forall j \in N \backslash \{0\} \quad (5.3)$$

$$B_i + l_i + t_{ij} - B_j \leq M \cdot (1 - x_{ij}^s), \qquad \forall i, j \in N \backslash \{0\}, s \in S \quad (5.4)$$

$$slack_{ij} \geq B_i + l_i + t_{ij} - M \cdot (1 - x_{ij}^s), \qquad \forall i, j \in N \quad (5.5)$$

$$slack_{ij} \geq 0, \qquad \forall i, j \in N \quad (5.6)$$

$$T_j \geq \sum_{i \in W \cup N \backslash \{j\}} slack_{ij}, \qquad \forall j \in N \quad (5.7)$$

$$Y_s - T_j \geq M \cdot (1 - x_{ij}^s), \qquad \forall i, j \in N \quad (5.8)$$

$$B_i + l_i - Z_s \leq M \cdot (1 - x_{ij}^s), \qquad \forall i, j \in N \quad (5.9)$$

The original constraint (3.6) is linearised to three constraints (5.1)-(5.3), correspondingly, the constraint (3.5) is reformulated to (5.4). Here $wait_j$ is a variable that has been introduced to linearise the function $max\{\}$, and M is a parameter with very large value. In constraints (5.5)-(5.7), the $slack_{ij}$ are variables to force T_j to be larger than the assigning B_i. In constraint (5.5), only the assigned B_i is positive value from this formulation. The non-assigning B_i will be deducted by M so that they are negative. Correspondingly, T_j is required to be larger than the summation of $slack_{ij}$ in constraint (5.7). Note that there is only one $slack_{ij}$ in i that is not zero, so $\sum_i slack_{ij}$ equals $\sum_i max(slack_{ij})$. This formulation reduces the number of constraints in the model, instead of enumerating every $slack_{ij}$ for every i. Constraints (5.8) and (5.9) replace the original constraints (3.7) and (3.8). We do not add extra constraints to the problem model, e.g. subtour elimination constraints, avoiding increasing the memory required for loading it to CPLEX.

In the preliminary experiments, the solver runs on a machine of 16 cores (2.6 GHz), 16 GB memory and 24 hours running time limit. The solver found feasible solutions for 7 out of 16 the 25% scaled down artificial instances and 2 out of 15 real-life instances, while on the other instances ran out of memory. Therefore, we increase the memory resource to 100 GB in the later experiments. The results are reported in Table 5.1 and Table 5.2. In these tables, the objective value of a solution is converted into *Heavy-Loaded Distance Rate* (HLDR), which is widely used by logistics companies in practice, see eq. (5.10). Note that the original objective (3.1) is equivalent to (5.10).

$$HLDR = Loaded\ Distance\ /\ (Loaded\ Distance + Unloaded\ Distance) \quad (5.10)$$

By using HLDR, the problem becomes a maximisation problem. In Tables 5.1 and 5.2, **NF** means no solution is found. **Bound** is the upper bound of HLDR obtained by CPLEX, while **Time** is the actual runtime. + indicates that the memory is insufficient to generate a feasible solution, and * represents that the search reached the memory limit while a feasible solution is obtained.

It can be found that, given a large amount of computing resources, it is still difficult to obtain optimal solutions on all the 25% scaled down instances. Except TU4-7, on other instances, the gaps between the obtained solutions and the upper bounds are quite large. It is no doubt that there must be other exact methods

Table 5.1: CPLEX solutions on the 25% scaled down real-life instances.

	NP4-1	NP4-2	NP4-3	NP4-4	NP4-5
HLDR	78.36%	65.14%	64.83%	54.39%	NF
Bound	92.36%	97.04%	100%	97.72%	100%
Time (hr)	24	24	24	24	24
	NP6-1	**NP6-2**	**NP6-3**	**NP6-4**	**NP6-5**
HLDR	NF	NF	54.30%	NF	66.11%
Bound	NF	NF	95.20%	NF	98.39%
Time (hr)	24	24	24	+	24
	NP8-1	**NP8-2**	**NP8-3**	**NP8-4**	**NP8-5**
HLDR	NF	NF	NF	NF	NF
Bound	98.98%	100%	100%	NF	100%
Time (hr)	24	24	24	+	24

Table 5.2: CPLEX solutions on the 25% scaled down artificial instances.

	LB4-1	LB4-2	TB4-3	TB4-4	LU4-5	LU4-6	TU4-7	TU4-8
HLDR	66.62%	76.41%	69.91%	69.30%	NF	58.65%	50.37%	55.36%
Bound	100%	94.87%	86.31%	83.51%	79.94%	73.90%	52.17%	66.38%
Time (hr)	24	24	13*	19*	24	24	1*	24
	LB8-1	**LB8-2**	**TB8-3**	**TB8-4**	**LU8-5**	**LU8-6**	**TU8-7**	**TU8-8**
HLDR	NF	NF	56.85%	52.40%	57.42%	NF	47.65%	50.74%
Bound	100%	100%	82.33%	88.75%	78.33%	86.84%	71.59%	70.43%
Time (hr)	24	24	24	24	24	24	24	24

may find better solution than CPLEX on this benchmark. However, it still can be concluded that, exact search would be time and resource-consuming in this large scale and nonlinearly constrained problem. The results obtained on the 25% instances provide the upper bounds of the optimal solutions, which can be used to estimate the performance of the other related heuristic algorithms. Therefore, approximate algorithms are developed for the PVRPTW-O in this study.

5.2 Solution Construction Heuristic

5.2.1 A Urgency Level-Based Insertion Heuristic

The construction heuristic for the feasible solution of PVRPTW-O is studied first. Among the various existing construction heuristics for VRPs (see Section 2.2.2), insertion-based heuristics are widely used as it can be easily integrated with diverse insertion selection strategies (i.e. selection criterion and insertion criterion). In addition, by adjusting the insertion strategy applied during the construction procedure, the number of vehicles used can be better controlled. At Ningbo Port, the nine dock container terminals are neither clustered nor uniformly located around the depot, and the triangle rules do not apply to the Ningbo Port transportation network due to the real traffics. These realistic conditions make some classic construction heuristics inapplicable or poor-performance under the circumstances, e.g. Saving Heuristic, Sweep Algorithm and so on. Considering the number of trucks

in this problem is limited (100), an Urgency Level-based Insertion constructive Heuristic (ULIH) is thus developed for PVRPTW-O.

In ULIH, the solution construction is done shift-by-shift. Because the number of containers to be transshipped is large every day in Ningbo Port, to complete all the tasks before their deadlines, the tasks are classified into two categories to specific shifts: *mandatory* and *optional*. For each shift, the mandatory tasks will be assigned first. The urgency level of a task varies along with the change of the operating shift. In brief, to a feasible task i of shift s, if it must be completed no later than s, then it is *mandatory* to shift s; otherwise, it is *optional*. This classification is executed before the solution construction. Tasks are inserted into the scheduled routes, which means being assigned to trucks. When no feasible insertion available to existing routes, a new route will be created, meaning the number of trucks used (k) is increased by one. Considering the *big* tasks (with long service time) are harder to assign in scheduling, the biggest task is selected first in the newly created route.

Another strategy often used in practice is that tasks should be assigned as early as possible to avoid leaving too many unassigned tasks to later shifts. In real-life, there may be new tasks submitted to the scheduling system in real-time, so leaving more free time slots and trucks for later shifts can also improve the system reliability. Thus, after all mandatory tasks being assigned, optional tasks will be inserted until no feasible insertion or free truck is available ($k \leq K$).

The insertion selection tactic determines the solution quality and constructive speed of insertion-based heuristics. Basically, the greedy selection strategies can generate higher quality routes but the computation time needed would be dramatically increased in large problems. To reduce the computation time, some researchers randomly select the next insertion or narrow the selection range in each iteration [261]. Three insertion selection tactics are proposed in ULIH, which are introduced below:

- *Greedy Tactic.* All unassigned tasks and insertion positions on all routes are evaluated, and the insertion which brings the lowest empty-load travel distance increase would be executed. This greedy tactic generates *tighter* routes but longer evaluation time is required.

- *First-Feasible Tactic.* In this case, all the tasks are sorted according to their closeness to their *deadlines*. Then, this tactic always inserts the first task in the unassigned task set (closest to deadline) into its first feasible position. This strategy can construct routes more quickly, but sacrificing the solution

quality as more trucks are required, and some mandatory tasks may not be assigned due to the lack of trucks. When K is small, this tactic should be used cautiously.

- *One-Route Tactic*. Considering all the unassigned tasks, the insertion with the lowest empty-load travel distance increase on the latest created route is executed with this tactic. This is a greedy insertion tactic on the newest route. It may cost more computation time than *First-Feasible* tactic, but produce tighter routes accordingly. This tactic trades off construction speed and solution quality.

5.2.2 Construction Scheme Selection

Completing all tasks on time with the limited vehicle resource is the primary job to logistic companies. In the Ningbo Port, the truck is relatively sufficient. But when the number of tasks is large, efficiently using the limited trucks becomes critical. With the three insertion selection tactics introduced above, nine different combinations of tactics are proposed and tested on the 100% benchmark instances to choose the recommended heuristics. Comparison results are presented in Table 5.3.

Table 5.3: Comparison of nine different combinations of insertion selection tactics.

Constructive Heuristic	Mandatory Task Tactic	Optional Task Tactic	Average HLDR	Average Time (s)
1	Greedy	Greedy	59.16%	3,576
2	Greedy	First-Feasible	56.40%	445
3	Greedy	One-Route	58.54%	710
4	First-Feasible	Greedy	58.60%	2,302
5	First-Feasible	First-Feasible	56.16%	705
6	First-Feasible	One-Route	57.99%	753
7	One-Route	Greedy	59.16%	3,402
8	One-Route	First-Feasible	56.64%	599
9	One-Route	One-Route	58.54%	620

Among the nine heuristics in Table 5.3, the three tactics are applied to mandatory tasks and optional tasks, respectively. The runtime is obtained on a PC with i7 CPU 3.2GHz and 6 GB memory. It can be found that, heuristics 1 and 7 obtain the best solution quality (the highest average HLDR), while the runtime are significantly higher than the other heuristics. Besides, when Greedy is used on mandatory tasks and First-Feasible is applied to optional tasks (heuristic 2), the fastest constructive speed is obtained, while the obtained average HLDR is not the worst. Applying First-Feasible to both mandatory and optional tasks (heuristic 5) does not bring the fastest heuristic.

The details of constructed solution quality is presented in Tables 5.4 and 5.5. It can be found that applying the Greedy heuristics to optional tasks (heuristics

Table 5.4: HLDR of initial solutions constructed with nine construction heuristics (on 100% real-life instances). Best obtained HLDR in bold.

Constructive Heuristic	1	2	3	4	5	6	7	8	9
NP4-1	60.14%	**64.10%**	58.64%	60.11%	61.69%	57.63%	60.14%	61.88%	58.64%
NP4-2	**58.42%**	54.13%	58.36%	**58.42%**	54.60%	58.36%	**58.42%**	54.60%	58.36%
NP4-3	**60.51%**	55.88%	59.22%	59.48%	54.73%	57.96%	**60.51%**	55.46%	59.22%
NP4-4	**56.37%**	50.86%	**56.37%**	**56.37%**	50.29%	**56.37%**	**56.37%**	50.29%	**56.37%**
NP4-5	**69.93%**	57.45%	69.68%	**69.93%**	59.98%	69.68%	**69.93%**	59.98%	69.68%
NP6-1	**62.28%**	54.90%	62.18%	62.03%	56.64%	62.06%	**62.28%**	56.74%	62.18%
NP6-2	60.10%	54.94%	**60.18%**	60.06%	54.55%	60.14%	60.10%	54.58%	**60.18%**
NP6-3	**49.66%**	46.79%	49.60%	**49.66%**	47.17%	49.60%	**49.66%**	47.17%	49.60%
NP6-4	**66.99%**	59.16%	63.20%	64.59%	56.92%	62.78%	**66.99%**	57.52%	63.20%
NP6-5	**56.90%**	54.82%	56.48%	56.39%	55.58%	55.52%	**56.90%**	56.22%	56.48%
NP8-1	64.05%	**64.56%**	63.19%	61.07%	62.08%	60.99%	64.05%	64.38%	63.19%
NP8-2	**64.83%**	60.59%	64.74%	63.42%	59.10%	63.21%	**64.83%**	61.01%	64.74%
NP8-3	68.56%	58.45%	68.22%	**68.98%**	58.39%	68.63%	68.56%	58.22%	68.22%
NP8-4	**57.27%**	55.53%	55.74%	57.16%	**57.27%**	55.88%	**57.27%**	56.83%	55.74%
NP8-5	55.42%	55.57%	55.35%	54.88%	54.96%	54.90%	55.42%	**55.58%**	55.35%

Table 5.5: HLDR of initial solutions constructed with nine construction heuristics (on 100% artificial instances). Best obtained HLDR in bold.

Constructive Heuristic	1	2	3	4	5	6	7	8	9
LB4-1	**59.79%**	54.41%	58.76%	59.42%	53.78%	58.75%	**59.79%**	54.39%	58.76%
LB4-2	**67.30%**	65.07%	66.29%	67.28%	66.92%	66.83%	**67.30%**	65.93%	66.29%
TB4-3	63.14%	63.06%	60.57%	**63.32%**	62.13%	58.95%	63.14%	63.06%	60.57%
TB4-4	**58.19%**	57.67%	57.95%	57.89%	57.55%	57.74%	**58.19%**	57.84%	57.95%
LU4-5	53.84%	52.66%	**53.89%**	51.88%	51.36%	52.68%	53.84%	52.69%	**53.89%**
LU4-6	**61.35%**	59.67%	60.69%	53.79%	52.95%	53.57%	**61.35%**	60.66%	60.69%
TU4-7	48.12%	**48.14%**	47.93%	48.10%	48.11%	47.66%	48.12%	**48.14%**	47.93%
TU4-8	**53.20%**	**53.20%**	52.97%	52.79%	52.66%	52.49%	**53.20%**	**53.20%**	52.97%
LB8-1	**58.32%**	54.64%	**58.32%**	**58.32%**	54.81%	**58.32%**	**58.32%**	54.81%	**58.32%**
LB8-2	65.88%	61.10%	**65.88%**	**65.88%**	61.08%	**65.88%**	65.88%	61.08%	**65.88%**
TB8-3	**59.93%**	59.67%	59.69%	58.74%	58.35%	58.43%	**59.93%**	59.67%	59.69%
TB8-4	54.73%	54.48%	53.59%	54.83%	**55.42%**	54.11%	54.73%	55.07%	53.59%
LU8-5	59.67%	59.70%	57.81%	**62.90%**	62.89%	60.59%	59.67%	59.70%	57.81%
LU8-6	49.63%	49.39%	48.87%	50.77%	**51.00%**	48.66%	49.63%	50.10%	48.87%
TU8-7	56.64%	56.28%	**57.65%**	56.59%	56.29%	57.63%	56.64%	56.28%	**57.65%**
TU8-8	**52.65%**	**52.65%**	**52.65%**	51.70%	51.70%	51.56%	**52.65%**	**52.65%**	**52.65%**

1, 4 and 7) obtains significantly better objective value than those of First-Feasible (heuristics 2, 5 and 8). The HLDR of tactic One-Route group (heuristics 3, 6 and 9) is between the above-mentioned two groups. However, to mandatory tasks, no tactic shows obvious better performance than the others. It can be concluded that, the tactic used on optional tasks is the key issue in devising a constructive heuristic. This is because, in practice, optional tasks always account for a major proportion of total unassigned tasks in one shift, and also mandatory tasks have less available insertion options due to their closeness to their deadlines. Therefore, the tactic for optional tasks has a greater influence on solution construction.

These constructive heuristics have not shown bias on specific features of instances. Overall, the obtained HLDR values span from 46.79% to 69.93%, higher than 50% on most instances. It should be noted that, two pairs of heuristics (1, 7) and (3, 9) have constructed the same solutions while heuristics 7 and 9 spend less computation time. It is because of that, when the candidate tasks are few, if the existing routes are too tight to insert other tasks, Greedy tactic can only assign a

task to the newest route. The same thing occurs to tactic One-Route in the same case, but its computation time is less since it only evaluates one route. As there are few mandatory tasks which are always assigned first in each shift and no randomisation in this heuristic, applying tactic Greedy or One-Route to mandatory tasks may generate the same partial routes. In this case, if the tactics applied to optional tasks are the same, their final routes (solutions) generated would be the same as well.

In metaheuristics, the quality of initial solutions does not always directly determine the quality of final improved solutions. This study also finds that better initial solutions cannot guarantee better final solutions in VNS-RLS. Considering all the nine heuristics can produce feasible solutions on the benchmark instances, in normal cases, the construction heuristic 2 is recommended, as it has in general the fastest construction speed.

Table 5.6: Quality of solutions on large instances.

Constructive Heuristic	HLDR	Time (hr)	Additional Trucks Requirement	No. of Failed Tasks
1	73.38%	35.7	0	56
2	65.49%	23.1	116	98
3	73.24%	30.0	4	59
4	73.18%	30.3	0	67
5	65.01%	22.1	100	118
6	73.31%	22.8	1	57
7	73.38%	28.5	0	56
8	65.00%	22.7	98	118
9	73.24%	23.2	4	59

The two main constraints of PVRPTW-O are the time constraints and the limited trucks. To evaluate and identify effective heuristics in tight constrained scenario, the performances of the nine constructive heuristics is compared on the largest instance of 2,614 tasks, results presented in Table 5.6. It can be found that, when tactic First-Feasible is applied to optional tasks (heuristics 2, 5 and 8), the quality of solutions is poor (lower HLDR and more violations). Heuristics 1 and 7 produce the same best solutions, while heuristic 7 takes less computation time. Heuristic 7 is thus recommended for this kind of very large instances. Other solutions with acceptable quality are generated when tactic Greedy or One-Route is applied to optional tasks (heuristics 3, 4, 6 and 9).

5.3 A Variable Neighbourhood Search Algorithm with Reinforcement Learning

An improved VNS algorithm for the PVRPTW-O is introduced in this section. Reinforcement Learning is applied to the Adaptive Operator Selection in the proposed algorithm. In metaheuristics, Sampling can be used to measure the fitness

landscape surrounding a solution, providing guidance for the search. It shows powerful performance in Evolutionary Computation [159]. A sampling scheme is also implemented in this study to investigate its impact on single solution-based metaheuristic methods.

5.3.1 Algorithmic Details

5.3.1.1 Algorithm Framework

The framework of the VNS with Reinforcement Learning and Sampling (VNS-RLS) is presented in this subsection, see Algorithm 3. After inputting the instance data and parameters, an initial feasible solution S is constructed with ULIH in ***Step 2***. Then, at each VNS iteration, *Shaking* generates a new feasible solution S' as the starting solution by perturbing S. In ***Step 3.2***, with S', *Sampling* initializes a set of weights (WS) for all the operators in the neighbourhood operator set (NS). With the invoked probability distribution of operators, which is generated according to WS, *Local Search* explores better solutions iteratively (***Step 3.3***). These steps are repeated until the objective value has not been improved by 0.01% after a predefined number of iterations.

Algorithm 3 The VNS-RLS framework

```
Step 1: Input the problem instance data.
Step 2: Produce an initial feasible solution S with the ULIH.        // Section 5.2.
Step 3:
    while Stopping Criterion is not met do
        Step 3.1: S' ← Shaking(S).                                   // Section 5.3.1.2.
        Step 3.2: WS ← Sampling(S', NS).                             // Section 5.3.1.3.
        Step 3.3: S ← Local Search(S, S', WS).                       // Section 5.3.1.4.
    end while
Step 4: Output S.
```

Eight improved neighbourhood operators, which require relatively less computation time, are adopted in VNS-RLS. These operators are devised considering the specific solution structure of the PVRPTW-O. The solutions of classic VRPs are 2-dimensional. However, as mentioned before, the solution of PVRPTW-O has one more dimension of shift. Previous research has shown that properly using operators with diverse degrees of perturbation can significantly improve search performance [59]. To systematically increase the diversification of the search in this tightly constrained problem, we specify the level of perturbation for each operator. The eight improved operators work at different levels (intra-route, intra-shift and inter-shift) with diverse levels of perturbations, details explained in Table 5.7. Considering the time constraints in PVRPTW-O, the directions of the operated

strings of tasks in the routes are kept in all the eight neighbourhood structures to reduce the constraint violation.

Table 5.7: Operations of the eight operators within *NS* in VNS-RLS.

Operator	Description
Intra-Shift 2-opt*	Execute 2-opt* exchange, where the two chosen routes are randomly selected from the same shift.
Inter-Shift 2-opt*	Execute 2-opt* exchange, where the two chosen routes are randomly selected from different shifts.
Intra-Route Or-opt	A string of task nodes is repositioned in the original route .
Intra-Shift Or-opt	A string of task nodes is repositioned from one route to another route in the same shift.
Inter-Shift Or-opt	A string of task nodes is repositioned from one route to another route in a different shift.
Intra-Route CROSS	Swap two strings of task nodes in the same route.
Intra-Shift CROSS	Swap two strings of task nodes from different routes in the same shift.
Inter-Shift CROSS	Swap two strings of task nodes from two routes which are from different shifts.

5.3.1.2 Shaking

Shaking in VNS algorithms is often used to escape from the local optimum and diversify the search. The commonly used Shaking schemes include: randomly invoking neighbourhood operators, designing specific Shaking Operators and so on. To jump to farther areas from the current search region, in VNS-RLS, four neighbourhood operators which bring larger changes are employed in Shaking. Inter-Shift 2-opt*, Intra-Shift 2-opt*, Inter-Shift Or-opt and Inter-Shift CROSS are sequentially used to generate the starting solution of *Local Search*. Each operator is executed once, and the solution generated is passed to the following operator as the seed solution. Note that the new solution generated will be accepted in Shaking as long as it is feasible, even if its quality is worse. The aim of Shaking is to increase diversification and avoid search cycle, rather than myopic solution improvement.

5.3.1.3 Sampling of Neighbourhood

Sampling of surrounding environment can provide guidance for the search and has improved the search performance in many population-based algorithms. The impact of sampling on the search trajectory of single solution-based algorithms is investigated in this study. A sampling scheme is applied before the *Local Search* phase in VNS-RLS.

Based on the perturbed solution S' generated by *Shaking*, every candidate neighbourhood operator NS_i is sampled, producing d sampling solutions ($S^s_{ij} \in S^s_i, j \in \{1, ..., d\}$). Preliminary experiments show that, with more samples, the

output results would be more stable but the number of iterations executed greatly increases, while the solution quality is without obvious further improvement. In light of the previous trials, each operator is sampled 50 times ($d = 50$) to balance the evaluation time and the metric approximation. The obtained sampling solutions are then measured, while the feedback will be used to generate the initial weights of operators *WS*, providing an initial search direction to *Local Search*.

Three scalars (eqs. 5.11 - 5.13) are defined to measure the sampling solutions. To operator NS_i, the first scalar E_i^a indicates the probability of finding a solution which is not worse than the current solution. E_i^b presents the average solution quality of the solutions generated. ρ is a parameter for balancing the impact of E_i^a and E_i^b, which is set to 10 empirically. In the preliminary experiments, it is observed that more than half of the neighbourhood moves produce infeasible solutions. To reduce the infeasible moves in the search, a third indicator E_i^c is proposed in VNS-RLS, to indicate the probability of obtaining a feasible solution with operator NS_i.

$$E_i^a = \frac{\{|S_i^s| \mid TD(S_{ij}^s) \leq TD(S')\}}{d} \tag{5.11}$$

$$E_i^b = \rho \cdot \left(\frac{TD(S)}{\sum_{j=1}^{d} TD(S_{ij}^s) \ / \ d} - 1 \right) \tag{5.12}$$

$$E_i^c = \frac{\{|S_i^s| \mid S_{ij}^s \ is \ feasible\}}{d} \tag{5.13}$$

For each operator NS_i, the overall evaluation value (E_i) is a combination of the above three metrics. Based on the preliminary experiment results, the weights of indicators are set as $\alpha = 0.5$, $\beta = 0.2$ and $\gamma = 0.3$.

$$E_i = \alpha \cdot E_i^a + \beta \cdot E_i^b + \gamma \cdot E_i^c \qquad (s.t.\ \alpha + \beta + \gamma = 1) \tag{5.14}$$

The weight (W_i) determines the probability of a neighbourhood operator NS_i being invoked in the Local Search, stored in *WS*. To each operator, the initial W_i is calculated according to the feedback of Sampling (E_i), see eq. (5.15). The sum of the initial W_i of the P operators is 1. Let E_{min} be the minimal E_i among all operators, an operator with larger E_i would have a larger initial W_i. To avoid some of the weights being too small and the associated operators never being invoked, a minimal initial weight W_{min} is adopted, which is empirically set to 5% in VNS-RLS.

$$W_i = W_{min} + (1 - P \cdot W_{min})(E_i - E_{min}) / \sum_{i=1}^{P} (E_i - E_{min}) \tag{5.15}$$

5.3.1.4 Local Search with Reinforcement Learning

In VNS-RLS, the *Local Search* is an iterative search with reinforcement learning, see Algorithm 4. Firstly, the initial invoked probabilities are distributed to operators according to W_i. Using the *Roulette Wheel scheme* as the selection rule, in ***Step 3.2***, one neighbourhood operator is selected and applied once to the current solution (S'), generating a new solution (S''). In *Move or Not*, S'' is evaluated to decide whether it is accepted as the new current solution. W_i is then updated according to the evaluation feedback. These steps are repeated until no improvement is obtained after a predefined number (L_{max}) of evaluations. Based on the preliminary experiments, L_{max} is set to 150.

Algorithm 4 Local Search(S, S',WS)

Step 1: Input the current best solution S, starting solution S' and the initial weight set $WS = \{W_1, W_2 ... W_P\}$.
Step 2: Set $L \leftarrow 0$.
Step 3:
while $(L < L_{max})$ **do**
Step 3.1: Generate the invoked probability of each operator based on its weight (W_i).

$Pr_i \leftarrow W_i / \sum_{i=1}^{P} W_i$
Step 3.2: Neighbourhood Search
Execute a neighbourhood operator NS_i selected with *Roulette Wheel Scheme* based on Pr_i, generating a new solution (S''): $S'' \leftarrow NS_i(S')$, $L \leftarrow L + 1$.
Step 3.3: Move or Not
if S'' is infeasible **then** // Feasibility Indicator: FI
$W_i \leftarrow W_i * 0.9$
else if $Evaluation(S'') < Evaluation(S)$ **then** // Quality Indicator: TD (3.1)
$S \leftarrow S'', S' \leftarrow S'', L \leftarrow 0.$
$W_i \leftarrow W_i * 1.1$
else if $Evaluation(S'') - Evaluation(S') < DEVIATION$ **then** // RRT
$S' \leftarrow S''.$
else
$W_i \leftarrow W_i * 0.9$
end if
end while
Step 4: Output the best found solution S.

In the *Move or Not* phase, the weight of NS_i is updated during the search. In the literature, most learning indicators focus on the solution quality, while the feasibility of search attracts less attention. When updating the weight, a feasibility indicator (FI) is employed in VNS-RLS, which is just the feasibility of the newly generated solution. To reduce the computing time in calculating the updated weights, a simple Credit Assignment Mechanism is employed. If S'' is infeasible or cannot be accepted due to its low solution quality, a penalty of 10% reduction of weight will be applied to NS_i. Otherwise, if the quality of S'' is better

than the current best solution S, the weight will be increased 10% as a reward. The penalty and reward rates are set based on the preliminary experiments. The feasible solutions with worse quality are accepted with a Record-to-Record Travel (RRT) scheme. In VNS-RLS, *DEVIATION* of the RRT scheme is set to 2 on *TD* empirically. In practice, less than 2 km increase in total travel distance is acceptable.

The contribution of the feasibility indicator and different components in VNS-RLS are analysed and discussed, and the results presented in subsection 5.3.2.2.

5.3.2 Experiment Results and Analysis

5.3.2.1 Impact of the Sampling

To evaluate the contribution of *Sampling* introduced in subsection 5.3.1.3, a variant of VNS-RLS without the *Sampling* scheme and with equal initial weights for all candidate operators is implemented. This variant of VNS with Reinforcement Learning (VNS-RL) is tested on benchmark instances of different sizes. The results of 30 runs on 25% scaled down instances are presented in Tables 5.8 and 5.9. It can be summarised that VNS-RLS performs better than VNS-RL in both the best found and average HLDR on 19/31 of all instances, but more evaluations were conducted. On 26/31 instances, both VNS-RLS and VNS-RL obtain better results (higher HLDR) than CPLEX with previously mentioned configuration, obtaining feasible solutions on all instances while CPLEX fails on 13 instances.

Results of large 100% instances are presented in Tables 5.10 and 5.11. In 16/31 instances, solutions with better best-found and average HLDR are found by VNS-RLS. In [16], the HLDR for the relaxed problem removing the unloaded travels from and to the depot are calculated as the upper bounds of the benchmark. Since PVRPTW-O can be relaxed to the same form, those upper bounds are also adopted in this study. It can be found that, the gaps between the obtained solutions and upper bounds are less than 5% on 14 instances. Especially, in the instances of NP4-2, NP4-5, NP6-4, NP8-4 and TU8-8, the objective values of solutions generated are very close to the upper bounds.

However, the contribution of *Sampling* on improving the objective value is not significant. The t-test results on VNS-RLS and VNS-RL (see Tables 5.12 and 5.13) show that the improvement is significant on only 25% instances. This observation indicates that, by sampling on the search trajectory of single solution-based algorithm, the information collected may not sufficiently reflect the neighbourhood environment or provide valuable guidance for the search. In other words, limited *Sampling* points in single solution-based approaches is not so helpful in improving

Table 5.8: Comparison on 25% scaled down artificial instances. **Eval**: average evaluation times; **S.D.**: standard deviation of obtained HLDR. Best heuristic solutions are in bold.

Instance		LB4-1	LB4-2	TB4-3	TB4-4	LU4-5	LU4-6	TU4-7	TU4-8
VNS-RLS	Best	76.68%	82.67%	69.28%	66.04%	60.16%	60.92%	**48.75%**	**54.97%**
	Average	74.35%	81.54%	**68.05%**	64.70%	59.09%	60.64%	**48.56%**	54.52%
	Eval	277,040	315,961	273,910	272,605	295,389	278,076	167,317	164,856
	S.D.	0.88%	0.83%	0.75%	0.65%	0.52%	0.22%	0.30%	0.31%
VNS-RL	Best	75.82%	83.08%	68.64%	66.33%	60.31%	60.91%	**48.75%**	**54.97%**
	Average	74.38%	81.33%	67.51%	64.83%	**59.35%**	60.43%	48.43%	54.61%
	Eval	134,970	129,368	117,319	133,301	141,104	117,165	75,732	81,194
	S.D.	0.90%	0.91%	0.86%	0.72%	0.48%	0.33%	0.32%	0.29%
VNS-RLS No Ec	Best	**76.92%**	**83.42%**	69.08%	66.41%	**60.71%**	**61.08%**	**48.75%**	**54.97%**
	Average	74.80%	**81.61%**	67.78%	**64.95%**	59.29%	60.62%	48.54%	**54.68%**
	Eval	313,707	280,849	286.059	298,651	321,835	290,082	166,248	193,536
	S.D.	0.95%	1.09%	0.65%	0.75%	0.64%	0.29%	0.30%	0.33%
VNS-RLS No FI	Best	75.59%	**83.42%**	**69.37%**	**66.45%**	60.29%	60.96%	**48.75%**	**54.97%**
	Average	74.40%	81.58%	67.75%	64.79%	59.17%	**60.69%**	48.55%	54.54%
	Eval	261,407	333,130	292,332	320,959	297,749	310,406	171,264	185,505
	S.D.	0.57%	1.05%	0.85%	0.77%	0.67%	0.20%	0.31%	0.34%
VNS-RL No FI	Best	**76.92%**	83.08%	68.78%	66.00%	59.74%	60.87%	**48.75%**	**54.97%**
	Average	**74.85%**	81.58%	67.52%	64.78%	59.27%	60.56%	48.53%	54.55%
	Eval	107,783	132,351	132,658	142,240	164,529	131,225	75,934	76,202
	S.D.	1.05%	1.12%	0.76%	0.84%	0.38%	0.24%	0.30%	0.17%
CPLEX		66.62%	76.41%	69.91%	69.30%	NF	58.65%	50.37%	55.36%
Upper Bound		100%	94.87%	86.31%	83.51%	79.94%	73.90%	52.17%	66.38%

Instance		LB8-1	LB8-2	TB8-3	TB8-4	LU8-5	LU8-6	TU8-7	TU8-8
VNS-RLS	Best	91.28%	93.60%	62.87%	65.28%	**65.76%**	66.37%	**56.72%**	**52.34%**
	Average	89.58%	91.90%	**61.81%**	62.98%	64.78%	65.39%	55.98%	**51.97%**
	Eval	538,438	508,352	280,373	448,218	465,806	430,572	319,333	265,267
	S.D.	0.99%	1.07%	0.59%	1.17%	0.54%	0.51%	0.37%	0.21%
VNS-RL	Best	91.25%	93.11%	62.49%	64.26%	65.61%	66.05%	56.54%	52.12%
	Average	88.84%	91.20%	61.74%	62.54%	64.37%	65.29%	55.90%	51.93%
	Eval	234,779	217,003	129,619	180,111	187,423	179,388	123,896	124,926
	S.D.	1.42%	1.13%	0.51%	0.90%	0.71%	0.37%	0.35%	0.13%
VNS-RLS No Ec	Best	91.25%	93.56%	**63.05%**	**66.31%**	**65.76%**	**66.58%**	56.46%	52.29%
	Average	**89.76%**	92.09%	61.78%	**63.25%**	**64.86%**	65.58%	55.79%	51.93%
	Eval	492,628	547,853	296,837	517,855	438,295	439,782	269,164	281,479
	S.D.	0.95%	0.87%	0.54%	1.16%	0.44%	0.49%	0.29%	0.18%
VNS-RLS No FI	Best	**91.94%**	**93.70%**	62.62%	65.82%	65.68%	66.03%	56.63%	52.16%
	Average	89.6%	91.88%	61.79%	62.96%	64.79%	65.46%	55.95%	51.94%
	Eval	463,923	473,201	287,761	425,639	410,844	363,277	305,785	259,536
	S.D.	1.19%	1.08%	0.51%	0.99%	0.41%	0.41%	0.33%	0.15%
VNS-RL No FI	Best	90.54%	93.62%	62.62%	63.41%	65.33%	66.37%	56.46%	52.11%
	Average	89.27%	**92.34%**	61.64%	62.49%	64.42%	**65.71%**	**56.01%**	51.94%
	Eval	189,883	257,521	134,118	184,205	195,563	200,338	155,815	110,828
	S.D.	0.67%	0.75%	0.78%	0.75%	0.51%	0.35%	0.37%	0.13%
CPLEX		NF	NF	56.85%	52.40%	57.42%	NF	47.65%	50.74%
Upper Bound		100%	100%	82.33%	88.75%	78.33%	86.84%	71.59%	70.43%

Table 5.9: Comparison on 25% scaled down real-life instances.

Instance		NP4-1	NP4-2	NP4-3	NP4-4	NP4-5
VNS-RLS	Best	**82.89%**	62.17%	75.78%	60.50%	79.45%
	Average	80.88%	61.19%	75.00%	59.28%	78.62%
	Eval	387,006	285,553	298,205	292,664	412,737
	S.D.	1.05%	0.53%	0.64%	0.65%	0.41%
VNS-RL	Best	**82.89%**	62.42%	75.74%	60.52%	79.14%
	Average	80.29%	61.28%	74.88%	59.34%	78.56%
	Eval	169,470	134,063	149,373	134,128	199,220
	S.D.	1.34%	0.60%	0.60%	0.56%	0.37%
VNS-RLS No Ec	Best	**82.89%**	62.32%	75.64%	59.76%	79.24%
	Average	81.51%	61.42%	74.92%	59.18%	78.48%
	Eval	469,233	311,885	319,202	347,134	326,956
	S.D.	1.16%	0.60%	0.62%	0.35%	0.42%
VNS-RLS No FI	Best	82.85%	**62.91%**	75.98%	**61.12%**	**79.61%**
	Average	**80.89%**	**61.59%**	74.93%	**59.42%**	**78.71%**
	Eval	442,311	320,768	306,145	310,331	425,826
	S.D.	0.81%	0.53%	0.63%	0.75%	0.43%
VNS-RL No FI	Best	81.50%	61.95%	**76.11%**	59.88%	79.18%
	Average	79.70%	61.16%	**75.19%**	59.27%	78.41%
	Eval	153,843	145,666	171,405	154,865	153,056
	S.D.	1.22%	0.42%	0.69%	0.43%	0.48%
CPLEX		78.36%	65.14%	64.83%	54.39%	NF
Upper Bound		92.36%	97.04%	100%	97.72%	100%

Instance		NP6-1	NP6-2	NP6-3	NP6-4	NP6-5
VNS-RLS	Best	76.88%	73.23%	62.48%	**80.56%**	82.56%
	Average	74.85%	72.63%	61.99%	79.70%	80.60%
	Eval	615,170	537,578	246,889	564,762	446,762
	S.D.	0.93%	0.39%	%0.29	0.51%	1.39%
VNS-RL	Best	76.31%	73.09%	**62.77%**	80.41%	**83.59%**
	Average	75.01%	72.50%	61.86%	79.35%	81.01%
	Eval	305,856	232,937	128,263	267,918	216,844
	S.D.	0.86%	0.32%	0.37%	0.78%	1.44%
VNS-RLS No Ec	Best	76.24%	**73.39%**	62.32%	80.50%	82.44%
	Average	74.99%	**72.83%**	**62.06%**	**79.84%**	80.53%
	Eval	698,514	624,078	253,037	541,548	365,435
	S.D.	0.96%	0.41%	0.20%	0.41%	1.72%
VNS-RLS No FI	Best	**77.61%**	73.35%	**62.77%**	80.41%	82.79%
	Average	**75.52%**	72.70%	61.91%	79.64%	**81.78%**
	Eval	653,283	485,454	274,900	497,273	445,694
	S.D.	1.21%	0.37%	0.30%	0.67%	0.92%
VNS-RL No FI	Best	76.00%	73.13%	62.58%	**80.56%**	83.06%
	Average	74.87%	72.61%	61.99%	79.33%	81.74%
	Eval	290,428	232,629	107,558	270,913	196,576
	S.D.	1.10%	0.47%	0.31%	0.69%	0.68%
CPLEX		NF	NF	54.30%	NF	66.11%
Upper Bound		NF	NF	95.20%	NF	98.39%

Instance		NP8-1	NP8-2	NP8-3	NP8-4	NP8-5
VNS-RLS	Best	**77.05%**	77.91%	75.66%	62.26%	72.27%
	Average	**75.35%**	77.00%	75.05%	62.08%	71.56%
	Eval	735,909	481,493	388,220	415,835	540,940
	S.D.	0.83%	0.50%	0.34%	0.09%	0.47%
VNS-RL	Best	76.33%	**78.27%**	75.69%	62.25%	72.27%
	Average	74.54%	76.97%	75.03%	62.08%	71.25%
	Eval	323,590	234,179	193,351	174,499	228,491
	S.D.	0.96%	0.47%	0.29%	0.11%	0.52%
VNS-RLS No Ec	Best	76.91%	77.76%	75.35%	60.90%	72.27%
	Average	74.72%	77.16%	74.93%	60.47%	**71.68%**
	Eval	607,961	525,479	442,103	430,962	516,872
	S.D.	1.20%	0.37%	0.31%	0.32%	0.36%
VNS-RLS No FI	Best	76.88%	78.08%	75.72%	**62.26%**	72.25%
	Average	74.84%	**77.28%**	74.86%	**62.11%**	71.54%
	Eval	656,798	588,599	352,809	354,340	541,743
	S.D.	1.06%	0.37%	0.43%	0.13%	0.40%
VNS-RL No FI	Best	76.77%	77.66%	**75.82%**	62.21%	**72.31%**
	Average	75.15%	76.88%	**75.19%**	62.06%	71.40%
	Eval	323,912	211,260	213,402	186,282	197,478
	S.D.	0.74%	0.49%	0.34%	0.15%	0.46%
CPLEX		NF	NF	NF	NF	NF
Upper Bound		98.98%	100%	100%	NF	100%

Table 5.10: Comparison on 100% artificial instances.

Instance		LB4-1	LB4-2	TB4-3	TB4-4	LU4-5	LU4-6	TU4-7	TU4-8
VNS-RLS	Best	73.57%	78.02%	69.52%	**72.91%**	64.64%	67.89%	53.07%	53.78%
	Average	72.79%	77.52%	68.78%	**72.09%**	64.22%	67.50%	52.90%	53.58%
	Eval	645,863	612,594	666,776	738,772	677,499	816,750	316,288	234,027
	S.D.	0.51%	0.37%	0.53%	0.51%	0.24%	0.26%	0.19%	0.09%
VNS-RL	Best	73.46%	78.60%	**69.90%**	72.19%	64.65%	67.92%	53.09%	53.77%
	Average	72.59%	77.25%	68.61%	71.27%	64.21%	67.54%	52.90%	53.56%
	Eval	295,938	346,393	335,523	319,619	335,523	358,255	191,666	141,046
	S.D.	0.48%	0.68%	0.69%	0.48%	0.27%	0.44%	0.22%	0.10%
VNS-RLS No Ec	Best	73.52%	78.08%	69.32%	72.24%	64.67%	**68.12%**	**53.21%**	**53.80%**
	Average	72.93%	77.70%	68.54%	71.42%	**64.38%**	67.52%	**53.03%**	**53.61%**
	Eval	642,796	617,656	616,237	635,130	724,154	782,608	399,970	290,599
	S.D.	0.32%	0.32%	0.42%	0.49%	0.20%	0.40%	0.16%	0.08%
VNS-RLS No FI	Best	**73.76%**	**78.75%**	69.44%	72.52%	64.50%	68.08%	53.18%	53.68%
	Average	**72.99%**	**77.79%**	**68.84%**	71.95%	64.01%	**67.71%**	52.79%	53.53%
	Eval	702,902	691,981	878,641	883,267	655,225	887,895	360,106	266,630
	S.D.	0.52%	0.47%	0.52%	0.40%	0.31%	0.37%	0.29%	0.08%
VNS-RL No FI	Best	73.51%	78.31%	69.36%	72.39%	**64.75%**	68.11%	53.12%	53.77%
	Average	72.80%	77.44%	68.26%	71.34%	64.25%	67.60%	52.78%	53.57%
	Eval	276,675	316,636	316,561	326.226	342,380	341,844	202,185	143,319
	S.D.	0.44%	0.41%	0.66%	0.87%	0.36%	0.26%	0.34%	0.14%
Upper Bound		79.47%	86.33%	84.05%	88.74%	74.11%	74.47%	64.05%	63.50%

Instance		LB8-1	LB8-2	TB8-3	TB8-4	LU8-5	LU8-6	TU8-7	TU8-8
VNS-RLS	Best	85.86%	**94.94%**	69.41%	66.08%	67.95%	68.40%	**59.72%**	54.36%
	Average	83.48%	**93.21%**	69.01%	65.24%	67.24%	67.87%	59.31%	54.23%
	Eval	1,481,728	1,362,552	619,999	1,422,110	990,621	888,131	586,452	887,790
	S.D.	1.46%	0.82%	0.29%	0.81%	0.52%	0.29%	0.28%	0.12%
VNS-RL	Best	85.16%	93.19%	69.15%	**67.12%**	68.00%	68.32%	59.64%	54.24%
	Average	82.74%	91.96%	68.73%	65.21%	66.96%	67.84%	59.29%	53.99%
	Eval	687,880	583,113	309,823	665,151	523,243	411,686	324,946	348,373
	S.D.	1.41%	0.85%	0.33%	1.29%	0.51%	0.40%	0.19%	0.17%
VNS-RLS No Ec	Best	85.49%	94.03%	69.59%	66.85%	67.81%	68.41%	59.60%	**54.50%**
	Average	84.11%	92.83%	**69.04%**	65.70%	67.20%	68.07%	59.21%	54.23%
	Eval	1,443,635	1,134,773	669,136	1,478,978	1,114,876	1,026,285	572,065	859,770
	S.D.	0.95%	1.05%	0.38%	0.76%	0.34%	0.21%	0.21%	0.16%
VNS-RLS No FI	Best	**87.49%**	94.12%	**69.90%**	66.66%	**68.07%**	**68.65%**	59.36%	54.42%
	Average	**85.17%**	92.71%	68.90%	65.66%	**67.29%**	**68.25%**	59.11%	**54.27%**
	Eval	1,324,345	1,216,936	524,024	1,713,200	1,058,132	1,043,078	499,058	842,941
	S.D.	0.26%	0.79%	0.39%	0.82%	0.37%	0.29%	0.16%	0.12%
VNS-RL No FI	Best	85.90%	93.85%	69.46%	66.58%	67.32%	68.64%	59.52%	54.38%
	Average	84.25%	92.35%	68.92%	**65.82%**	66.61%	**68.25%**	**59.33%**	54.05%
	Eval	662,849	607,075	330,893	704,743	409,184	520,407	337,106	403,569
	S.D.	1.04%	0.82%	0.35%	0.51%	0.36%	0.30%	0.17%	0.14%
Upper Bound		98.26%	97.97%	87.06%	92.44%	74.27%	71.36%	70.29%	56.54%

objective value comparing to its impact in evolutionary approaches. The further discussion on the impact of Sampling indicators is presented in subsection 5.3.2.2.

The same observation is found on the medium size 50% scaled down instances (see Tables 5.14 and 5.15). VNS-RLS outperforms VNS-RL on 16 out of 31 instances and higher stability (smaller standard deviation) is obtained on 17 instances, which is not significant better. In this single solution-based algorithm, no significant solution quality improvement is achieved by adopting *Sampling*.

5.3.2.2 Feasibility Indicators & Feasible Solution Space Exploration

In *Sampling* and *Local Search*, feasibility measure is employed aiming to improve the search efficiency. The hypothesis is that, by using the feasibility indicators, the infeasible solutions explored during the search will decrease. To validate this hypothesis, the feasibility indicators in *Sampling* (E^c) and *Local Search* (FI) are

Table 5.11: Comparison on 100% real-life instances.

Instance		NP4-1	NP4-2	NP4-3	NP4-4	NP4-5
VNS-RLS	Best	82.99%	69.78%	73.13%	66.76%	**80.82%**
	Average	**81.88%**	69.33%	72.11%	**66.06%**	80.37%
	Eval	612,487	383,815	518,193	811,218	487,919
	S.D.	0.53%	0.19%	0.53%	0.42%	0.16%
VNS-RL	Best	81.87%	69.72%	72.85%	66.51%	80.74%
	Average	80.90%	69.38%	**72.44%**	65.76%	80.35%
	Eval	358,785	260,903	371,269	497,124	354,045
	S.D.	0.80%	0.26%	0.28%	0.49%	0.23%
VNS-RLS No Ec	Best	**83.29%**	**69.85%**	72.90%	66.61%	80.65%
	Average	**81.88%**	**69.56%**	72.20%	65.91%	80.48%
	Eval	779,504	575,894	661.384	923,891	718,219
	S.D.	0.55%	0.16%	0.38%	0.47%	0.17%
VNS-RLS No FI	Best	82.89%	69.72%	**73.54%**	**66.78%**	80.78%
	Average	81.69%	69.51%	72.41%	65.99%	**80.55%**
	Eval	818,229	436,507	590,170	955,539	737,512
	S.D.	0.64%	0.16%	0.49%	0.48%	0.20%
VNS-RL No FI	Best	81.89%	69.67%	73.04%	66.27%	80.75%
	Average	81.08%	69.42%	72.10%	65.66%	80.49%
	Eval	446,173	245,862	308,416	438,464	411,071
	S.D.	0.51%	0.19%	0.65%	0.50%	0.20%
Upper Bound		90.43%	70.23%	79.58%	73.72%	81.20%

Instance		NP6-1	NP6-2	NP6-3	NP6-4	NP6-5
VNS-RLS	Best	79.60%	74.10%	58.86%	**80.19%**	**80.15%**
	Average	78.96%	73.77%	58.39%	79.29%	78.44%
	Eval	828,469	913,906	412,097	1,098,792	928,097
	S.D.	0.43%	0.23%	0.21%	0.49%	0.69%
VNS-RL	Best	79.54%	**74.23%**	58.95%	79.86%	80.09%
	Average	78.92%	73.59%	**58.81%**	79.32%	78.50%
	Eval	546,222	666,544	301,455	654,549	442,473
	S.D.	0.42%	0.30%	0.10%	0.47%	1.02%
VNS-RLS No Ec	Best	79.64%	74.14%	58.94%	79.52%	79.99%
	Average	79.07%	73.72%	58.62%	79.10%	78.36%
	Eval	1,030,825	1,163,006	513,974	1,053,682	984,987
	S.D.	0.47%	0.21%	0.23%	0.53%	0.99%
VNS-RLS No FI	Best	**79.69%**	74.10%	59.00%	80.01%	80.11%
	Average	**79.31%**	73.74%	58.62%	**79.45%**	79.34%
	Eval	1,163,262	1,015,544	621,863	1,095,930	1,162,651
	S.D.	0.28%	0.29%	0.20%	0.26%	0.49%
VNS-RL No FI	Best	79.64%	74.20%	**59.11%**	79.84%	79.67%
	Average	79.20%	**73.80%**	58.79%	79.23%	**78.94%**
	Eval	508,061	568,328	317,112	599,814	466,904
	S.D.	0.26%	0.24%	0.20%	0.44%	0.52%
Upper Bound		83.93%	76.67%	66.90%	80.97%	84.30%

Instance		NP8-1	NP8-2	NP8-3	NP8-4	NP8-5
VNS-RLS	Best	73.69%	75.09%	74.31%	61.94%	73.28%
	Average	73.10%	74.52%	73.77%	61.85%	72.84%
	Eval	1,091,477	917,666	857,388	636,706	1,031,595
	S.D.	0.26%	0.32%	0.40%	0.05%	0.21%
VNS-RL	Best	73.28%	75.25%	74.43%	62.00%	73.60%
	Average	72.87%	74.69%	74.08%	61.91%	73.00%
	Eval	567,895	460,155	514,308	398,169	487,284
	S.D.	0.29%	0.23%	0.23%	0.07%	0.32%
VNS-RLS No Ec	Best	**73.80%**	75.27%	74.20%	61.97%	73.62%
	Average	**73.48%**	74.86%	73.96%	61.91%	73.26%
	Eval	1,498,392	978,695	867,663	693,779	1,189,550
	S.D.	0.15%	0.28%	0.22%	0.06%	0.35%
VNS-RLS No FI	Best	73.70%	75.44%	74.60%	62.01%	**73.86%**
	Average	73.32%	**74.90%**	**74.13%**	61.90%	**73.42%**
	Eval	1,324,345	1,133,670	974,108	578,979	1,033,072
	S.D.	0.26%	0.31%	0.24%	0.06%	0.37%
VNS-RL No FI	Best	73.77%	**75.67%**	**74.65%**	**62.05%**	73.50%
	Average	73.12%	74.80%	73.94%	**61.95%**	73.14%
	Eval	645,318	497,384	534,775	406,094	468,859
	S.D.	0.39%	0.48%	0.37%	0.06%	0.30%
Upper Bound		77.04%	77.55%	78.82%	62.53%	76.09%

Table 5.12: T-test between VNS-RLS and VNS-RL on 25% scaled down and 100% real-life instances. (Y means significantly different, while N means not.)

	NP4-1		NP4-2		NP4-3		NP4-4		NP4-5	
Instance Scale	25%	100%	25%	100%	25%	100%	25%	100%	25%	100%
T-test result	N	Y	N	N	N	Y	N	N	N	N
	NP6-1		NP6-2		NP6-3		NP6-4		NP6-5	
Instance Scale	25%	100%	25%	100%	25%	100%	25%	100%	25%	100%
T-test result	N	N	N	N	N	Y	Y	N	N	N
	NP8-1		NP8-2		NP8-3		NP8-4		NP8-5	
Instance Scale	25%	100%	25%	100%	25%	100%	25%	100%	25%	100%
T-test result	Y	Y	N	N	N	N	N	N	Y	N

Table 5.13: T-test between VNS-RLS and VNS-RL on 25% scaled down and 100% artificial instances.

	LB4-1		LB4-2		TB4-3		TB4-4		LU4-5		LU4-6		TU4-7		TU4-8	
Scale	**25%**	**100%**	**25%**	**100%**	**25%**	**100%**	**25%**	**100%**	**25%**	**100%**	**25%**	**100%**	**25%**	**100%**	**25%**	**100%**
T-test	N	N	N	N	Y	N	N	Y	N	N	Y	N	N	N	N	N
	LB8-1		**LB8-2**		**TB8-3**		**TB8-4**		**LU8-5**		**LU8-6**		**TU8-7**		**TU8-8**	
Scale	**25%**	**100%**	**25%**	**100%**	**25%**	**100%**	**25%**	**100%**	**25%**	**100%**	**25%**	**100%**	**25%**	**100%**	**25%**	**100%**
T-test	Y	N	Y	Y	N	N	N	N	Y	N	N	N	N	N	N	Y

Table 5.14: Comparison on 50% scaled down artificial instances.

Instance		**LB4-1**	**LB4-2**	**TB4-3**	**TB4-4**	**LU4-5**	**LU4-6**	**TU4-7**	**TU4-8**
VNS-RLS	**Best**	74.57%	**76.86%**	68.49%	**69.41%**	61.03%	67.28%	55.63%	55.32%
	Average	73.68%	**76.06%**	67.39%	**67.95%**	60.29%	66.36%	55.48%	55.06%
	Eval	411,992	352,122	312,086	427,614	505,017	469,358	240,935	241,367
	S.D.	0.66%	0.37%	0.69%	0.73%	0.42%	0.44%	0.17%	0.15%
VNS-RL	**Best**	74.37%	76.34%	68.39%	68.57%	61.05%	66.94%	**55.98%**	**55.45%**
	Average	73.46%	75.74%	67.44%	67.59%	60.42%	66.53%	**55.49%**	55.02%
	Eval	214,815	178,610	182,916	179,805	232,713	242,746	140,668	104,870
	S.D.	0.98%	0.35%	0.57%	0.58%	0.44%	0.33%	0.26%	0.18%
VNS-RLS No Ec	**Best**	74.46%	76.40%	**68.60%**	68.92%	**61.71%**	**67.43%**	55.63%	55.33%
	Average	73.72%	75.94%	67.45%	67.89%	60.58%	**66.74%**	55.44%	54.98%
	Eval	401,541	341,333	445,478	362,031	542,568	575,862	233,207	198,381
	S.D.	0.41%	0.40%	0.64%	0.73%	0.61%	0.37%	0.17%	0.16%
VNS-RLS No FI	**Best**	74.35%	76.44%	68.46%	68.73%	61.22%	67.13%	55.66%	55.32%
	Average	**73.84%**	75.96%	**67.64%**	67.75%	60.67%	66.73%	55.47%	**55.09%**
	Eval	412,689	375,605	442,350	427,133	494,563	505,842	260,674	269,741
	S.D.	0.34%	0.51%	0.50%	0.57%	0.33%	0.35%	0.13%	0.12%
VNS-RL No FI	**Best**	**74.64%**	76.82%	68.18%	68.81%	61.45%	67.00%	55.76%	**55.45%**
	Average	73.50%	75.96%	67.49%	67.67%	**60.97%**	66.63%	55.46%	54.98%
	Eval	194,032	174,671	227,019	189,330	231,117	241,260	129,217	106,969
	S.D.	0.58%	0.59%	0.57%	0.84%	0.40%	0.22%	0.25%	0.20%
Instance		**LB8-1**	**LB8-2**	**TB8-3**	**TB8-4**	**LU8-5**	**LU8-6**	**TU8-7**	**TU8-8**
VNS-RLS	**Best**	89.16%	93.65%	67.91%	65.89%	67.75%	67.81%	60.89%	53.90%
	Average	86.40%	92.25%	**67.23%**	64.56%	**67.28%**	67.01%	60.51%	53.73%
	Eval	771,657	767,629	489,994	781,730	563,840	740,547	325,656	379,196
	S.D.	2.01%	1.13%	0.49%	0.76%	0.34%	0.51%	0.20%	0.08%
VNS-RL	**Best**	89.46%	92.72%	67.65%	65.39%	67.53%	67.34%	60.99%	53.76%
	Average	87.03%	91.44%	67.18%	64.69%	66.84%	66.71%	**60.67%**	53.56%
	Eval	463,459	354,493	210,610	390,148	219,171	282,208	165,852	147,020
	S.D.	1.53%	0.97%	0.41%	0.53%	0.36%	0.38%	0.22%	0.14%
VNS-RLS No Ec	**Best**	89.08%	93.66%	67.61%	65.78%	67,59%	**67.87%**	60.73%	**53.99%**
	Average	87.50%	**92.36%**	67.06%	64.73%	67.09%	67.24%	60.49%	**53.79%**
	Eval	913,824	764,714	386,613	715,519	551,729	649,507	312,937	405,527
	S.D.	0.69%	0.89%	0.29%	0.60%	0.47%	0.50%	0.18%	0.13%
VNS-RLS No FI	**Best**	**90.51%**	**93.90%**	**67.99%**	**66.59%**	**67.93%**	67.69%	**61.00%**	53.87%
	Average	**88.21%**	92.31%	67.10%	**65.09%**	67.32%	**67.03%**	60.51%	53.67%
	Eval	1,049,083	838,955	433,102	789,905	630,593	617,863	288,587	350,326
	S.D.	1.66%	1.29%	0.63%	0.88%	0.46%	0.60%	0.24%	0.17%
VNS-RL No FI	**Best**	90.08%	93.06%	67.76%	65.96%	67.49%	67.81%	60.89%	53.85%
	Average	87.00%	91.88%	67.10%	64.80%	66.78%	66.94%	60.60%	53.70%
	Eval	385,955	377,490	189,565	372,753	225,925	312,747	187,888	211,534
	S.D.	1.42%	0.65%	0.33%	0.81%	0.49%	0.54%	0.16%	0.08%

Table 5.15: Comparison on 50% scaled down real-life instances.

Instance		NP4-1	NP4-2	NP4-3	NP4-4	NP4-5
VNS-RLS	Best	75.94%	**65.02%**	**73.18%**	62.49%	**79.91%**
	Average	74.56%	**63.33%**	72.15%	61.46%	79.36%
	Eval	460,790	568,601	378,158	421,254	582,828
	S.D.	0.86%	0.88%	0.57%	0.66%	0.31%
VNS-RL	Best	75.69%	64.54%	72.88%	62.72%	79.64%
	Average	74.55%	62.84%	71.96%	61.76%	79.23%
	Eval	241,986	224,495	184,163	259,344	272,266
	S.D.	0.72%	1.02%	0.66%	0.74%	0.31%
VNS-RLS No Ec	Best	76.06%	63.74%	72.77%	63.63%	79.68%
	Average	75.17%	63.00%	72.22%	61.84%	79.47%
	Eval	534,540	513,641	376,939	518,689	555,183
	S.D.	0.62%	0.45%	0.44%	0.78%	0.14%
VNS-RLS No FI	Best	75.80%	63.71%	72.91%	**62.80%**	79.77%
	Average	74.83%	63.04%	**72.30%**	**61.97%**	**79.51%**
	Eval	477,073	497,737	457,465	562,605	577,746
	S.D.	0.59%	0.49%	0.44%	0.51%	0.11%
VNS-RL No FI	Best	**76.58%**	64.53%	72.67%	62.49%	79.83%
	Average	**75.18%**	63.25%	72.14%	61.37%	79.35%
	Eval	293,465	229,114	206,959	213,268	277,655
	S.D.	0.73%	0.66%	0.63%	0.54%	0.35%
Instance		**NP6-1**	**NP6-2**	**NP6-3**	**NP6-4**	**NP6-5**
VNS-RLS	Best	74.22%	71.86%	60.18%	78.22%	78.82%
	Average	73.33%	71.48%	**59.79%**	77.62%	77.13%
	Eval	790,909	755,430	366,196	922,226	737,902
	S.D.	0.58%	0.24%	0.22%	0.53%	1.11%
VNS-RL	Best	74.09%	72.13%	60.04%	78.38%	77.84%
	Average	73.05%	71.56%	59.76%	77.62%	77.23%
	Eval	368,065	419,235	178,172	386,823	318,487
	S.D.	0.58%	0.35%	0.30%	0.54%	0.80%
VNS-RLS No Ec	Best	**75.34%**	**72.03%**	**60.21%**	78.15%	78.05%
	Average	**73.86%**	**71.68%**	59.72%	77.73%	77.30%
	Eval	936,874	826,244	398,098	833.024	808,691
	S.D.	0.72%	0.25%	0.37%	0.38%	0.81%
VNS-RLS No FI	Best	75.23%	71.82%	**60.21%**	**78.66%**	78.98%
	Average	73.62%	71.45%	59.70%	**77.88%**	77.45%
	Eval	833,619	604,648	340,517	913,326	769,152
	S.D.	0.73%	0.32%	0.26%	0.41%	0.85%
VNS-RL No FI	Best	74.66%	71.98%	60.16%	78.02%	**80.02%**
	Average	73.68%	71.54%	59.75%	77.26%	**78.53%**
	Eval	471,835	354531	174,494	360,102	365,608
	S.D.	0.50%	0.21%	0.41%	0.59%	1.18%
Instance		**NP8-1**	**NP8-2**	**NP8-3**	**NP8-4**	**NP8-5**
VNS-RLS	Best	**73.83%**	74.95%	74.54%	62.39%	72.71%
	Average	72.69%	73.97%	73.93%	**62.32%**	72.04%
	Eval	949,066	916,414	685,003	277,510	937,414
	S.D.	0.62%	0.57%	0.37%	0.06%	0.40%
VNS-RL	Best	73.40%	74.29%	74.33%	62.35%	72.82%
	Average	72.53%	73.86%	**74.01%**	62.20%	72.04%
	Eval	528,557	342,240	332,628	277,510	397,483
	S.D.	0.53%	0.36%	0.29%	0.09%	0.41%
VNS-RLS No Ec	Best	73.66%	74.66%	74.32%	62.36%	72.76%
	Average	**72.80%**	74.07%	73.99%	62.30%	72.00%
	Eval	1,174,959	839.055	624,092	574,786	380,812
	S.D.	0.51%	0.30%	0.26%	0.06%	0.48%
VNS-RLS No FI	Best	73.29%	**74.99%**	**74.87%**	**62.40%**	72.16%
	Average	72.68%	**74.32%**	74.14%	62.30%	71.76%
	Eval	1,124,131	771,843	768,217	502,949	729,571
	S.D.	0.55%	0.45%	0.43%	0.07%	0.27%
VNS-RL No FI	Best	73.54%	74.88%	74.65%	62.36%	**73.33%**
	Average	**72.80%**	74.30%	**74.01%**	62.25%	**72.13%**
	Eval	480,227	338,567	341,863	244,973	380,812
	S.D.	0.68%	0.56%	0.34%	0.06%	0.53%

removed respectively leading to three new algorithm variants (VNS-RLS-NoEc, VNS-RLS-NoFI, VNS-RL-NoFI). The detailed experiment results are also presented in Tables 5.9 - 5.14. Comparing the best and average solutions, VNS-RLS-NoFI produces the most best results on medium and large instances, while VNS-RLS-NoEc outperforms the other variants on small instances.

The amount of infeasible solutions explored over every 10,000 evaluations during the search is recorded to present the trend of infeasible solutions explored by the five algorithm variants. Figure 5.1 presents the typical results of two instances. It is easy to see that, without *Sampling*, both VNS-RL and VNS-RL-NoFI explore much more infeasible solutions than the other three variants throughout the search. The infeasible rates of the three VNS-RLS variants decrease at the beginning of search and stay at a low level afterwards, showing the effects of Sampling on reducing infeasible neighbourhood moves. At each VNS iteration, by estimating the surrounding environment of the starting solution, Sampling leads Local Search to the search regions with high feasibility. Using feasibility indicators in both Sampling and Local Search, VNS-RLS achieves the lowest infeasible rate dring the search. In addition, in Figure 5.1, the two lines of VNS-RL and VNS-RL-NoFI are shorter than the other three VNS-RLS variants, indicating the two variants converge earlier.

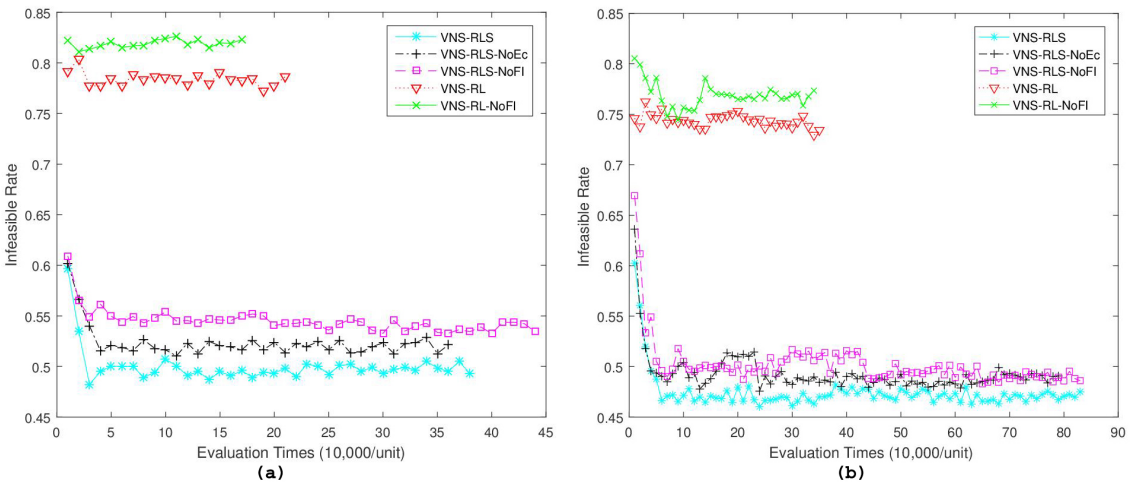


Figure 5.1: Comparison of infeasible rates on 25% NP4-3 (a) and 100% LB4-1 (b).

It can be concluded that, both *Sampling* and *Feasibility indicator* can reduce the infeasible rate of neighbourhood moves, while *Sampling* brings a greater impact, obtaining better solutions within the same computational time. *Sampling*, even without the feasibility indicator E^c, can significantly decrease the infeasible rate. This indicates that the other two indicators (E^a and E^b) apt to guide the search

to feasible solution regions, meaning the high quality neighbourhood has higher feasibility. The use of feasibility indicator in *Sampling* and *Local Search* contributes to a higher probability of exploring feasible neighbourhood moves during the search.

It should be noted that, the lowest infeasible rate does not necessarily contribute to the best objective function value. The removal of feasibility indicators may also increase the flexibility and diversification of the search, thus improves the search performance, e.g. on the LB8-1 100% instance, VNS-RLS-NoFI is the only one variant which finds the solution with objective value 87.49%. It is almost 2% higher than the other four variants. Overall, VNS-RLS-NoEc and VNS-RLS-NoFI produce the most best-found solutions and average solutions in all the five variants.

5.3.2.3 Comparison with the general VNS and State-of-the-art Algorithms

PVRPTW-O is a new model in the literature, thus there is no existing algorithms applied to it directly. A general VNS algorithm (VND) [147] with the same operators as VNS-RLS is implemented. Besides, considering the similarity between PVRP and PVRPTW-O, two state-of-the-art methodologies for PVRP, namely RVNS [244] and FVNS [150], are also modified (e.g. initial solution construction, side constraints consideration and problem specific parameter setting) and applied to PVRPTW-O for comparison in our research. Both RVNS and FVNS use the VNS framework as well. Apart from the different operators used (i.e. different shaking operators and neighbourhood structures), the main difference between them is that the shaking operators are randomly selected in RVNS but applied in a fixed sequence in FVNS. In addition, due to the time constraints and multiple periods in PVRPTW-O, algorithms for Open-VRP and Open-VRPTW are either inapplicable or not suitable for PVRPTW-O, thus they are not compared in this study.

Table 5.16: Result deterioration comparing with VNS-RLS.

		NP4	NP6	NP8	LB4	LB8	TB4	TB8	LU4	LU8	TU4	TU8
VND	Best	-0.05%	-0.22%	-0.28%	-0.24%	-0.33%	-0.38%	-0.66%	-0.71%	-0.48%	-0.14%	-0.31%
	Average	-0.31%	-0.29%	-0.36%	-0.46%	-1.36%	-0.48%	-1.57%	-0.62%	-0.47%	-0.62%	-0.55%
	S.D.	-0.03%	-0.10%	0%	-0.15%	-0.19%	-0.13%	-0.34%	-0.18%	-0.04%	-0.06%	-0.04%
RVNS	Best	-1.77%	-2.36%	-1.73%	-3.42%	-5.66%	-1.00%	-2.58%	-1.58%	-1.39%	-0.47%	-0.55%
	Average	-2.23%	-3.43%	-2.11%	-4.21%	-6.62%	-0.74%	-2.19%	-2.52%	-2.02%	-0.42%	-0.80%
	S.D.	-0.33%	-0.50%	-0.34%	-0.49%	-0.51%	0.06%	0.20%	-0.20%	-0.21%	0.15%	0.04%
FVNS	Best	-1.86%	-2.26%	-1.64%	-2.67%	-5.59%	-1.55%	-3.16%	-2.07%	-1.99%	-0.47%	-0.73%
	Average	-2.53%	-3.94%	-2.58%	-4.65%	-7.21%	-0.94%	-2.27%	-2.83%	-2.50%	-0.48%	-0.83%
	S.D.	-0.35%	-0.60%	-0.50%	-0.71%	-0.71%	0.20%	0.24%	-0.15%	-0.11%	0.16%	0.07%

The comparison results are presented in Table 5.16, which gives the deterioration comparing to the results of VNS-RLS-NoEc. Generally, our algorithm outperforms the other three algorithms on most categories of instances on both the solution quality and stability. VND produces significantly better results than RVNS and FVNS, while RVNS and FVNS virtually tie. It indicates that the customised neighbourhood operators in VNS-RLS contribute much to the improvement of algorithm performance in this problem, since RVNS and FVNS use different neighbourhood operators. The operators in VNS-RLS bring systematical exploration and perturbation at different levels, cooperating with the proposed reinforcement learning scheme and the effective indicators, thus exploration and exploitation are better balanced in the search. The solution deterioration is smaller on tight and unbalance instances (TU), which shows the stronger search ability of VNS-RLS on the larger search space caused by the loose time windows and balanced workload. However, RVN and FVNS show better stability on tight instances (TB4, TB8, TU4 and TU8) than VNS-RLS.

5.3.2.4 Comparison with the Previous Scheduling Scheme

Comparing the PVRPTW-O model with the study of [60, 16], the biggest difference is that the shift-change between day-shift and night-shift can occur at any terminal in PVRPTW-O. The open routes in each shift may lead to the increase of HLDR to the fleet. Table 5.17 presents the increase of HDLR obtained by applying PVRPTW-O. The values in the table are the differences of HLDR between the best-found results on both models. It can be found that the use of PVRPTW-O increases the utility of vehicle to varying degrees, while the improvement on artificial instances is larger. This scheduling scheme can further save operating cost and promote efficiency for the company.

Table 5.17: The growth of HLDR brought by the PVRPTW-O scheduling scheme.

Instances	**NP4**	**NP6**	**NP8**	**LB4**	**TB4**	**LU4**	**TU4**	**LB8**	**TB8**	**LU8**	**TU8**
Growth	0.59%	1.44%	0.29%	0.80%	4.48%	3.39%	4.31%	1.98%	4.51%	5.11%	6.26%

5.4 A Variable-Depth Adaptive Large Neighbourhood Search Algorithm

It can be found that, in the experimental results of VNS-RLS on the benchmark, the gaps between the obtained objective values and upper bounds are still large on several instances. Considering that the operators used in VNS-RLS bring relatively

small changes to solutions, techniques with larger perturbation and higher diversification are worthy of further investigation. Therefore, a Variable-Depth Adaptive Large Neighbourhood Search algorithm (VD-ALNS) is devised for PVRPTW-O, providing a systematic exploration with large neighbourhood structures in the solution space.

5.4.1 Algorithmic Details

Adaptive Large Neighbourhood Search (ALNS) algorithms use variable *Destroy* and *Repair* operators, bringing larger perturbation to the search than traditional local search approaches. Considering the multiple dimensional structure in the solution of PVRPTW-O, a deterministic neighbourhood depth changing scheme is devised and applied to the ALNS algorithm, enhancing the intensification of search. The algorithmic details of the proposed VD-ALNS algorithm are presented.

5.4.1.1 Algorithm Framework

The framework of VD-ALNS is shown in Algorithm 5. The initial solution in this algorithm is generated by ULIH as well. Starting from the initial solution, four destroy operators and three repair operators are then used to produce new solutions by modifying the current solution ($S_{current}$), pursuing solutions with higher quality.

The $Weight$ and $Score$ in ***Step 3*** are two scalars used to guide the subsequent search. Specifically, $Score_i$ records the contributions of operator i in solution improvement within a fixed number of iterations (so called a $Segment$). $Score_i$ is used to update the value of $Weight_i$, which determines the probability of operator i being adopted during search in the next Segment. Their values are set as the same for all operators at the beginning, and then updated during the search. The algorithm iteratively explores the solution space until the Stopping Criterion is met, i.e. the quality of the best-found solution (S) has not been improved in the last $UNIMPR_{MAX}$ iterations, or the improvement is less than 1% in the last $INCRE_{MAX}$ iterations.

In ***Step 4.1***, $Depth$ is the range the operators work upon. It is systematically switched between the whole planning horizon (*HORIZON*) and a specified shift (*SHIFT*) to balance the exploration and exploitation. In ***Step 4.2***, a pair of a destroy operator (D_i) and a repair operator (R_j) are used to generate a new solution (S').

Every single operator in ALNS has its own weight ($Weight_i$). However, a research issue here is whether an operator should be assigned two different weights for two different depths to separately record its contribution of improvement at

depths *HORIZON* and *SHIFT*, or only one weight is sufficient to record all previous contribution. In the literature this question has been addressed in VNS and VDNS [246]. Using two independent weights separately records knowledge collected during the search employing two independent operators at different depths, thus would prevent the knowledge collected at the other depth from being used. However, in terms of the neighbourhood depth, it is found that search experience at different operation depths can contribute and promote each other in our preliminary experiments. Thus, VD-ALNS adopts operators in both scenarios, and records an operator's information with only one scalar.

Algorithm 5 The VD-ALNS framework

Step 1: Input the problem instance, Stopping Criterion, ITE_{MAX} and $LEN_{SEGMENT}$.
Step 2: Produce an initial feasible solution S with the ULIH. // Section 5.2.
Step 3: **Set up the initial parameters.**
$Weight \leftarrow \{1, \cdots, 1\}, Score \leftarrow \{0, \ldots, 0\}, S_{current} \leftarrow S, Depth \leftarrow HORIZON$.
Step 4:
while Stopping Criterion is not met **do**
Step 4.1: **Variable-Depth Setting.** // Section 5.4.1.2.
if S is not improved in the last ITE_{max} iterations **then**
if $Depth = HORIZON$ **then**
$Depth \leftarrow SHIFT$.
else
$Depth \leftarrow HORIZON$.
end if
end if
Step 4.2: **Operators Selection and Execution.** // Section 5.4.1.3.
Select a Destroy Operator (D_i) and a Repair Operator (R_j) based on $Weight$.
Execute D_i and R_j at $Depth$, obtaining a new solution: $S' \leftarrow R_j(D_i(S_{current}))$.
Step 4.3: **Accept or Reject.** // Section 5.4.1.4.
Use a Record-to-Record Travel algorithm to determine if the newly generated solution is acceptable: $S_{current} \leftarrow S'$.
If the quality of S' is better than S, update the best-found solution: $S \leftarrow S'$.
Step 4.4: **Weight Adjustment.** // Section 5.4.1.5.
At every iteration, the Scores of D_i and R_j ($Score_i$ and $Score_j$) are updated according to the quality of S'.
At every $LEN_{SEGMENT}$ iteration, $Weight$ is updated based on the accumulated $Score$, then $Score$ is reset.
end while
Step 5: Output the improved solution S.

A pair of operators is selected by *Roulette Wheel* based on the weights of operators in *Step 4.2*. The probability of an operator i being selected is calculated with eq. (5.16), where h is the number of candidate operators.

$$Pr_i = \frac{Weight_i}{\sum_j^h Weight_j} \tag{5.16}$$

Step 4.3 decides if S' is accepted as new $S_{current}$ and S is updated, while ***Step 4.4*** adjusts the scores and weights of operators according to the quality of

S'. These adaptive weights guide the search to promising solution regions. More details are introduced in following subsections.

5.4.1.2 Variable-Depth Setting

Variable search depth endows a balanced search performance. When *Depth* is *SHIFT*, the destroy operators remove a number of nodes from one specified shift, while the repair operators reinsert them back into that shift. All the shifts are specified and checked sequentially. When *Depth* is *HORIZON*, the removal and reinsertion happen within the whole planning horizon. Obviously, *HORIZON* is a greater depth than *SHIFT*, and leads to a greater change in a solution, thus improves the diversification of search. Contrarily, using a *Depth* of *SHIFT* modifies routes in a single shift by locally optimising the solution, thus increases the intensification of search.

Depth is systematically switched to seek a trade-off between exploration and exploitation. Searching with smaller depth exploits a relatively smaller solution area intensively, while larger search depth avoids search trapping into local optima. In the proposed algorithm, *Depth* would be switched to the other value when S is not improved in ITE_{MAX} iterations, to keep both the diversification and intensification in searching the large scale tightly constrained solution space.

5.4.1.3 Operators of Destroy and Repair

Four destroy operators and three repair operators are developed in VD-ALNS.

Destroy Operators

In each iteration, q nodes are removed by a destroy operator (Removal Heuristic). The value of q increases by 5 when the solution is not improved in the last iteration. As a too small q will hardly bring change to a solution, while a too large q will significantly increase repair operation time and the algorithm becomes a random search, a lower bound of $max\{0.1n, 10\}$ and an upper bound of $min\{0.5\text{n}, 60\}$ are set for q, where n is the total number of nodes.

- *Random Removal*: The q nodes to be removed are randomly selected.
- *Worst Removal*: This is a greedy heuristic, where the top q nodes causing the greatest cost will be removed. In other words, removing the q task nodes brings the greatest cost reduction to the solution.

- *Worst Edge Removal*: This is also a greedy heuristic, which deletes q nodes adjacent to arcs of the highest cost.

- *Related Removal*: Shaw [277] proposes this operator based on the observation that, if nodes relate to one another are removed together, there would be an opportunity to interchange them in the later repaired solution. In VD-ALNS, we define the Relatedness of two task nodes (i and j) from five aspects: Service Time (R_{ij}^{ST}), Time window (R_{ij}^{TW}), Service Starting Time (R_{ij}^{SST}), Vehicle used (R_{ij}^{V}) and Source and Destination (R_{ij}^{SD}) as follows.

$$R_{ij}^{ST} = \frac{|l_i - l_j|}{0.5 \cdot (l_i + l_j)} \tag{5.17}$$

$$R_{ij}^{TW} = \frac{0.5 \cdot (|a_i - a_j| + |b_i - b_j|)}{max\{b_i, b_j\} - min\{a_i, a_j\}} \tag{5.18}$$

$$R_{ij}^{SST} = \frac{|B_i - B_j|}{Length\ of\ Planning\ Horizon} \tag{5.19}$$

$$R_{ij}^{V} = \begin{cases} 0 & i \text{ and } j \text{ are serviced by the same vehicle.} \\ 0.5 & i \text{ and } j \text{ are serviced in the same shift, but by different vehicles.} \\ 1 & otherwise. \end{cases} \tag{5.20}$$

$$R_{ij}^{SD} = \begin{cases} 0 & i \text{ and } j \text{ have the same source } AND \text{ destination.} \\ 0.5 & i \text{ and } j \text{ have the same source } OR \text{ destination.} \\ 1 & otherwise. \end{cases} \tag{5.21}$$

Correspondingly, the relatedness of two task nodes (R_{ij}) is a linear combination of the five components above-mentioned (eq. (5.22)). The values of the five linear coefficients are discussed in Section 5.4.2.1. In *Relatedness Removal*, the first node to be removed is randomly selected, then the other nodes are sorted in ascending order of their relatedness R_{ij} to the first node.

$$R_{ij} = \alpha \cdot R_{ij}^{ST} + \beta \cdot R_{ij}^{TW} + \gamma \cdot R_{ij}^{SST} + \delta \cdot R_{ij}^{V} + \epsilon \cdot R_{ij}^{SD} \quad (s.t.\ \alpha+\beta+\gamma+\delta+\epsilon = 1) \tag{5.22}$$

The rest $q-1$ nodes to be removed are randomly selected with a preference of smaller R_{ij}, where the nodes with the index of $\lceil N\rho^D \rceil$ will be removed. Here, N is the number of candidate nodes, ρ is a random number between 0 and 1, and D is a constant greater or equal to 1. The greater D is, the stronger the preference would be, while D is empirically set to 3 in VD-ALNS.

Repair Operators

The nodes removed in the Destroy phase will be reinserted back into the solution following the below specific rules of each repair operator (Insertion Heuristic).

- *Random Insertion*: The removed nodes are randomly inserted into feasible positions.
- *Greedy Insertion*: The removed nodes are inserted into their best feasible positions. Here the best position causing the least cost increase.
- *Regret2 Insertion*: This greedy insertion heuristic is proposed by Pisinger and Ropke [245], which always inserts firstly the node of the largest *REGRET* into its best feasible position. The *REGRET* of a node is the cost difference between inserting the node to its best and second best feasible positions.

5.4.1.4 Acceptance Criterion

Record-to-Record Travel acceptance criterion is also used in VD-ALNS to determine if the newly generated solution (S') is acceptable during the search. If S' is better than the best-found solution S (i.e. $COST(S') < COST(S)$), S' will be accepted as the current solution ($S_{current}$). In addition, a new solution worse than $S_{current}$ is also acceptable as long as the gap between their $COST$ is less than a *DEVIATION* threshold (i.e. $0.01 \cdot COST(S)$).

5.4.1.5 Weight Adjustment

In each iteration, the employed operator i is rewarded a value $\sigma \geq 0$ based on the quality of the generated solution S' (see eq. 5.23). The effect of σ is further studied in Section 5.4.2.1.

$$\sigma = \begin{cases} \sigma_1 & S' \; is \; accepted \; AND \; COST(S') < COST(S) \\ \sigma_2 & S' \; is \; accepted \; AND \; COST(S) < COST(S') < COST(S_{current}) \\ \sigma_3 & S' \; is \; accepted \; AND \; COST(S_{current}) < COST(S') \\ \sigma_4 & S' \; is \; rejected \end{cases} \tag{5.23}$$

After a fixed number ($LEN_{SEGMENT}$) of iterations (a Segment), the total accumulated reward saved in $Score_i$ in the current Segment $t-1$ is used to update the weight of operator i for the next Segment t (see eq. (5.24)). In eq. (5.24), the reaction factor r controls how quickly the adjustment scheme reacts. u_i is the

number of times operator i is used in Segment $t-1$. After updating $Weight_i^t$, $Score_i$ is reset to zero to start the accumulation of reward in Segment t.

$$Weight_i^t = r \cdot Weight_i^{t-1} + (1-r) \cdot \frac{Score_i}{u_i} \tag{5.24}$$

5.4.2 Experiment Results and Analysis

5.4.2.1 Parameter Tuning and Analysis

Parameters in VD-ALNS are studied one at a time, fixing the other parameters. It is easy to understand that, higher $UNIMPR_{MAX}$ and $INCRE_{MAX}$ lead to more iterations in search, so might bring better solutions but at the cost of longer time. ITE_{MAX} represents the times of one *Depth* value would be continuously used. The trade-off between the solution quality and running time needs to be considered to strike a balance between effectiveness and efficiency of the search. The values of parameters used in VD-ALNS are presented in Table 5.18.

Table 5.18: Parameters in VD-ALNS.

Parameter	σ_1	σ_2	σ_3	σ_4	$UNIMPR_{MAX}$	$INCRE_{MAX}$	ITE_{MAX}
Value	30	15	5	0	150	200	4*No Of Shifts
Parameter	α	β	γ	δ	ϵ	$LEN_{SEGMENT}$	r
Value	0.3	0.2	0.1	0.2	0.2	70	0.4

In adaptive weight adjustment, the values of rewards represent the contributions in solution improvement. To obtain the best setting of reward values, σ_4 is set to zero, which indicates $Score_i$ stays the same when S is rejected. Besides, σ_3 is set to 5 as a base unit. Different σ_1 and σ_2 are tested in parameter tuning experiments to find the setting generating the best solutions. It is observed that a too large σ_1 would cause premature search. The best solutions are obtained when the reward to producing a new best solution (σ_1) is two times of that of generating an acceptable solution better than $S_{current}$ (σ_2), and six times of that of obtaining an acceptable solution worse than $S_{current}$ (σ_3).

When tuning the definition of *Relatedness* (eq. (5.22)), all the five components are firstly assigned equal weights ($\alpha = \beta = \gamma = \delta = \epsilon = 0.2$). Then, each coefficient is gradually increased to reflect the contribution of the associated component to the total relatedness. It is found that when the weight of Service Time Relatedness (R_{ij}^{ST}) is high, the quality of solutions is higher. This indicates that reassigning two tasks with a higher similarity of Service Time leads to a higher possibility to produce a better solution. Since the Service Staring Time of a task may change for various reasons (e.g., a task is assigned to a new truck, and a precedent task is

reassigned, etc.), R_{ij}^{SST} can hardly represent the relatedness of two tasks and shows low contribution in tuning tests. A lower coefficient is given to R_{ij}^{SST}.

A too small $LEN_{SEGMENT}$ changes the weights of operators frequently and thus the search may converge prematurely. On the other hand, a large $LEN_{SEGMENT}$ cannot update the guidance information in time. Our preliminary experiments show that the best performance is found when $LEN_{SEGMENT}$ is between 50 and 80. In eq. (5.24), the higher r is, the slower the algorithm reacts to the latest guidance information. VD-ALNS performs the best when r is between 0.4 and 0.6.

5.4.2.2 Comparison of Solution Algorithms

To demonstrate the contribution of variable depth, a standard ALNS for OPVRPTW is also implemented, where the Destroy and Repair operators are only used at the depth of *HORIZON* in global searching. Comparing to other meta-heuristics using small change operators, both VD-ALNS and ALNS have a stronger ability to escape from local optima in a tightly constrained solution space. They are compared with VNS-RLS-NoEc, which uses neighbourhood operators with small changes producing the best results among all VNS-RLS variants.

Table 5.19: HLDR comparison on 25% scaled down artificial instances.

Instance		LB4-1	LB4-2	TB4-3	TB4-4	LU4-5	LU4-6	TU4-7	TU4-8
VNS-RLS	Best	76.92%	**83.42%**	**69.08%**	66.41%	**60.71%**	61.08%	48.75%	54.97%
	Average	74.80%	**81.61%**	**67.78%**	64.95%	**59.29%**	**60.62%**	48.54%	**54.68%**
	Eval	313,707	280,849	286.059	298,651	321,835	290,082	166,248	193,536
	S.D.	0.95%	1.09%	0.65%	0.75%	0.64%	0.29%	0.30%	0.33%
ALNS	Best	78.85%	81.85%	68.41%	66.94%	58.87%	59.35%	**49.42%**	54.12%
	Average	77.84%	80.08%	67.36%	66.06%	57.84%	58.60%	48.87%	53.35%
	Eval	438	421	426	410	396	287	371	287
	S.D.	0.67%	1.01%	0.51%	0.39%	0.52%	0.37%	0.39%	0.43%
VD-ALNS	Best	**79.16%**	**83.42%**	68.92%	**67.01%**	59.84%	**61.16%**	**49.42%**	**55.31%**
	Average	**77.98%**	80.92%	67.45%	**66.22%**	58.74%	59.37%	**49.05%**	54.19%
	Eval	445	448	457	443	472	477	411	448
	S.D.	0.75%	0.95%	0.65%	0.36%	0.47%	0.46%	0.38%	0.48%
Lower Bound		66.62%	76.41%	69.91%	69.30%	NF	58.65%	50.37%	55.36%
Upper Bound		100%	94.87%	86.31%	83.51%	79.94%	73.90%	52.17%	66.38%
Instance		LB8-1	LB8-2	TB8-3	TB8-4	LU8-5	LU8-6	TU8-7	TU8-8
VNS-RLS	Best	**91.25%**	**93.56%**	63.05%	66.31%	**65.76%**	**66.58%**	**56.46%**	52.29%
	Average	**89.76%**	**92.09%**	61.78%	63.25%	**64.86%**	**65.58%**	**55.79%**	**51.93%**
	Eval	492,628	547,853	296,837	517,855	438,295	439,782	269,164	281,479
	S.D.	0.95%	0.87%	0.54%	1.16%	0.44%	0.49%	0.29%	0.18%
ALNS	Best	87.37%	87.87%	63.61%	66.12%	64.84%	60.34%	55.37%	51.89%
	Average	83.02%	84.41%	62.75%	64.89%	63.61%	58.13%	54.69%	51.28%
	Eval	398	396	403	461	437	318	334	385
	S.D.	2.40%	1.40%	0.59%	0.74%	0.54%	0.73%	0.23%	0.42%
VD-ALNS	Best	88.71%	89.62%	**64.37%**	**67.01%**	65.30%	63.08%	55.52%	**52.41%**
	Average	84.32%	84.35%	**62.99%**	**65.26%**	63.93%	59.95%	54.78%	51.81%
	Eval	515	499	549	535	598	590	482	577
	S.D.	1.87%	1.95%	0.59%	0.54%	0.57%	1.29%	0.14%	0.39%
Lower Bound		NF	NF	56.85%	52.40%	57.42%	NF	47.65%	50.74%
Upper Bound		100%	100%	82.33%	88.75%	78.33%	86.84%	71.59%	70.43%

The comparison results on the 25% scaled down dataset are presented in Tables 5.19 and 5.20. The three algorithms are compared from four aspects: best-found

Table 5.20: HLDR comparison on 25% scaled down real-life instances.

Instance		NP4-1	NP4-2	NP4-3	NP4-4	NP4-5
VNS-RLS	Best	**82.89%**	62.32%	**75.64%**	59.76%	**79.24%**
	Average	**81.51%**	61.42%	**74.92%**	59.18%	**78.48%**
	Eval	469,233	311,885	319,202	347,134	326,956
	S.D.	1.16%	0.60%	0.62%	0.35%	0.42%
ALNS	Best	81.15%	**65.51%**	75.17%	61.86%	77.14%
	Average	79.80%	65.08%	73.60%	61.47%	76.15%
	Eval	385	500	458	499	395
	S.D.	0.72%	0.33%	0.80%	0.27%	0.57%
VD-ALNS	Best	81.74%	65.45%	75.54%	**62.53%**	77.67%
	Average	79.61%	**65.16%**	74.15%	**61.75%**	77.03%
	Eval	483	529	503	549	573
	S.D.	1.20%	0.25%	0.82%	0.27%	0.53%
Lower Bound		78.36%	65.14%	64.83%	54.39%	NF
Upper Bound		92.36%	97.04%	100%	97.72%	100%
Instance		NP6-1	NP6-2	NP6-3	NP6-4	NP6-5
VNS-RLS	Best	76.24%	**73.39%**	62.32%	**80.50%**	82.44%
	Average	74.99%	**72.83%**	62.06%	**79.84%**	80.53%
	Eval	698,514	624,078	253,037	541,548	365,435
	S.D.	0.96%	0.41%	0.20%	0.41%	1.72%
ALNS	Best	79.07%	70.28%	65.00%	78.43%	82.15%
	Average	78.03%	69.42%	64.26%	77.07%	**80.58%**
	Eval	420	449	412	426	450
	S.D.	0.69%	0.49%	0.42%	0.80%	0.69%
VD-ALNS	Best	**79.95%**	70.75%	**65.31%**	78.26%	**82.75%**
	Average	**78.33%**	69.85%	**64.40%**	77.07%	80.34%
	Eval	549	537	553	515	496
	S.D.	0.92%	0.49%	0.47%	0.76%	1.19%
Lower Bound		NF	NF	54.30%	NF	66.11%
Upper Bound		NF	NF	95.20%	NF	98.39%
Instance		NP8-1	NP8-2	NP8-3	NP8-4	NP8-5
VNS-RLS	Best	**76.91%**	**77.76%**	**75.35%**	60.90%	**72.27%**
	Average	**74.72%**	**77.16%**	**74.93%**	60.47%	**71.68%**
	Eval	607,961	525,479	442,103	430,962	516,872
	S.D.	1.20%	0.37%	0.31%	0.32%	0.36%
ALNS	Best	74.74%	74.32%	75.08%	61.85%	71.60%
	Average	73.90%	73.07%	74.29%	61.66%	71.05%
	Eval	445	444	442	421	439
	S.D.	0.54%	0.49%	0.59%	0.14%	0.29%
VD-ALNS	Best	75.50%	74.76%	75.09%	**61.92%**	71.58%
	Average	74.22%	73.53%	74.53%	**61.70%**	71.10%
	Eval	579	524	528	456	527
	S.D.	0.57%	0.58%	0.36%	0.14%	0.31%
Lower Bound		NF	NF	NF	NF	NF
Upper Bound		98.98%	100%	100%	NF	100%

solution (Best), average solution (Ave), evaluation times (Eval) and standard deviation (S.D.). All the results are obtained from 30 runs. The lower and upper bounds of optimal solutions are obtained by CPLEX introduced in Section 5.1. NF in the tables means no feasible solution is found by CPLEX.

From the experiment results we can find that, VD-ALNS beats ALNS in almost all instances, indicating that the variable depth scheme does improve the performance of ALNS. This scheme enhances the exploitation in local areas, leading to increased total evaluation times in ALNS. Comparing with VNS-RLS, on 6 of 15 real-life instances and half of artificial instances, VD-ALNS finds better or equally good solutions, showing no significantly difference. However, VD-ALNS takes remarkably less evaluation times and 90% computational time of VNS-RLS to obtain those results. All the three methods have similar stability of a difference on S.D. lower than 1%.

Tables 5.21 and 5.22 present results on the original Ningbo Port instances. It

Table 5.21: HLDR comparison on 100% artificial instances.

Instance		LB4-1	LB4-2	TB4-3	TB4-4	LU4-5	LU4-6	TU4-7	TU4-8
VNS-RLS	Best	73.52%	**78.08%**	**69.32%**	72.24%	**64.67%**	**68.12%**	**53.21%**	**53.80%**
	Average	72.93%	**77.70%**	**68.54%**	71.42%	**64.38%**	**67.52%**	**53.03%**	**53.61%**
	Eval	642,796	617,656	616,237	635,130	724,154	782,608	399,970	290,599
	S.D.	0.32%	0.32%	0.42%	0.49%	0.20%	0.40%	0.16%	0.08%
ALNS	Best	75.98%	77.28%	68.68%	73.03%	61.11%	64.45%	52.75%	53.39%
	Average	75.41%	76.68%	68.05%	71.52%	60.59%	63.85%	52.01%	53.39%
	Eval	328	193	222	257	316	202	242	106
	S.D.	0.48%	0.35%	0.43%	1.26%	0.35%	0.30%	0.43%	0.00%
VD-ALNS	Best	**76.05%**	77.15%	69.03%	**73.66%**	61.04%	65.33%	52.88%	53.66%
	Average	**75.14%**	76.83%	68.51%	**72.78%**	60.40%	64.80%	52.49%	53.47%
	Eval	379	253	309	315	400	255	294	151
	S.D.	0.60%	0.18%	0.38%	0.64%	0.43%	0.49%	0.39%	0.10%
Upper Bound		79.47%	86.33%	84.05%	88.74%	74.11%	74.47%	64.05%	63.50%
Instance		LB8-1	LB8-2	TB8-3	TB8-4	LU8-5	LU8-6	TU8-7	TU8-8
VNS-RLS	Best	85.49%	**94.03%**	**69.59%**	**66.85%**	**67.81%**	**68.41%**	**59.60%**	**54.50%**
	Average	84.11%	**92.83%**	**69.04%**	**65.70%**	**67.20%**	**68.07%**	**59.21%**	**54.23%**
	Eval	1,443,635	1,134,773	669,136	1,478,978	1,114,876	1,026,285	572,065	859,770
	S.D.	0.95%	1.05%	0.38%	0.76%	0.34%	0.21%	0.21%	0.16%
ALNS	Best	**91.22%**	92.98%	68.60%	63.76%	66.95%	61.68%	59.26%	53.78%
	Average	83.01%	84.98%	67.80%	63.33%	65.28%	60.12%	58.86%	53.18%
	Eval	231	212	236	232	275	225	242	210
	S.D.	3.44%	3.35%	0.49%	0.28%	0.34%	0.57%	0.15%	0.32%
VD-ALNS	Best	88.71%	89.74%	69.53%	64.95%	67.01%	62.30%	58.99%	54.31%
	Average	**85.96%**	86.67%	68.52%	63.78%	65.38%	61.29%	58.77%	53.10%
	Eval	339	347	427	336	280	343	251	175
	S.D.	2.43%	1.77%	0.55%	0.75%	0.53%	0.76%	0.15%	0.50%
Upper Bound		98.26%	97.97%	87.06%	92.44%	74.27%	71.36%	70.29%	56.54%

Table 5.22: HLDR comparison on 100% real-life instances.

Instance		NP4-1	NP4-2	NP4-3	NP4-4	NP4-5
VNS-RLS	Best	**83.29%**	**69.85%**	72.90%	66.61%	80.65%
	Average	**81.88%**	**69.56%**	72.20%	65.91%	80.48%
	Eval	779,504	575,894	661.384	923,891	718,219
	S.D.	0.55%	0.16%	0.38%	0.47%	0.17%
ALNS	Best	81.68%	69.08%	**74.72%**	66.63%	78.16%
	Average	80.21%	68.62%	**74.06%**	66.11%	77.78%
	Eval	212	281	288	271	267
	S.D.	0.99%	0.36%	0.49%	0.29%	0.22%
VD-ALNS	Best	82.30%	69.13%	73.94%	**67.05%**	78.96%
	Average	81.42%	68.83%	73.01%	**66.28%**	78.11%
	Eval	313	501	243	345	297
	S.D.	0.58%	0.21%	0.86%	0.56%	0.49%
Upper Bound		90.43%	70.23%	79.58%	73.72%	81.20%
Instance		NP6-1	NP6-2	NP6-3	NP6-4	NP6-5
VNS-RLS	Best	79.64%	**74.14%**	58.94%	**79.52%**	**79.99%**
	Average	**79.07%**	**73.72%**	58.62%	**79.10%**	**78.36%**
	Eval	1,030,825	1,163,006	513,974	1,053,682	984,987
	S.D.	0.47%	0.21%	0.23%	0.53%	0.99%
ALNS	Best	76.73%	69.16%	**65.27%**	77.99%	77.43%
	Average	76.27%	64.76%	64.79%	77.11%	76.64%
	Eval	265	44	251	236	274
	S.D.	0.29%	3.04%	0.35%	0.49%	0.56%
VD-ALNS	Best	**81.74%**	71.73%	65.16%	78.67%	77.39%
	Average	77.04%	70.95%	**64.84%**	77.86%	76.52%
	Eval	483	300	303	381	387
	S.D.	1.20%	0.69%	0.24%	0.50%	0.54%
Upper Bound		83.93%	76.67%	66.90%	80.97%	84.30%
Instance		NP8-1	NP8-2	NP8-3	NP8-4	NP8-5
VNS-RLS	Best	**73.80%**	**75.27%**	**74.20%**	**61.97%**	**73.62%**
	Average	**73.48%**	**74.86%**	**73.96%**	**61.91%**	**73.26%**
	Eval	1,498,392	978,695	867,663	693,779	1,189,550
	S.D.	0.15%	0.28%	0.22%	0.06%	0.35%
ALNS	Best	69.53%	71.88%	74.02%	61.13%	72.63%
	Average	68.58%	71.56%	73.22%	61.00%	72.05%
	Eval	113	253	227	322	290
	S.D.	0.45%	0.23%	0.40%	0.09%	0.45%
VD-ALNS	Best	70.13%	72.48%	74.02%	61.17%	73.07%
	Average	69.72%	71.39%	73.67%	60.98%	72.59%
	Eval	303	284	338	306	365
	S.D.	0.31%	0.28%	0.23%	0.09%	0.34%
Upper Bound		77.04%	77.55%	78.82%	62.53%	76.09%

can be found that, with the variable depth scheme, VD-ALNS outperforms ALNS again from the aspects of both the average and best-found solution. New best solutions are generated by VD-ALNS on 7 out of 31 benchmark instances.

5.4.2.3 Contributions of Operators

Table 5.23 provides statistics on the Destroy and Repair operators. On the scaled down dataset, one single operator is excluded at a time in VD-ALNS to record the resulting solution quality deterioration. The second and third columns show the average deterioration on the best-found solution and average solution, while the last two columns give the maximum deterioration on the dataset.

Table 5.23: Contributions of each operator

Operator	Best sol. deg.	Avg. sol. deg.	Max best sol. deg.	Max avg. sol. deg.
Random Removal	0.15%	0.23%	1.08%	0.13%
Worst Removal	0.33%	0.60%	2.18%	2.14%
Related Removal	0.09%	0.08%	1.32%	0.68%
Worst Edge Removal	0.55%	0.56%	2.87%	2.14%
Random Insertion	0.21%	0.12%	1.80%	1.09%
Greedy Insertion	4.84%	5.34%	9.64%	7.69%
Regret2 Insertion	0.54%	0.25%	4.07%	1.31%

The results indicate the contributions of each operator in VD-ALNS. It can be found that, Worst Edge Removal is the most efficient destroy operator, followed by Worst Removal. Related Removal contributes the least. Among all repair operators, Greedy Insertion is the most useful, followed by Regret2 Insertion. Overall, greedy heuristics provide effective complement on search intensification and outperform the others in VD-ALNS.

5.4.2.4 Analysis of Runtime

The Destroy and Repair operators in ALNS bring greater changes than the traditional neighbourhood operators by operating on more nodes and making greater perturbation. Therefore, the computation time spent on choosing removal nodes and insertion positions is considerable. The evaluation times of ALNS and VD-ALNS to obtain these results are significantly less than that of VNS-RLS, but the computational time of VD-ALNS compared with VNS-RLS is around 17% more on the original instances, and slightly less on small instances. This observation indicates that scalability of the runtime of VD-ALNS is worse (increases faster) than VNS-RLS along with the instance size.

Choosing the insertion position is time-consuming. Actually, the computational time of the repair operators accounts for a larger proportion of the overall time, around 3.5 times of the destroy operators on scaled down instances. Whats more,

on the original dataset, the repair operation may spend more than 95% of the total computing time.

5.5 Summary

This chapter studies the solution methodologies for the short-haul container transshipment problem in Ningbo Port. By solving the associated mathematical problem model with CPLEX, it is found that exact search approaches are unrealistic to the PVRPTW-O. Based on this observation, two metaheuristic algorithms are developed. The first algorithm introduces the schemes of reinforcement learning and sampling into operator selection in a VNS framework (VNS-RLS). Eight improved neighbourhood operators with relatively small perturbation are developed, making a large contribution to the solution improvement. The second proposed algorithm (VD-ALNS) adopts the adaptive large neighbourhood search method for the larger perturbation during the search. A variable depth scheme associated to the specific solution structure of the PVRPTW-O is devised in VD-ALNS for better exploitation of solution space. Comparing with the existing scheduling scheme on the benchmark instances, this study reveals that the proposed PVRPTW-O model and the two solution techniques can significantly improve the fleet utilisation and cut costs for the company. This observation proves the feasibility of promoting the fleet efficiency without substantial hardware upgrade and greatly changing the whole existing scheduling system in EDI.

The findings on modelling and designing methodologies in this research bring helpful guidances for studying other theoretical and practical problems sharing similar features, e.g. the extended container transportation problem introduced in the next chapter. Some observations and managerial insights are concluded:

- In real-life scenarios, if the urgency level of task is considered in a construction heuristic, the performance and effectiveness of the heuristic would be mainly determined by the insertion tactic applied to optional tasks. When there are sufficient trucks, applying the *First-Feasible* tactic to optional tasks is of the fastest. When the number of tasks is extremely large, the heuristic of using *One-Route* tactic on mandatory tasks and *Greedy* tactic on optional tasks is recommended to produce better solutions within acceptable computation time.

- In single solution-based metaheuristics, the validity period of the guidance provided by Sampling is short, and Sampling is not as powerful as it is in

evolutionary approaches. However, the use of Sampling in local search may greatly reduce the number of infeasible solutions generated, while using the feasibility indicator in local search makes further contributions.

- When the neighbourhood structures used bring large perturbation in the search, variable depth scheme can be a good tool to handle the trade-off between exploration and exploitation. According to the specific features in problems, different neighbourhood depths and the associated changing scheme properly designed can strike a balance between the intensification and diversification of search.

In addition, further analysis is conducted to develop a deep understanding of this problem to explore some directions for further research. Some more analyses are presented as follows:

- In the Ningbo Port problem, one commodity may include multiple identical transportation tasks. The problem definition based on the task-based route representation facilitates the application and development of neighbourhood search approaches, however, the resulting many-to-one mapping from solutions to objective values may lead to poor performance search.

- The tasks real-time added are not considered in the current PVRPTW-O model, although there are cases when a few urgent requests submitted 24 hours before their deadlines in practice. If the online request submission is accepted, both the problem model and solution methods have to be modified to adapt to the online environment. Correspondingly, the uncertainty factor in customer request has to be taken into account as well, requiring more research on dynamic and stochastic problems.

- The objective function can be further extended. Although HLDR is widely used in practice, it does not reveal all the important information of solutions, e.g. the solutions with the same HLDR may be greatly different. In addition, HLDR does not sufficiently reflect the operational cost to the company, as it only considers the travel distance. Multi-objective models can be established for the Ningbo Port problem to further improve the company's management efficiency. This extension is done at the second stage research of the Ningbo Port problem in the next chapter.

The research on PVRTW-O conducts a preliminary attempt to develop efficient automatic container transportation routing and scheduling system for the Ningbo

Port problem. The experience and further study directions from this research motivate the following research in the next chapter.

Chapter 6

A Hyper-Heuristic with Two Guidance Indicators for MS-VRPTW

This chapter researches the solution methodologies for Mixed-Shift Vehicle Routing Problem with Time Windows (MS-VRPTW) derived from the Ningbo Port problem considering both the long-haul and the short-haul container transshipments. The problem mathematical model is introduced in Section 3.2. Based on the research on the PVRPTW-O, this problem is extended to a bi-objective problem to better describe the problem solution and cut down costs. The two objectives in MS-VRPTW are minimising the operational cost of driver salary (DP) and minimising the distance of unloaded travelling (TD). Instead of a problem-specific algorithm, this chapter aims to develop an efficient generic method for various multi-objective vehicle routing problems, and a hyper-heuristic algorithm is proposed accordingly.

This chapter is organised as follows: Section 6.1 discusses the exact search results for the MS-VRPTW, where the problem model is converted to be solved with CPLEX on a high performance computer. The experiment results indicate the need for massive computational resources for the exact search in the MS-VRPTW. Section 6.2 introduces an initial feasible solution construction heuristic considering the distinct task types and shift setting in the MS-VRPTW. The performance of this insertion-based construction heuristic is investigated. A generic algorithm for multi-objective VRP is presented in Section 6.3, which is a selective perturbation hyper-heuristic with five improved classic Low Level Heuristics (LLH). Two specific guidance indicators are developed dealing with the multiple objectives in the problem. The proposed algorithm is tested in the MS-VRPTW and the classic VRPTW, producing the promising results presented in Section 6.4. Section 6.5 summarises the contributions and findings of this chapter.

6.1 Exact Search for MS-VRPTW

In this study, exact search is first implemented to address MS-VRPTW by using CPLEX as well. To address this bi-objective MIP problem with CPLEX, the objectives of the mathematical model has to be slightly modified since CPLEX is not a tool for multi-objective models. To this end, three different configurations are employed to linearly combine the two objectives into one, see formula (6.1)). The configurations represent three scenarios in the modified objective: 1) *DP* has the same weight as *TD*, 2) *DP* dominates *TD* and 3) *TD* dominates *DP*. Considering the different ranges of *DP* and *TD*, the three configurations are $\{a = 200, b = 1\}$, $\{a = 10000, b = 1\}$ and $\{a = 4, b = 1\}$, respectively. In some research, this procedure is called *decomposition*, where the objective space is decomposed by sampling with diverse weighted sum of objective components [100]. In addition, those nonlinear constraints in the model need to be converted into the linear form before inputting them to adapt to CPLEX. This conversion is similar as that of in Section 5.1 and it does not change the problem complexity, thus, the details are not presented for saving space.

$$Minimize \qquad a \cdot DP + b \cdot TD \tag{6.1}$$

The CPLEX script of exact search was run on a high performance computer system. Considering the scale of this problem, a large number of computation resources have been assigned, which were 16 cores (2.6 GHz), 100 GB memory and 24 hours runtime limit for each experiment instance. However, the output of CPLEX shows that even with this large amounts of computation resources, it is still very hard to obtain satisfying solutions for MS-VRPTW with exact search methods. CPLEX was out of memory within 10 minutes in all the three configurations. This observation indicates that exact search is not realistic for solving this large-scale tightly constrained nonlinear problem due to massive computation resources required for computation time and memory. It is no doubt that there may exist exact methods which can work better than CPLEX in this problem, however, the requirement of extensive computation resource still remains. Therefore our studies focus on developing efficient approximate approaches for MS-VRPTW.

6.2 Initial Solution Construction Heuristic

Considering the similarity on problem background of the PVRPTW-O and the MS-VRPTW, the urgency level-based insertion heuristic (ULIH) introduced in Section

5.2 is extended to implement the initial feasible solution construction heuristic for MS-VRPTW, namely EBIH. The definitions of urgency level for the Long-Distance Task (LDT) and the Short-Distance Task (SDT) are inherited from ULIH. When a task i can be completed in shift s according to its time window, the task is either *optional* or *mandatory*. To be precise, if i can be completed in s and later shift(s), i is an *optional* task in shift s; otherwise, i is a *mandatory* task to s. So, to each shift, four sets of available tasks would be assigned, which are *mandatory LDT*, *optional LDT*, *mandatory SDT*, and *optional SDT*.

The four sets of tasks are considered in order in EBIH. It is easy to understand that we should assign *mandatory* tasks first. Because the delay of tasks may cause the containers missing the vessel appointed and greatly increase the operational cost of the company. Besides, SDT can be completed with both *one-driver* trucks and *double-driver* trucks while LDT can only use *double-driver* trucks, which means SDT have more insertion options than LDT when constructing a solution. Therefore, LDT is relatively harder to assign than SDT and should be assigned earlier. In EBIH, after arranging all mandatory tasks, if there still are available trucks in the fleet, optional tasks will be inserted to the current shift until all trucks are ran out. The order of task sets being assigned shift by shift is: *mandatory LDT* $\rightarrow$ *mandatory SDT* $\rightarrow$ *optional LDT* $\rightarrow$ *optional SDT*.

To keep the consistency in the scheduling system, the same insertion selection heuristics of Section 5.2 are followed here. In EBIH, *Greedy Strategy* is adopted for mandatory tasks. This setting aims to guarantee the urgent tasks' assignment first. On the other hand, to avoid long computation time, *First Feasible Strategy* is applied to the insertion of optional tasks. In addition, because the tasks with long service time are often too *big* to be inserted into the routes with existing tasks, the task with the longest service time will be selected as the first task in the newly created new route.

The performance of EBIH is tested on instances with diverse sizes and features. Table 6.1 presents the initial solutions produced by EBIH, obtained on a PC with i7-3820 3.60GHz CPU and 16.0 GB memory. Feasible solutions can be obtained within an acceptable time for all instances. The computation time of generating a solution grows rapidly along with the number of tasks in the instance. The highest requirement of truck happens on instance TU2-3, where 71 *one-driver* trucks and 171 *double-driver* trucks are used.

Table 6.1: Initial solutions produced by EBIH.

	LB2-1	LB2-2	LB2-3	LU2-1	LU2-2	LU2-3
DP	83.5	393	456.5	221	341.5	520.5
TD	22955.5	144926	140842.5	69096	93044	174560.5
time (s)	6	369	869	125	464	1176
	TB2-1	**TB2-2**	**TB2-3**	**TU2-1**	**TU2-2**	**TU2-3**
DP	147.5	315.5	484.5	252.5	374.5	554.5
TD	46032	98749	153719.5	71517.5	109143.5	145165
time (s)	29	215	1017	87	284	1407
	LB4-1	**LB4-2**	**LB4-3**	**LU4-1**	**LU4-2**	**LU4-3**
DP	131.5	382	547.5	294.5	475.5	671.5
TD	47028	113011	169075.5	101569	149.47	234230.5
time (s)	5	220	841	130	374	1711
	TB4-1	**TB4-2**	**TB4-3**	**TU4-1**	**TU4-2**	**TU4-3**
DP	289	411	579	349.5	533	569
TD	76690	125146.5	150932	105797.5	144385	188010.5
time (s)	52	229	1063	92	198	799

6.3 A Selective Perturbation Hyper-Heuristic with Two Guidance Indicators

To further reduce the operational cost of the company, based on the initial solution generated by EBIH, an improvement Hyper-Heuristic with Two Guidance Indicators (GIHH) is developed. GIHH is a selective perturbation hyper-heuristic, which selects perturbative low level heuristics (LLH) adaptively based on the change of problem scenario. Two guidance indicators are proposed to guide the selection of LLH. Considering the large scale and complex multi-level solution structure in MS-VRPTW, only one solution is updated in each algorithm iteration (single solution-based).

6.3.1 High-Level Heuristic

Algorithm 6 introduces the high level framework of GIHH. The input contains an initial feasible solution, a set of given LLH (H, introduced in subsection 6.3.3) and the stopping criterion. To this bi-objective problem, GIHH is a *Pareto Method* whose output is a solution archive ($ARCH$) consisting of *non-dominated* solutions. The small range of objective DP reduces the diversity of DP, leading to a relatively small number of non-dominated solutions. Thus, no limit is set to the size of $ARCH$, which means all non-dominated solutions found will be stored. In addition, to increase the diversification of the search, different solutions with the same objective values are stored in $ARCH$.

Algorithm 6 The GIHH framework

Step 1: Input an initial feasible solution (S) produced by EBIH, a set of LLH (H), Stopping Criterion (*NONIMP*).

Step 2. **Set up the initial parameters and *ARCH*.**

$Weight \leftarrow \{1, ..., 1\}$, $ScoreA \leftarrow \{0, ..., 0\}$, $ScoreB \leftarrow \{0, ..., 0\}$,$ARCH \leftarrow S$.

Step 3.

while Stopping Criterion is not met **do**

Step 3.1: **Solution Selection**

Randomly select a solution from $ARCH$ as the current solution S_c.

Step 3.2: **Low Level Heuristic Selection and Execution.**

Select a LLH (h_i) from H according to $weight_i$ with *Roulette Wheel Rule*, see Eq. (6.2); Execute h_i on the current solution, obtaining a new solution: $S' \leftarrow h_i(S_c)$

Step 3.3: **Accept or Reject (Hill Climbing).**

if S' is non-dominated in $ARCH$ **then**

Add S' into $ARCH$ and remove all dominated solutions.

else

Reject S'.

end if

Update $ScoreA_i$ and $ScoreB_i$, recording the contribution of h_i to the solution improvement. More details are presented in Section 6.3.2.

Step 3.4: **Weight Adjustment.**

After a predefined number (*SEG*) of iterations, *Weight* of H is updated according to *ScoreA* and *ScoreB*. Details are introduced in Section 6.3.2.

end while

Step 4: Output a solution set $ARCH$.

$$Pr_i = \frac{weight_i}{\sum_{j \in H} weight_j} \qquad \forall i \in H \tag{6.2}$$

In each iteration, one LLH is chosen and applied to a chosen solution (S_c), generating an updated solution. During the loop, to diversify the search, S_c is randomly selected from $ARCH$ in ***Step 3.1***. The stopping criterion is set as when $ARCH$ is not being updated in a predefined number (*NONIMP*) of iterations.

In GIHH, three scalars (*Weight*, *ScoreA* and *ScoreB*) are defined to guide the selection of LLH, generating better problem solutions. The LLH executed in an iteration is chosen with the *Roulette Wheel Rule* (***Step 3.2***). To avoid the probabilities of LLH converging to zero and the corresponding LLH never being called at all, a minimal probability limit of 5% is applied to every LLH. *ScoreA* and *ScoreB* are two guidance indicators, which record the performance of LLH in previous search history from two different aspects respectively. *Weight* is updated based on *ScoreA* and *ScoreB*. All these three scalars are adjusted adaptively during search (in ***Step 3.3*** and ***3.4***), details in Section 6.3.2.

Because the ranges of the two objectives in MS-VRPTW are significantly different, that is, the range of DP is remarkably smaller than that of TD, a small change on DP is usually accompanied by a great fluctuation on TD in a solution.

To further investigate this issue, in addition to the Hill Climbing acceptance criteria, a *Record-to-Record Travel* (RRT) [96] acceptance criterion is also implemented in our study. RRT accepts the worse solutions (S') of deteriorated quality from the current solution (S_c) in a predefined range. The comparison of experiment results are presented in Section 6.4.

6.3.2 Guidance Indicators and Weight Adjustment Scheme

$ScoreA_i$ stores the accumulated rewards to h_i according to the change of objective values from S_c to S', recording the performance of h_i on improving solution quality. In each iteration, if S' is acceptable, a reward 1 is added to $ScoreA_i$, otherwise no reward is added. Therefore, a larger $ScoreA_i$ represents a greater contribution of h_i to generating new non-dominated solutions. This indicator emphasises LLH's contribution on solution quality improvement.

$ScoreB_i$ is a specially designed indicator for this bi-objective problem, which indicates which objective h_i inclines to improve (improvement direction). In MS-VRPTW, a Pareto Solution Set with uniform distribution and good convergence on the Pareto Front is expected, instead of the solutions within local regions. During the search, the improvement on both of the two objectives is pursued. When updating $ScoreB_i$, the objective values of S_c and S' are compared. If S' is better than S_c on *DP*, $ScoreB_i$ is increased by one; If S' is better than S_c on *TD*, $ScoreB_i$ is decreased by one. A positive $ScoreB_i$, thus, means the inclination of improving *DP* (generated more improved solutions on *DP*) to h_i, while a negative one indicates that of improving *TD*.

$Weight_i$ is updated once in every *SEG* iterations (called a *Segment*) to avoid over-fitting. It is adjusted according to the feedback from the search history ($ScoreA_i$ and $ScoreB_i$). The update is a two-phase procedure. The first phase is guided by $ScoreA_i$, see eq. (6.3).

$$weight_i^t = \alpha \cdot weight_i^{t-1} + \beta \cdot \frac{ScoreA_i}{Applied\ Times\ of\ h_i} \tag{6.3}$$

In the second update phase, to find the improvement *DEV* (eq. (6.4)) between the two objectives, the newly generated non-dominated solutions are compared with the first S_c in the last Segment, obtaining the number of the non-dominated solutions with improved *DP* (*DP_IMP*) and that of improved *TD* (*TD_IMP*). If *DP* was improved more times in the last Segment ($DEV > 0$), then the $weight_i$ of those LLH with *TD* inclination should be increased by using eq. (6.5), obtaining a higher

probability being selected in the current Segment. The similar operations are made when $DEV < 0$. This procedure aims to balance the improvement direction.

$$DEV = \frac{DP_IMP - TD_IMP}{(DP_IMP + TD_IMP) \cdot 0.5} \tag{6.4}$$

$$\begin{gathered} weight_i^t \mathrel{+}= \gamma \cdot \frac{ScoreB_i}{Applied\ Times\ of\ h_i} \cdot DEV \\ when\ (DEV < 0\ And\ ScoreB_i < 0)\ Or\ (DEV > 0\ And\ ScoreB_i > 0) \end{gathered} \tag{6.5}$$

The three coefficients (α, β, γ) in eqs. (6.3) and (6.5) determine the response speed to the search feedback and the influence of each guidance component on updating $weight_i$, subject to $\alpha + \beta + \gamma = 1$.

6.3.3 Low-Level Heuristics

Five LLH are adopted in GIHH. Each LLH changes the current solution to a certain extent, obtaining updated solutions. Heuristics with large changes perturb the operated solution dramatically. They increase the search diversity and avoid trapping to local optimum, but longer computation time is needed usually to produce a new feasible solution. Heuristics with small changes use relatively less computation time in each iteration, however, their common deficits are easy to stuck to local optimum and premature search. Previous research shows that properly combining heuristics can improve the performance of search [59].

- ***Inter-Route 2-opt****. Lin [201] proposes the λ-*opt* route improvement heuristic which removes and reconnects λ edges in a route. This classic heuristic brings relatively small changes to a solution, obtaining good results in various VRPs. Potvin and Rousseau [249] develop an improved *2-opt* heuristic (*2-Opt**) which keeps the direction of each route segment during reconnection. This heuristic is devised for *Traveling Salesman Problem* at first, but shows excellent performance in various routing problems with time windows. In GIHH, *Inter-Route 2-Opt** removes two edges from different routes and reconnects them while keeping the directions of associated route segments. Notice that the edges modified can be the starting or ending points of routes, which means two routes being connected into one route is possible.

- ***Inter-Route CROSS-exchange***. Taillard et al. [286] propose *CROSS-exchange* which swaps two route segments from two different routes while keeping their directions. This heuristic brings relatively small perturbation

as well. The length of route segment can be zero, e.g. when one of the two operated route segments is empty, the execution of *Inter-Route CROSS-exchange* actually relocates a route segment from one route to another route.

- ***Intra-Route CROSS-exchange***. In this heuristic, the swapping strategy of *CROSS-exchange* is applied to one single route.

- ***Large Neighbourhood Search (LNS)***. In GIHH, Random Selection is used in the *Destroy* heuristic of LNS to remove q randomly chosen tasks. Then the removed tasks are reinserted into the destroyed solution using a greedy *Repair* heuristic. This heuristic always executes the insertion causing the least increase on the travel distance. Obviously, comparing all possible insertion positions for all the q tasks is time-consuming. To balance the solution quality and the computation time, the value of q is defined as $min\{5\% \cdot n, 10\}$, where n is the total number of tasks.

- ***Guided Ejection Search (GES)***. To further reduce the number of trucks used and optimise *DP*, GES is employed in GIHH. The main ideas of GES have been summarised in subsection 2.2.3.2. Using LNS and GES obtains larger change to solutions and greater perturbation in search, at the cost of longer execution time.

6.4 Experiments and Results Analysis

6.4.1 Parameter Setting and Complexity Discussion

GIHH adaptively employs LLH according to the search, with relatively few parameters to set. The parameters are tuned one by one, while the others are fixed.

In eqs. (6.3) and (6.5), a large α means a low response speed to the change in the search space, often leading to slow convergence. However, high-quality solutions may be skipped over when the response speed is too high. On the other hand, high response speed usually leads to premature convergence. Our preliminary experiments show that, the setting of $\alpha = 0.5$ makes a good trade-off between convergence speed and solution quality. The values of β and γ determine the influence of the two guidance indicators to update $weight_i$. The setting of $\beta = 0.4, \gamma = 0.1$ is adopted based on preliminary experiments, indicating that $ScoreA$ has a greater influence than $ScoreB$ in GIHH.

When updating $weight_i$, a smaller *SEG* would change $weight_i$ more frequently, while when *SEG* is too large, the feedbacks cannot effect in time. *SEG* is set to 80

in GIHH empirically. In addition, $NONIMP = 150$ is used as the stopping criterion to strike a balance between the computation time and the quality of results.

When assessing the computational complexity of metaheuristics and hyper-heuristics, time complexity cannot be determined since these approximate algorithms do not guarantee finding the global optimal solution within a given time limit. When the algorithm procedure would terminate depends on the applied problem and specific definition of its stopping criterion (e.g. the definition of *NONIMP* in GIHH). Therefore, the CPU time and objective function evaluations on benchmark are often used to compare the computational complexity of approximate methods in research. In this study, the algorithms with the above parameter setting are compared from the aspects of computational time and iterations at high level, and the results and associated analysis are presented in the next subsection. As only one solution is updated in each iteration, with the task node-based solution representation, the space complexity of GIHH is $O(K \cdot |S| \cdot n)$, where K is the fleet size, $|S|$ is the length of the planning horizon and n is the number of tasks to be assigned.

6.4.2 Impacts of the Guidance Indicators

To evaluate the influence of the two proposed guidance indicators in GIHH, two variants (GIHH-A and RHH) of GIHH with different guidance indicator settings are developed for comparison. In GIHH-A, only $ScoreA$ is adopted, while in RHH, LLH are randomly chosen without any guidance. Our preliminary experiments show that increasing the computation time does not improve the results significantly, so all the three algorithms use the same stopping criterion.

Table 6.2 presents the comparison of GIHH, GIHH-A and RHH. All the results are obtained in 20 runs. In the literature, to compare the performance of Pareto Methods, various quality indicators are proposed. Most of them focus on the comparison on the Pareto Set approximation [198]. One of the most widely used indicators is *Hyper-Volume*, which considers the convergence, uniformity and spread over the Pareto Front produced. Previous studies have shown that a Pareto Set with a larger hyper-volume is likely to have a better trade-off among multiple objectives [41]. To compare the three algorithm variants, the hyper-volumes of the $ARCH$s obtained are calculated and presented in Table 6.2. In our study, the *reference points* used in calculating hyper-volume are the initial feasible solutions generated by EBIH. It can be found that, comparing the three algorithms from multiple aspects, most of the best results are produced by GIHH.

Table 6.2: Results comparison among algorithms with different indicator configurations. **Ave HV** and **Best HV** are the average and best hyper-volumes, respectively. **Best DP** and **Best TD** are the best found objective values, while **S.D.** is standard deviations. **Iteration** is the average iterations in the 20 runs. Best results are in bold.

		LB2-1	LB2-2	LB2-3	LU2-1	LU2-2	LU2-3	TB2-1	TB2-2
RHH	Iteration	1479	4831	5557	4096	5782	9012	2459	4919
	Ave HV	0.8176	1.0964	0.6745	0.6652	0.4826	0.8086	0.2598	0.4856
	Best HV	0.8966	1.1303	0.6936	0.6924	0.5472	0.8504	0.2941	0.5132
	Best DP	**53**	**247**	**336**	**166**	**256.5**	**369.5**	131	**224.5**
	Best TD	11685	53531.5	73506	34004	57693	76342.5	30498	68167.5
	S.D.	4.32%	2.63%	**1.03%**	2.14%	2.90%	2.20%	1.65%	1.98%
GIHH-A	Iteration	1322	4911	3701	3121	5441	9320	2973	4945
	Ave HV	0.8456	1.1277	0.6943	0.6842	**0.5299**	**0.8632**	0.2754	0.4950
	Best HV	0.8933	1.1424	**0.7201**	0.7093	**0.5575**	**0.8837**	**0.3044**	0.5259
	Best DP	53.5	250	339	166.5	259	371	**130.5**	226
	Best TD	11491.5	50565	68909	**32365**	**54509**	**71227**	**29811**	**66195**
	S.D.	3.08%	1.21%	1.70%	2.80%	**1.73%**	1.39%	1.66%	2.11%
GIHH	Iteration	1538	4586	5125	3486	4732	7958	2837	6516
	Ave HV	**0.8578**	**1.1321**	**0.6967**	**0.6914**	0.5134	0.8539	**0.2792**	**0.5036**
	Best HV	**0.9203**	**1.1624**	0.7182	**0.7234**	0.5516	0.8771	0.3000	**0.5275**
	Best DP	**53**	248.5	339	**166**	261	372	131	227.5
	Best TD	**11332**	**50382.5**	**67728**	32409	55549	71609.5	29961	66208.5
	S.D.	**3.00%**	**0.85%**	1.51%	**1.30%**	1.86%	**1.34%**	**1.31%**	1.42%

		TB2-3	TU2-1	TU2-2	TU2-3	LB4-1	LB4-2	LB4-3	LU4-1
RHH	Iteration	6763	2821	5170	7334	1648	4827	8474	2880
	Ave HV	0.5908	0.4493	0.6217	0.4574	0.6443	0.7181	0.8271	0.3262
	Best HV	0.6146	0.4710	0.6385	0.4888	**0.6823**	0.7507	0.8560	0.3574
	Best DP	**357.5**	183	268	**430.5**	**91**	**268**	374.5	237.5
	Best TD	88763.5	51263.5	64919	92812	27610.5	58122	80004.5	74178
	S.D.	2.02%	1.42%	1.60%	2.09%	2.08%	**1.66%**	**2.20%**	**1.34%**
GIHH-A	Iteration	6228	2800	4933	9713	1389	4199	6691	2257
	Ave HV	**0.6005**	**0.4649**	0.6463	**0.5357**	0.6429	0.7429	0.8360	0.3407
	Best HV	**0.6305**	0.4824	0.6531	**0.5396**	0.6665	0.7629	**0.8784**	0.3625
	Best DP	364.5	**181**	268	432.5	93	269	**374**	236
	Best TD	**84585**	50878	62694	**85413.5**	27465	57612	75555	73566
	S.D.	**1.66%**	1.29%	**0.52%**	**0.40%**	**1.57%**	1.87%	3.01%	1.70%
GIHH	Iteration	5360	2592	4975	8695	1371	3883	6009	3071
	Ave HV	0.5972	0.4622	**0.6480**	0.5176	**0.6517**	**0.7495**	**0.8362**	**0.3423**
	Best HV	0.6203	**0.4863**	**0.6700**	0.5333	0.6728	**0.7747**	0.8692	**0.3838**
	Best DP	364	**181**	**267**	431	92	**268**	380	**233.5**
	Best TD	85127	**50544**	**62691**	85813.5	**27409**	**57437**	**75285**	**73073**
	S.D.	1.67%	**1.21%**	1.10%	1.16%	1.72%	2.16%	2.25%	2.05%

		LU4-2	LU4-3	TB4-1	TB4-2	TB4-3	TU4-1	TU4-2	TU4-3
RHH	Iteration	5032	8915	1180	3807	7649	2349	3508	7349
	Ave HV	0.7547	0.9387	0.3778	0.3634	0.6021	0.2371	0.3648	0.6134
	Best HV	0.7824	0.9765	0.3885	0.3831	0.6357	0.2454	0.3800	0.6334
	Best DP	**321.5**	**476.5**	226	**323.5**	**409**	287	397	**405.5**
	Best TD	80535.5	75804	55991	90690.5	92126	88214	111090	112465
	S.D.	1.52%	2.57%	0.78%	1.06%	1.95%	0.50%	1.22%	1.71%
GIHH-A	Iteration	3029	6789	921	2817	5487	2229	2386	4919
	Ave HV	0.7659	0.9759	0.3770	0.3761	0.6247	**0.2577**	0.3743	0.6239
	Best HV	0.7962	0.9891	0.3909	0.3871	0.6643	**0.2620**	0.3888	0.6470
	Best DP	327.5	484.5	**225**	325.5	**409**	**284.5**	398	412
	Best TD	**76664**	71859	55920	89808.5	88744	86548	**110743**	**106921**
	S.D.	2.18%	**1.00%**	0.78%	**0.54%**	1.97%	**0.32%**	0.85%	1.80%
GIHH	Iteration	3701	5341	1131	3098	5745	2122	2871	5055
	Ave HV	**0.7774**	**0.9768**	**0.3865**	**0.3795**	**0.6343**	0.2554	**0.3756**	**0.6246**
	Best HV	**0.8050**	**0.9997**	**0.3974**	**0.3981**	**0.6674**	0.2613	**0.3938**	**0.6480**
	Best DP	322.5	480	**224.5**	**323.5**	412	285.5	**393.5**	407
	Best TD	**75138**	**70811.5**	**55606**	**89397**	**86690**	**86482**	110926	107892
	S.D.	**1.50%**	1.38%	**0.73%**	0.57%	1.82%	0.37%	**0.74%**	**1.34%**

Among the three variants, RHH produced the worst hyper-volumes with the most iterations, while its standard deviation obtained is the largest. This shows that, when the High-Level Heuristic is random selection with no guidance, the algorithm would take more iterations to converge with a lower stability. However, it may have a higher probability of finding better solutions against objective (3.20), i.e. with the best *DP*.

It can be found that from Table 6.2, GIHH-A and GIHH obtained remarkably better solutions (higher **HV**) than RHH. Using $ScoreA$ significantly improves the quality of the produced solution set. Generally, GIHH-A and GIHH used less iterations but longer computation time to obtain the output. This can be observed in Figure. 6.1. GIHH-A and GIHH may have less average iterations than RHH (blue columns), but their computation time (red crosses) are longer on all the eight sample instances. Because the unit computation time of LNS and GES are significantly longer than the other LLH, this observation indicates that, compared to RHH, GIHH-A and GIHH employed these two LLH with greater perturbation more frequently during the search.

Between GIHH-A and GIHH, the latter obtained a higher average and the best hyper-volume on most instances with the guidance of $ScoreB$, while no obvious increase on iteration time and computation time is found. This can also be observed from Figure. 6.1. GIHH promotes the overall search performance and stability with the help of the two proposed guidance indicators.

With regard to the features of instances, *Loose* instances have broader time windows than *Tight* instances, which means more scheduling options and larger solution space. Thus, when the sizes of instances are similar, the *Loose* instances require more iterations and computation time to converge in all the three algorithms. In addition, comparing the iteration time, GIHH-A and GIHH work better on *Loose* instances, see Figure. 6.2. It can be found that, compared to RHH, the reduction of iterations is higher on *Loose* instances than on *Tight* instances, except GIHH on the LB4 instances. When the feature of throughput imbalance at terminals changes, no obvious difference is found.

Note that, in the *ARCH* generated by GIHH, each non-dominated point on the Pareto Front may have 20-40 different solutions on average. The number of different solutions with the same objectives stored does not effect the value of hyper-volume. Experiment results show that storing different solutions with the same objective values does not significantly increase the hyper-volume of a solution archive, but it boosts the diversification of the solution set. Those solutions provide logistics company coordinator more reference solutions.



Figure 6.1: The iteration times and computation time of the three algorithms on eight sample instances.

6.4.3 Impacts of Solution Selection and Acceptance Criterion

In each iteration of GIHH, the solution to be operated (S_c) is randomly selected from *ARCH*, aiming to increase the diversity in the search. To justify the function of the random selection scheme, an algorithm with deterministic selection of S_c (named GIHH-D) is also implemented in our research. With this deterministic

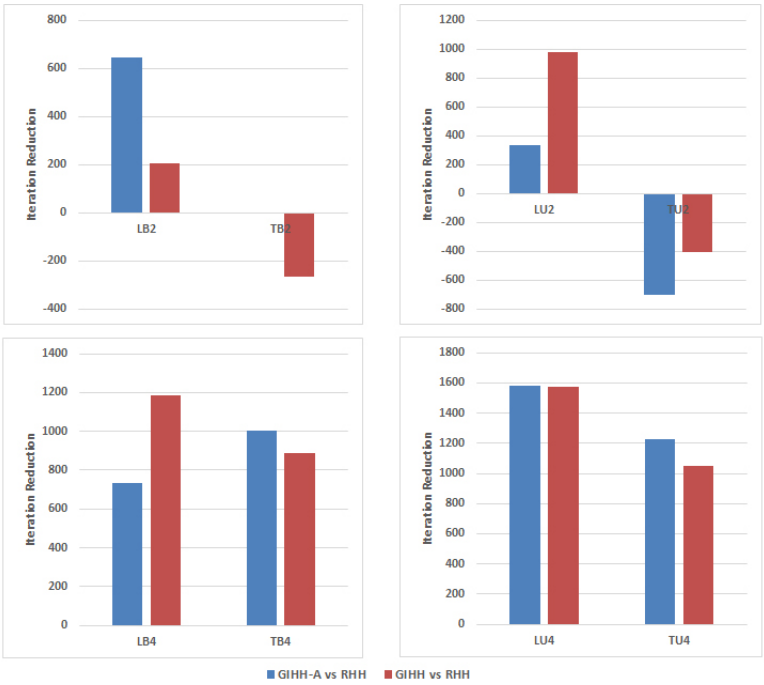


Figure 6.2: Comparison of the iteration time reduction. The bars indicate the number of reduction of iterations. Longer bars represent greater reduction, while the negative values indicate more iterations than RHH.

scheme, in *ARCH*, the solution farthest from the reference point will be selected as S_c. Because all solutions are derived from the initial solution (reference point), this deterministic scheme means that the solution with the highest improvement on both objectives will be selected. T-test is conducted on the output of GIHH and GIHH-D. The results are presented in Table 6.3.

In addition, as mentioned in Section 6.3.1, another variant adopting the Record-to-Record Travel acceptance criterion (GIHH-RRT) is also compared with GIHH. In GIHH-RRT, comparing to S_c, a worse solution would be accepted as long as the deterioration of objective value is less than $0.01 \cdot TD(S_c)$ on TD and less than 1.5 on DP. Acceptance criterion in a perturbative algorithm should balance the diversification and intensification of search, while RRT can increase the diversification of search greatly. Its output is compared with that of *Hill Climbing* criterion presented in Table 6.3.

From Table 6.3, it can be found that GIHH outperforms the other two algorithms. On the one hand, using the deterministic scheme to select the solution to be updated (GIHH-D) decreases the diversity of search, leading to significantly worse output than GIHH on most instances (19/24). On the other hand, accepting worse solutions (GIHH-RRT) does not improve the final search result on all instances. As the two objectives have remarkably different ranges, accepting worse

Table 6.3: T-test results with GIHH. Y means GIHH generates significantly better solutions, while N represents it does not.

	LB2-1	**LB2-2**	**LB2-3**	**LU2-1**	**LU2-2**	**LU2-3**	**TB2-1**	**TB2-2**	**TB2-3**	**TU2-1**	**TU2-2**	**TU2-3**
GIHH-D vs GIHH	Y	Y	Y	Y	Y	N	N	Y	Y	Y	Y	Y
GIHH-RRT vs GIHH	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y
	LB4-1	**LB4-2**	**LB4-3**	**LU4-1**	**LU4-2**	**LU4-3**	**TB4-1**	**TB4-2**	**TB4-3**	**TU4-1**	**TU4-2**	**TU4-3**
GIHH-D vs GIHH	N	Y	Y	N	Y	Y	N	Y	Y	Y	Y	Y
GIHH-RRT vs GIHH	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y

solutions would bring great fluctuation and deterioration to S_c in the search. This observation indicates that, in MOVRP, when the difference in the ranges of objectives is big, accepting solutions of lower quality does not improve the search. Besides, our experiments also show that GIHH is more stable than the other two algorithms with smaller standard deviations.

6.4.4 Comparison with State-of-the-art Algorithms

MS-VRPTW is a newly introduced model in the literature, there is thus no existing algorithm applied to it yet. Three state-of-the-art algorithms (RVNS [244], FVNS [150] and ALNS [261, 212]) are adopted and applied to MS-VRPTW in our study. Both RVNS and FVNS use the Variable Neighbourhood Search framework and produce the best solutions in PVRP. Apart from the different neighbourhood structures used, a main difference between them is that the order of shaking operators employed is fixed in FVNS, while they are randomly selected in RVNS. ALNS produces the best results for VRPPD with Adaptive Large Neighbourhood Search. The experiments show that GIHH outperforms the three algorithms on both solution quality and computation time in MS-VPRTW, especially on larger instances. Their result deterioration is presented in Table 6.4.

Possible causes for these results include the following. Firstly, the neighbourhood structure employed in GIHH are highly effective. FVNS and RVNS only use the small perturbation neighbourhood operators (e.g. λ-opt, CROSS, relocation). With these smaller neighborhood structures, it is hard or needs long time to escape from the local optimum in this nonlinear constrained problem. On average, 65% more computation time is required by FVNS and RVNS comparing to GIHH. Large perturbation operators are used in ALNS but are lacking of intensive exploitation. Secondly, without the guidance of specific indicators, e.g. $ScoreB$, the solutions generated are more likely to cluster, leading to a low hyper-volume. In addition, the three algorithms compared are problem specific metaheuristics. Different from hyper-heuristics, their performance may decline drastically for different instances

Table 6.4: Solution deterioration comparing with the results of GIHH. The values in table are the objective differences divided by the GIHH objective values.

		Average HV	Best HV	Best DP	Best TD	S.D.
FVNS	LB2	-4.86%	-4.76%	-3.19%	-0.88%	-0.54%
	LU2	2.44%	1.80%	-0.27%	1.13%	0.34%
	TB2	-15.13%	-15.90%	-3.39%	-0.40%	0.31%
	TU2	-9.25%	-9.75%	-2.76%	-2.01%	0.04%
	LB4	-5.70%	-5.82%	-2.29%	-1.04%	0.16%
	LU4	-1.30%	-2.59%	-1.19%	-1.22%	-0.67%
	TB4	-9.85%	-7.51%	-1.14%	-2.42%	-0.63%
	TU4	-13.85%	-12.14%	-3.50%	-0.72%	-0.30%
RVNS	LB2	-13.14%	-16.50%	-5.76%	-12.41%	-4.17%
	LU2	-10.27%	-12.93%	-2.31%	-8.50%	-9.05%
	TB2	-19.36%	-24.08%	-4.90%	-5.67%	-1.03%
	TU2	-14.92%	-18.46%	-5.63%	-6.04%	-2.52%
	LB4	-22.80%	-15.06%	-5.09%	-8.64%	-3.73%
	LU4	-29.90%	-23.98%	-4.49%	-11.38%	-4.40%
	TB4	-30.88%	-19.17%	-2.67%	-7.42%	-2.29%
	TU4	-27.02%	-21.87%	-3.99%	-4.61%	-1.04%
ALNS	LB2	-5.48%	-5.33%	-0.07%	-7.97%	0.22%
	LU2	-8.82%	-6.97%	1.17%	-9.59%	-0.48%
	TB2	-6.72%	-7.31%	0.57%	-3.77%	0.33%
	TU2	-8.47%	-7.69%	-1.03%	-4.68%	-0.19%
	LB4	-5.70%	-5.03%	-1.28%	-3.38%	0.59%
	LU4	-7.09%	-10.43%	-1.05%	-5.22%	0.39%
	TB4	-4.64%	-4.59%	-0.11%	-2.29%	0.08%
	TU4	-18.55%	-17.37%	-2.28%	-4.20%	-0.23%

even in the same problem. For example, both FVNS and ALNS obtain better results than GIHH on LU2 instances.

An observation from the results of FVNS and RVNS is that, they both produce many more solutions with the same objective values than GIHH. The small perturbation operators tend to generate a large number of solutions with small differences but of the same objective values in the solution archive. Comparing FVNS and RVNS, the former performs better in MS-VRPTW with a fixed order of the neighbourhood operators of low perturbation to high perturbation. ALNS outperforms FVNS and RVNS on the objective DP with the help of large perturbation, while has a higher stability than GIHH.

6.4.5 Results on VRPTW Benchmarks

To evaluate its performance in other problems, GIHH is applied to classic VRPTW on the Solomon Benchmarks [282]. The VRPTW is the basis of many other complex VRPs, while the Solomon Benchmarks have been extended and adopted in the research of many other VRP variants as well. An equal priority is given to the two objectives, the number of vehicles used (NV) and the total travel distance (TD), in the VPRTW model of our study. The results obtained are compared with the best known solutions to date, see Table 6.5. It can be found that, GIHH obtains solutions the same as or close to the best known solutions (which are optimal

actually) on the instances with clustered customers (C1 and C2). On the randomly and mixed distributed instances (R1, R2 and RC1, RC2), GIHH produces solutions close to the best known ones, and nine new non-dominated solutions are found. Considering that most of those best known solutions are generated by customised problem-specific algorithms with sufficient computation resource, the results of GIHH are satisfying.

Table 6.5: Results of GIHH on the Solomon's instances. The solutions equal to best known results and the newly found non-dominated solutions are shown in bold.

Instance	Best Known			GIHH	
				Best Found	
	NV	TD	Ref.	NV	TD
C101	10	828.94	[280]	**10**	**828.94**
C102	10	828.94	[280]	**10**	**828.94**
C103	10	828.06	[280]	10	828.94
C104	10	824.78	[280]	10	825.65
C105	10	828.94	[280]	**10**	**828.94**
C106	10	828.94	[280]	**10**	**828.94**
C107	10	828.94	[280]	**10**	**828.94**
C108	10	828.94	[280]	**10**	**828.94**
C109	10	828.94	[280]	**10**	**828.94**
C201	3	591.56	[280]	**3**	**591.56**
C202	3	591.56	[280]	**3**	**591.56**
C203	3	591.17	[280]	**3**	**591.17**
C204	3	590.6	[280]	**3**	**590.6**
C205	3	588.88	[280]	**3**	**588.88**
C206	3	588.49	[280]	**3**	**588.49**
C207	3	588.29	[280]	**3**	**588.29**
C208	3	588.32	[280]	**3**	**588.32**
R101	19	1650.80	[280]	**19**	**1650.80**
	20	1642.87	[5]	20	1643.34
R102	17	1486.12	[280]	17	1489.33
	18	1476.06	[59]	18	1490.72
R103	13	1292.67	[280]	13	1367.27
				14	**1219.89**
R104	9	1007.31	[280]		
	10	974.24	[288]	10	1007.27
	11	971.5	[178]	11	994.85
R105	14	1377.11	[280]	14	1381.88
	15	1346.12	[167]	15	1360.78
R106	12	1252.03	[280]	12	1270.45
	13	1234.6	[72]	13	1243.72
R107	10	1104.66	[280]		
	11	1051.84	[167]	11	1077.24
				12	**1050.06**
R108	9	960.88	[280]		
	10	932.1	[230]	10	956.22
R109	11	1194.73	[280]		
	12	1013.2	[62]	12	1168.18
	13	1151.84	[5]	13	1157.61
R110	10	1118.84	[280]		
	11	1112.21	[230]	11	1153.83
	12	1068	[72]	12	1081.88
R111	10	1096.72	[280]	**11**	**1087.5**
	12	1048.7	[72]	12	1062.58
R112	9	982.14	[280]		
	10	953.63	[260]	10	958.7
R201	4	1252.37	[280]	4	1282.75
	5	1190.52	[59]		
R202	3	1191.7	[280]	3	1239.82
	4	1091.21	[288]	4	1098.06
R203	3	939.503	[280]	3	968.67
	4	905.72	[59]	4	935.55
R204	2	825.52	[280]		
	3	766.91	[59]	3	767.52
R205	3	994.42	[280]	3	1059.91
	5	954.16	[5]	**4**	**964.02**
R206	3	906.142	[280]	3	930.80
R207	2	890.61	[280]		
	3	814.78	[260]	3	843.88
R208	2	726.82	[280]	2	741.75
	4	698.88	[300]	**3**	**708.9**

(continued)

Instance	Best Known			GIHH	
				Best Found	
	NV	TD	Ref.	NV	TD
R209	3	909.16	[280]	3	962.08
	5	860.11	[5]	4	871.63
R210	3	939.37	[280]	3	978.11
	4	935.01	[59]	4	948.95
R211	2	885.71	[280]		
	3	794.04	[59]	3	804.16
	4	761.1	[230]		
RC101	14	1696.94	[280]		
	15	1619.8	[300]	15	1633.10
RC102	12	1554.75	[280]		
	13	1470.26	[288]	13	1497.43
	14	1466.84	[5]	14	1467.25
RC103	11	1261.67	[280]	11	1265.86
RC104	10	1135.48	[280]	10	1136.49
RC105	13	1629.44	[280]		
	14	1589.91	[288]	14	1623.54
	15	1513.7	[5]	15	1524.14
RC106	11	1424.73	[280]	**12**	**1396.59**
	13	1371.69	[288]	13	1376.99
RC107	11	1230.48	[280]	11	1254.68
	12	1212.83	[5]	12	1233.58
RC108	10	1139.82	[280]	10	1200.69
	11	1117.53	[5]	11	1131.23
RC201	4	1406.94	[280]	4	1457.87
	6	1134.91	[288]	**5**	**1310.44**
RC202	3	1365.64	[280]	3	1546.3
	4	1181.99	[230]	4	1192.54
RC203	3	1049.62	[280]	3	1097.32
	4	957.10	[59]		
RC204	3	798.46	[280]	3	829.13
RC205	4	1297.65	[280]	4	1298.90
	5	1233.46	[59]	5	1240.45
RC206	3	1146.32	[280]	3	1156.06
	4	1107.40	[59]	**4**	**1107.19**
RC207	3	1061.14	[280]	3	1135.61
	4	1032.78	[59]	4	1033.78
RC208	3	828.14	[280]	3	830.06

6.5 Summary

This chapter investigates the solution methodologies for the bi-objective Mixed-Shift Vehicle Routing Problem with Time Windows (MS-VRPTW), which arises from the container transportation problem between short-distance and long-distance terminals faced by the Ningbo Port company. Due to the large difference in the service time of the transportation tasks, two types of shifts (*long-shift* and *short-shift*) with different operational costs are considered in this problem. The investigation shows that it is unrealistic to tackle MS-VRPTW with exact search approaches even if a huge amount of computation resources is given. A hyper-heuristic is thus developed for MS-VRPTW. It is capable of efficiently handling the container transportation over the terminals of inland dry ports and seashore docks in the Ningbo Port problem, showed to increase the utilisation rate of trucks and reduce the operational cost of the logistics company.

In the proposed method, firstly, an initial feasible solution is generated using an extended urgency level-based insertion construction heuristic (EBIH). Then, a Hyper-Heuristic with two Guidance Indicators (GIHH) is implemented to improve the solutions, while a Pareto solution set is kept. GIHH is a selection perturbation hyper-heuristic, adapting a set of Low-Level Heuristics (LLH) with different extents of perturbation to the problem solution. The two indicators are proposed to guide

the LLH selection adaptively. They evaluate LLH's contribution to the solution quality improvement and the improvement direction, respectively. To test the generality and performance, the proposed algorithms are tested on the benchmark instances of the MS-VRPTW and the VRPTW. On all the benchmark instances of MS-VRPTW, EBIH produced feasible solutions within an acceptable time. The experiment results also show that:

- In different searching environments, the two proposed guidance indicators in GIHH significantly improve the performance and stability of search for the bi-objective problem, producing solutions with higher hyper-volumes.
- In terms of the acceptance criterion, it is shown that, when the ranges of objectives are vastly different in the multi-objective vehicle routing problem, the *Hill Climbing* acceptance criterion outperforms the acceptance criterion of accepting worse solutions (*Record-to-Record Travel*).
- With regard to the selection scheme of solution, the research finds that, in the MS-VRPTW, randomly selecting the next current solution from the solution population can increase the diversity of search, bringing better results than deterministic selection.
- GIHH outperforms three state-of-the-art algorithms for PVRP and VRPPD on both the computation time and the quality of solutions generated. Compared to the best known solutions to date, GIHH also produces promising results in the classic VRPTW.

The MS-VRPTW model and the associated solution method bring efficient automatic routing and scheduling for the Ningbo Port problem. As a generic automatic scheduling method, GIHH is promising to other multi-objective VRPs. With slight modifications (e.g. different definitions of artificial nodes) the MS-VRPTW model can be easily extended to other mixed-shift problems, and the proposed algorithms (EBIH, GIHH) can also be applied to more practical, complex, multi-objective optimisation problems for their high stability in diverse scenarios. However, the experiment results on some instances of the VRPTW benchmark still have obvious gaps to the best-known solutions on either TD or NV. The promotion of search intensification in GIHH deserves deeper investigation in future research.

Chapter 7

Conclusions and Future Research

7.1 Research Summary and Conclusions

This thesis has studied the optimisation models and algorithms for transportation routing and scheduling problems, aiming to address a container transshipment problem faced by a port company in Ningbo, China. In this problem, the optimal truck routing and driver scheduling are pursued to accommodate the commodity transshipment requests with a limited number of trucks and satisfying the time constraints. The planning horizon consists of multiple shifts, while the time windows of servicing the requests vary from hours to several shifts. The current scheduling system (EDI) used by the company shows low efficiency when facing a large number of transshipment requests in busy seasons. Thus, this study aims to establish a feasible phased scheme for upgrading EDI, providing an efficient intelligent task assignment system.

The transshipment requests in the Ningbo Port problem can be divided into two classes of short-haul and long-haul. The short-haul requests are embodied as the container transportation tasks among nine relatively close port districts, located on the east coast of Ningbo City. The long-haul requests consider the long-distance container transportation related to the inland dry ports. Two independent scheduling systems were set up in EDI to accommodate these two types of requests separately. To improve scheduling efficiency and vehicle utilisation, this thesis explores the upgrading mechanism for the current planning system, which is conducted in three stages as follows.

- The research starts from investigating the classic VRPTW, which is a basic model to various transportation routing and scheduling problems. This first stage research explores the basic principle and features of vehicle routing problems, developing a metaheuristic algorithm which produces the promising results on the classic VPRTW benchmark.

- The second stage of this research focuses on the short-haul of the Ningbo Port problem. A scheduling scheme with a new shift-change rule is proposed in this study. The experiment results on historical real instances and artificial instances show that the proposed scheme brings a more efficient use of the fleet.

- Then, at the third stage, considering the trucks in Ningbo Port are identical, the scheduling and routing of both types of tasks are integrated into one planning system for better fleet utilisation. A mixed-shift vehicle routing problem is then defined to assign different types of tasks in one algorithm framework efficiently.

The main research contents and the associated contributions of this thesis are summarised below:

7.1.1 Mathematical Models of the Ningbo Port Problem and Benchmark Instances

In Chapter 3, the two proposed mathematical models are introduced. The short-haul inter dock container transshipment problem in Ningbo Port is investigated first. Based on this study, the global container transportation optimisation is researched by combining the scheduling of long-distance container transshipment tasks.

- An improved scheduling scheme was proposed, formulated as a Periodic Vehicle Routing Problem with Time Windows and Open Routes (PVRPTW-O), further improving the usage efficiency of the fleet in port districts. With the new scheme, the relaxation on the constraint of driver shift-change location promotes the HLDR on truck routes. This setting explores a local upgrading solution as the first step in the overall upgrade scheme of EDI.

- Following the PVRPTW-O study, a new mathematical problem model of Mixed-Shift Vehicle Routing Problem with Time Windows (MS-VRPTW) is therefore established, integrating the long-haul tasks.

- Artificial nodes are introduced in both models with different definitions. The node-based solution representation facilitates the application of many existing successful neighbourhood search methods.

- Benchmark dataset are generated for the two problems, considering the instance diversity and dataset generality. On these benchmark instances, the proposed algorithms were tested and compared with state-of-the-art methods. The experiment results were presented in Chapters 5 and 6.

7.1.2 Vehicle Routing Problem with Time Windows and Variable Compounded Neighbourhood Search

The classic VRPTW shares many common features in various vehicle routing problems, and it has been extended to model many real-life transportation routing and scheduling problems. To develop high-performance solution methodologies for the Ningbo Port problem, the VRPTW is investigated in Chapter 4, providing deeper understanding of the characteristics of VRPs and the associated solution methods.

- A variable neighbourhood search algorithm with compounded neighbourhood (VNS-C) is proposed for the VRPTW in this chapter. In this bi-objective problem, as a single solution-based metaheuristic approach, VNS-C efficiently improves both objectives simultaneously producing promising results on the benchmark.

- Three types of shaking operators and three compounded neighbourhood operators are developed in the phases of Shaking and Local Search. In the proposed compounded neighbourhood structures, a new way of compounding independent operator is used. This way provides a new way of constructing neighbourhood structures, striking a balance between diversification and intensification of the search.

- This study provides helpful guidances for future research on the Ningbo Port problem. It shows that the trade off between exploitation and exploration in the solution space is a critical factor to successful metaheuristic algorithms.

7.1.3 Inter-Dock Container Transshipment Problem and Metaheuristic Solution Methods

Chapter 5 studies the solution methodologies for PVRPTW-O intensively.

- Exact search is tried first. After some necessary modifications on the mathematical model, the scaled down benchmark instances are input to CPLEX. The experiment results show that, even if massive computation resources are given, addressing PVRPTW-O with exact search methods is still unrealistic.

- An improved VNS algorithm for PVRPTW-O is explored first. In the proposed algorithm VNS-RLS, after constructing a feasible initial solution with an urgency level based insertion heuristic, a reinforcement learning scheme is introduced to guide the neighbourhood operator selection in the VNS framework by estimating the historical performance of the operators during the search. During the solution improvement phase, three metrics are devised to evaluate the operator performance from the perspectives of objective value and solution feasibility. The experiment results on benchmark show that the three proposed evaluating metrics and associated guiding scheme can promote the HLDR and solution feasibility. In addition, the impact of sampling scheme to single solution-based metaheuristics is also investigated in this study. It is found that the sampling scheme does not show significant improvement to solution quality as it does in population-based metaheuristics, but it greatly increases the solution feasibility during search.

- A variable-depth adaptive large neighbourhood search algorithm (VD-ALNS) is later implemented for PVRPTW-O. VD-ALNS brings larger changes to the solution operated in each iteration. To balance the diversification and the intensification of the search, a deterministic neighbourhood depth changing scheme is proposed to a trade off for the great perturbation brought by the ALNS framework and local exploitation. A number of new best solutions have been found by this algorithm with slightly less computation time on small instances and 17% more on big instances. In addition, the research also shows that, to both destroy heuristic and repair heuristic, greedy approaches would contribute the most to solution quality improvement in ALNS algorithms.

- Chapter 5 provides a feasible upgrade scheme for promoting the HLDR of inter-dock container transshipment in Ningbo Port. The experiment results indicates that, with relaxing the constraints on shift-change location, the proposed new scheduling scheme can bring higher HLDR on all types of instances. In addition, this study explores the possibility of routing and scheduling in longer shifts, facilitating the follow-up research which takes the dry port-related tasks into account.

7.1.4 Mixed-Shift Container Transportation Problem and a Hyper-Heuristic Method

In Chapter 6, a generic method for multi-objective routing problems is introduced and applied to the MS-VRPTW.

- The proposed method GIHH is a selection perturbation hyper-heuristic, where the low-level heuristic (LLH) library consists of four heuristics with diverse degrees of perturbation and a state-of-the-art heuristic for minimising the number of routes. Based on the ULIH method introduced in Chapter 5, an extended construction heuristic is proposed to generate the initial feasible solution as the seed solution in GIHH. Two indicators are proposed in the high-level heuristic of GIHH to guide the selection of LLH, where one evaluates the contribution to solution quality improvement and the other concerns to the improvement direction.

- Experiments show that the MS-VRPTW model and the GIHH algorithm can provide the Ningbo Port company satisfying solutions of this large scale problem with nonlinear constraints. According to the experiment results, it can be found that the proposed guidance indicators bring a self-adapting employment of LLH, significantly improving the solution quality and the solution distribution on Pareto Front. GIHH shows good performance in solving the VPRTW as well, proving the generality of this method, while it performs better on instances with tighter time constraints.

- It is observed that, when the ranges of objectives are greatly different in multi-objective problems, the Hill Climbing acceptance criterion would outperform the acceptance criteria allowing inferior solutions (e.g. Record-to-Record Travel).

- The study shows that the MS-VRPTW model and the solution methodology of GIHH can be an efficient mechanism for updating the overall scheduling and routing performance in the current system EDI. Besides, GIHH may be used in other scheduling scenarios in Ningbo Port for its generality, saving the development time of upgrading other sub-systems in EDI, e.g. vessel parking optimisation and crane scheduling.

7.2 Future Research Directions

7.2.1 Path Planning

In some studies, the exact path between two waypoints is also considered. This task is called Path Planning [171]. Path planning is low-level motion planning. In long-distance transportation problems, planning the detailed path for each vehicle according to specific objective conditions may further improve the efficiency of

fleet. For example, because traffic conditions change over time in practical conditions, specifying the paths between two terminals by hour can reduce the vehicle waiting time caused by congestion. In addition, some large terminals have more than one gate. Planning the specific path including the entrance gate for every vehicle, instead of representing a terminal with a simple waypoint, may promote the efficiency of both fleet and terminals, reducing the long queue at gates.

Since the paths among the terminals and the depot are fixed by the company in the Ningbo Port problem, the drivers cannot change the fixed path when completing a task or when going to the next task. Vehicle routing considering path planning presents an interesting and different integrated problem, which can be investigated in future research.

7.2.2 Dynamic and Stochastic Factors

In real-world multi-shift vehicle routing problems, some information cannot be obtained in advance or it is not static. For example, a few declaration forms might be submitted within 24 hours ahead of their deadlines. In busy shifts, simply inserting these urgent transportation tasks into the current schedule can result in failing an entire task set of relevant trucks. In addition, the vessel docking time and truck travelling time may be updated when executing the current schedule. This underlying stochastic feature has greater impact on long-distance task scheduling. When vessels dock ahead of schedule or the truck travelling time is longer than expected, not only will the utilisation of fleet decrease, but the feasibility of the routing plan will be changed. Although the company coordinators can adjust the current routes manually and there is sufficient outsource truck resource in Ningbo Port to handle urgent tasks, the overall system robustness and efficiency would be promoted if the dynamic and stochastic factors are considered from the beginning.

7.2.3 Other Real-Life Objectives

As discussed in Section 5.5, HLDR does not reveal all the important solution details and the driver salary represents only one essential part of the company's operational costs. To better estimate the quality of solutions and provide advanced decision support to coordinators, more objective functions are expected to be defined related to experience. For example, with increasing environmental concerns and carbon tax in recent years, carbon emission and petrol consumption have risen more and more attention in the VRP community, contributing to the Pollution-Routing Problem and Green Vehicle Routing Problem [104]. Forkenbrock [114] discusses the external

costs and benefits outside normal market processes, e.g. the costs of accidents, emissions, noise and the benefits that are not reflected in prices.

7.2.4 Vessel Routing Problem and Maritime Location Based Services

With the rapid growth of intermodal industry, the higher requirements to the global planning are brought forward. With the help of big data in maritime location-based services, the vessel docking time can be predicted precisely leading to more efficient vehicle route planning. The intermodal global planning would reduce the container stay time and vehicle parking time, providing the all-the-way tracking. Accordingly, the overall operational costs will decrease as well. During my PhD study, as the major contributor, I completed a project of "a tool of statistic analysis and visualisation for big data of maritime location based service", which is a sub-project of a Knowledge Transfer Partnerships project supported by INNOVATEUK. The implemented functionalities of this tool are introduced in Appendix A in detail. In the future research, the achievements of this project could be integrated to the EDI, constructing a more advanced planning system in Ningbo Port.

Both historical achievements and the latest update in the research of the Ningbo Port problem can be found on https://sites.google.com/nottingham.ac.uk/port-management

Appendix A

A Tool of Statistical Analysis and Visualisation for Big Data Maritime Location-Based Services

A picture is worth a thousand words. The Visualisation for Big Data Maritime Location-Based Services is an INNOVATEUK sponsored project using maritime AIS (Automatic Identification System) data. As an essential part of the project, the work introduced in this section has achieved a stand-alone tool, written with MathWork Matlab. An application named Visual_GlobaVista is implemented, providing the following graphical functions of:

- visualising connections between the ports;
- calculating the statistic by port and by vessels.

This appendix introduces the features of Visual_GlobaVisata. The content of this appendix can be used as the user guide of this application. Figure A.1 shows the main interface. After importing the geographic information of ports file and the trip record file, a brief information summary will be presented on the main interface and the message box. With the big data imported, the application provides three sub-interfaces with various data retrieval and visualisation functionalities.

A.1 Geographic Information View

On the main interface, click the "View Geographic Information" button to open the window presented in Figure A.2. In the popup window, the routes specified by the imported data can be visualised. The radio buttons are used to specify a user's selection of departure port or arrival port. Clicking the "Draw Specified Trips" button draws out the specified routes on the map. By scrolling the mouse

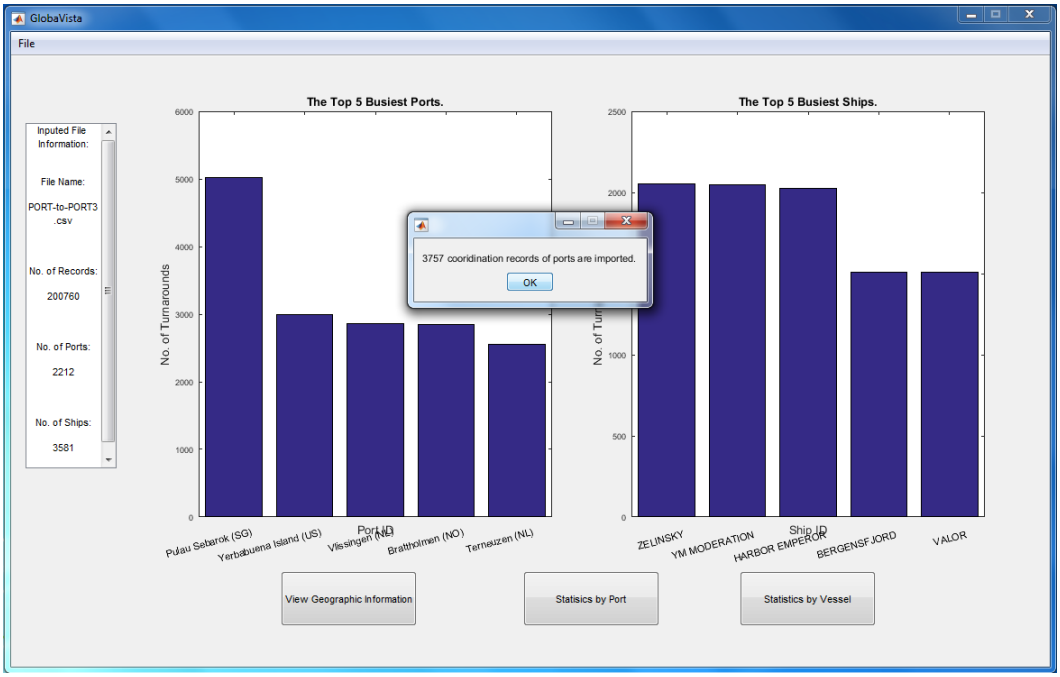


Figure A.1: Main interface of Visual_GlobaVista.

wheel or positioning the mouse cursor where a user wishes the centre of the plot to be, the user can zoom in and out the plot. It is noticeable that, in the plot, the number at the end of a route is the repeated times of the route, i.e. the number of trips with this route (Figure A.3).

Clicking "Clean All Trips" button removes all trips on the map. The obtained filtered data can be exported to a CSV file by clicking the "Export Specified Data" button. The obtained plot can also be saved, in png, eps, pdf or jpg format with the "Save Plot" button.

A.2 Statistics by Port

Clicking "Statistic by Port" button pops up a window that plots the data for specified ports (Figure A.4). Using the time filter on the up right corner, constraints on departure and arrival time of trips can be set. By clicking the "Before" or "After" button, a calendar control will pop up to choose the constraint date. Note that, in the edit boxes, any content that does not match the date format "dd-mmm-yyyy" will be ignored and no constraint is considered.

After selecting a plot type from the plot list and a port from a port list, the specified plot can be generated with the "Generate Plot" button. Users can reset the value of N (the number of items to be calculated) or specify a vessel type when the plot type is confirmed. The detailed description of each type of plot is

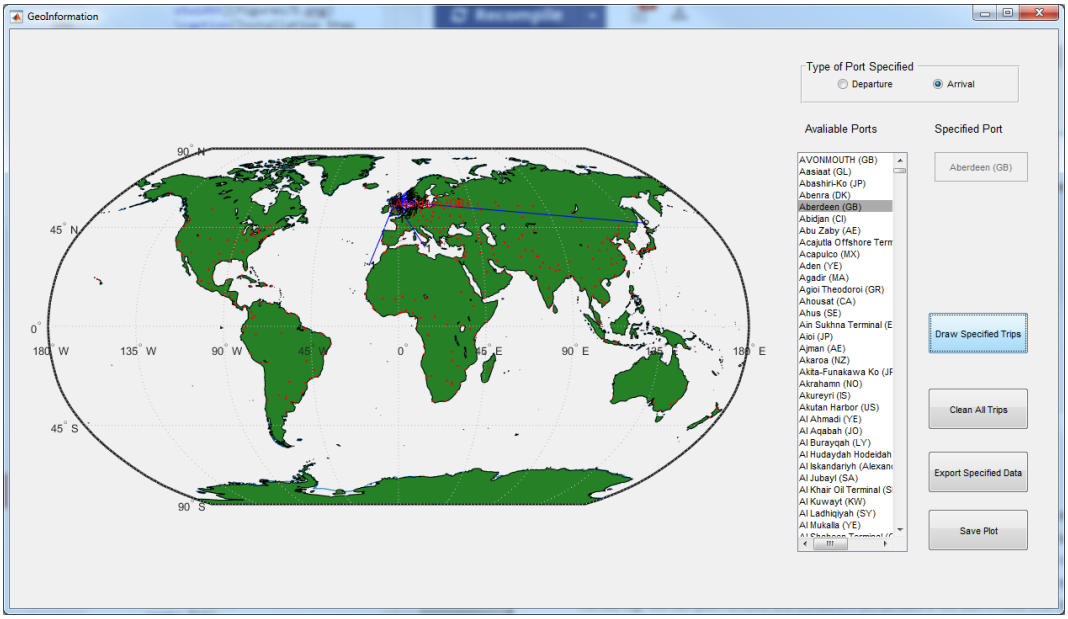


Figure A.2: Port geographic distribution and the related routes.

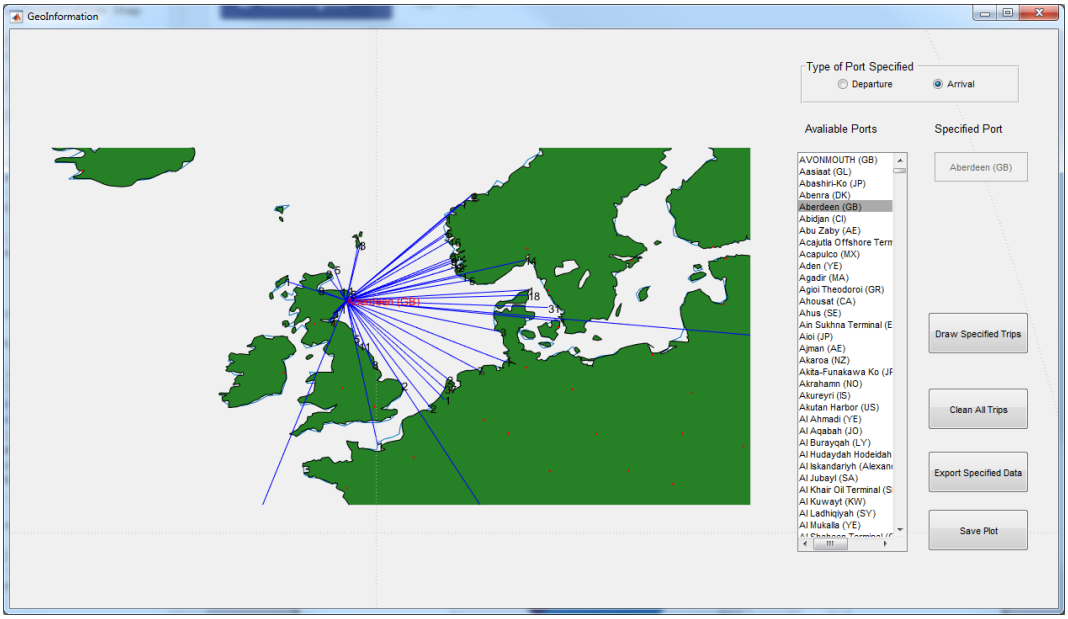


Figure A.3: Zoom view of the geographic information.

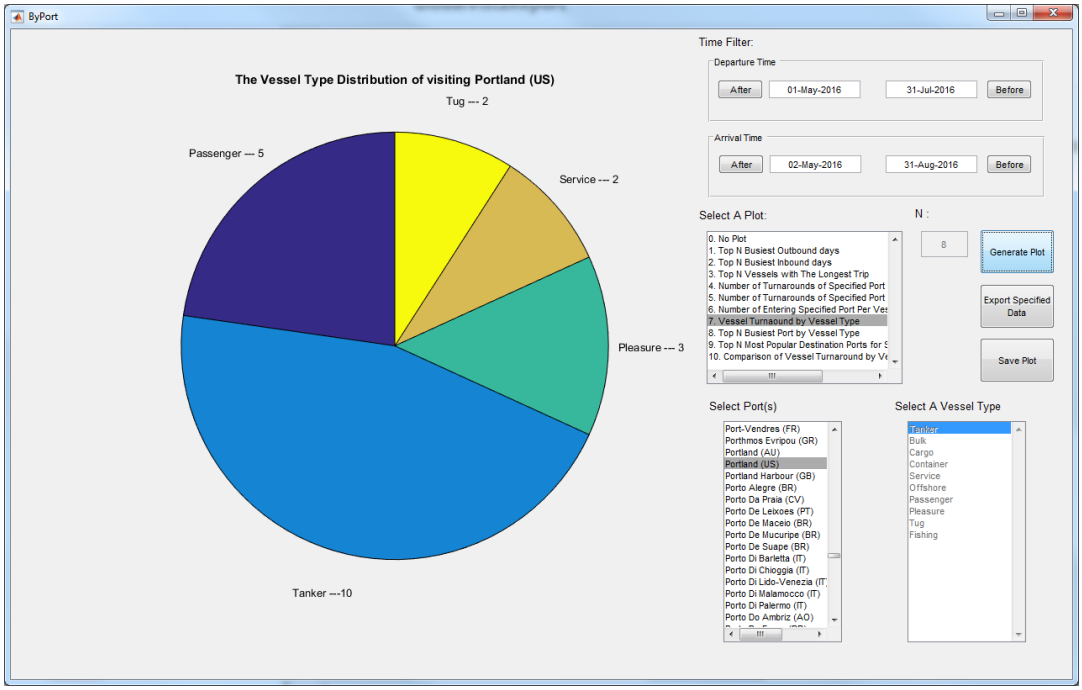


Figure A.4: The interface for retrieving information by port.

presented in Table A.1. The functions of the "Export Specified Data" and "Save Plot" buttons are the same as the two introduced in the last interface.

Table A.1: Plot descriptions of the statistics by port.

SN	Description	N's Editability	Vessel Type Specification	Number of Ports Selected
1	**Top N busiest outbound days**. This plot reveals the top N busiest outbound days at the specified port. The x-axis gives the number of ships departing that port on a specific date, while the y-axis indicates the specific dates.	Yes	No	Single
2	**Top N busiest inbound days**. This plot reveals the top N busiest inbound days at the specified port. The x-axis gives the number of ships arriving at that port on a specific date, while the y-axis indicates the specific dates.	Yes	No	Single
3	**Top N vessels with the longest trip**. This plot reveals the N ships departing the specified port with the longest trips to their destination ports. The x-axis indicates the lengths of the trips, while the y-axis indicates the names of the N ships.	Yes	No	Single
4	**Number of turnarounds of specified port (by day)**. This plot reveals the number of ships arriving the specified port by day during the specified period. The number of lines in the plot is decided by the range of Arrival Time interval appointed. If there is no arrival time constraint placed, the default number of lines (53) will be applied, which is the number of weeks in one single year. The x-axis indicates the days of a week, and the y-axis indicates the number of ships entering the specified port.	No	No	Single

(continued)

SN	Description	N's Editability	Vessel Type Specification	Number of Ports Selected
5	**Number of turnarounds of specified port (by week)**. This plot reveals the number of ships arriving the specified port by week during the specified period. The number of nodes in the plot is decided by the range of Arrival Time interval appointed. If there is no arrival time constraint placed, the default number of nodes (53) will be applied, which is the number of weeks in one single year. The x-axis indicates the indices of weeks in a year, and the y-axis indicates the number of ships entering the specified port during a corresponding week.	No	No	Single
6	**Number of entering a specified port per vessel**. This plot presents the number of arrivals at the specified port by ship. The x-axis indicates the times of arrivals for each ship, while the y-axis shows the names of all ships having visited the port during the appointed period. It is noticeable that, the y-axis labels would be crowded when the number of ships is large.	No	No	Single
7	**Vessel turnaround by vessel type**. A pie chart showing the distribution of the type of vessels visiting the specified port.	No	No	Single
8	**Top N busiest port by vessel type**. This plot reveals the N busiest ports to the specified type of vessels. e.g the 8 most popular destination ports to cargo vessels. The x-axis shows the names of ports, while the y-axis presents the times of being visited by ships of the specified type.	Yes	Yes	Single
9	**Top N most popular destination ports for specified source port**. This plot reveals the N most popular destination ports when the specified port is used as a source port. The x-axis shows the names of ports, while the y-axis presents the times of being visited by ships from the specified port.	Yes	No	Single
10	**Comparison of vessel turnaround by vessel type**. This plot presents the comparison on the number of ships visiting the specified ports, while the type of ships is specified as well. The x-axis lists the names of specified ports, while the y-axis presents the numbers of visits to those ports. At least two ports should be selected before generating the plot. Users have to press and hold the "Ctrl" key when choosing multiple ports from the port list.	No	Yes	Multiple

A.3 Statistics by Vessel

In the window of statistics by vessel, clicking "Statistic by Vessel" button pops up a window of specified vessel(s) (Figure A.5). The functions and conditions of the time filter are the same as the one introduced in Section A.2. In this interface, users can either specify one single ship by ticking the radio button "Select by Vessel Name", or specify a set of ships by ticking the radio button "Select by Parameters". Then users can choose a plot to be generated from the plot list and click the "Generate Plot" button to generate the associated plot.

When the radio button of "Select by Vessel Name" is selected, users can specify the ship either by picking a ship from the candidate ships of a port chosen, or type in the name of the ship to the edit box of "Vessel Specified" directly.

On the other hand, when the radio button of "Select by Parameter" is selected, users can specify a set of ships by setting constraints on Net Tonnages and Lengths

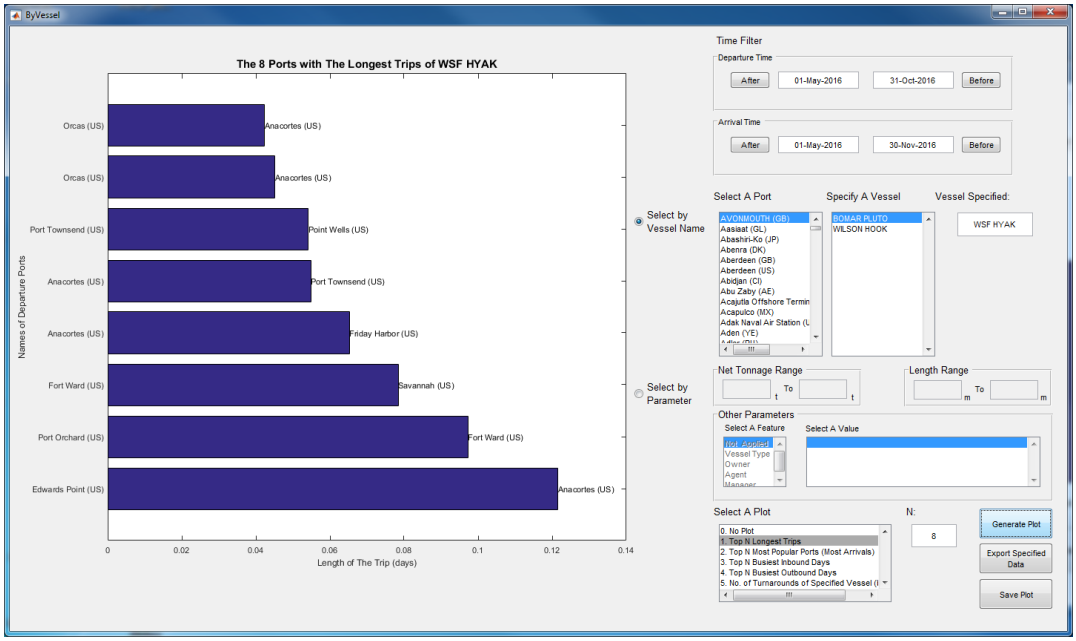


Figure A.5: The interface for retrieving information by vessel.

of ships. However, the constraints on these two features are not very valuable in practice as the data of these two features are missing on many records. In addition, one extra constraint can be selected from Vessel Type, Owner, Agent and Manager of ships. When the extra feature is selected, the candidate value list for the feature will update automatically.

The five types of plots provided in this interface are summarised in Table A.2.

Table A.2: Plot descriptions of the statistics by vessel

SN	Description	N's Editablity
1	**Top N longest trips**. This plot reveals the N trips of the specified vessel(s). In the generated plot, The x-axis indicates the duration of trips, while the y-axis shows the names of departure ports of the N trips. When users selected one specific vessel, the names of the associated arrival ports of the longest trips will be shown on the right side of generated bars. In contrast, when the "Select by Parameter" radio button is ticked, those names on the right side of the bars will be the names of the N ships having the longest trips.	Yes
2	**Top N most popular ports (most arrivals)**. This plot reveals the N most popular destinations of the specified vessel(s). The x-axis indicates the number of visits, and the y-axis lists the names of ports being visited.	Yes
3	**Top N busiest inbound days**. This plot reveals the N days with the most arrivals of the specified vessel(s). The x-axis indicates the number of visits, while the y-axis lists the dates of that N days.	Yes
4	**Top N busiest outbound days**. This plot reveals the N days with the most departures of the specified vessel(s). The x-axis indicates the number of departures, while the y-axis lists the dates of that N days.	Yes
5	**No. of turnarounds of specified vessels (by day)**. This plot reveals the number of arrivals of the specified vessel(s) on each day. The range of the period to be plotted is set by the arrival time in the time filter. When the number of days during that period is less than 35, the detailed date information will be listed on the x-axis, otherwise, only the indices of days will be given. The y-axis shows the number of arrivals.	No

Bibliography

[1] Ravindra K Ahuja. *Network flows: theory, algorithms, and applications*. Pearson Education, 2017.

[2] Attahiru Sule Alfa, Sundresh S Heragu, and Mingyuan Chen. A 3-opt based simulated annealing algorithm for vehicle routing problems. *Computers & Industrial Engineering*, 21(1-4):635–639, 1991.

[3] Tony Allard and Charles Gretton. A realistic multi-modal cargo routing benchmark. In *Workshops at the Twenty-Ninth AAAI Conference on Artificial Intelligence*, 2015.

[4] Samira Almoustafa, Said Hanafi, and Nenad Mladenović. New exact method for large asymmetric distance-constrained vehicle routing problem. *European Journal of Operational Research*, 226(3):386–394, 2013.

[5] Guilherme Bastos Alvarenga, Geraldo Robson Mateus, and G De Tomi. A genetic and set partitioning two-phase approach for the vehicle routing problem with time windows. *Computers & Operations Research*, 34(6):1561–1584, 2007.

[6] Jardar Andersen, Teodor Gabriel Crainic, and Marielle Christiansen. Service network design with management and coordination of multiple fleets. *European Journal of Operational Research*, 193(2):377–389, 2009.

[7] Enrico Angelelli, M Grazia Speranza, and Martin WP Savelsbergh. Competitive analysis for dynamic multiperiod uncapacitated routing problems. *Networks*, 49(4):308–317, 2007.

[8] Enrico Angelelli, Nicola Bianchessi, Renata Mansini, and Maria G Speranza. Short term strategies for a dynamic multi-period routing problem. *Transportation Research Part C: Emerging Technologies*, 17(2):106–119, 2009.

[9] Shoshana Anily, Michel Gendreau, and Gilbert Laporte. The preemptive swapping problem on a tree. *Networks*, 58(2):83–94, 2011.

[10] David L Applegate. *The traveling salesman problem: a computational study*. Princeton university press, 2006.

[11] Claudia Archetti and Maria Grazia Speranza. The split delivery vehicle routing problem: a survey. *The Vehicle Routing Problem: Latest Advances and New Challenges*, pages 103–122, 2008.

[12] Ph Augerat, Jose Manuel Belenguer, Enrique Benavent, A Corberán, D Naddef, and G Rinaldi. Computational results with a branch-and-cut code for the capacitated vehicle routing problem. *Rapport de Recherche- IMAG*, 1998.

[13] Nabila Azi, Michel Gendreau, and Jean-Yves Potvin. An adaptive large neighborhood search for a vehicle routing problem with multiple routes. *Computers & Operations Research*, 41:167–173, 2014.

[14] Ruibin Bai, Jacek Blazewicz, Edmund K Burke, Graham Kendall, and Barry McCollum. A simulated annealing hyper-heuristic methodology for flexible decision support. *4OR: A Quarterly Journal of Operations Research*, 10(1): 43–66, 2012.

[15] Ruibin Bai, Stein W Wallace, Jingpeng Li, and Alain Yee-Loong Chong. Stochastic service network design with rerouting. *Transportation Research Part B: Methodological*, 60:50–65, 2014.

[16] Ruibin Bai, Ning Xue, Jianjun Chen, and Gethin Wyn Roberts. A set-covering model for a bidirectional multi-shift full truckload vehicle routing problem. *Transportation Research Part B: Methodological*, 79:134–148, 2015.

[17] Egon Balas and Paolo Toth. Branch and bound methods for the traveling salesman problem. Technical report, CARNEGIE-MELLON UNIV PITTSBURGH PA MANAGEMENT SCIENCES RESEARCH GROUP, 1983.

[18] Roberto Baldacci, Maria Battarra, and Daniele Vigo. Routing a heterogeneous fleet of vehicles. *The Vehicle Routing Problem: Latest Advances and New Challenges*, pages 3–27, 2008.

[19] Michel L Balinski and Richard E Quandt. On an integer program for a delivery problem. *Operations Research*, 12(2):300–304, 1964.

[20] Shumeet Baluja. Population-based incremental learning. a method for integrating genetic search based function optimization and competitive learning. Technical report, Carnegie-Mellon Univ Pittsburgh Pa Dept Of Computer Science, 1994.

[21] Shumeet Baluja. An empirical comparison of seven iterative and evolutionary heuristics for static function optimization. In *Proc. of the 11th Int. Conf. on System Engineering*, pages 692–697. Citeseer, 1995.

[22] RaúL Baños, Julio Ortega, ConsolacióN Gil, Antonio L MáRquez, and Francisco De Toro. A hybrid meta-heuristic for multi-objective vehicle routing problems with time windows. *Computers & Industrial Engineering*, 65(2): 286–296, 2013.

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

[24] Cynthia Barnhart, Ellis L Johnson, George L Nemhauser, Martin WP Savelsbergh, and Pamela H Vance. Branch-and-price: Column generation for solving huge integer programs. *Operations Research*, 46(3):316–329, 1998.

[25] James C Bean. Genetic algorithms and random keys for sequencing and optimization. *ORSA Journal on Computing*, 6(2):154–160, 1994.

[26] J Christopher Beck, Patrick Prosser, and Evgeny Selensky. Vehicle routing and job shop scheduling: what's the difference? In *ICAPS*, pages 267–276, 2003.

[27] Tolga Bektaş and Gilbert Laporte. The pollution-routing problem. *Transportation Research Part B: Methodological*, 45(8):1232–1250, 2011.

[28] Edward J Beltrami and Lawrence D Bodin. Networks and vehicle routing for municipal waste collection. *Networks*, 4(1):65–94, 1974.

[29] Gerardo Berbeglia, Jean-François Cordeau, Irina Gribkovskaia, and Gilbert Laporte. Static pickup and delivery problems: a classification scheme and survey. *Top*, 15(1):1–31, 2007.

[30] Jean Berger, Mohamed Barkaoui, and Olli Bräysy. A route-directed hybrid genetic approach for the vehicle routing problem with time windows. *INFOR: Information Systems and Operational Research*, 41(2):179–194, 2003.

[31] Eshetie Berhan, Birhanu Beshah, Daniel Kitaw, and Ajith Abraham. Stochastic vehicle routing problem: a literature survey. *Journal of Information & Knowledge Management*, 13(03):1450022, 2014.

[32] Dimitris Bertsimas, John Tsitsiklis, et al. Simulated annealing. *Statistical Science*, 8(1):10–15, 1993.

[33] Dimitris J Bertsimas and David Simchi-Levi. A new generation of vehicle routing research: robust algorithms, addressing uncertainty. *Operations Research*, 44(2):286–304, 1996.

[34] Patrick Beullens, Luc Muyldermans, Dirk Cattrysse, and Dirk Van Oudheusden. A guided local search heuristic for the capacitated arc routing problem. *European Journal of Operational Research*, 147(3):629–643, 2003.

[35] Patrick Beullens, Dirk Van Oudheusden, and Luk N Van Wassenhove. Collection and vehicle routing issues in reverse logistics. In *Reverse Logistics*, pages 95–134. Springer, 2004.

[36] Christian Blum and Andrea Roli. Metaheuristics in combinatorial optimization: Overview and conceptual comparison. *ACM Computing Surveys (CSUR)*, 35(3):268–308, 2003.

[37] Brian Boffey, Francisco Ramón Fernández García, Gilbert Laporte, Juan A Mesa, and Blas Pelegrín Pelegrín. Multiobjective routing problems. *Top*, 3 (2):167–220, 1995.

[38] Ralf Borndörfer, Martin Grötschel, Fridolin Klostermeier, and Christian Küttner. Telebus berlin: Vehicle scheduling in a dial-a-ride system. In *Computer-Aided Transit Scheduling*, pages 391–422. Springer, 1999.

[39] Nathalie Bostel, Pierre Dejax, Pierre Guez, and Fabien Tricoire. Multiperiod planning and routing on a rolling horizon for field force optimization logistics. *The Vehicle Routing Problem: Latest Advances and New Challenges*, pages 503–525, 2008.

[40] Robert Bowerman, Brent Hall, and Paul Calamai. A multi-objective optimization approach to urban school bus routing: Formulation and solution method. *Transportation Research Part A: Policy and Practice*, 29(2):107–123, 1995.

[41] Lucas Bradstreet. *The hypervolume indicator for multi-objective optimisation: calculation and use*. University of Western Australia, 2011.

[42] Kris Braekers, An Caris, and Gerrit K Janssens. A deterministic annealing algorithm for a bi-objective full truckload vehicle routing problem in drayage operations. *Procedia-Social and Behavioral Sciences*, 20:344–353, 2011.

[43] Kris Braekers, An Caris, and Gerrit K Janssens. Time-dependent routing of drayage operations in the service area of intermodal terminals. In *Proceedings of the 14th International Conference on Harbor, Maritime and Multimodal Logistics Modeling and Simulation*. DIME Università di Genova, 2012.

[44] Kris Braekers, An Caris, and Gerrit K Janssens. Integrated planning of loaded and empty container movements. *OR Spectrum*, 35(2):457–478, 2013.

[45] Kris Braekers, An Caris, and Gerrit K Janssens. Bi-objective optimization of drayage operations in the service area of intermodal terminals. *Transportation Research Part E: Logistics and Transportation Review*, 65:50–69, 2014.

[46] Julien Bramel and David Simchi-Levi. A location based heuristic for general routing problems. *Operations Research*, 43(4):649–660, 1995.

[47] José Brandão. A tabu search algorithm for the open vehicle routing problem. *European Journal of Operational Research*, 157(3):552–564, 2004.

[48] Jürgen Branke, Martin Middendorf, Guntram Noeth, and Maged Dessouky. Waiting strategies for dynamic vehicle routing. *Transportation Science*, 39 (3):298–312, 2005.

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

[50] Olli Bräysy and Michel Gendreau. Metaheuristics for the vehicle routing problem with time windows. *Report STF42 A*, 1025:3–38, 2001.

[51] Julio Brito, F Javier Martínez, JA Moreno, and José L Verdegay. An a-co hybrid metaheuristic for close–open vehicle routing problems with time windows and fuzzy constraints. *Applied Soft Computing*, 32:154–163, 2015.

[52] Geir Brønmo, Marielle Christiansen, Kjetil Fagerholt, and Bjørn Nygreen. A multi-start local search heuristic for ship schedulinga computational study. *Computers & Operations Research*, 34(3):900–917, 2007.

[53] Edmund Burke, Graham Kendall, Jim Newall, Emma Hart, Peter Ross, and Sonia Schulenburg. Hyper-heuristics: An emerging direction in modern search technology. In *Handbook of Metaheuristics*, pages 457–474. Springer, 2003.

[54] Edmund K Burke, Matthew Hyde, Graham Kendall, Gabriela Ochoa, Ender Özcan, and John R Woodward. A classification of hyper-heuristic approaches. In *Handbook of Metaheuristics*, pages 449–468. Springer, 2010.

[55] Edmund K Burke, Michel Gendreau, Gabriela Ochoa, and James D Walker. Adaptive iterated local search for cross-domain optimisation. In *Proceedings of the 13th Annual Conference on Genetic and Evolutionary Computation*, pages 1987–1994. ACM, 2011. ISBN 1450305571.

[56] Edmund K Burke, Michel Gendreau, Matthew Hyde, Graham Kendall, Gabriela Ochoa, Ender Özcan, and Rong Qu. Hyper-heuristics: A survey of the state of the art. *Journal of the Operational Research Society*, 64(12): 1695–1724, 2013.

[57] Edmund K Burke, Graham Kendall, et al. *Search methodologies: introductory tutorials in optimization and decision support techniques*. Springer, 2014.

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

[59] Binhui Chen, Rong Qu, Ruibin Bai, and Hisao Ishibuchi. An investigation on compound neighborhoods for vrptw. In *International Conference on Operations Research and Enterprise Systems*, pages 3–19. Springer, 2016.

[60] Jianjun Chen, Ruibin Bai, Rong Qu, and Graham Kendall. A task based approach for a real-world commodity routing problem. In *IEEE Workshop on Computational Intelligence In Production And Logistics Systems (CIPLS), 2013*, pages 1–8. IEEE, 2013.

[61] Raymond K Cheung, Judy H Tong, and Brian Slack. The transition from freight consolidation to logistics: the case of hong kong. *Journal of Transport Geography*, 11(4):245–253, 2003.

[62] Wen-Chyuan Chiang and Robert A Russell. A reactive tabu search metaheuristic for the vehicle routing problem with time windows. *INFORMS Journal on Computing*, 9(4):417–430, 1997.

[63] Darren M Chitty and Marcel L Hernandez. A hybrid ant colony optimisation technique for dynamic vehicle routing. In *Genetic and Evolutionary Computation Conference, GECCO 2004*, pages 48–59. Springer, 2004.

[64] Eunjeong Choi and Dong-Wan Tcha. A column generation approach to the heterogeneous fleet vehicle routing problem. *Computers & Operations Research*, 34(7):2080–2095, 2007.

[65] Nicos Christofides. The vehicle routing problem. *Revue française d'automatique, Informatique, Recherche Opérationnelle. Recherche Opérationnelle*, 10(V1):55–70, 1976.

[66] Nicos Christofides, Aristide Mingozzi, and Paolo Toth. The vehicle routing problem. *Combinatorial Optimization*, pages 315–338, 1979.

[67] Lukáš Chrpa, Daniele Magazzeni, Keith McCabe, Thomas L McCluskey, and Mauro Vallati. Automated planning for urban traffic control: Strategic vehicle routing to respect air quality limitations. *Intelligenza Artificiale*, 10 (2):113–128, 2016.

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

[69] Maurice Clerc. *Particle swarm optimization*, volume 93. John Wiley & Sons, 2010.

[70] Carlos A Coello Coello, Gary B Lamont, David A Van Veldhuizen, et al. *Evolutionary algorithms for solving multi-objective problems*, volume 5. Springer, 2007.

[71] Alberto Colorni, Marco Dorigo, Vittorio Maniezzo, et al. Distributed optimization by ant colonies. In *Proceedings of the First European Conference on Artificial Life*, volume 142, pages 134–142. Paris, France, 1991.

[72] William Cook and Jennifer L Rich. A parallel cutting-plane algorithm for the vehicle routing problem with time windows. *Computational and Applied Mathematics Department, Rice University, Houston, TX, Technical Report*, 1999.

[73] Angel Corberán, Elena Fernández, Manuel Laguna, and RAFAEL Marti. Heuristic solutions to the problem of routing school buses with multiple objectives. *Journal of the Operational Research Society*, 53(4):427–435, 2002.

[74] Jean-François Cordeau. A branch-and-cut algorithm for the dial-a-ride problem. *Operations Research*, 54(3):573–586, 2006.

[75] Jean-François Cordeau and Gilbert Laporte. A tabu search algorithm for the site dependent vehicle routing problem with time windows. *INFOR: Information Systems and Operational Research*, 39(3):292–298, 2001.

[76] Jean-François Cordeau and Gilbert Laporte. The dial-a-ride problem (darp): Variants, modeling issues and algorithms. *4OR*, 1(2):89–101, 2003.

[77] Jean-François Cordeau and Gilbert Laporte. A tabu search heuristic for the static multi-vehicle dial-a-ride problem. *Transportation Research Part B: Methodological*, 37(6):579–594, 2003.

[78] Jean-François Cordeau, Gilbert Laporte, Martin WP Savelsbergh, and Daniele Vigo. Vehicle routing. *Handbooks in Operations Research and Management Science*, 14:367–428, 2007.

[79] Jean-François Cordeau, Gilbert Laporte, and Stefan Ropke. Recent models and algorithms for one-to-one pickup and delivery problems. *The Vehicle Routing Problem: Latest Advances and New Challenges*, 43:327–357, 2008.

[80] Jean-Franois Cordeau, Gilbert Laporte, and Anne Mercier. A unified tabu search heuristic for vehicle routing problems with time windows. *Journal of the Operational Research Society*, 52(8):928–936, 2001. ISSN 0160-5682.

[81] Jean-Franois Cordeau, Gilbert Laporte, and Anne Mercier. Improved tabu search algorithm for the handling of route duration constraints in vehicle routing problems with time windows. *Journal of the Operational Research Society*, 55(5):542–546, 2004. ISSN 0160-5682.

[82] Gerard Cornuejols and Farid Harche. Polyhedral study of the capacitated vehicle routing problem. *Mathematical Programming*, 60(1-3):21–52, 1993.

[83] Teodor Gabriel Crainic. Service network design in freight transportation. *European Journal of Operational Research*, 122(2):272–288, 2000.

[84] John R Current and David A Schilling. The median tour and maximal covering tour problems: Formulations and heuristics. *European Journal of Operational Research*, 73(1):114–126, 1994.

[85] Zbigniew J Czech and Piotr Czarnas. Parallel simulated annealing for the vehicle routing problem with time windows. In *Proceedings. 10th Euromicro Workshop on Parallel, Distributed and Network-based Processing, 2002.*, pages 376–383. IEEE, 2002.

[86] Anna Danandeh, MAHDI Ghazanfari, Reza Tavakoli-Moghaddam, and Mehdi Alinaghian. A swift heuristic algorithm based on capacitated clustering for the open periodic vehicle routing problem. In *Proceedings of the 9th WSEAS international conference on Artificial intelligence, knowledge engineering and data bases, Stevens Point, Wisconsin, USA*, pages 208–214. World Scientific and Engineering Academy and Society (WSEAS), 2010.

[87] Emilie Danna and Claude Le Pape. Branch-and-price heuristics: A case study on the vehicle routing problem with time windows. In *Column Generation*, pages 99–129. Springer, 2005.

[88] George B Dantzig and John H Ramser. The truck dispatching problem. *Management Science*, 6(1):80–91, 1959.

[89] Iman Dayarian, Teodor Gabriel Crainic, Michel Gendreau, and Walter Rei. An adaptive large-neighborhood search heuristic for a multi-period vehicle routing problem. *Transportation Research Part E: Logistics and Transportation Review*, 95:95–123, 2016.

[90] Bruno De Backer, Vincent Furnon, Paul Shaw, Philip Kilby, and Patrick Prosser. Solving vehicle routing problems using constraint programming and metaheuristics. *Journal of Heuristics*, 6(4):501–523, 2000.

[91] Ulrich Derigs, Markus Pullmann, Ulrich Vogel, Marco Oberscheider, Manfred Gronalt, and Patrick Hirsch. Multilevel neighborhood search for solving full truckload routing problems arising in timber transportation. *Electronic Notes in Discrete Mathematics*, 39:281–288, 2012.

[92] Martin Desrochers and TW Verhoog. A matching based savings algorithm for the vehicle routing problem. *Cahiers du GERAD*, 1989.

[93] Martin Desrochers, Jacques Desrosiers, and Marius Solomon. A new optimization algorithm for the vehicle routing problem with time windows. *Operations Research*, 40(2):342–354, 1992.

[94] Jacques Desrosiers, Yvan Dumas, and François Soumis. A dynamic programming solution of the large-scale single-vehicle dial-a-ride problem with time windows. *American Journal of Mathematical and Management Sciences*, 6 (3-4):301–325, 1986.

[95] Karl Doerner, Axel Focke, and Walter J Gutjahr. Multicriteria tour planning for mobile healthcare facilities in a developing country. *European Journal of Operational Research*, 179(3):1078–1096, 2007.

[96] Gunter Dueck. New optimization heuristics: The great deluge algorithm and the record-to-record travel. *Journal of Computational Physics*, 104(1):86–92, 1993.

[97] Russell C Eberhart and James Kennedy. Particle swarm optimization. In *Proceeding of IEEE International Conference on Neural Network*, volume 4, pages 1942–1948, 1995.

[98] Francis Ysidro Edgeworth. *Mathematical psychics: An essay on the application of mathematics to the moral sciences*, volume 10. Kegan Paul, 1881.

[99] Richard Eglese and Daniel Black. Optimizing the routing of vehicles. *2010) Green Logistics: Improving the Environmental Sustainability of Logistics, KoganPage*, pages 215–228, 2010.

[100] Matthias Ehrgott. *Multicriteria optimization*. Springer Science & Business Media, 2006.

[101] Burak Eksioglu, Arif Volkan Vural, and Arnold Reisman. The vehicle routing problem: A taxonomic review. *Computers & Industrial Engineering*, 57(4): 1472–1483, 2009.

[102] Nasser El-Sherbeny. Resolution of a vehicle routing problem with multi-objective simulated annealing method. *Faculté Polytechnique de Mons*, 2001.

[103] Gary Eppen and Linus Schrage. Centralized ordering policies in a multi-warehouse system with lead times and random demand. *Multi-level Production/Inventory Control Systems: Theory and Practice*, 16:51–67, 1981.

[104] Sevgi Erdoğan and Elise Miller-Hooks. A green vehicle routing problem. *Transportation Research Part E: Logistics and Transportation Review*, 48 (1):100–114, 2012.

[105] Özlem Ergun, James B Orlin, and Abran Steele-Feldman. Creating very large scale neighborhoods out of smaller ones by compounding moves. *Journal of Heuristics*, 12(1-2):115–140, 2006.

[106] Dominique Feillet. A tutorial on column generation and branch-and-price for vehicle routing problems. *4OR*, 8(4):407–424, 2010.

[107] Matteo Fischetti and Paolo Toth. An additive bounding procedure for the asymmetric travelling salesman problem. *Mathematical Programming*, 53 (1-3):173–197, 1992.

[108] Matteo Fischetti, Paolo Toth, and Daniele Vigo. A branch-and-bound algorithm for the capacitated vehicle routing problem on directed graphs. *Operations Research*, 42(5):846–859, 1994.

[109] Marshall L Fisher. A polynomial algorithm for the degree-constrained minimum k-tree problem. *Operations Research*, 42(4):775–779, 1994.

[110] Marshall L Fisher. The lagrangian relaxation method for solving integer programming problems. *Management Science*, 50(12_supplement):1861–1871, 2004.

[111] Marshall L Fisher and Ramchandran Jaikumar. A generalized assignment heuristic for vehicle routing. *Networks*, 11(2):109–124, 1981.

[112] Marshall L Fisher, Kurt O Jörnsten, and Oli BG Madsen. Vehicle routing with time windows: Two optimization algorithms. *Operations Research*, 45 (3):488–492, 1997.

[113] Krzysztof Fleszar, Ibrahim H Osman, and Khalil S Hindi. A variable neighbourhood search algorithm for the open vehicle routing problem. *European Journal of Operational Research*, 195(3):803–809, 2009.

[114] David J Forkenbrock. External costs of intercity truck freight transportation. *Transportation Research Part A: Policy and Practice*, 33(7-8):505–526, 1999.

[115] Brian A Foster and David M Ryan. An integer programming approach to the vehicle scheduling problem. *Journal of the Operational Research Society*, 27(2):367–384, 1976.

[116] Peter M Francis, Karen R Smilowitz, and Michal Tzur. The period vehicle routing problem and its extensions. *The Vehicle Routing Problem: Latest Advances and New Challenges*, pages 73–102, 2008.

[117] Zhuo Fu, Richard Eglese, and Leon YO Li. A new tabu search heuristic for the open vehicle routing problem. *Journal of the Operational Research Society*, 56(3):267–274, 2005.

[118] Ricardo Fukasawa, Humberto Longo, Jens Lysgaard, Marcus Poggi de Aragão, Marcelo Reis, Eduardo Uchoa, and Renato F Werneck. Robust branch-and-cut-and-price for the capacitated vehicle routing problem. *Mathematical Programming*, 106(3):491–511, 2006.

[119] Delbert Ray Fulkerson. Blocking and anti-blocking pairs of polyhedra. *Mathematical Programming*, 1(1):168–194, 1971.

[120] Birger Funke, Tore Grünert, and Stefan Irnich. Local search for vehicle routing and scheduling problems: Review and conceptual integration. *Journal of Heuristics*, 11(4):267–306, 2005.

[121] Luca Maria Gambardella, Éric Taillard, and Giovanni Agazzi. Macs-vrptw: A multiple colony system for vehicle routing problems with time windows. In *New Ideas in Optimization*. Citeseer, 1999.

[122] Abel Garcia-Najera and John A Bullinaria. An improved multi-objective evolutionary algorithm for the vehicle routing problem with time windows. *Computers & Operations Research*, 38(1):287–300, 2011.

[123] Pablo Garrido and María Cristina Riff. Dvrp: a hard dynamic combinatorial optimisation problem tackled by an evolutionary hyper-heuristic. *Journal of Heuristics*, 16(6):795–834, 2010.

[124] Hermann Gehring and Jörg Homberger. A parallel hybrid evolutionary metaheuristic for the vehicle routing problem with time windows. In *Proceedings of EUROGEN99*, volume 2, pages 57–64. Citeseer, 1999.

[125] Michel Gendreau and Christos D Tarantilis. *Solving large-scale vehicle routing problems with time windows: The state-of-the-art*. Cirrelt Montreal, 2010.

[126] Michel Gendreau, Alain Hertz, and Gilbert Laporte. New insertion and postoptimization procedures for the traveling salesman problem. *Operations Research*, 40(6):1086–1094, 1992.

[127] Michel Gendreau, Alain Hertz, and Gilbert Laporte. A tabu search heuristic for the vehicle routing problem. *Management Science*, 40(10):1276–1290, 1994.

[128] Michel Gendreau, Gilbert Laporte, and René Séguin. Stochastic vehicle routing. *European Journal of Operational Research*, 88(1):3–12, 1996.

[129] Michel Gendreau, Gilbert Laporte, and René Séguin. A tabu search heuristic for the vehicle routing problem with stochastic demands and customers. *Operations Research*, 44(3):469–477, 1996.

[130] Hassan Ghaziri. Supervision in the self-organizing feature map: Application to the vehicle routing problem. In *Meta-Heuristics*, pages 651–660. Springer, 1996.

[131] Keivan Ghoseiri and Seyed Farid Ghannadpour. Multi-objective vehicle routing problem with time windows using goal programming and genetic algorithm. *Applied Soft Computing*, 10(4):1096–1107, 2010.

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

[133] Fred Glover. Future paths for integer programming and links to artificial intelligence. *Computers & Operations Research*, 13(5):533–549, 1986.

[134] Fred Glover. *Multilevel tabu search and embedded search neighborhoods for the traveling salesman problem.* Graduate School of Business, University of Colorado, 1991.

[135] Fred Glover. Ejection chains, reference structures and alternating path methods for traveling salesman problems. *Discrete Applied Mathematics*, 65(1-3): 223–253, 1996.

[136] Fred Glover and Manuel Laguna. Tabu search. In *Handbook of Combinatorial Optimization*, pages 3261–3362. Springer, 2013.

[137] Fred Glover and Eric Taillard. A user's guide to tabu search. *Annals of Operations Research*, 41(1):1–28, 1993.

[138] David E. Goldberg and John H. Holland. Genetic algorithms and machine learning. *Machine Learning*, 3(2):95–99, Oct 1988. ISSN 1573-0565.

[139] Bruce L Golden, Subramanian Raghavan, and Edward A Wasil. *The vehicle routing problem: latest advances and new challenges*, volume 43. Springer Science & Business Media, 2008.

[140] Peter Greistorfer. A tabu scatter search metaheuristic for the arc routing problem. *Computers & Industrial Engineering*, 44(2):249–266, 2003.

[141] Li Guiyun. An improved ant colony algorithm for open vehicle routing problem with time windows. In *International Conference on Information Management, Innovation Management and Industrial Engineering, 2009*, volume 2, pages 616–619. IEEE, 2009. ISBN 0769538762.

[142] Li Guiyun. Research on open vehicle routing problem with time windows based on improved genetic algorithm. In *International Conference on Computational Intelligence and Software Engineering, 2009. CiSE 2009.*, pages 1–5. IEEE, 2009. ISBN 1424445078.

[143] Gregory Gutin and Abraham P Punnen. *The traveling salesman problem and its variations*, volume 12. Springer Science & Business Media, 2006.

[144] Yacov Y Haimes. On a bicriterion formulation of the problems of integrated system identification and system optimization. *IEEE Transactions on Systems, Man, and Cybernetics*, 1(3):296–297, 1971.

[145] Pierre Hansen and Nenad Mladenović. Variable neighborhood search. In *Handbook of Metaheuristics*, pages 145–184. Springer, 2003.

[146] Pierre Hansen and Nenad Mladenović. Variable neighborhood search. In *Search Methodologies*, pages 313–337. Springer, 2014.

[147] Pierre Hansen, Nenad Mladenović, and José A Moreno Pérez. Variable neighbourhood search: methods and applications. *Annals of Operations Research*, 175(1):367–407, 2010.

[148] Michael Held and Richard M Karp. The traveling-salesman problem and minimum spanning trees. *Operations Research*, 18(6):1138–1162, 1970.

[149] Michael Held, Philip Wolfe, and Harlan P Crowder. Validation of subgradient optimization. *Mathematical Programming*, 6(1):62–88, 1974.

[150] Vera C Hemmelmayr, Karl F Doerner, and Richard F Hartl. A variable neighborhood search heuristic for periodic routing problems. *European Journal of Operational Research*, 195(3):791–802, 2009.

[151] Vera C Hemmelmayr, Jean-François Cordeau, and Teodor Gabriel Crainic. An adaptive large neighborhood search heuristic for two-echelon vehicle routing problems arising in city logistics. *Computers & Operations Research*, 39 (12):3215–3228, 2012.

[152] Alain Hertz, Gilbert Laporte, and Michel Mittaz. A tabu search heuristic for the capacitated arc routing problem. *Operations Research*, 48(1):129–135, 2000.

[153] Arild Hoff, Arnt-Gunnar Lium, Arne Løkketangen, and Teodor Gabriel Crainic. A metaheuristic for stochastic service network design. *Journal of Heuristics*, 16(5):653–679, 2010.

[154] Markus Höhfeld and Günter Rudolph. Towards a theory of population-based incremental learning. In *Proceedings of the 4th IEEE Conference on Evolutionary Computation*. Citeseer, 1997.

[155] John Holland. An introductory analysis with applications to biology, control, and artificial intelligence. *Adaptation in Natural and Artificial Systems. First Edition, The University of Michigan, USA*, 1975.

[156] John J Hopfield and David W Tank. neural computation of decisions in optimization problems. *Biological Cybernetics*, 52(3):141–152, 1985.

[157] George Ioannou, Manolis Kritikos, and G Prastacos. A greedy look-ahead heuristic for the vehicle routing problem with time windows. *Journal of the Operational Research Society*, 52(5):523–537, 2001. ISSN 0160-5682.

[158] Stefan Irnich and Daniel Villeneuve. The shortest-path problem with resource constraints and k-cycle elimination for k 3. *INFORMS Journal on Computing*, 18(3):391–406, 2006.

[159] Yaochu Jin. A comprehensive survey of fitness approximation in evolutionary computation. *Soft Computing-A Fusion of Foundations, Methodologies and Applications*, 9(1):3–12, 2005.

[160] David S Johnson. The np-completeness column: an ongoing guide. *Journal of Algorithms*, 6(3):434–451, 1985.

[161] Nicolas Jozefowiez, Frédéric Semet, and El-Ghazali Talbi. Parallel and hybrid models for multi-objective optimization: Application to the vehicle routing problem. In *International Conference on Parallel Problem Solving from Nature*, pages 271–280. Springer, 2002.

[162] Nicolas Jozefowiez, Frédéric Semet, and El-Ghazali Talbi. The bi-objective covering tour problem. *Computers & Operations Research*, 34(7):1929–1942, 2007.

[163] Nicolas Jozefowiez, Frédéric Semet, and El-Ghazali Talbi. Multi-objective vehicle routing problems. *European Journal of Operational Research*, 189(2): 293–309, 2008.

[164] Hossein Jula, Maged Dessouky, Petros Ioannou, and Anastasios Chassiakos. Container movement by trucks in metropolitan networks: modeling and optimization. *Transportation Research Part E: Logistics and Transportation Review*, 41(3):235–259, 2005.

[165] Voratas Kachitvichyanukul et al. A particle swarm optimization for the capacitated vehicle routing problem. *International journal of logistics and SCM systems*, 2(1):50–55, 2007.

[166] Voratas Kachitvichyanukul et al. A particle swarm optimization for the vehicle routing problem with simultaneous pickup and delivery. *Computers & Operations Research*, 36(5):1693–1702, 2009.

[167] Brian Kallehauge, Jesper Larsen, and Oli BG Madsen. Lagrangian duality applied to the vehicle routing problem with time windows. *Computers & Operations Research*, 33(5):1464–1487, 2006.

[168] Sašo Karakatič and Vili Podgorelec. A survey of genetic algorithms for solving multi depot vehicle routing problem. *Applied Soft Computing*, 27:519–532, 2015.

[169] CP Keller and MF Goodchild. The multiobjective vending problem: a generalization of the travelling salesman problem. *Environment and Planning B: Planning and Design*, 15(4):447–460, 1988.

[170] James Kennedy. Particle swarm optimization. In *Encyclopedia of Machine Learning*, pages 760–766. Springer, 2011.

[171] Scott Kiesel, Ethan Burns, Christopher Makoto Wilt, and Wheeler Ruml. Integrating vehicle routing and motion planning. In *ICAPS*, 2012.

[172] Philip Kilby, Patrick Prosser, and Paul Shaw. Guided local search for the vehicle routing problem with time windows. In *Meta-Heuristics*, pages 473–486. Springer, 1999.

[173] Jung-Ug Kim and Yeong-Dae Kim. A decomposition approach to a multi-period vehicle scheduling problem. *Omega*, 27(4):421–430, 1999.

[174] Seongmoon Kim, Mark E Lewis, and Chelsea C White. Optimal vehicle routing with real-time traffic information. *IEEE Transactions on Intelligent Transportation Systems*, 6(2):178–188, 2005.

[175] Niklas Kohl and Oli BG Madsen. An optimization algorithm for the vehicle routing problem with time windows based on lagrangian relaxation. *Operations Research*, 45(3):395–406, 1997.

[176] Niklas Kohl, Jacques Desrosiers, Oli BG Madsen, Marius M Solomon, and Francois Soumis. 2-path cuts for the vehicle routing problem with time windows. *Transportation Science*, 33(1):101–116, 1999.

[177] George Kontoravdis and Jonathan F Bard. A grasp for the vehicle routing problem with time windows. *ORSA Journal on Computing*, 7(1):10–23, 1995.

[178] İlker Küçükoğlu and Nursel Öztürk. An advanced hybrid meta-heuristic algorithm for the vehicle routing problem with backhauls and time windows. *Computers & Industrial Engineering*, 86:60–68, 2015.

[179] V Sivaram Kumar, MR Thansekhar, R Saravanan, and S Miruna Joe Amali. Solving multi-objective vehicle routing problem with time windows by faga. *Procedia Engineering*, 97:2176–2185, 2014.

[180] Wasakorn Laesanklang and Dario Landa-Silva. Decomposition techniques with mixed integer programming and heuristics for home healthcare planning. *Annals of Operations Research*, 256(1):93–127, 2017.

[181] Rahma Lahyani, Mahdi Khemakhem, and Frédéric Semet. Rich vehicle routing problems: From a taxonomy to a definition. *European Journal of Operational Research*, 241(1):1–14, 2015.

[182] Ailsa H Land and Alison G Doig. An automatic method of solving discrete programming problems. *Econometrica: Journal of the Econometric Society*, pages 497–520, 1960.

[183] Gilbert Laporte and Yves Nobert. Exact algorithms for the vehicle routing problem. *North-Holland Mathematics Studies*, 132:147–184, 1987.

[184] Gilbert Laporte and Frédéric Semet. Classical heuristics for the capacitated vrp. In *The Vehicle Routing Problem*, pages 109–128. SIAM, 2002.

[185] Gilbert Laporte, Yves Nobert, and Martin Desrochers. Optimal routing under capacity and distance restrictions. *Operations Research*, 33(5):1050–1073, 1985.

[186] Gilbert Laporte, Hélène Mercure, and Yves Nobert. An exact algorithm for the asymmetrical capacitated vehicle routing problem. *Networks*, 16(1): 33–46, 1986.

[187] Gilbert Laporte, Yves Nobert, and Serge Taillefer. A branch-and-bound algorithm for the asymmetrical distance-constrained vehicle routing problem. *Mathematical Modelling*, 9(12):857–868, 1987.

[188] Gilbert Laporte, Michel Gendreau, Jean-Yves Potvin, and Frédéric Semet. Classical and modern heuristics for the vehicle routing problem. *International Transactions in Operational Research*, 7(4-5):285–300, 2000.

[189] Gilbert Laporte, Roberto Musmanno, and Francesca Vocaturo. An adaptive large neighbourhood search heuristic for the capacitated arc-routing problem with stochastic demands. *Transportation Science*, 44(1):125–135, 2010.

[190] Tzong-Ru Lee and Ji-Hwa Ueng. A study of vehicle routing problems with load-balancing. *International Journal of Physical Distribution & Logistics Management*, 29(10):646–657, 1999.

[191] Jan Karel Lenstra and AHG Kan. Complexity of vehicle routing and scheduling problems. *Networks*, 11(2):221–227, 1981.

[192] Adam N Letchford, Jens Lysgaard, and Richard W Eglese. A branch-and-cut algorithm for the capacitated open vehicle routing problem. *Journal of the Operational Research Society*, 58(12):1642–1651, 2007.

[193] Feiyue Li, Bruce Golden, and Edward Wasil. Very large-scale vehicle routing: new test problems, algorithms, and results. *Computers & Operations Research*, 32(5):1165–1179, 2005.

[194] Feiyue Li, Bruce Golden, and Edward Wasil. The open vehicle routing problem: Algorithms, large-scale test problems, and computational results. *Computers & Operations Research*, 34(10):2918–2930, 2007.

[195] Feiyue Li, Bruce Golden, and Edward Wasil. A record-to-record travel algorithm for solving the heterogeneous fleet vehicle routing problem. *Computers & Operations Research*, 34(9):2734–2742, 2007.

[196] Haibing Li and Andrew Lim. Local search with annealing-like restarts to solve the vrptw. *European Journal of Operational Research*, 150(1):115–127, 2003.

[197] Leon YO Li and Richard W Eglese. An interactive algorithm for vehicle routeing for wintergritting. *Journal of the Operational Research Society*, 47 (2):217–228, 1996.

[198] Miqing Li, Shengxiang Yang, and Xiaohui Liu. Diversity comparison of pareto front approximations in many-objective optimization. *IEEE Transactions on Cybernetics*, 44(12):2568–2584, 2014. ISSN 2168-2267.

[199] Andrew Lim and Xingwen Zhang. A two-stage heuristic with ejection pools and generalized ejection chains for the vehicle routing problem with time windows. *INFORMS Journal on Computing*, 19(3):443–457, 2007.

[200] Qiuzhen Lin, Zhiwang Liu, Qiao Yan, Zhihua Du, Carlos A. Coello Coello, Zhengping Liang, Wenjun Wang, and Jianyong Chen. Adaptive composite operator selection and parameter control for multiobjective evolutionary algorithm. *Information Sciences*, 339:332–352, 2016. ISSN 0020-0255.

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

[202] John DC Little, Katta G Murty, Dura W Sweeney, and Caroline Karel. An algorithm for the traveling salesman problem. *Operations Research*, 11(6): 972–989, 1963.

[203] Ran Liu and Zhibin Jiang. The close–open mixed vehicle routing problem. *European Journal of Operational Research*, 220(2):349–360, 2012.

[204] Arnt-Gunnar Lium, Teodor Gabriel Crainic, and Stein W Wallace. A study of demand stochasticity in service network design. *Transportation Science*, 43(2):144–157, 2009.

[205] Tobie Lourens. *Using population-based incremental learning to optimize feasible distribution logistic solutions*. PhD thesis, Stellenbosch: University of Stellenbosch, 2005.

[206] Miranda Lundy and Alistair Mees. Convergence of an annealing algorithm. *Mathematical Programming*, 34(1):111–124, 1986.

[207] Google Maps. Google maps. https://www.google.com/maps/d/u/0/edit?hl=en&mid=1F7Ap9EO3MyzFudUQ4_Mu48ZlhSY&ll=30.08120233597248%2C117.0881652999999&z=7, 2018. Accessed: 2018-05-11.

[208] Google Maps. Google maps. https://www.google.co.uk/maps/@29.8715435,121.8372319,12z/data=!3m1!4b1!4m2!6m1!1s1IPQurvRAx3x96-V7XEUw6h9kmFs, 2018. Accessed: 2018-05-11.

[209] Yannis Marinakis. Multiple phase neighborhood search-grasp for the capacitated vehicle routing problem. *Expert Systems with Applications*, 39(8): 6807–6815, 2012.

[210] Yannis Marinakis, Magdalene Marinaki, and Georgios Dounias. A hybrid particle swarm optimization algorithm for the vehicle routing problem. *Engineering Applications of Artificial Intelligence*, 23(4):463–472, 2010.

[211] Yannis Marinakis, Georgia-Roumbini Iordanidou, and Magdalene Marinaki. Particle swarm optimization for the vehicle routing problem with stochastic demands. *Applied Soft Computing*, 13(4):1693–1704, 2013.

[212] Renaud Masson, Fabien Lehuédé, and Olivier Péton. An adaptive large neighborhood search for the pickup and delivery problem with transfers. *Transportation Science*, 47(3):344–355, 2013.

[213] Yasuo Matsuyama. Self-organization via competition, cooperation and categorization applied to extended vehicle routing problems. In *International Joint Conference on Neural Networks, 1991., IJCNN-91-Seattle*, volume 1, pages 385–390. IEEE, 1991.

[214] Redouane Mechti, Stephane Poujade, Catherine Roucairol, and Bernard Lemarié. Global and local moves in tabu search: a real-life mail collecting application. In *Meta-Heuristics*, pages 155–174. Springer, 1999.

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

[216] Ryszard S Michalski, Jaime G Carbonell, and Tom M Mitchell. *Machine learning: An artificial intelligence approach*. Springer Science & Business Media, 2013.

[217] Donald L Miller. A matching based exact algorithm for capacitated vehicle routing problems. *ORSA Journal on Computing*, 7(1):1–9, 1995.

[218] Ministry of Tranport, China. Cargo and passenger throughput of ports above designated size. `http://zizhan.mot.gov.cn/zfxxgk/bnssj/zhghs/201804/t20180417_3011025.html`, 2018. Accessed: 2018-04-09.

[219] Snežana Mitrović-Minić and Ramesh Krishnamurti. The multiple tsp with time windows: vehicle bounds based on precedence graphs. *Operations Research Letters*, 34(1):111–120, 2006.

[220] Snežana Mitrović-Minić, Ramesh Krishnamurti, and Gilbert Laporte. Double-horizon based heuristics for the dynamic pickup and delivery problem with time windows. *Transportation Research Part B: Methodological*, 38 (8):669–685, 2004.

[221] Abdolhamid Modares, Samerkae Somhom, and Takao Enkawa. A self-organizing neural network approach for multiple traveling salesman and vehicle routing problems. *International Transactions in Operational Research*, 6(6):591–606, 1999.

[222] RH Mole and Stanley R Jameson. A sequential route-building algorithm employing a generalised savings criterion. *Journal of the Operational Research Society*, 27(2):503–511, 1976.

[223] Magalie Mourgaya. The periodic vehicle routing problem: planning before routing. *Laboratoire de Mathématiques Appliquées de Bourdeaux*, 2004.

[224] Magalie Mourgaya and François Vanderbeck. Column generation based heuristic for tactical planning in multi-period vehicle routing. *European Journal of Operational Research*, 183(3):1028–1041, 2007.

[225] Tadahiko Murata and Ryota Itai. Multi-objective vehicle routing problems using two-fold emo algorithms to enhance solution similarity on non-dominated solutions. In *International Conference on Evolutionary Multi-Criterion Optimization*, pages 885–896. Springer, 2005.

[226] Yuichi Nagata and Olli Bräysy. A powerful route minimization heuristic for the vehicle routing problem with time windows. *Operations Research Letters*, 37(5):333–338, 2009.

[227] Yuichi Nagata and Shigenobu Kobayashi. Guided ejection search for the pickup and delivery problem with time windows. In *European Conference on Evolutionary Computation in Combinatorial Optimization*, pages 202–213. Springer, 2010.

[228] Yuichi Nagata and Satoshi Tojo. Guided ejection search for the job shop scheduling problem. In *European Conference on Evolutionary Computation in Combinatorial Optimization*, pages 168–179. Springer, 2009.

[229] Jenny Nossack and Erwin Pesch. A truck scheduling problem arising in intermodal container transportation. *European Journal of Operational Research*, 230(3):666–680, 2013.

[230] Beatrice Ombuki, Brian J Ross, and Franklin Hanshar. Multi-objective genetic algorithms for vehicle routing problem with time windows. *Applied Intelligence*, 24(1):17–30, 2006.

[231] Ilhan Or. *Traveling salesman-type combinatorial problems and their relation to the logistics of regional blood banking*. Xerox University Microfilms, 1976.

[232] Ibrahim H Osman and Gilbert Laporte. Metaheuristics: A bibliography, 1996.

[233] Ibrahim Hassan Osman. Metastrategy simulated annealing and tabu search algorithms for the vehicle routing problem. *Annals of operations research*, 41(4):421–451, 1993.

[234] Joaquín Pacheco and Rafael Martí. Tabu search for a multi-objective routing problem. *Journal of the Operational Research Society*, 57(1):29–37, 2006.

[235] Ana Cristina Paixao and Peter Bernard Marlow. Fourth generation ports–a question of agility? *International Journal of Physical Distribution & Logistics Management*, 33(4):355–376, 2003.

[236] Dimitris C Paraskevopoulos, Panagiotis P Repoussis, Christos D Tarantilis, George Ioannou, and Gregory P Prastacos. A reactive variable neighborhood tabu search for the heterogeneous fleet vehicle routing problem with time windows. *Journal of Heuristics*, 14(5):425–455, 2008.

[237] Vilfredo Pareto. Course of political economy, 1896.

[238] Junhyuk Park and Byung-In Kim. The school bus routing problem: A review. *European Journal of Operational Research*, 202(2):311–319, 2010. ISSN 0377-2217.

[239] Yang B Park and C Patrick Koelling. An interactive computerized algorithm for multicriteria vehicle routing problems. *Computers & Industrial Engineering*, 16(4):477–490, 1989.

[240] Sophie N Parragh, Karl F Doerner, and Richard F Hartl. A survey on pickup and delivery problems part i: Transportation between customers and depot. *Journal für Betriebswirtschaft*, 58(1):21–51, 2008.

[241] Sophie N Parragh, Karl F Doerner, and Richard F Hartl. A survey on pickup and delivery problems part ii: Transportation between pickup and delivery locations. *Journal für Betriebswirtschaft*, 58(2):81–117, 2008.

[242] AAN Perwira Redi, Meilinda FN Maghfiroh, and Vincent F Yu. An improved variable neighborhood search for the open vehicle routing problem with time windows. In *IEEE International Conference on Industrial Engineering and Engineering Management (IEEM), 2013*, pages 1641–1645. IEEE, 2013.

[243] Victor Pillac, Michel Gendreau, Christelle Guéret, and Andrés L Medaglia. A review of dynamic vehicle routing problems. *European Journal of Operational Research*, 225(1):1–11, 2013.

[244] Sandro Pirkwieser and Günther R Raidl. A variable neighborhood search for the periodic vehicle routing problem with time windows. In *Proceedings of the 9th EU/Meeting on Metaheuristics for Logistics and Vehicle Routing, Troyes, France*, pages 23–24, 2008.

[245] David Pisinger and Stefan Ropke. A general heuristic for vehicle routing problems. *Computers & Operations Research*, 34(8):2403–2435, 2007.

[246] David Pisinger and Stefan Ropke. Large neighborhood search. In *Handbook of Metaheuristics*, pages 399–419. Springer, 2010.

[247] Michael Polacek, Richard F Hartl, Karl Doerner, and Marc Reimann. A variable neighborhood search for the multi depot vehicle routing problem with time windows. *Journal of Heuristics*, 10(6):613–627, 2004.

[248] Jean-Yves Potvin and Jean-Marc Rousseau. A parallel route building algorithm for the vehicle routing and scheduling problem with time windows. *European Journal of Operational Research*, 66(3):331–340, 1993.

[249] Jean-Yves Potvin and Jean-Marc Rousseau. An exchange heuristic for routeing problems with time windows. *Journal of the Operational Research Society*, 46(12):1433–1446, 1995.

[250] Jean-Yves Potvin, Tanguy Kervahut, Bruno-Laurent Garcia, and Jean-Marc Rousseau. The vehicle routing problem with time windows part i: tabu search. *INFORMS Journal on Computing*, 8(2):158–164, 1996.

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

[252] Rong Qu, Edmund K Burke, Barry McCollum, Liam TG Merlot, and Sau Y Lee. A survey of search methodologies and automated system development for examination timetabling. *Journal of Scheduling*, 12(1):55–89, 2009.

[253] Alireza Rahimi-Vahed, Teodor Gabriel Crainic, Michel Gendreau, and Walter Rei. Fleet-sizing for multi-depot and periodic vehicle routing problems using a modular heuristic algorithm. *Computers & Operations Research*, 53:9–23, 2015. ISSN 0305-0548.

[254] Ted K Ralphs, Leonid Kopman, William R Pulleyblank, and Leslie E Trotter. On the capacitated vehicle routing problem. *Mathematical Programming*, 94 (2):343–359, 2003.

[255] Cesar Rego. A subpath ejection method for the vehicle routing problem. *Management Science*, 44(10):1447–1459, 1998.

[256] Jacques Renaud, Fayez F Boctor, and Gilbert Laporte. An improved petal heuristic for the vehicle routeing problem. *Journal of the Operational Research Society*, 47(2):329–336, 1996.

[257] Panagiotis P Repoussis, Christos D Tarantilis, and George Ioannou. The open vehicle routing problem with time windows. *Journal of the Operational Research Society*, 58(3):355–367, 2007. ISSN 0160-5682.

[258] Rita Ribeiro and Helena Ramalhinho Dias Lourenço. A multi-objective model for a multi-period distribution management problem. In *Metaheuristic International Conference 2001*, pages 91–102, 2001.

[259] Ulrike Ritzinger, Jakob Puchinger, and Richard F Hartl. A survey on dynamic and stochastic vehicle routing problems. *International Journal of Production Research*, 54(1):215–231, 2016.

[260] Yves Rochat and Éric D Taillard. Probabilistic diversification and intensification in local search for vehicle routing. *Journal of Heuristics*, 1(1):147–167, 1995.

[261] Stefan Ropke and David Pisinger. An adaptive large neighborhood search heuristic for the pickup and delivery problem with time windows. *Transportation Science*, 40(4):455–472, 2006. ISSN 0041-1655.

[262] Stefan Ropke and David Pisinger. A unified heuristic for a large class of vehicle routing problems with backhauls. *European Journal of Operational Research*, 171(3):750–775, 2006.

[263] Violeta Roso, Johan Woxenius, and Kenth Lumsden. The dry port concept: connecting container seaports with the hinterland. *Journal of Transport Geography*, 17(5):338–345, 2009.

[264] Louis-Martin Rousseau, Michel Gendreau, and Gilles Pesant. Using constraint-based operators to solve the vehicle routing problem with time windows. *Journal of Heuristics*, 8(1):43–58, 2002.

[265] Robert Russell and Wayne Igo. An assignment routing problem. *Networks*, 9(1):1–17, 1979.

[266] David M Ryan, Curt Hjorring, and Fred Glover. Extensions of the petal method for vehicle routeing. *Journal of the Operational Research Society*, 44 (3):289–296, 1993.

[267] Nasser R Sabar, Masri Ayob, Graham Kendall, and Rong Qu. Grammatical evolution hyper-heuristic for combinatorial optimization problems. *Strategies*, 3:4, 2012.

[268] Nasser R Sabar, Masri Ayob, Graham Kendall, and Rong Qu. A dynamic multiarmed bandit-gene expression programming hyper-heuristic for combinatorial optimization problems. *IEEE Transactions on Cybernetics*, 45(2): 217–228, 2015.

[269] Dimitrios Sariklis and Susan Powell. A heuristic method for the open vehicle routing problem. *Journal of the Operational Research Society*, pages 564–573, 2000.

[270] Martin Savelsbergh. A branch-and-price algorithm for the generalized assignment problem. *Operations Research*, 45(6):831–841, 1997.

[271] Martin Savelsbergh and Marc Sol. Drive: Dynamic routing of independent vehicles. *Operations Research*, 46(4):474–490, 1998.

[272] Martin WP Savelsbergh. The vehicle routing problem with time windows: Minimizing route duration. *ORSA Journal on Computing*, 4(2):146–154, 1992.

[273] Kristian Schopka and Herbert Kopfer. *An Adaptive Large Neighborhood Search for the Reverse Open Vehicle Routing Problem with Time Windows*, pages 243–257. Springer, 2016. ISBN 3319208624.

[274] Linus Schrage. Formulation and structure of more complex/realistic routing and scheduling problems. *Networks*, 11(2):229–232, 1981.

[275] Gerhard Schrimpf, Johannes Schneider, Hermann Stamm-Wilbrandt, and Gunter Dueck. Record breaking optimization results using the ruin and recreate principle. *Journal of Computational Physics*, 159(2):139–171, 2000. ISSN 0021-9991.

[276] Weerapat Sessomboon, Kazuyuki Watanabe, Takashi Irohara, and Kazuho Yoshimoto. A study on multi-objective vehicle routing problem considering customer satisfaction with due-time (the creation of pareto optimal solutions by hybrid genetic algorithm). *Transaction of the Japan Society of Mechanical Engineers*, 1998.

[277] Paul Shaw. A new local search algorithm providing high quality solutions to vehicle routing problems. *APES Group, Dept of Computer Science, University of Strathclyde, Glasgow, Scotland, UK*, 1997.

[278] Paul Shaw. 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, 1998.

[279] Yuhui Shi and Russell C Eberhart. Empirical study of particle swarm optimization. In *Proceedings of the 1999 Congress on Evolutionary Computation CEC99*, volume 3, pages 1945–1950. IEEE, 1999.

[280] SINTEF. Best known solution values for solomon benchmark. `http://www.sintef.no/Projectweb/TOP/VRPTW/Solomon-benchmark/100-customers/`, 2018. Accessed: 2018-05-11.

[281] Jim E Smith and Terence C Fogarty. Operator and parameter adaptation in genetic algorithms. *Soft Computing*, 1(2):81–87, 1997. ISSN 1432-7643.

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

[283] Marius Mihai Solomon. Vehicle routing and scheduling with time window constraints: Models and algorithms. Technical report, Northeastern University, 1984.

[284] Jagiełło Szymon and Żelazny Dominik. Solving multi-criteria vehicle routing problem by parallel tabu search on gpu. *Procedia Computer Science*, 18: 2529–2532, 2013.

[285] Éric Taillard. Parallel iterative search methods for vehicle routing problems. *Networks*, 23(8):661–673, 1993.

[286] Éric Taillard, Philippe Badeau, Michel Gendreau, François Guertin, and Jean-Yves Potvin. A tabu search heuristic for the vehicle routing problem with soft time windows. *Transportation Science*, 31(2):170–186, 1997.

[287] El-Ghazali Talbi. *Metaheuristics: from design to implementation*, volume 74. John Wiley & Sons, 2009.

[288] Kay Chen Tan, Yong Han Chew, and Loo Hee Lee. A hybrid multiobjective evolutionary algorithm for solving vehicle routing problem with time windows. *Computational Optimization and Applications*, 34(1):115–151, 2006.

[289] Christos D Tarantilis and Chris T Kiranoudis. Boneroute: An adaptive memory-based method for effective fleet management. *Annals of Operations Research*, 115(1-4):227–241, 2002.

[290] Christos D Tarantilis, Danae Diakoulaki, and Chris T Kiranoudis. Combination of geographical information system and efficient routing algorithms for real life distribution operations. *European Journal of Operational Research*, 152(2):437–453, 2004.

[291] Christos D Tarantilis, George Ioannou, Chris T Kiranoudis, and Gregory P Prastacos. A threshold accepting approach to the open vehicle routing problem. *RAIRO-Operations Research*, 38(04):345–360, 2004. ISSN 1290-3868.

[292] Christos D Tarantilis, George Ioannou, Chris T Kiranoudis, and Gregory P Prastacos. Solving the open vehicle routeing problem via a single parameter metaheuristic algorithm. *Journal of the Operational Research Society*, 56(5): 588–596, 2005. ISSN 0160-5682.

[293] Mohit Tawarmalani and Nikolaos V Sahinidis. A polyhedral branch-and-cut approach to global optimization. *Mathematical Programming*, 103(2): 225–249, 2005.

[294] Mandayam AL Thathachar and P Shanti Sastry. Varieties of learning automata: an overview. *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, 32(6):711–722, 2002.

[295] Dirk Thierens. An adaptive pursuit strategy for allocating operator probabilities. In *Proceedings of the 7th Annual Conference on Genetic and Evolutionary Computation*, pages 1539–1546. ACM, 2005. ISBN 1595930108.

[296] Paul M Thompson and Harilaos N Psaraftis. Cyclic transfer algorithm for multivehicle routing and scheduling problems. *Operations Research*, 41(5): 935–946, 1993.

[297] Paul Michael Thompson, James B Orlin, et al. The theory of cyclic transfers, 1989.

[298] Paolo Toth and Daniele Vigo. *The vehicle routing problem*. SIAM, 2002.

[299] EL Ulungu, Jacques Teghem, Philippe Fortemps, and Daniel Tuyttens. Mosa method: a tool for solving multiobjective combinatorial optimization problems. *Journal of Multicriteria Decision Analysis*, 8(4):221, 1999.

[300] Ziauddin Ursani, Daryl Essam, David Cornforth, and Robert Stocker. Localized genetic algorithm for vehicle routing problem with time windows. *Applied Soft Computing*, 11(8):5375–5390, 2011.

[301] Alex Van Breedam. Improvement heuristics for the vehicle routing problem based on simulated annealing. *European Journal of Operational Research*, 86 (3):480–490, 1995.

[302] HRG Van Landeghem. A bi-criteria heuristic for the vehicle routing problem with time windows. *European Journal of Operational Research*, 36(2):217–226, 1988.

[303] Nadarajen Veerapen, Jorge Maturana, and Frédéric Saubion. An exploration-exploitation compromise-based adaptive operator selection for local search. In *Proceedings of the 14th Annual Conference on Genetic and Evolutionary Computation*, pages 1277–1284. ACM, 2012.

[304] Thibaut Vidal, Teodor Gabriel Crainic, Michel Gendreau, and Christian Prins. A hybrid genetic algorithm with adaptive diversity management for a large class of vehicle routing problems with time-windows. *Computers & Operations Research*, 40(1):475–489, 2013.

[305] Thibaut Vidal, Teodor Gabriel Crainic, Michel Gendreau, and Christian Prins. A unified solution framework for multi-attribute vehicle routing problems. *European Journal of Operational Research*, 234(3):658–673, 2014.

[306] Christos Voudouris, Edward PK Tsang, and Abdullah Alsheddy. Guided local search. In *Handbook of Metaheuristics*, pages 321–361. Springer, 2010.

[307] James D Walker, Gabriela Ochoa, Michel Gendreau, and Edmund K Burke. Vehicle routing and adaptive iterated local search within the hyflex hyper-heuristic framework. In *LION*, pages 265–276. Springer, 2012.

[308] Shanshan Wan and Dongwei Qiu. Vehicle routing optimization problem with time constraint using advanced pbil algorithm. In *IEEE International Conference on Service Operations and Logistics, and Informatics, 2008. IEEE/SOLI 2008.*, volume 1, pages 1394–1398. IEEE, 2008.

[309] Shanshan Wan, Ying Hao, and Dongwei Qiu. Research on vrp using advanced probability learning based evolutionary algorithm. In *Proceedings of the First ACM/SIGEVO Summit on Genetic and Evolutionary Computation*, pages 517–520. ACM, 2009.

[310] Kang-Ping Wang, Lan Huang, Chun-Guang Zhou, and Wei Pang. Particle swarm optimization for traveling salesman problem. In *International Conference on Machine Learning and Cybernetics, 2003*, volume 3, pages 1583–1585. IEEE, 2003.

[311] Xiubin Wang and Amelia C Regan. Local truckload pickup and delivery with hard time window constraints. *Transportation Research Part B: Methodological*, 36(2):97–112, 2002.

[312] Peter Wark and John Holt. A repeated matching heuristic for the vehicle routeing problem. *Journal of the Operational Research Society*, 45(10):1156–1167, 1994.

[313] Min Wen, Jean-François Cordeau, Gilbert Laporte, and Jesper Larsen. The dynamic multi-period vehicle routing problem. *Computers & Operations Research*, 37(9):1615–1623, 2010.

[314] Nicole Wieberneit. Service network design for freight transportation: a review. *OR Spectrum*, 30(1):77–112, 2008.

[315] JAG Willard. Vehicle routing using r-optimal tabu search. *Master's thesis, The ManagementSchool, ImperialCollege, London*, 1989.

[316] Sanne Wøhlk. A decade of capacitated arc routing. *The vehicle routing problem: latest advances and new challenges*, pages 29–48, 2008.

[317] Anthony Wren and Alan Holliday. Computer scheduling of vehicles from one or more depots to a number of delivery points. *Journal of the Operational Research Society*, 23(3):333–344, 1972.

[318] Bin Yu and Zhong Zhen Yang. An ant colony optimization model: The period vehicle routing problem with time windows. *Transportation Research Part E: Logistics and Transportation Review*, 47(2):166–181, 2011. ISSN 1366-5545.

[319] Emmanouil E Zachariadis and Chris T Kiranoudis. An open vehicle routing problem metaheuristic for examining wide solution neighborhoods. *Computers & Operations Research*, 37(4):712–723, 2010.

[320] Günther Zäpfel and Michael Bögl. Multi-period vehicle routing and crew scheduling with outsourcing options. *International Journal of Production Economics*, 113(2):980–996, 2008.

[321] Ruiyou Zhang, Won Young Yun, and Ilkyeong Moon. A reactive tabu search algorithm for the multi-depot container truck transportation problem. *Transportation Research Part E: Logistics and Transportation Review*, 45(6):904–914, 2009.

[322] Ruiyou Zhang, Won Young Yun, and Herbert Kopfer. Heuristic-based truck scheduling for inland container transportation. *OR Spectrum*, 32(3):787–808, 2010.