THE UNIVERSITY *of* EDINBURGH

Applications and Improvements of the Adaptive Large Neighbourhood Search

Syu-Ning Johnn

Doctor of Philosophy
University of Edinburgh
May 31, 2024

Declaration

I declare that this thesis was composed by myself and that the work contained therein is my own, except where explicitly stated otherwise in the text.

(*Syu-Ning Johnn*)

To…gether

Abstract

The Adaptive Large Neighbourhood Search (ALNS) is a popular metaheuristic widely recognised for efficiently solving complex combinatorial optimisation problems within reasonable computing time. The general principle behind the metaheuristic is "destroy-and-repair", which involves a perturbation process that iteratively deconstructs and reconstructs an initial solution to form new incumbents to explore unvisited regions of the solution space in the search for better local optima. This thesis concentrates on the implementation and enhancement of this metaheuristic, intending to further optimise its effectiveness and robustness under diverse settings and real-life scenarios.

Since its initial introduction almost two decades ago, the ALNS metaheuristic has been renowned for its effectiveness in generating good-quality solutions, particularly when addressing routing-related problems. The first direction of this thesis aims to validate this promising efficacy reported in the literature and demonstrate the advantages of the ALNS metaheuristic in tackling challenging real-life problems involving complex and dynamic decision-making processes. We begin with an in-depth analysis of the overall methodological process of the method, scrutinising the impact of the embedded heuristic components and hyper-parameter values on the general performance of the ALNS metaheuristic. Afterwards, we utilise the ALNS metaheuristic to tackle two real-life applications, where the first is a two-echelon location-routing problem that pertains to the food supply chain industry, specifically focusing on the design of a distribution network that incorporates heterogeneous vehicle types, facility and driver-based districts, and multiple commodities. A mixed-integer linear programming (MILP) formulation is proposed and solved with a two-stage decision-making heuristic involving ALNS as an improvement method. The second application arises in the home healthcare industry with integrated service team fleet size, assignment, routing and scheduling decisions under travel time, service time and customer cancellation uncertainties, referred to as the "H-SARA" problem. We model the H-SARA problem using a stochastic MILP formulation and employ Benders decomposition to tackle small to medium-sized instances. Based on the natural partition of our two-stage stochastic model, we can decompose the original problem into a routing master problem and scenario-based scheduling subproblems. In addition, we introduced an ALNS-based warm-start, several valid inequalities, and a closed-form primal formulation to accelerate the computational progress of the Benders algorithm. For larger instances, we develop a tailored two-stage decision support system built upon ALNS, which leverages the increased level of available information at each stage to construct time-sensitive decisions. The results from the computational experiments indicate that our proposed two-stage heuristic is competitive with CPLEX's exact solution methods in terms of providing time and cost-effective decisions.

In the second part of the thesis, we focus on the mechanisms of the ALNS heuristic itself. The ALNS perturbation process is based on sequentially executing a selected pair of destroy and repair operators, each transforming one solution to another in its neighbourhood. This perturbation is governed by a Roulette Wheel mechanism that probabilistically selects operators to determine which neighbourhood of the current solution should be visited. The operator selection mechanism incorporates an adaptive layer, which tracks the historical performance of each operator and periodically adjusts its selection probability to favour the most efficient operator. However, recent studies have indicated that such an adaptive layer within the ALNS framework has limited capability to dynamically choose the best operators, despite being intentionally engineered for this purpose. We follow this up with a study of using Reinforcement

Learning (RL) to improve the performance of the ALNS adaptive layer. We formulate the choice of operators as a Markov Decision Process and propose a hybrid operator selection mechanism based on Deep RL and Graph Neural Networks to enhance the performance of the classical ALNS framework. A key insight and contribution is the proposal of an RL-based operator selection process that is conditioned on the individual solution, which compared to the classical operator selector that is updated periodically by the Roulette Wheel algorithm, enables us to gain insight into the dynamic contribution of each operator in navigating the search towards promising neighbourhoods. We also discuss important considerations such as the size of the operator portfolio and the impact of the choice of operator scales. Lastly, we show that our learning-based ALNS is capable across a variety of routing-related variants with considerable instance scales. Notably, our approach can save significant time and labour costs for handcrafting problem-specific operator configurations.

Lay Summary

In today's rapid-evolving business landscape, industries face the imperative of efficiently processing and analysing ever-increasing volumes of data that are complex and dynamic. Several real-life factors cast challenges for this process, including the constant changes that occur in the customer list, the inherent variability in the demand quantities, and the uncertainties associated with travel and service times. These factors can quickly render any previously established model and decision obsolete. Consequently, it is essential to form an agile and efficient decision-making process for making prompt and proficient decisions, thus ensuring a continuous refinement of decisions in a changing environment.

Given the fast-changing and intricate nature of real-life instances, the emphasis of such a decision-making process is usually placed on achieving good-quality solutions that are reasonably close to the absolute best solution, the so-called optimal solution, with reasonable time and computational effort. In contrast, the pursuit of identifying optimal solutions takes significantly lesser precedence but can be used as the benchmark to evaluate the quality of metaheuristic performance. This leads us to the field of metaheuristics, which focusses on problem-independent solution methods with the goal of finding solutions that are reasonably good, however without a guarantee of optimality.

This thesis revolves around the investigation of a renowned metaheuristic known as the Adaptive Large Neighbourhood Search (ALNS) and presents four key components related to the implementation and improvement of this metaheuristic. We begin with an introduction of the methodological process of the algorithm, providing an analysis of the impact of each embedded component on the general performance of the ALNS metaheuristic. Afterwards, we dive into two real-life industry applications in the field of home healthcare service planning and food supply chain network design, utilising the ALNS metaheuristic to tackle multi-decision problems arising from these two applications. Exact methods (Cutting-plane method and Benders Decomposition) are implemented as benchmark methods. Lastly, recognising the potential room for improvement of this metaheuristic in achieving better performance stability, we integrate Reinforcement Learning techniques as a means to finely regulate and accelerate the direction of the solution-searching process to discover good-quality decisions. Experiment results demonstrate that our learning-based metaheuristic algorithm is able to save significant time and labour costs for handcrafting problem-specific algorithm designs.

Publications and Conferences

Some portions of material in this thesis are drawn from the following papers:

- SN Johnn, VA Darvariu, J Handl, J Kalcsics. (2024). A Graph Reinforcement Learning Framework for Neural Adaptive Large Neighbourhood Search. *Computers and Operations Research [under revision]*
- SN Johnn, VA Darvariu, J Handl, J Kalcsics. (2023). Graph Reinforcement Learning for Operator Selection in the ALNS Metaheuristic. In *International Conference on Optimization and Learning (OLA 2023)* (pp. 200-212). Cham: Springer Nature Switzerland. An alternative version can be found at arXiv: 2302.14678
- SN Johnn, A Miniguano-Trujillo, Y Zhu, A Gupte, J Kalcsics. (2023). Stochastic Programming for an Integrated Assignment, Routing and Scheduling Problem. *EURO Journal on Computational Optimization [under revision]*. The preprint can be found at https://optimization-online.org/?p=21936
- SN Johnn, A Miniguano-Trujillo, Y Zhu, A Gupte. (2021). Solving the Home Service Assignment, Routing, and Appointment Scheduling (H-SARA) Problem with Uncertainties. In *21st Symposium on Algorithmic Approaches for Transportation Modelling, Optimization, and Systems (ATMOS 2021)*. Schloss-Dagstuhl-Leibniz Zentrum für Informatik.

The content in Chapter 3 has been revised from the initial submission to the AIMMS-MOPTA Modeling Competition [213] at which the author and her teammates, Yiran Zhu and Andrés Miniguano-Trujillo, were jointly awarded the First Prize. The ALNS-based two-stage heuristic in Section 3.7 was published in Johnn et al. [113]. In a follow-up work, the authors improved the Benders techniques in Section 3.4 that were applied in Johnn et al. [113] and developed a customised Benders decomposition framework.

An abbreviated rendition of the content in Chapter 5 has been published in Johnn et al. [114]. Selected content from Chapter 6 has been submitted to the Computers and Operations Research journal that is currently under revision.

The contents related to Reinforcement Learning in Chapter 5 and Chapter 6 have been proofread and revised by Dr.Victor-Alexandru Darvariu based on the two collaboration manuscripts above. The relevant paragraphs and graphs have been included in this thesis with his permission. The codes to produce numerical and graphical results for Chapter 5 and Chapter 6 have been collaboratively developed by the author and Dr.Victor-Alexandru Darvariu. The content relates to the lifted formulation in Section 3.6.2 is based on the individual work of Yiran Zhu, and the content relates to the closed-form formulation in Section 3.6.3 is based on the individual work of Andrés Miniguano-Trujillo. Both parts have been included in this thesis with their permissions

for the sake of continuity. The codes produced to generate results in Chapter 3 have been collaboratively developed by Yiran Zhu and the author. The plots and charts in Chapter 3 have been produced by Andrés Miniguano-Trujillo.

I have presented different parts of this research in its different phases of development at the following events in reversed chronological order:

- International Symposium on Locational Decisions (ISOLDE) XVI and EURO Working Group on Locational Analysis (EWGLA) XXVIII, Kaiserslautern and Baden-Baden, Germany, June 2023: presented *'A Hybrid ALNS-Deep Reinforcement Learning Approach for Solving A Location And Routing Problem'*, including complete theory in Chapter 5 and a specific variant in Chapter 6. Awarded the EURO Working Group on Locational Analysis Grant.

- International conference on Optimization and Learning (OLA), Malaga, Spain, May 2023: presented *'Graph Reinforcement Learning for Operator Selection in the ALNS Metaheuristic'*, including part of the theory in Chapter 5 with simplified training and evaluation results. Co-presented with Dr.Victor-Alexandru Darvariu.

- EURO Working Group on Location Analysis (EWGLA), Aveiro, Portugal, September 2022: presented *'Solving the Location-Routing-Districting Problem with Metaheuristic and Reinforcement Learning'*, including part of Chapter 4 and the theory in Chapter 5.

- Operational Research Applied to Health Cares (ORAHS), Bergamo, Italy, July 2022: presented *'Solving the Home Service Assignment Routing and Appointment Scheduling (H-SARA) Problem with Uncertainties'*, including the simplified version of Chapter 3. Presented the same content at a separate poster session.

- EURO Summer Institute on Location Science (ESI), Edinburgh, United Kingdom, June 2022: presented *'Solving the Location-Routing-Districting Problem with a Reinforcement Learning-enhanced ALNS Metaheuristic'*, including mostly content from Chapter 4 with future perspective and preliminary model training results from Chapter 5. Selected and awarded as a conference laureate.

- Approaches for Transportation Modelling, Optimization, and Systems (ATMOS), Portugal, September 2021 [online due to pandemic]: presented *'Solving the Home Service Assignment Routing and Appointment Scheduling (H-SARA) Problem with Uncertainties'*, including the heuristical approach in Chapter 3.

- Modeling and Optimization: Theory and Applications (MOPTA), Lehigh, United States, August 2021 [online due to pandemic]: presented *'Home Service Assignment Routing and Appointment Scheduling Problem'*, including the initial model and results from Chapter 3 as the finalist for the competition.

Acknowledgement

I am deeply thankful to my principal supervisor, Jörg Kalcsics, whose unwavering support has guided me through the uncharted and adventurous realms of the academic world. Jörg has provided invaluable guidance and feedback for my determined and sometimes impulsive research approach, making me feel exceptionally supported in my academic journey for the past four years. His encouragement extended to my engagement in diverse research domains when I exhibited the enthusiasm of a child in a candy shop with a knapsack ever so small. Thank you so much, Jörg!

My four years of research have been marked by numerous ups and downs, zig-zagging, and many exciting collaborations. Reflecting on this path, I would like to express my sincere gratitude to all the people with whom I had the privilege of collaborating: Victor, back then as a Turing doctoral student, whose participation in my proposed project turned the mini project into a two-year interdisciplinary journey of learning, exploration and published work, and Julia, whose kind and generous guidance as an external advisor has been instrumental in ensuring the steady progress of this project. I extend my sincere thanks to the Alan Turing Institute for the Enrichment Scheme, which provided an excellent platform for nurturing this project and expanding my connections with early stage researchers. Many thanks to the fellow PhD students Yiran and Andrés, my teammate at the MOPTA competition back in 2021 and subsequently collaborators in our extended work. Thank you Akshay, for kindly agreeing to be our advisor and supporting both the competition and the collaboration project. Thank you to the MOPTA programme and committee from Lehigh University.

My sincere gratitude to my grandpa, who turns 83 years old this year but nonetheless possesses great curiosity in my research work. I am deeply thankful to my parents, a pair of academic people who have infused my life with academic and research from a young age. Thank you Sunny my 9-year-old cousin for making my life full of laughter during weekends, and particularly, for playing Rainbow Friends with me at the most intense hours of thesis writing and journal submissions. Last but not least, thank you Filippo, for accompanying me through the PhD journey with your countless annoyingly funny jokes, to lighten my day and balance my seriousness. How on Earth could we meet? Thank you, higher dimensional spirit, God or big Buddha, or just pure unbelievably good luck, for bringing this nerd into my life.

Thank you to School of Mathematics for supporting my research, thank you to the Math studentship and to the Maxwell Institute, thank you to the EURO group and the lecturers, particularly Sergio, Julian, Jacek, Ken and Andreas, I still vividly remember the first time seeing all of you back in 2017 when I started my MSc in Edinburgh, thank you for contributing to a wonderful research group that I feel a strong sense of belonging. Thank you to fORum and the fellow PhD students, Tom, Mook, Claire, Barbara, Malte, Monse, Josh, Ivona and Paula. I am proud and overjoyed, though feeling a bit nostalgic, to witness the growth of this family over the years, welcoming new members and students.

List of Figures

List of Tables

Contents

Chapter 1

Introduction

1.1 Motivation

In today's rapid-evolving business landscape, supply chain industries encounter challenges in their decision-making processes due to the complexities of processing and analysing ever-increasing volumes of complex and dynamic data. A spectrum of real-life factors with different time horizons contributes to the potential hurdles in this endeavour, including the constant changes that occur in the customer list, the inherent variability in the demand quantities, the workload balance of service providers and warehouses, the multi-period service consistency requirement, the uncertainties associated with travel and service times, and the last-minute cancellation uncertainty. If not appropriately captured and considered in the model, these factors can quickly render any previously established decision obsolete.

To navigate these challenges, industries face the imperative of forming an agile and efficient decision-making process to ensure a continuous refinement of decisions in a changing environment. Given the intricate nature of real-life instances, the emphasis of such a decision-making process is usually placed on achieving good-quality solutions that are reasonably close to the absolute best solution, the so-called *optimal solution*, which can be achieved with reasonable time and computational effort. Metaheuristics is a field focusing on building problem-independent solution methods that are prompt and proficient, with the goal of finding solutions that are reasonably good, even without a guarantee of optimality.

The Adaptive Large Neighbourhood Search (ALNS) is a renowned metaheuristic that, since its initial introduction almost two decades ago, has been widely recognised for its effectiveness in seeking good-quality solutions to complex Combinatorial Optimisation Problems (COPs) while maintaining a reasonable computational time. The general principle behind this metaheuristic is "destroy-and-repair", which involves a perturbation process that iteratively deconstructs and reconstructs a solution to form new incumbents. The underlying algorithm is local search-based, which is driven by sequentially executing a selected pair of destroy and repair operators that transforms one solution to another in its neighbourhood. Two components sufficiently govern the perturbation: the first is a Roulette Wheel (RW) operator selection mechanism that probabilistically selects operators to determine which neighbourhood of the current solution should be visited to intensify the search. The mechanism incorporates an adaptive layer that dynamically adjusts each operator's selection probability based on its historical performance to favour the most efficient operator. The second is a Simulated Annealing (SA) solution acceptance mechanism that determines whether to

accept a newly-formed incumbent to diversify the search. In this way, ALNS explores unvisited regions of the solution space in the search for better local optima.

This thesis revolves around the investigation of ALNS and presents three key aspects related to the terminology, implementation, and enhancement of this metaheuristic. We begin with an in-depth analysis of the methodological process of the algorithm, scrutinising the impact of the embedded heuristic components and hyper-parameter values on the overall performance of the metaheuristic. Afterwards, we dive into real-life applications to validate ALNS's promising performance and efficacy, as reported in the literature, especially when addressing large-scale problems with routing-related aspects. For that, we utilise the ALNS metaheuristic to tackle two multi-decision problems arising from industry applications.

The first application arises in the home healthcare industry with integrated service team fleet size, assignment, routing and scheduling decisions under travel time, service time and customer cancellation uncertainties, referred to as the "H-SARA" problem. The problem is modelled using deterministic and stochastic Mixed-integer Linear Programming (MIP) formulations. Since the two-stage stochastic model possesses a natural partition that allows the whole problem to be decomposed into a routing master problem and a set of scenario-based scheduling subproblems, we introduce the Benders' decomposition algorithm to tackle small-to-medium-sized instances. To accelerate the Benders algorithm, we present an ALNS-based warm-start and a closed-form primal solution for the subproblem to allow the computational time to increase linearly with the instance size. A set of valid inequalities is also proposed for the master problem that proved beneficial for reducing the number of feasibility cuts needed to be added and overall for speeding up the algorithm's convergence. To tackle large-scale instances, we develop a tailored two-stage decision support system built upon ALNS, which leverages the increased level of available information at each stage to construct time-sensitive decisions. The two-stage system is able to derive near-optimal routing and scheduling plans for 150 customers with 100 scenarios in under 30 minutes.

The second is a two-echelon Location-Routing-Districting (LoRD) problem in the field of food supply chain network design incorporating a number of model features such as heterogeneous vehicle types, facility and driver-based districts, and multiple commodities. A MIP formulation is presented and solved with CPLEX on small instances. In addition, several acceleration techniques including cutting planes and root-node pre-processing are introduced to speed up the computational progress of the exact solution method. To tackle larger instances, a problem-tailored two-stage decision-making heuristic with an embedded ALNS improvement phase has been proposed. Experiment results show that the proposed heuristic gives competitive performances in solving small-scale instances compared to the exact method and is able to tackle larger instances scaling up to 200 customers within 2 minutes.

From the two applications, the good computational results achieved by the ALNS-empowered heuristics indicate their strong competitiveness compared to the benchmarking exact methods, particularly in terms of providing time and cost-effective decisions. The good solution quality and efficiency demonstrated by ALNS, nonetheless, requires substantial domain knowledge and often comes at the expense of a significant development effort and time in hand-engineering the implementation to fit individual problem structures, which unavoidably comes with a plethora of preliminary testings conducted through trial-and-error before securing good performance. For instance, we show in Section 5.6 that determining the size and composition of the list of ALNS operators for a simple capacitated vehicle-routing problem can already take considerable effort to fine-tune. This points to a set of broader issues emerging in the research field

of metaheuristics, especially the tendency to create handcrafted methods that are so often overfitted to individual applications [221]. Moreover, taking the Roulette Wheel adaptive layer as an example, despite being intentionally engineered to dynamically choose the best operators, recent studies have indicated that such an adaptive layer within the ALNS framework still has limited capability for this purpose [232].

Therefore, the final topic we focus on is the mechanism of the ALNS metaheuristic itself. We integrate Reinforcement Learning (RL) techniques as a means to improve the performance of the ALNS adaptive layer and enhance its adaptability to address a wider range of routing-related problems, rather than being limited to any specific one. We formulate the choice of operators as a Markov Decision Process and propose a hybrid operator selection mechanism based on Deep RL and Graph Neural Networks. A key insight and contribution is the proposal of an RL-based operator selection process that is conditioned on the individual solution, which compared to the classical operator selector that is updated periodically by the Roulette Wheel algorithm, enables us to gain insight into the dynamic contribution of each operator in navigating the search towards promising neighbourhoods. We also discuss important considerations such as the size of the operator portfolio and the impact of the choice of operator scales. Finally, experiment results demonstrate that our learning-based ALNS is capable across a variety of routing-related variants with considerable instance scales. Notably, our approach can save significant time and labour costs for handcrafting problem-specific operator configurations.

1.2 Organisation of the thesis

The overflow of this paper includes six chapters. The introduction is followed by Chapter 2, which gives a general introduction of ALNS metaheuristic mechanism and how each of its embedded components influences the performance of the algorithm. The following Chapter 3 and Chapter 4 introduce how ALNS is applied to two applications in the field of home healthcare service design and food supply chain network design. To strive for further improvement in ALNS performance, we integrate the ALNS framework with RL in Chapter 5 to form a hybrid learning-based ALNS model that achieves better solution accuracy and model robustness compared to the classic ALNS algorithm. Finally, Chapter 6 extends the hybrid learning model to several routing-related variants and showcases its good adaptability as well as performance.

Chapter 2

Adaptive Large Neighbourhood Search

In the following, we present the fundamental methodology of local search-based heuristic search in Section 2.1 and problem-independent metaheuristics in Section 2.2, followed by a description of the terminology and inherent mechanisms of the Large Neighbourhood Search metaheuristic in Section 2.3 as a simplified variant of Adaptive Large Neighbourhood Search metaheuristic, which is introduced in Section 2.4. We provide a comprehensive description on the embedded parameters as well as a thorough examination of the parameter impact on the search performance, which is given in Section 2.5.

2.1 Local Search-based Heuristic

We consider an instance of an optimisation problem alongside with a corresponding set of feasible solutions that satisfy all the constraints of the specified problem. Seeking the optimal solution s^* amongst all feasible solutions by an exhaustive enumeration is computationally challenging, particularly for *NP-hard* optimisation problems of substantial size. Instead of applying such a brute-force exhaustive search, a more efficient alternative is a neighbourhood-based local search by minimising (or maximising) a cost function f so that the quality of a solution can be measured via the cost function with respect to a particular set of criteria represented by constraints. In a minimisation problem, any two solutions satisfying $f(s_1) < f(s_2)$ indicate that solution s_1 is better than s_2.

The search space of an optimisation problem, denoted by S, consists of all the feasible solutions to the problem. The neighbourhood of a solution $s \in S$ is a set of feasible solutions that share a significant amount of similarity in their structures. It can be defined as $N(s) = \{s' \in S : s' = \theta(s)\}$, where θ is the transformation function applied to the original solution s that results in the generation of its neighbourhood solution s'. That is, any solutions belonging to the same neighbourhood can be reached by making changes from solution s via some transformation function. For example, in the Travelling Salesman Problem with a feasible solution represented by a single tour that sequentially connects all n vertices through the visit, the neighbourhood of a valid tour $s = [1, 2, 3..., n]$ can be generated using an *insert* function that moves any selected vertex from its original position to another position, an *exchange* function that swaps the visiting sequence of any two vertices, or a *2-opt* function that inverses part of the visiting sequence to eliminate any overlap tour. Consequently, different transformation can yield distinct neighbourhoods.

Local search, also known as neighbourhood search, operates based on the fundamental principle of neighbourhood explorations within the search space. Steepest descent is a basic version of local search that starts from an initial solution and repeatedly applies local changes. In each iteration of the algorithm, a transformation is applied to the incumbent to generate a corresponding neighbourhood. Subsequently, any neighbouring solution that exhibits a smaller total cost function value in a minimisation problem is deemed superior and replaces the previously recorded best solution. Local search-based algorithms seek to converge towards (near-)optimality by iteratively examining and improving upon the neighbouring solutions. It creates a search trajectory that navigates the local search process to scan through multiple adjacent neighbourhoods to discover the best candidate solution. A candidate solution s is considered a *local optimum* in a minimisation setting when it satisfies $f(s) \leq f(s')$ among all the s' in a given neighbourhood $N(s)$, with all its neighbouring solutions being worse off.

2.2 Metaheuristic

Local search algorithms generate and seek for better solutions in the neighbourhood of the current one are referred to as *exploitation* or *intensification*. However, as a heuristic it can only ensure the identification of a local optimum and cannot provide a guarantee of global optimality, for it has a fundamental weakness of terminating prematurely due to potentially being stuck within a neighbourhood and incapable of escaping from the current neighbourhood to explore the rest of the solution space. Conducting a more extensive search allows visits to other unexplored areas of the search space, enabling the escape from the confines of the current neighbourhood and expanding the scope of potential solutions in the pursuit of better results. This is referred to as *exploration* or *diversification*. Since the steepest descent mechanism gets trapped at the first local optimal it visits, an exploration technique can be introduced to mitigate this drawback, allowing the search to continue until the termination criteria is met.

Exploration and exploitation represent two conflicting yet collaborative objectives that involve a delicate trade-off: On the one hand, exploration encourages the examination of unexplored areas of the search space and therefore, has a higher chance of generating significantly different solutions to the current ones and escaping from a local optimum. On the other hand, exploitation focuses on neighbouring candidates of the current solutions, allowing the algorithm to thoroughly explore the neighbourhoods to seek for a better solution.

Metaheuristic algorithms are proposed as problem-independent advanced search methods to improve the performance of a local search heuristic. Their algorithmic framework is embedded with exploitation and exploration as major ingredients to minimise the chance of getting stuck in the clutches of a low-quality local optimum. According to Silver [217], a metaheuristic is an iterative high-level problem-independent algorithm that guides the subordinate heuristics to construct near-optimal solutions efficiently. Based on the classification of the search strategy, metaheuristics can be classified mainly into *local search-based* and *population-based*. The former contains the neighbourhood-based mechanism for modifying and improving a single candidate solution. A few popular algorithms of this type include tabu search [76], simulated annealing [202], large neighbourhood search [212], variable neighbourhood search [94], and greedy randomised adaptive search procedures [71]. Population-based metaheuristics mimic the principle of the natural evolution process by iteratively combining existing solutions from a population of solutions and extracting the predominately high-quality

solutions for the next round. Some popular metaheuristics of this category include particle swarm optimisation [123], genetic algorithms [122], and ant colony optimisation [59].

2.3 Large Neighbourhood Search

In a local search-based heuristic, one must define a neighbourhood structure to design of a neighborhood search approach. Expanding the neighborhood size enhances solution quality, albeit at the cost of increased complexity. Consequently, efficient exploration strategies become pivotal when searching larger neighborhoods. In the literature, various Large-scale Neighbourhood Search (LNS) algorithms have been meticulously devised to address this challenge. These methodologies span across broad classes of methods, such as variable-depth methods employing heuristic search techniques, network flow-based algorithms, poly-time exact techniques leveraging mixed-integer linear programming or constraints programming solvers for exhaustive search of the neighbourhoods, and dynamic programming for specially constructed neighbourhoods, each offering unique approaches and insights. The readers are referred to the work of Ahuja et al. [5] for a more detailed reivew.

In this thesis, we delve into a renowned variant of Large Neighborhood Search, abbreviated as LNS, which is a metaheuristic first introduced by Shaw [212] with the general principle of "relax-and-reoptimise" that iteratively deconstructs and reconstructs a part of the solution to explore unvisited solution space in the search for more promising local optimal. This transformation process is based on sequentially executing a fixed pair of destroy and repair operators, each of which specifies a way of mapping one solution to another solution in its neighbourhood. Such iterative improvement procedure serves as a guide to explore unvisited solution space.

Compared to many other neighbourhood-based metaheuristics that work with a relatively small search space, LNS applies larger changes to the solution and therefore works with large solution neighbourhoods, as indicated by its name. The LNS pseudocode is presented in Algorithm 1.

Algorithm 1 Basic mechanism of LNS

1: $s \leftarrow$ `InitialSolution` and $s^{\text{best}} = s \in S$
2: **while** stopping criteria not met **do**
3: $\quad s' \leftarrow \mathrm{r}(\mathrm{d}(s, o^-, q), o^+)$
4: $\quad \texttt{obj}(s') = \texttt{sum}$ total cost value
5: $\quad$ **if** s' satisfies an acceptance criterion **then**
6: $\quad\quad s \leftarrow s'$
7: $\quad\quad$ **if** $\texttt{obj}(s) < \texttt{obj}(s^{\text{best}})$ **then**
8: $\quad\quad\quad s^{\text{best}} \leftarrow s$
9: **return** s^{best}

The LNS algorithm commences by constructing an initial solution s and recording as the best solution s^{best} encountered so far from the search space S. The neighbourhood of the current solution is created by the consecutive destroy and repair operator pair represented by o^- and o^+, correspondingly. The destroy function $d(\cdot)$ takes in the current solution s and returns an incomplete solution, which becomes the input for the repair function $r(\cdot)$ that returns a feasible solution constructed from the destroyed one. The destroy budget q, or destroy scale, determines the extent of modification applied

to the solution. It can be randomly drawn or set as a hyper-parameter to describe the proportion of the solution that undergoes an iteration of the relaxing and re-optimising search process.

For each iteration, the transformation iteratively finds in the neighbourhood of the current solution s a temporary solution s' that can be promoted or discarded by the acceptance criterion. Various acceptance criteria have been proposed, the first implemented by Shaw [212] being a greedy hill-climbing criterion to ensure that in each iteration, only an improving solution with a strictly better objective value is recorded as the new best solution. The algorithm updates the best solution s^{best} by comparing it to the current solution s during the search. Finally, the algorithm terminates when a stopping criterion is met, typically when the recorded best solution does not change within a certain number of iterations or when a total number of iterations is reached.

2.4 Adaptive Large Neighbourhood Search

The Adaptive Large Neighbourhood Search (ALNS) metaheuristic was first introduced by Ropke and Pisinger [201]. Based on the principle of Shaw's LNS in Section 2.3, ALNS is equipped with two additional mechanisms that determine the performance of the perturbation process. First, unlike LNS with only a single pair of destroy and reinsertion algorithms, ALNS contains a collection of predefined operators and an adaptive layer that iteratively selects and applies destroy and repair operators based on their past success rates. Such an adaptive layer involves an operator selection mechanism, typically an embedded Roulette Wheel (RW) algorithm [159]. The second distinction lies in the solution acceptance mechanism, wherein conventional options such as simulated annealing (SA), record-to-record travel, and threshold acceptance have been commonly employed in the ALNS framework [206], aiming to reduce the chance of the search process getting trapped in non-promising local optima. This is an improvement from the classical LNS framework with a greedy acceptance criterion that accepts only strictly-improving solutions and, as a result, helps to strengthen the exploration of the unvisited solution space. The ALNS pseudocode is presented in Algorithm 2, with each step explained in detail in the remainder of the chapter.

Algorithm 2 Basic mechanism of ALNS

$s \leftarrow$ `InitialSolution`, $s^{\text{best}} = s$, $w \leftarrow$ `InitialScore`
while stopping criteria not met **do**
  $o_i^- \leftarrow$ `RouletteWheel`$(\mathcal{D}, w_i^-)$
  $o_i^+ \leftarrow$ `RouletteWheel`$(\mathcal{R}, w_i^+)$
  $s' \leftarrow r(d(s, o_i^-, q), o_i^+)$
  if $s' <$ `QualityThreshold` **then**
    $s' \leftarrow$ `LocalSearch`(s')
  `obj`(s') = `sum` total cost function value
  if s' satisfies an acceptance criterion **then**
    $s \leftarrow s'$
    if $s < s^{\text{best}}$ **then**
      $s^{\text{best}} \leftarrow s$
  update operators performance scores w
return s^{best}

In every iteration of a classic ALNS, an initial solution s is relaxed and re-optimised

through the iterative application of a pair comprising a destroy operator $o_i^- \in \mathcal{D}$ and a repair operator $o_i^+ \in \mathcal{R}$, which together yield the new incumbent solution s'. A comprehensive overview of various existing operators in the ALNS literature has been provided in Section 2.4.1. A local search improvement is selectively applied to newly-formed incumbents with promising solution quality that typically falls within a certain threshold of the best found solution. This is introduced in Section 2.4.4. The implemented RW operator selection mechanism determines the choice of o_i^- and o_i^+ based on their recorded performance scores w_i^- and w_i^+, which are periodically updated at the end of each iteration to reflect the up-to-date performance of each operator. The detail of the update formula is given in Section 2.4.2. Finally, the algorithm terminates when the stopping criteria is met. This is explained in Section 2.4.3. The following contents involve the Capacitated Vehicle Routing Problem (CVRP) as an example for describing the detailed ALNS mechanism.

2.4.1 Destroy and Repair Operators

An iteration in the ALNS perturbation process consists of the destroy and repair phases, during which a pre-defined number of nodes are removed from the solution, temporarily placed inside a pool, and inserted back into the incomplete solution.

Despite the variety of destroy and repair operators from the existing literature tailored for a variety of combinatorial optimisation problems and variants, the fundamental mechanisms behind remain relatively repetitive and can be classified into random-based, greedy-based and related-based operators.

A *random*-based destroy operator randomly removes nodes from the solution until the total removal target is met. The nodes are commonly referred to as customers, routes, periods, facilities, or requests under specific problem settings. Random removal was first introduced in the work of Ropke and Pisinger [201] and has been modified to fit different problem features in the ALNS literature. A few of its variants include the random removal of nodes from a single route [55], a set of routes [101], the open facilities [101], the same period [7], and removal with restrictions from historical records [143].

A *greedy*-based destroy operator removes the top q nodes with the highest proximity rankings with respect to specific criteria, which can be interpreted as the distance, time, cost, workload, demand level, inventory level, node removal gain, or time difference. A typical representation of the greedy-based destroy is the worst removal operator [201], which removes a set of nodes with the highest costs. A few variants include the removal of nodes with the highest time difference [55], the highest geographical distances from their preceding and following nodes [55], the maximum inventory utilisation [2], the smallest delivery demand [19], the highest deviations from historical records [10], the removal of routes whose distances deviate most from the average route length [55], and removal based on both historical and present information [227].

A *related*-based destroy operator is a hybrid of the previous two types, randomly selecting the first node and removing the remaining $(q-1)$ nodes with the highest relatedness to the first node. It originated from the Shaw removal [212] and is generally captured as the proximity-based destroy that removes a set of nodes which are similar in certain ways. A few variations include the removal of nodes with high proximity in geographical distance [160], Cartesian distance [66], inner-cluster relatedness [184], inner-rectangle cluster relatedness [10], travel time [142], service start time [184], demand [55], a combination of time and demand [55], customer-pair relatedness [157], or based on historical information [156]. In some instances where the first removal is not

strictly a single node, such as the pair removal operator [157] that randomly identify half the removal budget of nodes, then proceed to remove each node's geographically closest node and remove all the pairs collectively from the graph.

After the destruction process, all nodes inside the customer pool will be re-inserted by a repair operator. Similarly, the classification can be applied to the repair operators to divide them into random, greedy, and related-based node insertions. The *random*-based repair operator [190] iteratively inserts the previously-removed nodes back to the solution at a random position. A few variants for the *greedy*-based repairs include insertion with the least increment in objective value [201], the least global cost [199], restrictions from historical records [101], time windows [55], sorting of edges [140], regret value [201], and k-regret value [184]. Finally, a few examples from the *related*-based repairs include insertion with the least insertion cost and noise [101], 2-regret value and noise [55], and k-regret value and noise [6].

2.4.2 Roulette Wheel Selection Mechanism

The RW selection mechanism is a widely-used probabilistic approach employed to select the destroy and repair operators at each iteration while using an adaptive layer to dynamically update the selection probabilities of all operators based on their historical performance. RW is considered the standard selection mechanism due to its well-established popularity in the ALNS literature with an implementation rate of over 99%, according to a recent survey by Mara et al. [159].

In ALNS, the search process is divided into sequential *segments*, each of which begins by assigning an initial score $\psi_i = 0$ to each operator o_i. The score is incremented by a value δ every time a new incumbent solution is generated using an operator pair that includes the specific operator o_i. The increase in score varies depending on the quality of the newly-found incumbent: the score is increased by δ_1 if the incumbent is a new global best solution, by δ_2 for a newly-found incumbent better than the previous solution but not the global best, and by δ_3 for an accepted yet worse newly-found incumbent, where $\delta_1 > \delta_2 > \delta_3$. As each segment K concludes, the cumulated score for each operator i and the number of times N_i it was selected within the segment are utilised to compute a corresponding weight $w_{i,K}$ for this segment, which serves as an estimate of the operator's capability to discover promising solutions and is used to update the operator's weight in the subsequent segment. By incorporating this periodical weight-based estimation on the operator performance, the weight guides the subsequent operator selection as the segment progresses and optimises the overall search efficiency. The adaptive layer works as below. As shown in Equation (2.1), for each operator selected and employed in the current segment K, its weight for the next segment $K+1$ is updated by computing a weighted average of the historical weight $w_{i,K}$ aggregating its previous performance and its average performance in segment K:

$$w_{i,K+1} = \begin{cases} (1-\alpha_{\text{RW}}) \cdot w_{i,K} + \alpha_{\text{RW}} \cdot \frac{\psi_i}{N_i} & \text{if } \psi_i > 0, \\ w_{i,K} & \text{if } \psi_i = 0, \end{cases} \tag{2.1}$$

Within each iteration of the segment, the RW selection algorithm is employed to select a pair of destroy and repair operators based on their associated probabilities, computed by $w^-_{i,K} / \sum_{j\in\mathcal{D}} w^-_{j,K}$ and $w^+_{i,K} / \sum_{j\in\mathcal{R}} w^+_{j,K}$ respectively. These probabilities are determined by the weight $w^{\pm}_{i,K}$ associated with each operator i within any particular segment K. Initially, all operators are assigned an equal weight and hence share an identical selection probability to ensure that each operator is given an equal opportu-

nity to contribute to the search at the beginning. Throughout the search process, the operators accumulate different scores and their selection probabilities evolve to reflect their respective performance. Consequently, the selection algorithm is adjustable to the changing landscape of the search process by assigning higher probabilities to operators that have exhibited more robust capability, hence improving the general efficiency of the algorithm.

Another noteworthy aspect within the traditional ALNS weight adaption lies in the potentially disparate computation times required by different destroy and repair methods. Recent research has proposed more refined weight adaption strategies, such as retaining results based on a predetermined number of iterations or incorporating computational time into the calculation of operator weights to refine the operators quality evaluation. This factor can significantly influence the operator selection preferences when designing the heuristic, given that both solution quality and efficiency in generating solutions are crucial in any heuristical approach.

In this work, we prioritise assessing the solution quality attained by individual operators over their computational efficiency. Hence, we track the operator quality through their associated weights and analyse the efficacy of a sequence of operators by maintaining a fixed limit on the total number of allowed operators, rather than on computational time constraints. This setting is maintained for the remaining of the thesis.

2.4.3 Acceptance and Stopping Criteria

Upon the formation of a new solution through the operator selection process, the ALNS algorithm features an acceptance mechanism to determine whether the newly-formed solution should be accepted as the new incumbent for the subsequent iteration. This acceptance mechanism is typically implemented as the SA metaheuristic. The mechanism aims to diversify the search by allowing the probabilistic acceptance of worse solutions with the potential for further improvement, and therefore reduces the chance of the search process becoming trapped in local optima and terminating prematurely.

The SA mechanism resembles the steel annealing process with an increase or decrease of temperature. Mathematically, the new solution s' is accepted with probability

$$\exp\{\frac{f(s') - f(s)}{T}\} \tag{2.2}$$

where s is the current solution, s' is the newly-found solution, and T is the current temperature. The current temperature progressively decreases at a predetermined rate by multiplying its previous temperature with a cooling coefficient c_τ, resulting in an increasingly stringent acceptance mechanism that is more resistant to accepting worse solutions towards the end as the temperature decreases. With stochasticity incorporated into the acceptance decision, the algorithm can avoid premature convergence and reduce the chance of becoming trapped in a non-promising local neighbourhood. It can also promote ventures into unexplored areas of the solution space in search of better solutions, therefore balancing the algorithm's exploration and exploitation capabilities.

The search process continues until specific stopping criteria are met, including a predefined number of iterations, maximum computation time constraints, or the attainment of a satisfactory solution quality to ensure that the algorithm terminates appropriately while providing a solution meeting the desired requirements.

2.4.4 Hybridisation

The hybridisation of the ALNS metaheuristic with other algorithmic components can potentially enhance the overall metaheuristic framework performance. Such hybrid metaheuristic frameworks can be broadly classified into *pre-processing*, *intra-processing* and *post-processing* integration, based on the attempted strategies proposed by the research community.

As an improvement metaheuristic, ALNS performs as a local search strategy to further strengthen a given initial solution, with its effectiveness during the search process directly influenced by the quality of the initial solution. The attainment of a high-quality initial solution as a pre-processing stage of the ALNS framework can be achieved through the incorporation of exact methods including linear programming [90], constraint programming [105], and dynamic programming [210], and other metaheuristics including greedy randomized adaptive search procedure [36], tabu search [127], and memetic algorithm [96]. Various hybridisation strategies have been proposed to address challenges related to intra-processing in the metaheuristic framework, those include using tabu search to mitigate the low searching efficiency due to re-visiting previously-explored solutions [98], and replacing the repair phase by an exact method [90, 169] or a local search procedure [181]. Within the ALNS framework, local search can be integrated to improve the best solution found during the process after each destroy-repair operator pair. Specifically, local search methods *move*, *swap*, and *2-opt* are frequently applied to intensify the solution by exploring the search space surrounding the incumbent. These implementations are regarded as important components in the ALNS literature to enhance the overall effectiveness of the algorithm. However, since local search is usually computationally expensive, it is typically only applied to promising candidates whose objective values after the repair stage are within a limit of the best-found incumbent (default $x\%$ from initial experiments). Setting a loose limit means that a high proportion of candidates will go through the local search process to compete for the best solution, whereas a tight limit means that only a few outstanding candidates can proceed. A description of the common local search methods is given in Figure 2.1 below.

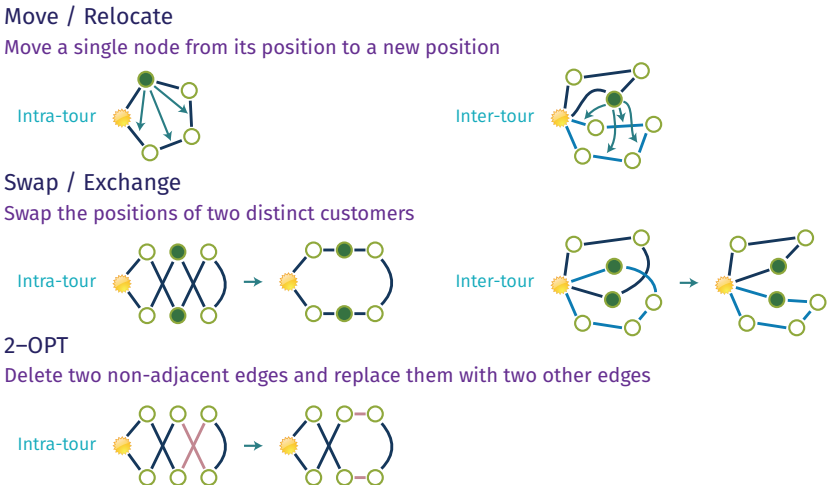


Figure 2.1: A graphical description of the *move*, *swap*, and *two-opt* local search methods for the CVRP.

In post-processing, the output produced by the ALNS metaheuristic can often be

further reinforced through alternative optimisation techniques [159], which can take the form of global refinement applied after the termination of ALNS algorithm as a post-optimisation technique.

2.5 Parameter Testings

In this section, we use CVRP as a sample model for sensitivity analysis to showcase the impact of different hyperparameter value settings on the ALNS performance. The dataset employed for the experimental studies in this section is derived from the Solomon benchmark dataset proposed by Solomon [218] with random (R), clustered (C), and random-clustered (RC) categories that have been widely applied for testing routing-related COPs. The dataset encompasses node-based information such as node coordination and service time for up to 100 customer nodes, together with global information such as vehicle capacity. In the following experiments, the dataset's RC category is chosen to conduct the parameter testings.

Since ALNS is an improvement heuristic that enhances the quality of a pre-constructed initial set of routes, the initial routes are randomly constructed such that the customers are first partitioned into groups of predetermined size and then randomly sequenced into tours that start and end at the same depot. Unless explicitly stated otherwise, the hyperparameters for the ALNS algorithm follow the predetermined values listed in Table 2.1. Nine destroy operators are applied: *random node destroy*, *worst distance destroy*, *proximity distance destroy*, *node neighbourhood destroy*, *route neighbourhood destroy*, *zone destroy*, *pair destroy*, *cluster destroy*, and *historical node-pair destroy*, together with four repair operators including: *greedy repair*, *greedy perturbation repair*, *deep greedy repair*, and *regret-2 repair*. A summary of the operators together with their descriptions is included in Table A.3. The local search considered within the LNS heuristic is conducted in a best-improvement manner until a local optimum is reached across all neighbourhoods.

Table 2.1: ALNS hyper-parameter default values summary

Symbols	**Parameter Names**	**Values**
$total\_its$	number of total iterations	600
$iter$	non-changing iteration stop criterion	50
q	destroy scale	5%
λ	local search threshold	30%
l_k	weight segment length	100
α_{RW}	reaction factor	0.5
c_τ	temperature cooling coefficient	0.99
T	initial temperature	100
δ_1	global best solution score	2
δ_2	local best solution score	1
δ_3	accepted worse solution score	0

The primary focus of this section is not to compute and present the optimal set of ALNS hyperparameter values for the given instance, but rather to illustrate how the distinct alterations in those values can exert influence on the algorithm's ability to navigate the solution space and its performance in terms of convergence speed, solution quality and stability in identifying good-quality solutions. Moreover, since different ALNS destroy and repair operators consume slightly different computational

times that do not directly reflect the algorithm's efficiency in the search process, we consider the number of iterations used by the algorithm as the total computational time, where a reduction in this number denotes enhanced efficiency in the ALNS algorithm.

2.5.1 ALNS Performance Curve

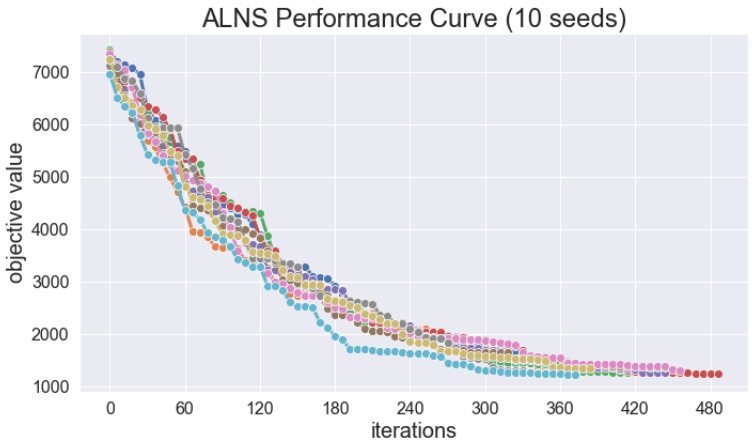


Figure 2.2: The dynamic progress of ALNS algorithm for CVRP instance with predefined hyperparameter values from Table 2.1. 10 repetitions are presented as curves inside the plot, each of which represents the ALNS search process starting from the initial solution until convergence to a best-found solution.

The performance curve depicted in Figure 2.2 showcases the progressive improvement in the total objective value throughout the search process for 10 repetitions of the same algorithm with identical parameters setting. The plot demonstrates a rapid decline in the objective value during the beginning iterations, signifying a steeper improvement in the quality of incumbents. As the simulation progresses, the algorithm's improvement speed slows down and the best-found solution eventually converges toward a certain value that triggers the stopping criteria.

In the subsequent experiments outlined within this section, we undertake alterations to each of the individual predefined ALNS hyperparameter values given in Table 2.1. In each distinct category, we apply adjustments and showcase the interrelation between the hyperparameter values and their impact on the qualities of the best-found solutions, as well as their influence on the efficacy and stability in identifying these solutions. A total of 100 repetitions is computed for each individual parameter value realisation.

2.5.2 Search Horizon Length Analysis

In the first experiment results shown in Figure 2.3, we conduct an ALNS search length analysis by alternating the total number of iterations allowed before the algorithm terminates. We relax the hyperparameters *total_its* and in each algorithm round assign a different value to it. We also set the value of *iter* to be equal to *total_its* so that the algorithm stops when reaching the predetermined number of iterations. Each round contains 100 repetitions that share the same parameters values.

An improvement in the overall solution quality can be observed as the allowed number of total iterations expands, with a noticeable improvement in the occurrence of maximal outliers. Yet this improvement is accompanied by an increased computational

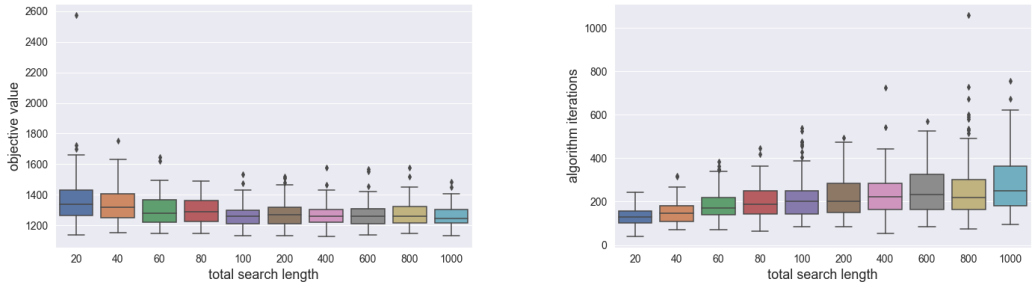


Figure 2.3: ALNS overall iteration length analysis. Each box plot contains 100 repetitions. Left: best-found objective values. Right: total number of algorithm iterations before termination.

time reflected by the growing number of total iterations from the right-hand-side graph. From the objective value graph on the left, we observe that a total search budget of 100+ becomes sufficient to ensure stable performance with a relatively consistent box plot span and maximal outlier value, given that the rest of the parameters remain fixed.

2.5.3 Non-changing Iteration Length Analysis

In this experiment, we examine the impact of the termination condition on the quality and stability of the best-found solution. Specifically, we define a consecutive number of iterations *iter* for which the best-found incumbent must remain unchanged, denoted as the "non-changing iterations" in short, and observe the impact from the length alternation. We allow $total\_its = 3000$ iterations for this experiment to prevent pre-terminating the search process without reaching the total *iter* requirement.

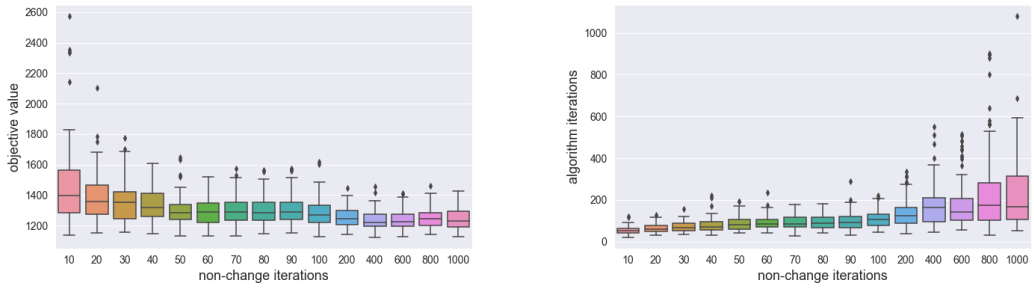


Figure 2.4: ALNS non-changing iteration number analysis. Each box plot contains 100 repetitions. Left: best-found objective values. Right: total number of algorithm iterations before termination.

As shown in Figure 2.4, an increasing length of non-changing iterations facilitates the identification of potentially better local optima, as evidenced by the decrease of average objective values and the box plot span. Meanwhile, the termination condition becomes more challenging to be satisfied when a stringent stopping criterion is imposed, necessitating a more extensive search before termination. This reflects in a rising computational demand, as indicated in the right-hand-side graph.

2.5.4 Local Search Threshold Analysis

The hyperparameter λ governs the threshold for filtering out the good-quality incumbents to apply further intensification using the local search heuristics in Figure 2.1. A lower λ value corresponds to the selection of a smaller proportion of incumbents for subsequent enhancement. We define $total\_its = 300$ to prevent the algorithm from exploiting sufficient iterations to reach solutions of similar level, in order to observe the improvement in best-found solution quality due to the threshold adjustment. The local search iterations are counted inside the total algorithm iteration but not consuming the $total\_its$ budget.

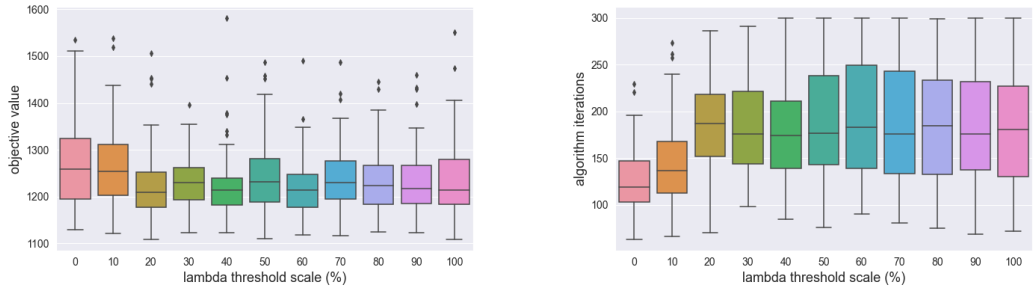


Figure 2.5: ALNS lambda value analysis. Each box plot contains 100 repetitions. Left: best-found objective values. Right: total number of algorithm iterations before termination.

As shown in Figure 2.5, upon increasing the magnitude of λ from 0% to 40%, the box plots become more compact, showing that the algorithm's further intensification on the most promising λ% of incumbents via local search is able to further enhance the best-found solution quality. However, when a larger portion of incumbents is considered, the intensification process is no longer exclusive to the most-promising incumbents and begins to improve a larger number of solutions with a reduced emphasis on their qualities, which significantly lengthens the computational time due to the local search process being applied more frequently.

2.5.5 Operator Destroy Scale Analysis

In this experiment, we aim to test the impact of the destroy scale q on the best-found solution quality. We define $total\_its = 300$ for the same reason as in the previous experiment.

In Figure 2.6, the escalation in destroy scale corresponds to a larger solution neighbourhood, thereby increasing the search space and the likelihood of discovering local optima of better qualities. Evidenced by the algorithm's total iterations on the right, a larger destroy scale help speed up the search progress, allowing the algorithm to navigate more efficiently towards a local optima using fewer iterations. Upon elevating the destroy scale from 50% onward, we observe that the box plots' average best-found objective values converge to a similar level with an almost identical box plot width. This suggests that the algorithm's performance converges towards a comparable level with the large destroy scale, emphasising the repair operators' reconstruction ability to build up the solution from the ground. On the other hand, keeping a rather small-scaled destroy budget stresses selecting and removing the most expensive nodes that contribute more significantly to better solution quality.

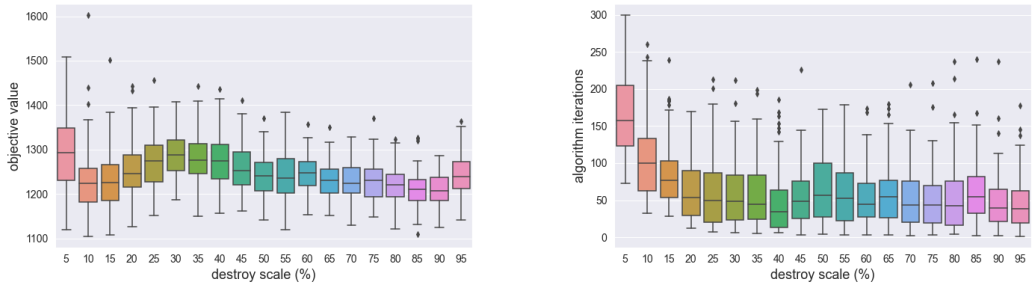


Figure 2.6: ALNS operator destroy scale analysis. Each box plot contains 100 repetitions. Left: best-found objective values. Right: total number of algorithm iterations before termination.

2.5.6 Roulette Wheel Segment Length Analysis

In this experiment, we explore the impact of the segment length adjustment on the ALNS performance. According to Equation (2.1), different values for RW segment length w influence the probability calculation for individual operator selection, which can impact the algorithm's behaviour in navigating the solution space. In the left-hand-

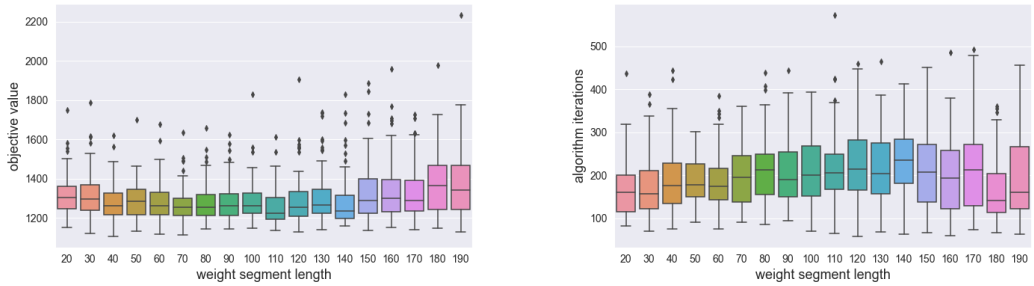


Figure 2.7: RW weight segment length analysis. Each box plot contains 100 repetitions. Left: best-found objective values. Right: total number of algorithm iterations before termination.

side graph in Figure 2.7, we observe a comparable level of average solution quality found across different segment length settings, although the oscillation in algorithm performance becomes more noticeable when the segment length exceeds 150 iterations, reflected by a larger box plot span.

The adjustment of segment length affects the balance between diversification and intensification during the search: a shorter length allows a narrower window for each selected operator to accumulate in-segment performance scores δ, leading to a similar scale for the in-segment total scores across operators and therefore a similar level of operator selection probability in the upcoming segment. This ensures that a broader range of operators share a similar selection likelihood, which limits single-operator domination and increases search diversification. An increased diversification can be two-sided as it helps the algorithm escape premature local optima but can slow down the overall convergence speed. Conversely, longer segments emphasise more on intensification and favour operators with better success rates, resulting in a diminished selection probability for operators not chosen within this segment or not sufficiently contribute to good-quality solutions. Since each operator imparts its unique influence on the solu-

tion transformation process, having a dominated operator therefore may intensify the search.

2.5.7 Roulette Wheel Reaction Factor Value Analysis

The RW mechanism determines each operator's selection probability based on a weighted balance of the historical and in-segment performances controlled by the reaction factor r_{RW}. As shown in Equation (2.1), a smaller reaction factor value emphasises the

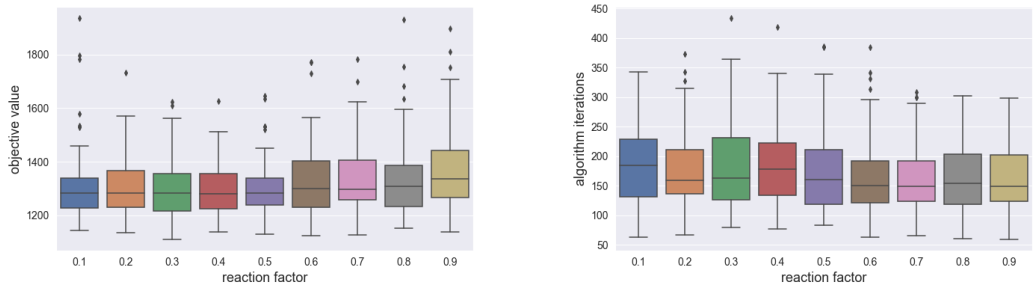


Figure 2.8: RW reaction factor value analysis. Each box plot contains 100 repetitions. Left: best-found objective values. Right: total number of algorithm iterations before termination.

operator's historical performance over the search process, mitigating the influence of any abrupt oscillations in the operator's performance within a short search period. In contrast, a larger value leads to a more intensive focus on the operator's performance within the current segment, which means that the operator's selection probability strongly hinges on its performance recorded in its short-term history. As shown in Figure 2.8, attaining a more balanced allocation of emphasis between historical and current segment performances, by setting the $r_{RW} = 0.5$, contributes to better solution quality in terms of the box plot compactness and outlier level.

2.5.8 Roulette Wheel Grid Search

Given the coexistence of the segment length and reaction factor in Equation (2.1), there is a reasonable presumption of a strong correlation between the two that contributes to the algorithm's performance collaboratively. As a result, we conduct a grid search analysis on the two values. Figure 2.9 showcases the average objective value in a grid search using different combinations of segment length and reaction factor values. In general, the average solution quality demonstrates an upward trend as the reaction factor value increases and the segment length approaches the central value of 100. The global optimal for the grid search appears at weight segment length $w = 120$ and reaction factor $r_{RW} = 0.8$. Nevertheless, we observe that the disparity in final average objective values across diverse parameter combinations is relatively minor, with a maximum marginal improvement of approximately 7% between the best and worst solutions found. Figure 2.10 showcases the exact same experiment with the minimal best-found solutions found in 100 repetitions.

2.5.9 Initial Temperature and Cooling Coefficient Analysis

The ALNS-embedded SA mechanism determines the degree of stochasticity regarding the acceptance of worse-quality incumbents, where a lower current temperature T in

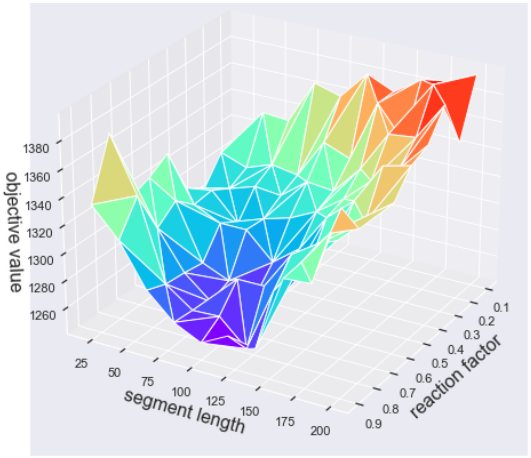


Figure 2.9: RW segment length-reaction factor combination grid search. Each grid is averaged from 100 repetitions of the average best-found solution. Red is higher (worse), purple is lower (better).

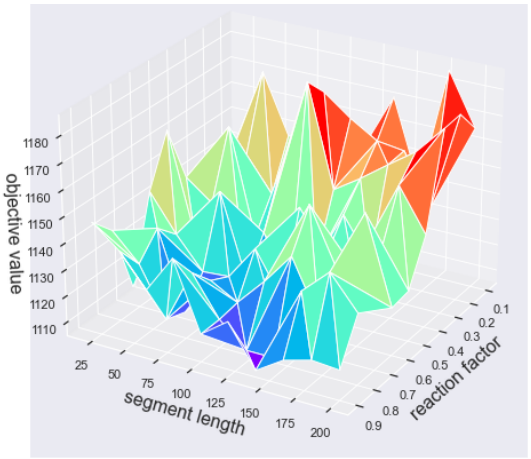


Figure 2.10: RW segment length-reaction factor combination grid search. Each grid is averaged from 100 repetitions of the minimum best-found solution. Red is higher (worse), purple is lower (better).

Equation (2.2) corresponding to a lower probability for worse-quality solution acceptance. In the following two sets of experiments shown in Figure 2.11 and Figure 2.12, we individually alternate the initial temperature T_0 and cooling coefficient c_τ, both of which control the computation of current temperature T in the SA mechanism. Due

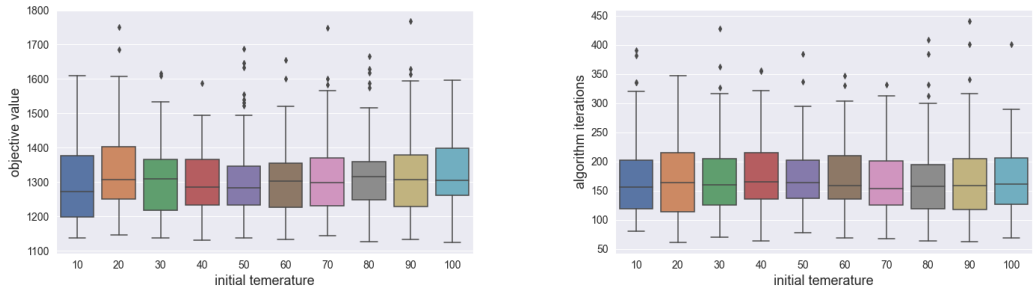


Figure 2.11: SA initial temperature analysis. Each box plot contains 100 repetitions. Left: best-found objective values. Right: total number of algorithm iterations before termination.

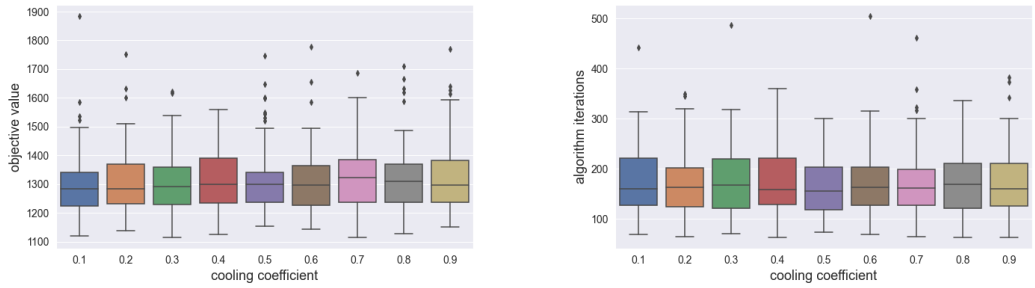


Figure 2.12: SA cooling coefficient analysis. Each box plot contains 100 repetitions. Left: best-found objective values. Right: total number of algorithm iterations before termination.

to the simplicity of the CVRP on Solomon dataset, we observe from both experiments that a change in the probability of accepting worse-quality solution candidates does not significantly alter the final solution quality. We therefore design another experiment in the following section to showcase a more noticeable impact of SA temperature lowering speed on the solution quality.

2.5.10 Simulated Annealing Grid Search

In this experiment, we investigate the collaborative impact of the SA initial temperature T_0 and cooling factor c_τ on the solution quality.

The average objective value in a grid search is given in Figure 2.13. We observe the solution variation between the best and worst solutions found by different combinations of initial temperatures and cooling factor to be approximately 4.5%. However, no obvious trend is found due to the stochastic nature of the acceptance process.

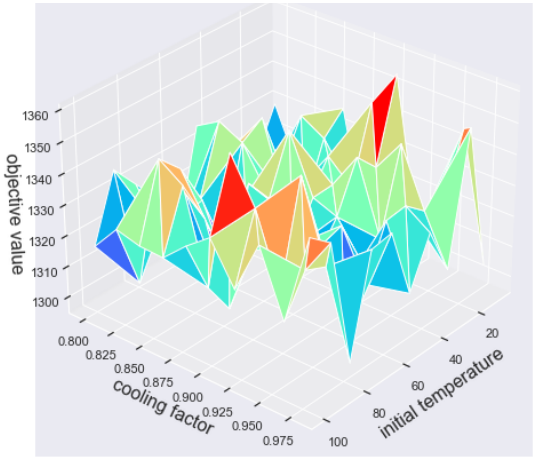


Figure 2.13: SA initial temperature-cooling factor combination grid search. Each grid is averaged from 100 repetitions of the average best-found solution. Red is higher (worse), purple is lower (better).

2.5.11 Cooling Frequency Analysis

In this experiment, we aim to test the impact of temperature lowering speed on the best-found solution quality. We have slightly adjusted the original setting stated in

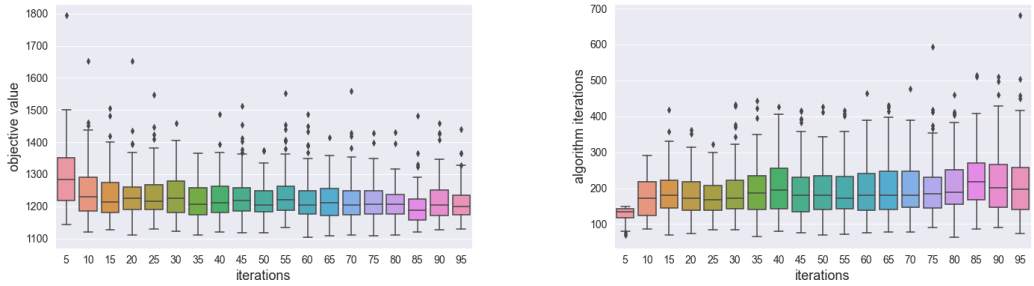


Figure 2.14: Current temperature periodical adjustment analysis. Each box plot contains 100 repetitions. Left: best-found objective values. Right: total number of algorithm iterations before termination.

the work of Mara et al. [159] from single-iteration-based to multi-iteration-based for the temperature update in order to observe a more pronounced impact on the solution quality variations. We define fre to be the number of iterations sharing the same current temperature before the update occurs.

From Figure 2.14, augmenting the number of iterations between successive applications of the cooling factor allows the algorithm to sustain elevated levels of diversification over a longer period, thereby facilitating a more extensive exploration of the solution space. This helps to bring up the average best-found solution quality marginally but unavoidably the computational time as well.

2.5.12 Roulette Wheel Credit Score Analysis

Within each RW segment, each selected operator accumulates scores depending on the newly-formed incumbent's quality. The operator selection probability for the next segment will be computed based on those cumulated scores. In this experiment, since we assume that the hierarchy of credit scores $\delta_1 > \delta_2 > \delta_3$ must hold, we explore the different ratios between the three with δ_1 fixed to 100. Consequently, the grid search only covers half the search space with $\delta_2 > \delta_3$. Notice that as long as the score ratio remains consistent, for instance, at $\delta_1 : \delta_2 = 2 : 1$, their actual values can encompass any numerical values, which in this case, $\delta_1 = 100, \delta_2 = 50$ and $\delta_1 = 4, \delta_2 = 2$ give identical results.

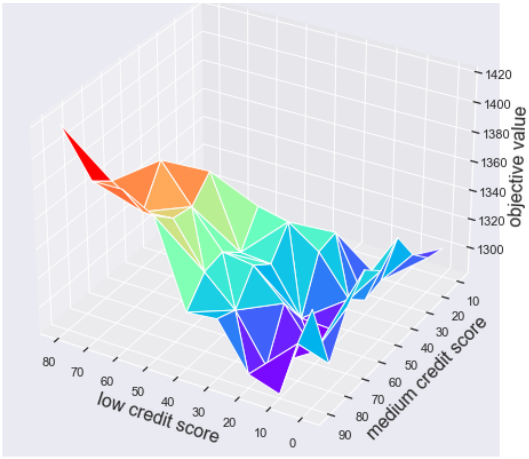


Figure 2.15: RW credit score grid search. Each grid is averaged from 100 repetitions of the **average** best-found solution. Red is higher (worse), purple is lower (better).

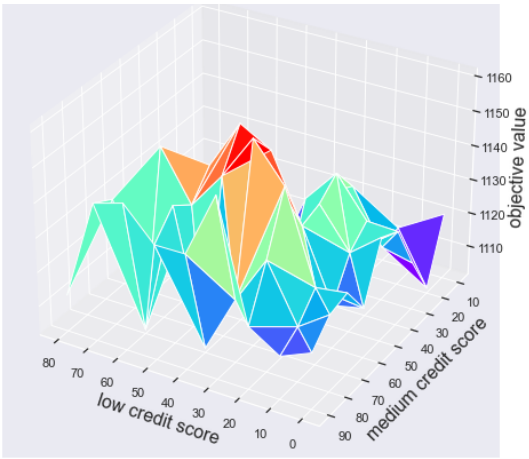


Figure 2.16: RW credit score grid search. Each grid is averaged from 100 repetitions of the **minimum** best-found solution. Red is higher (worse), purple is lower (better).

We observe from Figure 2.15 that when the gap between δ_3 and δ_2 becomes larger,

the score awarded to operators upon creation of worse-off incumbents decreases, which implies the search maintains a higher standard for selecting those operators that intensify good-quality solutions while mitigating the adverse effects of diversification that deviate the search progress. Together with Figure 2.16, we observe that a larger ratio of $\delta_1 : \delta_2 : \delta_3$ allows good ALNS performance Since the performance oscillation caused by different score values is within 5%, we opt for a simplified version of $\delta_1 : \delta_2 : \delta_3 = 2 : 1 : 0$ that gives good average and minimal objective values comparing to the other score settings.

2.6 Conclusion

The previous section summarises the influence of individual hyperparameter values on solution quality within the context of the Solomon instance. Since each COP has a unique solution space determined by its neighbourhood structure, fine-tuning hyperparameter values is generally necessary for individual problems.

Although figuring out the best values for individual hyperparameters is computationally achievable, especially using optimisation tools such as Irace [153], certain probabilistic hyperparameters may require more extensive testings due to their inherent probabilistic nature that poses challenges in terms of control. We show the operator portfolio design in the form of a factorial analysis in Chapter 5 and demonstrate it is difficult and time-demanding to manually select the destroy and repair operators. To cope with this, we propose a learning-based operator selection mechanism in Chapter 5 and test it on a set of COPs in Chapter 6.

ALNS has been applied to many problems, especially those with a routing-related aspect with good performance. In the following Chapter 3 and Chapter 4, we show that the ALNS algorithm can be applied to two complex COPs, each incorporating multiple decisions, and the algorithm is able to effectively derive good-quality solutions with good computational time compared to the exact methods. The parameters are predetermined using preliminary testing results on the given instances or those from the existing literature. The best combination of parameters can only be discovered via a complete factorial analysis based on all ALNS parameters, which is excessively challenging and time-consuming.

Chapter 3

Application of ALNS in Home Healthcare Problem

3.1 Introduction

The home service industry encompasses businesses that provide a range of services to individuals in their homes. Home Health Care (HHC), also known as domiciliary care or hospital-at-home services, is an important part of home services. This type of care involves providing medical, paramedical, and social services to patients in out-of-hospital settings such as private homes, care facilities, boarding homes, hospices, and shelters [134]. In recent years, the HHC sector has experienced significant growth due to various societal changes, including shifts in family structures, increased life expectancy, an ageing population, hospital overcrowding, and the spread of infectious diseases. All of these factors contribute to the expansion of the HHC industry. According to Genet et al. [79], 1% to 5% of the total public health budget in Europe is spent on HHC. In the US, the expenditure on HHC was over $102 billion in 2018, which is equivalent to a 30% increase from 2013 [95].

The typical business requirements for HHC organisations are to decide on the number of professional service teams to deliver services to geographically distributed customers, the assignment of the service teams to customers, the sequences of customer visits, and the scheduling of appointments to all customers with service demands. These four decisions form the *Home Service Assignment, Routing, and Appointment scheduling* (H-SARA) problem, which was presented for the 13th AIMMS-MOPTA Optimization Modeling Competition [213]. The H-SARA is a multifaceted optimisation problem that integrates planning decisions at tactical and operational levels and is related to a set of widely studied problems in academia and industry. It considers real-world complexities and uncertainties, such as travel times and service durations, making it a crucial problem to be addressed by HHC organisations.

Uncertainty and perturbation are inevitable in the healthcare system and cannot be neglected in real-life situations, where a planned schedule is often unlikely to be conducted throughout the planning horizon without any disruption. Amongst all uncertainties, variability in travel times, service times, and patients' availabilities are the most often appearing real-world unpredictabilities and have been widely studied in practice [58]. As a result, involving some uncertain aspects from real life within the healthcare decision framework is essential in producing practical solutions. The H-SARA problem considers the following real-life aspects: First, travel times can vary due to traffic congestion, potentially resulting in delays, especially during peak hours.

Second, the actual service duration at each patient's location is intrinsically random. Third, patient presence during the decision-planning stage can be uncertain. For instance, patients can cancel at short notice after the healthcare provider has already made the routing and scheduling decisions, leading to abrupt modifications of service plans and timetables. We identified three customer cancellation scenarios on the service day and their associated policies: in-time cancellation, last-minute cancellation, and no-presence, all of which are further explained in Section 3.7.2. All three uncertainties can result in service teams incurring idle time or overtime, or patients experiencing waiting times.

The above uncertainties encompass both continuous (travel and service times) and discrete types (customer cancellations). Given their distinct nature, we assume in the following that customer cancellations are revealed right before the planning, whereas travel and service time uncertainties are intrinsic to the planning. To account for the uncertainty of the former, we delineate the problem into two distinct stages following a typical chronological timeline, the initial planning stage and the tour refinement stage, plus a post-service evaluation step, as depicted in Figure 3.1. This temporal progression facilitates a seamless daily operations rundown. The initial planning stage is carried out on the afternoon of the day prior to the actual service day. The planning uses as input all currently scheduled customer visits for the following day, i.e., all customer cancellations are assumed to have been received, plus the uncertain travel and service times. As output it determines the number of service teams to be used and their routes as well as an appointment time window for each customer of a certain length, e.g., 4 hours, which is then communicated to the customers.

In the morning of the actual service day, the tour refinement stage is carried out by a certain cut-off time and before the teams head out on their tours, considering all further customer cancellations received by then and also possible updates to travel and service times (any customer cancellations received after the cut-off time will have to be addressed ad-hoc by each team). The tour refinement stage allows re-optimising the number of service teams, the team-customer allocations and the service routes in order to better balance workload and minimise overtime and waiting and idling times. However, it will generate and communicate narrower appointment time windows to customers, e.g., of 30 minutes length, which should be nestled within the time windows that was communicated to customers on the previous day. To ensure the robustness of the solution of the tour refinement stage, an out-of-sample simulation can be run that provides feedback to the planner. To complete the planning process, a post-service evaluation step takes place at the end of the service day, allowing decision-makers to evaluate the service teams' workload and, e.g., compensate teams which ran into overtime by assigning them fewer customers on the next day.

To solve the models for the initial planning and tour refinement stage, we propose an exact method and a heuristic. With respect to the three types of uncertainties, both methods consider travel and service time uncertainties included via a finite set of scenarios, but assume the list of customers to be served to be known in advance.

Concerning the exact method, given the distinct nature of uncertainties that encompass both continuous and linear types, we make the assumption that the model should be solved for once before the cut-off time after all the cancellations are known. Therefore, the stochastic model follows the in-time cancellation policy assuming deterministic customer cancellations that occur either in the previous day or on the actual service day before the cut-off time. Consequently, the stochastic model is solved only with non-cancelled customers to get a set of valid routes, with the second stage encountering various travel and service time scenarios.

For the heuristic approach, we follow the last-minute cancellation, so that in the first stage we include all customers in the routes before the actual service day for the decision-making. After receiving updates from customers in the second stage, we remove the customer set who cancel their service requirements on short notice and update the decisions correspondingly. Therefore, we consider all three types of uncertainties within the heuristic framework.

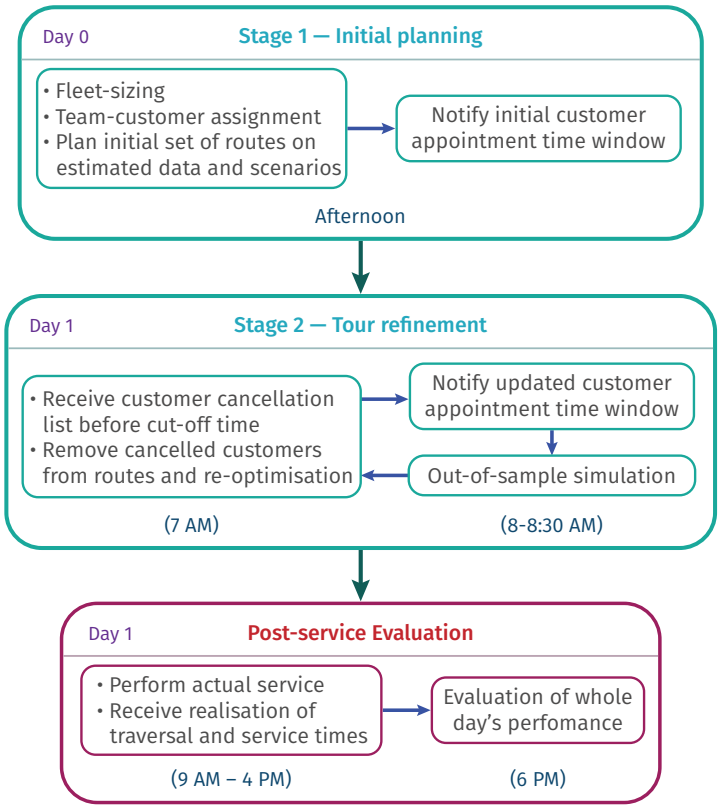


Figure 3.1: Daily operations rundown with chronological timeline

For the exact solution methods, a stochastic Mixed-Integer Linear Programming (MIP) model is proposed for the H-SARA problem. To get a main idea on the computational performance for a single scenario, we first simplify the model into its single-scenario deterministic counterpart and then use the CPLEX solver to tackle small-scale instances with two customised acceleration approaches to find the optimal solution. In order to gain a better understanding on the impact of uncertainties on the solution, we then resume the generalised multi-scenario model setting and develop an exact solution method using Benders' decomposition [24]. Realising the natural partition of our stochastic model, we take the first stage as the scenario-independent master problem that decides which set of routes to visit and treat each scenario inside the second-stage recourse model as a subproblem fixing the time-dependent decisions for the service teams and customers. Results show that the CPLEX built-in Benders algorithm delivers solutions with smaller gaps than the whole MIP model solved by CPLEX for small to medium-scale instances. In general, CPLEX Benders possesses a notable advantage, particularly as the number of subproblems increases. Yet, many existing works indicate that the CPLEX built-in Benders having worse performance than methods such as Branch-and-Cut, which emphasises potential of customising Benders framework based on specific problems.

Encouraged by the result, we develop a customised Benders framework and derive a problem-specific Benders' decomposition formulation tailored to the problem structure. Besides our proposed Benders optimality and feasibility cuts, we further introduce a set of valid inequalities for the master problem that proved to be very beneficial for reducing the number of feasibility cuts needed to be added and overall accelerating the convergence of the algorithm. Moreover, we include an ALNS-based warm-start to speed up the algorithm convergence process. We also present a closed-form primal formulation for the subproblem to allow the computational time to increase linearly with the instance size. Results show that our customised Benders algorithm, although with its capacity constrained by efficiency limitations of the Python language, can nevertheless rival the performance of the state-of-the-art competitive solver when the instance size increases. Overall, our Benders technique can deliver good-quality solutions with better efficiency than the whole MIP model solved by CPLEX. Our customised Benders is also competitive with CPLEX built-in Benders for small-scale instances.

To be able to solver larger instances in reasonable time, especially for the tour refinement stage, we also develop a tailored two-stage heuristic approach that provides high-quality and in-time solutions. The heuristic utilises the ALNS algorithm to create routes with balanced service teams workloads under reasonable computational time. We also introduce a two-stage appointment notification system to increase customer satisfaction. Results show that our two-stage heuristic is competitive to CPLEX's exact solution methods in providing time and cost-effective decisions for large-scale instances and can update previously-made decisions based on an increased level of information.

The organisation of this chapter is as follows. In Section 3.2, we review the studies related to human resource planning in HHC. In Section 3.3, we define the H-SARA problem first as a deterministic MIP model under a single scenario and then extend to a two-stage stochastic MIP model with multiple scenarios. In Section 3.4, we propose an exact method that employs a root node approach and acceleration techniques. The efficacy of our approach is demonstrated through implementation with CPLEX on the single-scenario deterministic model. Then, we present the Benders' decomposition for the multi-scenario MIP model, showcasing preliminary computational results from the CPLEX built-in Benders in Section 3.5. Recognising the inherent capabilities and advantageous problem-specific structures, as expounded in Section 3.6, we further dive into a customised Benders' decomposition framework formulated to address the multi-scenario MIP model. We propose several acceleration approaches in alignment with these distinctive structural characteristics to speed up the convergence process. We present a tailored two-stage heuristic solution method as a light-weighted decision-making process in Section 3.7. Computational results and performance analysis for both the exact methods and heuristic are presented in Section 3.8. Finally, the conclusions follow in Section 3.9.

3.2 Literature Review

Our problem consists of determining the number of required agents (in the context of H-SARA these would be caregivers) and their visiting routes to minimise the total hiring, service, and travel expenses while satisfying a set of constraints. It relates to a set of widely studied academic and industry problems. In this section, we summarise several relevant problems in the context of HHC.

3.2.1 Related Problems

The Vehicle Routing Problem (VRP) is a significant issue at an operational level. The VRP has been extensively studied in the literature and has several variants and applications [29, 135, 146]. The problem studied in this paper generalises the VRP to determine the necessary personnel to meet demand at the lowest cost. This approach includes a fleet-sizing aspect, commonly known as the staffing or personnel planning problem, to balance fixed team hiring costs and routing expenses [200]. Selecting a proper number of service teams is critical to profitability: as an inadequate number of teams will result in larger average workload and higher risk of experiencing service overtime, and redundant teams will lead to a low employee utilisation rate and unused resources. Many HHC problems also involve a patient-to-staff or patient-to-slot assignment aspect over a planning horizon [32], which is considered alongside other planning decisions.

Another related operational decision is *scheduling*, which refers to the chronological allocation of tasks to workers such that the list of tasks is accomplished within the shortest amount of time and with minimal time clashes. For example, in healthcare, *appointment scheduling* specifies the allocation of patients to staff for services, including care visits and elective surgeries. It aims at constructing staff schedules that satisfy patient requirements. From a service provider's perspective, each customer visit is scheduled as part of a service team's timetable in sequential order [87]. The VRP with Time Windows (VRP-TW) [56] is a VRP variant with a scheduling aspect, stressing that vehicle arrival or departure times must satisfy additional customer availability requirements. The difference between scheduling and the VRP-TW is the pro-activeness of the decision-makers in appointing the visiting time window due to the initial routing criteria instead of customer-imposed time requirements.

On an operational level, our problem can be formulated as the Routing and Scheduling Problem (RSP), which involves producing for every caregiver a job schedule and a visiting sequence at the minimum cost respecting regulatory and operational constraints [58]. RSP in the HHC context has received growing attention and has been the predominant research focus [174]. For comprehensive reviews of RSP in the context of HHC, we refer the readers to the work of Fikar and Hirsch [72] with a focus on the standard parameters, Cissé et al. [38] with analysis on OR models applied to RSP, and Di Mascolo et al. [58] with specification on the RSP constraints and objectives in RSP. Besides, Gutiérrez and Vidal [89] review the main models and decisions related to the design and delivery of HHC services. Furthermore, a more recent survey by Grieco et al. [86] gives a systematic review of a broader framework of OR literature in HHC logistics management. Finally, a comprehensive overview of planning decisions in health care can be found in Hulshof et al. [111].

3.2.2 Uncertainties

There is a growing trend in the recent HHC literature that deals with uncertainties such as travel times, service times variability, or customer cancellations. Yuan et al. [253], Shi et al. [214], Zhan and Wan [255] and Shi et al. [216] consider uncertain service times or demand for their RSP models. Rest and Hirsch [198] consider time-dependent travel time for the staff routing and rostering problem. Several works involve customer cancellation or departure in a routing and scheduling context. Eveborn et al. [67, 68] consider changes in staff and patients' availabilities in their staff routing and scheduling problem in the home care domain. Trautsamwieser et al. [230] considers personal

preferences from both patient and nurse that allow refusal of service. Nickel et al. [177], Fikar et al. [73], Cappanera et al. [31], and Gomes and Ramos [82] consider the uncertainty of patient existence due to patient cancellation or new requests. Finally, Lin et al. [147] considers sudden incidents, which include nurse leave, patients requesting changes in their assigned timeslot, and patient cancellation. Furthermore, a few more complex models in the literature have involved more than one type of the above uncertainties. Particularly, Shi et al. [215], Yuan et al. [254], Liu et al. [149], Naderi et al. [170], Nikzad et al. [178], Shahnejat-Bushehri et al. [211], and Yang et al. [251] consider travel and service times uncertainties. Han et al. [92] considers uncertainties in staff response time and customer cancellation in the form of no-shows. A review of other types of uncertainties apart from the above three can be found in the work of Di Mascolo et al. [58]. Nevertheless, to our best knowledge, no research integrates all three types of uncertainties together with the fleetsizing, assignment, routing, and appointment scheduling decisions in the context of HHC.

Many solution methods have been applied to handle uncertainty, amongst which *stochastic programming* has been one of the most selected methods that shows good performance in the HHC context [97, 256, 58]. A common approach applied in stochastic programming models is two-stage optimisation. In the first stage, an initial solution is created before the cancellation parameters are revealed in the second stage, such that first-stage decisions should possess sufficient flexibility to guarantee the feasibility of second-stage recourse actions. Although this work focuses on a problem emerging in home service delivery, decomposition methods, specifically two-stage optimisation, can be easily found in practice across service domains. Many international shippers including DPD [61] and Royal Mail [53] have now adapted to similar concepts in their last-mile deliveries: they first assign an estimated time slot to all customers based on the pre-collected information, then re-assign a narrower time slot on the actual day of delivery when more information is known (e.g., customer delivery sequence, cancellations). This multi-stage decomposition approach also suits the real-life circumstances in the HHC service industry, where last-minute service cancellations, i.e. customers cancelling their requests after being given an appointment time, are allowed.

3.2.3 Benders' Decomposition

The use of decomposition techniques, specifically Benders' decomposition, to solve integrated large-scale NP-hard problems like ours has gained considerable significance in recent years [170, 194]. One notable application of Benders' decomposition is the *L-shaped method* [194], which is designed for addressing stochastic problems. The method has been shown to be particularly effective in solving these kinds of problems and has therefore positioned itself in the field. The L-shaped method breaks the problem down into smaller sub-problems, which can then be solved separately and combined using cuts to find the overall solution. This approach can effectively tackle complex problems that would otherwise be difficult to tackle as a whole using traditional methods. Numerous extensions of the classical Benders framework have been developed to address a broader range of optimisation problems, including combinatorial Benders' decomposition [236], logic-based Benders' decomposition [107], and Benders' decomposition involving integer subproblems [136] in the first or second stage, as well as nonlinear functions and constraints [80]. In the HHC context, Hashemi Doulabi et al. [97], Zhan et al. [256] both consider stochasticity in their models. The former studies the stochastic VRP with synchronised staff visits and uncertain travel and service times and proposes an L-shaped algorithm as well as a branch-and-cut implementation as solution methods

for instances up to 20 customers. In contrast, the latter studies an integrates routing and scheduling problem with stochastic service times and proposed an L-shaped algorithm for instances up to 10 customers and a problem-specific heuristic for larger-sized instances.

3.2.4 Novelties and Contributions

Based on our literature review, the main contribution of this work is the mathematical treatment of a problem integrating the following four planning decisions: fleet-sizing, assignment, routing, and scheduling. These decisions are jointly integrated into the solution framework to contribute to a comprehensive decision-making process. Travel times, service duration, and cancellation rates are considered jointly as uncertain quantities, which to our best knowledge, has not been investigated in the literature before.

For the solution methods, we develop deterministic and stochastic MIP models that are tackled using CPLEX and its built-in Benders' decomposition. We then dive into the customised Benders' decomposition based on the natural partition of our two-stage stochastic model and compare the performance with CPLEX exact method and the built-in Benders on small-sized instances. Particularly, we propose a set of problem-specific Benders cuts and valid inequalities for the master problem, as well as several acceleration techniques to speed up the convergence of the algorithm, including the closed-form primal formulation for the subproblem to allow the computational time to increase linearly with the instance size and a metaheuristic-based warm-start algorithm to accelerate the convergence process. An alternative two-stage heuristic approach is proposed for tackling medium to large-sized instances, allowing decision-making based on imperfect information before the actual customer demands are revealed and updating existing solutions with increased information.

3.3 Mixed Integer Programming Model

To investigate the problem, we introduce two distinct MIP models, namely a deterministic single-scenario model referred to as H-SARA-1 and a stochastic counterpart considering multiple scenarios, denoted as H-SARA-2.

3.3.1 Problem Statement

Let a service area be represented by a directed graph $G := (V, A)$. The node set V encloses the customer set $[\![1, n]\!] := \{1, \ldots, n\}$, a single depot $\{0\}$, and its duplicate $\{n+1\}$. The arc set consists of all the ordered pairs linking each pair of customers, one link from the depot to each customer and another from each customer back to the duplicated depot; namely $A := \{(i,j) : i \neq j, \forall i, j \in [\![1, n]\!]\} \cup \{(0,j) : \forall j \in [\![1, n]\!]\} \cup \{(i, n+1) : \forall i \in [\![1, n]\!]\}$. Notice that any self-loop arc (i, i) is not formed or allowed in the model. The service for all n customers of known geographical location is provided by a group of no more than m homogeneous service teams, each of which makes a single trip starting from and returning to the depot. We aim to partition the set of customers into an optimal number of groups, each visited exactly once by an individual service team in an explicit visiting sequence, and to determine customer appointment timeslots prior to the actual visits. The solution should satisfy time and capacity constraints given by the customers and the service teams. Customers must be informed of their appointment time slots, either on the day before after the initial planning or on the service day before the cut-off time (8 am) or the departure

of the assigned service teams from the depot, whichever is earlier. Lastly, for the multi-scenario stochastic model (H-SARA-2) with different travel and service times realisations, the probability distributions associated with travel and service times are known, and all the instances are assumed to be independent.

3.3.2 Two-index *versus* Three-index Traversal Decision Variables

One core feature of the proposed models for H-SARA is to determine which set of arcs to traverse. The two-index traversal decision variable $x_{i,j}$ indicates if the arc (i, j) is traversed by a service team without identifying which vehicle makes the trip. The variable follows a two-index vehicle flow formulation as initially proposed in Laporte and Nobert [138], which is an extension of the model by Dantzig et al. [49]. A three-index vehicle flow formulation can be found in many existing works. It involves a more specified traversal variable $x_{i,j,k}$ indicating whether arc (i, j) is traversed by an indexed vehicle or team k. As a matter of fact, a two-index formulation has been applied more successfully than its three-index counterpart when it comes to computational complexity [135]. Avoiding the identification of a specific team by its index results in a significant reduction in the number of variables and constraints. Due to the homogeneous requirement for the service teams in H-SARA, we have selected a two-index formulation for our proposed models for its simplicity, although we acknowledge that a three-indexed model may also be an advantageous choice.

3.3.3 Deterministic MIP Model

We first propose a deterministic H-SARA-1 model to gain a general understanding on the model's computational performance for a single scenario.

Parameters

Let $s : [\![1, n]\!] \to \mathbb{R}_{\geq 0}$ be a non-negative function defined over the set of customers, representing expected service durations (from a known distribution). The expected traversal travel time on arc (i, j) is given by a positive matrix $\tau = [\tau_{i,j}]$ defined over the set of arcs A. Likewise, the distance from i to j is labelled $d_{i,j}$. In this formulation, symmetry of τ and d is not required, capturing possible discrepancies in the underlying road network; i.e., city topography and street layout. Let $p : [\![1, n]\!] \to \mathbb{R}_{\geq 0}$ be a probability mass function defined over the set of customers, such that for each customer $i \in [\![1, n]\!]$ the probability of service cancellation of i is given by p_i. The cost of hiring a homogeneous team $i \in [\![1, m]\!]$ is taken as a constant f_m, where m is the maximum number of service teams available despite the hiring status. To convert the unit travel time into cost, we use a constant $\lambda \in \{1/2, 1, 2\}$. The regular allowed working time per day is given by $L \geq 0$. Working times are expected to be within the interval $[0, L]$, yet we anticipate and allow overtime by an amount of up to $\theta > 0$. Any additional time beyond the regular working time L and within $L + \theta$ results in an overtime cost. Finally, let c_{wait}, c_{idle}, and c_{over} be the fixed non-negative unit customer waiting, team idling, and team overtime costs, respectively.

Decision Variables

The binary variable $x_{ij} \in \{0, 1\}$ is equal to 1 if and only if the arc $(i, j) \in A$ is traversed by a service team, otherwise it takes the value of zero. Notice that the fleet size can be derived using $x_{i,j}$ if we consider the total number of edges linking customers to the

depot is exactly twice the size of the fleet, and that $x_{i,i}$ due to the domain restriction is not allowed in the model. We use a continuous variable $0 \leq a_i \leq L$ for the team's arrival time at customer $i \in [\![1, n]\!]$, which is also the service starting time under the deterministic setting. Likewise, w_i and h_i are non-negative real-valued variables for the customer's waiting time, and the service team's idle time at customer $i \in [\![1, n]\!]$, respectively. In this model, we assume that a customer's service starting time is the same as their appointment time. Finally, g_i is also a real-valued variable measuring the overtime of a service team, registered at their arrival to the depot when returning from the last customer $i \in [\![1, n]\!]$ on their tour. A summary of the symbols used for the H-SARA-1 model can be found as part of the Table A.1.

Deterministic H-SARA-1 Formulation

$$\min \quad f_m \sum_{i \in [\![1,n]\!]} x_{i,n+1} + \lambda \sum_{(i,j) \in A} \tau_{i,j} x_{i,j} + c_{over} \sum_{i \in [\![1,n]\!]} g_i \tag{3.1a}$$

subject to

$$\sum_{i \in [\![1,n]\!]} x_{0,i} \leq m, \tag{3.1b}$$

$$\sum_{\substack{i \in [\![0,n]\!] \\ i \neq j}} x_{ij} = 1 \qquad \forall j \in [\![1, n]\!], \tag{3.1c}$$

$$\sum_{\substack{i \in [\![0,n]\!] \\ i \neq j}} x_{ij} = \sum_{\substack{i \in [\![1,n+1]\!] \\ i \neq j}} x_{j,i} \qquad \forall j \in [\![1, n]\!], \tag{3.1d}$$

$$\sum_{i \in [\![1,n]\!]} x_{0,i} = \sum_{i \in [\![1,n]\!]} x_{i,n+1}, \tag{3.1e}$$

$$a_i + s_i + \tau_{i,j} \leq a_j + M(1 - x_{i,j}) \qquad \forall (i,j) \in A, \tag{3.1f}$$

$$a_i + s_i + \tau_{i,j} \geq a_j - M(1 - x_{i,j}) \qquad \forall (i,j) \in A, \tag{3.1g}$$

$$a_i + s_i + \tau_{i,n+1} - L \leq g_i + \theta(1 - x_{i,n+1}) \qquad \forall i \in [\![1, n]\!], \tag{3.1h}$$

$$0 \leq g_i \leq \theta \qquad \forall i \in [\![1, n]\!], \tag{3.1i}$$

$$a_i \geq 0 \qquad \forall i \in [\![1, n]\!], \tag{3.1j}$$

$$x_{i,j} \in \{0, 1\} \qquad \forall (i,j) \in A. \tag{3.1k}$$

The objective function (3.1a) minimises the total traversal (time) costs, team hiring costs, and overtime costs. Note that in this context, no waiting or idle times occur, as the appointment times are exact. Constraint (3.1b) states that there are no more than m homogeneous service teams departing from the depot $\{0\}$. Constraints (3.1c) require that each customer must be visited once and only once by a service team. The flow conservation constraints (3.1d) require that a team travelling to any customer node must leave the node afterwards. This is complemented with (3.1e), which stresses that the number of teams leaving the depot must equal the number that returns. The functionality of constraints (3.1f) is two-fold. First they join constraints (3.1g) to link together the arrival time to the first customer, its service time and the traversal time to the next customer provided that the two customer visits are consecutive. Second it forbids the formation of subtours, which are cycles formed only by a group of customers without the depot. Constraints (3.1h) determine the consumed overtime amount when returning to the depot from the last customer. Constraints (3.1i) require that each

team's overworking time length cannot exceed an upper bound. Finally, (3.1j) and (3.1k) are the domain constraints for the corresponding variables.

Additionally, we can improve the lower bound imposing the simple constraint:

$$\sum_{i \in [\![1,n]\!]} x_{0,i} \geq 1, \tag{3.2}$$

which helps in the linear relaxation of the model by ensuring that at least one team will be needed to satisfy the service requirement. The following inequality can also be applied for subtour elimination, since it is more powerful to cut off fractional solutions during the branching process:

$$\sum_{(i,j) \in A} x_{i,j} \leq |S| - 1, \qquad \forall S \subset [\![i,n]\!], |S| \geq 2; \tag{3.3}$$

where S is any subset of customer set $[\![i,n]\!]$ that contains at least two nodes.

3.3.4 Stochastic MIP Model

In order to improve the model robustness, we introduce the *H-SARA-2* model with stochastic travel and service times. We derive a set of different scenarios, each with a different realisation of the travel and service durations. We decompose the problem into two interconnected stages based on scenario dependency, separating the integer arc traversal variables from the continuous time-related variables. Specifically, all decision variables relating to fleet size, customer-team allocation, team routing, and customer appointment time windows are scenario-independent and are kept in the first stage, whereas variables computing the actual arrival times as well as idling and waiting times are scenario-dependent are included in the second stage.

Unlike the deterministic formulation with complete information from the beginning, in the stochastic case, the decisions are made based on estimated travel and service durations from probabilistic scenarios before the exact values of the parameters are completely known. There are pros and cons for involving stochasticity: it provides an enriched closer-to-real perspective [78], but at the cost of being considerably more difficult to solve and evaluate than its deterministic counterpart.

Additional Parameters

In addition to the deterministic parameters introduced in Section 3.3.3, we have the following stochastic parameters. The set Ω denotes different scenarios ω, each associated with a different realisation of the travel and service times, and an occurring certain probability p_ω. We assume that Ω is a finite set and denote its cardinality by $|\Omega|$. Concerning the uncertainties, we assume that all cancellations are known by the time of planning, whereas the travel and service times remain uncertain, but with known probability distributions and the associated variables are assumed to be independent. We impose a traversal duration matrix $T = \tau_{ij}^{\omega}$ under scenario ω for any arc $(i,j) \in A$ based on the stochastic traversal variables, and same-wise impose a scenario-based service duration vector $S = s_i^{\omega}$ for customer i under scenario ω from the stochastic service variables.

Additional Decision Variables

Similar to the deterministic H-SARA-1 model, we continue using the binary traversal variable $x_{i,j} \in \{0,1\}$ that equals to 1 if and only if arc $(i,j) \in A$ is traversed by a service team. For the stochastic H-SARA-2 model, since the actual arrival time of a team under the stochastic setting could be different from the predicted one, we introduce a scenario-independent variable t_i that computes the expected arrival time at customer i. We then assume that the appointment time window is centred at t_i with a fixed width of $2W$, $W > 0$, i.e., $[t_i - W, t_i + W]$. The actual arrival of a service team before the scheduled appointment time window leads to team idling, whereas an arrival after the time window leads to the customer waiting. This is modelled in the second stage where we employ a set of continuous variables for each customer $i \in [\![1,n]\!]$ and each scenario $\omega \in \Omega$: $a_i^\omega \in [0,L]$ and $h_i^\omega \geq 0$ for the team's arrival and idling time at i, respectively; $w_i^\omega \geq 0$ for the customer's waiting time; and g_i^ω for the overtime of a service team registered at their arrival at the depot when returning from the last customer i in the tour.

Stochastic H-SARA-2 Formulation

The traversal variables $x_{i,j}$ and the appointment time variables t_i are our first-stage decisions. Fleet size, customer-team allocation, and team routing decisions can again be modelled through the single variable $x_{i,j}$. The team idling time h_i^ω, overtime g_i^ω, and customer waiting time w_i^ω are our second-stage decisions dependent on the different scenarios.

The first-stage formulation for the stochastic model is as follows:

$$(1^{\text{st}}\text{ Stage}) \qquad \min_{(x;t)} \quad f_m \sum_{i \in [\![1,n]\!]} x_{i,n+1} + \sum_{(i,j)\in A} d_{ij} x_{ij} + \mathbb{E}\left[Q(x,t,\omega)\right] \tag{3.4a}$$

subject to

$$\hat{s}_i + \hat{\tau}_{i,j} \leq L(1 - x_{i,j}) + t_j - t_i, \qquad \forall i,j \in [\![1,n]\!], i \neq j \tag{3.4b}$$

$$0 \leq t_i + \hat{s}_i + \hat{\tau}_{i,o} \leq L, \qquad \forall i \in [\![1,n]\!] \tag{3.4c}$$

(3.1b), (3.1c), (3.1d), (3.1e), (3.1k)

The objective function (3.4a) minimises the total team hiring cost, routing cost, and the expected second stage cost $\mathbb{E}\left[Q(x,t;\omega)\right] = \sum_{\omega\in\Omega} q^\omega \cdot Q(x,t;\omega)$ for idling, waiting, and overtime for any x and t satisfying the above equations and over all scenarios $\omega \in \Omega$ associated with probability q^ω. Constraints (3.4b) link the traversal variable $x_{i,j}$ to the expected customer arrival time t_i using the the expected travel time $\widehat{\tau}_{i,j}$ and the expected service time $\widehat{s}_i$. Constraints (3.4c) ensure that, on expectation, a service team can complete its tour within the regular working time L (we only allow overtime in the second stage). Additionally, (3.4b) are a variant of the Miller-Tucker-Zemlin (MTZ) subtour elimination constraints that rule out any cycles not containing the depot.

To strengthen the weak MTZ subtour elimination cut by (3.4b), we introduce the following *subtour elimination constraint*:

$$\sum_{(i,j)\in S'} x_{i,j} \leq |S'| - r(S'), \tag{3.5}$$

where S' refers to an infeasible inter-customer cycle not containing the depot, and $r(S')$ refers to the (minimum) fleet size needed to serve the customers in S'.

The second-stage formulation for the stochastic model is as follow:

(2^{nd} Stage)

$$Q(x,t,\omega) = \min_{w,h,g} \quad c_{wait}\sum_{i\in[\![1,n]\!]} w_i^\omega + c_{idle}\sum_{i\in[\![1,n]\!]} h_i^\omega + c_{over}\sum_{i\in[\![1,n]\!]} g_i^\omega \tag{3.6a}$$

subject to

$$a_i^\omega + h_i^\omega + s_i^\omega + \tau_{i,j}^\omega \leq a_j^\omega + M(1-x_{i,j}) \qquad \forall (i,j)\in A, \tag{3.6b}$$
$$a_i^\omega + h_i^\omega + s_i^\omega + \tau_{i,j}^\omega \geq a_j^\omega - M(1-x_{i,j}) \qquad \forall (i,j)\in A, \tag{3.6c}$$
$$a_i^\omega + s_i^\omega + \tau_{i,n+1}^\omega - L \leq g_i^\omega + \theta(1-x_{i,n+1}) \qquad \forall i\in[\![1,n]\!], \tag{3.6d}$$
$$h_i^\omega \geq (t_i - W) - a_i^\omega \qquad \forall i\in[\![1,n]\!], \tag{3.6e}$$
$$w_i^\omega \geq a_i^\omega - (t_i + W) \qquad \forall i\in[\![1,n]\!], \tag{3.6f}$$
$$g_i^\omega \leq \theta \qquad \forall i\in[\![1,n]\!], \tag{3.6g}$$
$$a_i^\omega, h_i^\omega, w_i^\omega, g_i^\omega \geq 0 \qquad \forall i\in[\![1,n]\!]. \tag{3.6h}$$
$$a_0^\omega = h_0^\omega = 0 \qquad \forall i\in[\![1,n]\!]. \tag{3.6i}$$

where the scenario-based objective function (3.6a) minimises the idling, waiting, and overtime costs under scenario ω. Constraints (3.6b) and (3.6c) link together the arrival time to the first customer, its service time, and the traversal time to the next customer, given that the two customer visits are consecutive. Hereby, M is a large number in these constraints, e.g., $L+\theta$ would be sufficient. Constraints (3.6d) determine the incurred overtime when returning to the depot from the last customer. Constraints (3.6e) and (3.6f) specify the idling and waiting times, respectively. Constraints (3.6g) restrict the extent of overtime permissible in the second stage due to uncertainties related to travel and service. Together with (3.6d) they make sure that, any second-stage tour must not exceed a total time length of $L+\theta$. Constraints (3.6h) provide domain restrictions for the relevant variables. Finally, an additional requirement of no arrival or idle times at the depot must be enforced by (3.6i).

3.4 CPLEX Exact Method

In this section, we propose to solve the H-SARA-1 and H-SARA-2 models via CPLEX built-in exact methods with acceleration approaches.

3.4.1 Bounding the number of service teams

The tightness of constraint (3.1b) is strongly dependent on the value of m, which is generally large given that the number of available service teams might be more than the number eventually hired. Therefore, we aim to provide a tighter restriction on m by establishing both upper and lower bounds for the service teams, such that the domain is effectively constrained. For notational simplicity, the hat decorator in the travel and service times involved in the following constraints are the expected values for each arc and node, namely $\hat{\tau}$ and $\hat{s}$. We can find an upper bound on the number of teams required to visit all customers by solving the following linear programme:

$$\min \ell_u \tag{3.7a}$$

subject to

$$\sum_{i\in[\![1,n]\!]} \widehat{s}_i + \frac{1}{2}\sum_{i\in V}\left(\max\{\widehat{\tau}_{i,j} : (i,j)\in A\} + \max\{\widehat{\tau}_{j,i} : (j,i)\in A\}\right) \leq \ell_u(L+\theta) \tag{3.7b}$$

$$1 \leq \ell_u \leq \widehat{m}, \qquad \text{and} \qquad \ell_u \in \mathbb{Z}; \tag{3.7c}$$

where ℓ_u is a decision variable representing the maximum number of required teams to satisfy, in a mean-worst-case scenario, all the transportation and service requirements. The travelling times to and from each node is considered the highest. Here, m is an upper limit on the number of available teams, which can be as large as the number of customers n, and an optimal solution of (3.7) determines a value over m. Observe that if we divide constraint (3.7b) by ℓ_u, the resulting expression distributes the routing task in two parts: There is a term averaging service time, and another term averaging the time required to travel between customers taking the most time-consuming paths. Notice that the optimal solution can be obtained using exhaustive enumeration in $\mathcal{O}(1)$ time.

Likewise, service times can be applied to provide a lower bound on the amount of time that all service teams spend on the road. Here, the travelling time to and from each node are assumed to the lowest. To do so, we define ℓ_l as the minimum number of teams required to distribute the aggregated service time and minimum transportation time. Thus, we need to solve the following nonlinear program:

$$\max \frac{1}{\ell_l} \tag{3.8a}$$

subject to

$$\sum_{i\in[\![1,n]\!]} \widehat{s}_i + \frac{1}{2}\sum_{i\in V}\left(\min\{\widehat{\tau}_{i,j} : (i,j)\in A\} + \min\{\widehat{\tau}_{j,i} : (j,i)\in A\}\right) \leq \ell_l(L+\theta) \tag{3.8b}$$

$$1 \leq \ell_l \leq \widehat{m}, \qquad \text{and} \qquad \ell_l \in \mathbb{Z}; \tag{3.8c}$$

Notice that if this problem is infeasible, then there are not enough teams to solve the problem with mean values for service and transportation times. As a result, we have an infeasibility certificate. Again, this problem can be solved in $\mathcal{O}(1)$ time.

3.4.2 Root Node Solution Method

After shrinking the fleet size domain, we apply the *root node* solution method to determine the best fleet size value $\hat{m}$ as the pre-set fleet size value before deriving the complete set of decisions.

The flow chart of this method is given in Figure 3.2, where it begins with limiting the fleet size options using the upper and lower bound given in the previous Section 3.4.1. After deriving the bounds, we enumerate through the list of possible fleet size options in $\{\ell_l, \ell_l+1, \ldots, \ell_u\}$ and for each fixed fleet size value, CPLEX solver is instructed to stop at the first integer solution found in the root node before the branching begins. After all the associated root nodes values are computed, we instruct CPLEX to identify the smallest root node value out of the list of integer solutions amongst all possible fleet sizes and return its associated fleet size $\hat{m}$, which is used to provide an initial solution for the warm-start. This way, we find a good starting point of the search with fleet size $\hat{m}$ for the specific travel and service times realisations.

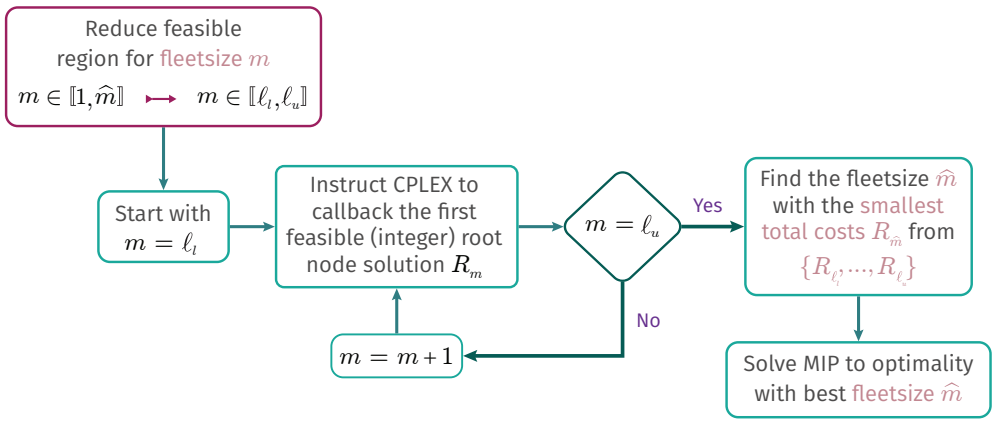


Figure 3.2: Visualisation for the root node solution method

3.4.3 Deterministic CPLEX Method with Acceleration Approaches

The deterministic model can be considered as a single-scenario stochastic model, where appointment time t_i is the same as the arrival time a_i with zero service team idling time $h_i = t_i - a_i = 0$ at customer $i \in [\![1, n]\!]$. Besides, the model has a pre-specified set of customer nodes, since we assume all cancelled customers are already removed.

We first attempted to solve the deterministic H-SARA-1 model to optimality. The model was inputted with a pre-specified number of customer nodes. The first deterministic model in Table 3.1 (first two rows) show the average CPU times and the average gap achieved using CPLEX 20.1.0 for 10 instances with a time limit of 1800 seconds. The gap indicates the solution's quality and is defined as the difference in percentage between the upper and the lower bounds. The upper bound describes the incumbent and the best-known feasible solution, whereas the lower bound describes the solution with model relaxation, which is typically better than the optimal. Nevertheless, while the

Table 3.1: Results for deterministic H-SARA-1 model using CPLEX

Type		**Number of customers** n				
		5	10	15	20	30
Deterministic	CPU-time	0.08	0.63	1.21	146.14	1800
	Gap	0%	0%	0%	0%	0.05%
Deterministic[1] (fixed m)	CPU-time	0.06	0.18	0.632	94.07	1568.16
	Gap	0%	0%	0%	0%	0%
Deterministic[2] (fixed m+ gap)	CPU-time	0.05	0.11	0.294	0.506	7.34
	Gap	0.8%	0.2%	0.6%	0.9%	1%

resolution of the smaller-scaled deterministic problem remains computationally manageable for small instances, the solver proves ineffective when confronted with large or even moderate-sized instances and fails to find feasible solutions within 30 minutes on average. To cope with the low computational efficiency, we have proposed two acceleration approaches to reduce the computing time for the deterministic H-SARA-1 model.

To start with, we apply the root node method introduced in Section 3.4.2 to address the trade-off between fixed hiring costs and travelling costs, opting for an "economical fleet size" [81]. Since service team hiring costs typically dominate the remaining costs, starting the search process with a near-optimal fleet size is crucial in discovering the

optimal solution or at least near-optimal solution candidates in reasonable computational time. To commence with the method, we fix the number of service teams as $\hat{m}$ in constraint (3.1b) and change its sense to strict equality so that the solver is no longer required to optimise the fleet size but rather treats it as an input parameter. To find this optimal number of service teams, we limit our fleet size options using the upper and lower bounds given in Section 3.4.1. By using the proposed root node method in Section 3.4.2, we observe a considerable improvement in computation speed without loss of solution quality, as shown in the third and fourth row of Table 3.1 under the Deterministic[1] category.

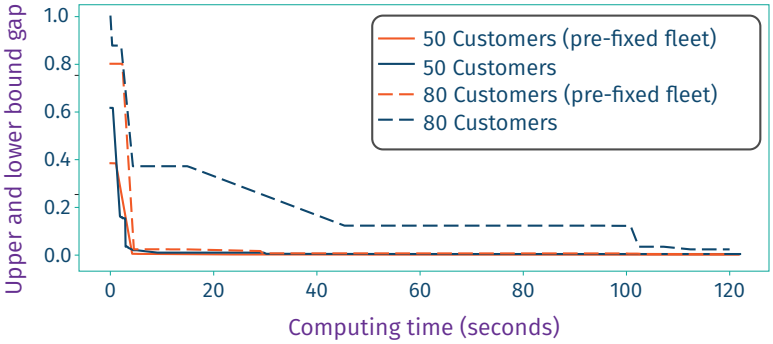


Figure 3.3: Deterministic model gap versus computing time

For the second acceleration approach, we observe from experimental testings that CPLEX's default heuristic solution method can reach an integer solution at the root node with reasonably good quality and within a concise computing time (less than 1 minute). Nevertheless, reaching a global optimum is difficult due to the time-consuming nature of the branch-and-bound process encoded in the solver. This trend is shown in Figure 3.3 ((marked by blue solid and dotted lines)) and can be observed visually during the solution process that the solver spends an extremely long time improving the visiting sequence of customers. Therefore, to further accelerate the speed of developing a near-optimal solution, we terminate the solving process if the MIP gap is less than 2%. The solution time and relative MIP gap reported by CPLEX for different customer sizes are presented in the last two rows of Table 3.1 under the Deterministic[2] category. Overall, the two acceleration approaches allow us to target a near-optimal solution within a considerably shorter computation time. The multi-scenario stochastic model is noticeably more challenging to tackle than its deterministic counterpart, which can be considered as a single-scenario stochastic model. Nonetheless, we observe that our stochastic H-SARA-2 model embodies a distinct structure that offers potential for the application of Benders framework that will be expanded in Section 3.5.

3.5 L-shaped Method

We realise the natural partitioning of our stochastic model, where the first stage is an integer MIP problem and the second-stage recourse model is linear. Furthermore, the second-stage problem is scenario-dependent, and therefore its structure suggests the application of Benders' decomposition [24], taking the first stage as the master problem that decides which set of paths to take, and treating each scenario inside the second-stage recourse model as a subproblem.

3.5.1 Benders' decomposition Algorithm

Benders' decomposition is a frequently-applied technique for solving large-scale MIP problems with a block-diagonal structure. It is called the L-shaped method when applied to two-stage problems. Benders' decomposition has a "row-generation" strategy that relaxes the original problem and iteratively generates cuts to strengthen the relaxed problem until convergence is achieved.

We introduce the Benders' algorithm below. A given stochastic programming model containing both global and local constraints can be represented as below:

$$z_{\text{MIP}} = \min\{cx + \sum_{\omega\in\Omega} dy^{\omega} : Ax + By^{\omega} \geq b, Dx \geq f, x \in \mathbb{Z}^n, y^{\omega} \in \mathbb{R}^m, \omega \in \Omega\} \quad (3.9)$$

Benders' decomposition separates the complex optimisation problem into the master problem (MP) and subproblem. It keeps the complicated linking variable x within the MP, which can be written as:

$$z_{\text{MP}} = \min\{cx + z_{\text{SP}}(x) : Dx \geq f, x \in \mathbb{Z}^n\} \quad (3.10)$$

and projects out the remaining variables in a composite format which yields the subproblem's objective value as the sum of different scenarios $z_{\text{SP}} = \sum_{\omega\in\Omega} z^{\omega}_{\text{SP}}$. Here, for a fixed first-stage decision vector $x = \overline{x}$, the remaining subproblem can be formulated as:

$$z^{\omega}_{\text{SP}}(\overline{x}) = \min\{dy^{\omega} : By^{\omega} \geq b - A\overline{x}, y^{\omega} \in \mathbb{R}^m\} \quad (3.11)$$

which can be solved independently for each scenario $\omega \in \Omega$. Since the second-stage variable y^{ω} is continuous for any scenario ω, each subproblem is a linear programming problem and therefore has a corresponding dual formulation as:

$$z^{\omega}_{\text{DP}}(\overline{x}) = \max\{u(b - A\overline{x}) : ub \leq d, u \geq 0\}, \quad (3.12)$$

where u is the vector of dual multipliers corresponding to the constraints $By^{\omega} \geq b - A\overline{x}$ from the primal problem. The polyhedron of (3.12) can be represented in terms of its extreme points and extreme rays. We begin by solving the master problem. For each iteration of the Benders algorithm, a vector $\overline{x}$ can be derived from the Reduced Master Problem (RMP). The subproblems are linked to the RMP through two types of additional cuts. In the first case, if the corresponding dual subproblem $z^{\omega}_{\text{DP}}(\overline{x})$ has a bounded and nonempty feasibility region, a Benders optimality cut

$$u(b - Ax) \leq \varphi \quad (3.13)$$

is generated using the extreme point u, which are the corner points of the feasible region that satisfy a particular subset of constraints. φ is a scalar value determined by the SP solution that represents the threshold for the violation of constraints at the extreme point that is allowed to satisfy optimality. The generated optimality cuts are added to the RMP for the consecutive iteration. Otherwise, if the subproblem dual is unbounded and determines that the first-stage realisation $\overline{x}$ is infeasible, a Benders feasibility cut

$$u(b - Ax) \leq 0 \quad (3.14)$$

is generated using the extreme ray u and added to the RMP. Compared to the single-cut Benders' decomposition that generates a single optimality cut integrating all the subproblems for each iteration, the multi-cut reformulation produces a cut from each

subproblem, which can lead to a more aggressive pruning of the solution space and therefore be potentially more efficient compared to the classical single-cut method. The RMP is iteratively updated and solved to generate new Benders cuts until the gap between the upper and lower bounds on the objective function value falls below the termination threshold.

3.5.2 Stochastic CPLEX Method

We first use CPLEX solver to solve the full H-SARA-2 model. The first and second row in Table 3.2 list the numerical results of solving the complete stochastic model as a cohesive whole, incorporating the fleet size pre-solving procedure described in the root node method (Section 3.4.2) and limiting the gap to 2%. The third and fourth row are with CPLEX built-in Benders' method. The empirical results show that CPLEX built-in Benders' decomposition may not be suitable for our models as it consumes much more computing time to provide worse results. Moreover, we notice that due

Table 3.2: Results for stochastic model with 10 scenarios

Type		**Number of customers** n				
		5	10	15	20	30
Stochastic	CPU-time	0.19	2.05	1421.18	25.38	183.31
	Gap	1.99%	1.97%	2.12%	1.99%	1.91%
Stochastic (benders)	CPU-time	0.52	11.28	1800*	1800*	1800*
	Gap	1.99%	2.00%	2.93%	2.12%	3.66%

to specific parameter scaling settings, we have the fleet size cost dominating the other costs. Specifically, the fleet size for 20 customers (2 vehicles) is always larger than that for the 10-customer set (1 vehicle), meaning an additional vehicle is required when the instance size reaches an intermediate number. The actual value for this threshold varies due to the inconsistency of the uncertainty realisation, which in our case results in the possibility of the 15 customer instances being served by either 1 or 2 service teams. This is the reason behind the long solution process before termination for the 15-customer instance, which incurs a significantly higher computational cost compared to the neighbouring instances, since hiring one vehicle may be just enough to satisfy all constraints or infeasible until all the routing decisions are explored before the more expensive fleet size decision is reconsidered.

In general, the CPLEX built-in Benders framework demonstrates suboptimal performance for small-scale instances with a limited number of scenarios. Nevertheless, the encouraging outcomes reported in the literature (e.g., see Hashemi Doulabi et al. [97]) inspire the exploration of a customised Benders approach that utilises problem-specific structures. Such customisation holds the promise of better aligning with the specific problem's characteristics and unlocking the full potential of the Benders' decomposition technique.

3.6 Customised L-shaped Method

3.6.1 Benders Framework Overview

We visualise our customised Benders' decomposition process in a flowchart depicted in Figure 3.4. The process commences with the upper centre box in red on the flowchart,

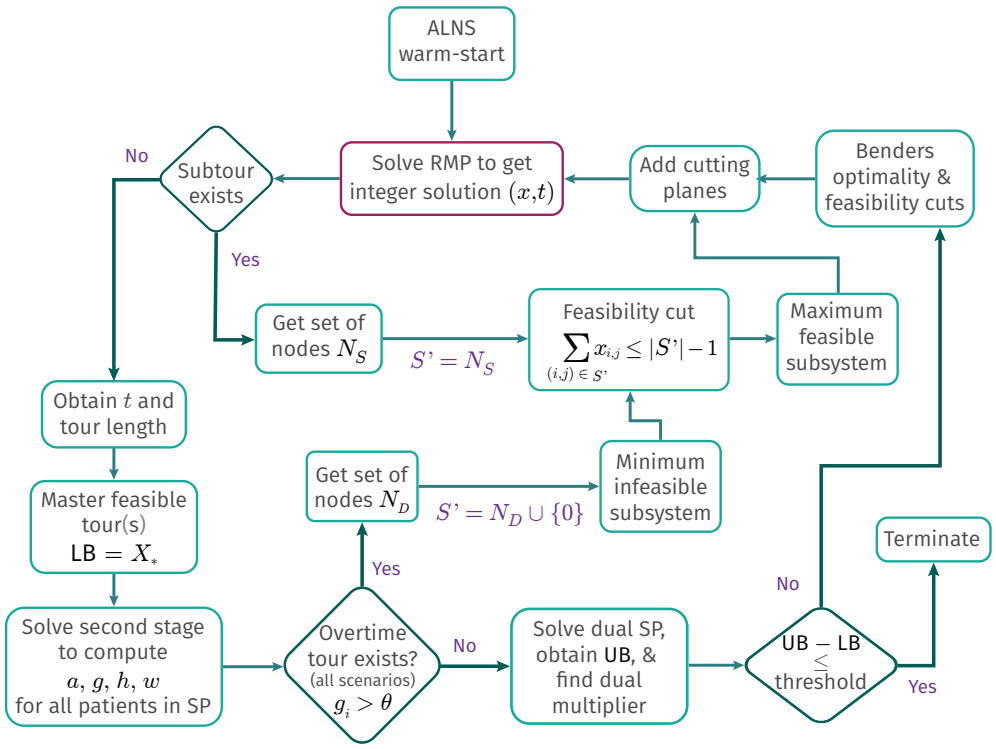


Figure 3.4: customised Benders' decomposition flowchart

where we first relax the MIP formulation for our stochastic programming model by separating the first-stage traversal decision from the time-related decisions that are scenario-dependent in the second stage. We form the initial RMP based on constraints (3.1b)–(3.1e), (3.1k) and (3.4b), solving the resulting MIP model yields a set of visiting sequences x and appointment times t for each customer. After validating the tours for feasibility and eliminating any illegal subtours that involve sequences without visiting the depot, we proceed to derive the master solution and record the lower bound. A preprocessing warm start can be included prior to solving the RMP and this is also explained as part of the master problem in Section 3.6.2. After validating the tours for feasibility and eliminating any illegal subtours that involve sequences without visiting the depot, we proceed to derive the master solution and record the lower bound. At the master problem level, any detected subtour formed by a group of customer nodes N_s without the inclusion of a depot is used to generated subtour elimination constraints and added to the RMP.

Afterwards, we determine the second-stage time-related variables a, g, h, w for the scenario-based subproblems (3.6a)–(3.6h), which are used to verify the feasibility of the master-level traversal decision and generate the corresponding Benders cuts. At this stage, we check for any group of nodes N_D causing overtime infeasibility using (3.4c), which are again used to generate constraints that are added to the RMP.

Moving on in the flowchart, to derive solutions for a large number of subproblems more efficiently, we develop a closed-form formulation explained in Section 3.6.3 for each primal subproblem with realisations of scenario-dependent travel and service time.

The generations of Benders optimality and feasibility cuts are explained in Section 3.6.4. Specifically, if the master-level traversal decision is feasible for a certain scenario, a Benders optimality cut is generated using the dual multipliers from the subproblem. For any infeasible scenario, we generate a feasibility cut to forbid the

formation of an overtime tour visiting an oversized group of customers. While the first-stage solution does not allow overtime, this is based on the expected travel and service times. Due to travel and service time uncertainty, for a specific scenario the actual working time of a team may not only lead to overtime, but even exceed $L + \theta$. To cope with the latter, we have introduced feasibility cuts in the form of a subtour elimination constraint.

Furthermore, we provide some acceleration techniques using the minimum infeasible subsystem (MIS) and maximum feasible subsystem (MFS) for generating optimality cuts based on infeasible scenarios and strengthening feasibility cuts will be explained in Section 3.6.5 and Section 3.6.6, respectively.

Finally, the iteration terminates when the master problem lower bound and subproblem upper bound satisfy a predetermined threshold. Regarding the overall Benders method, we utilise a single tree approach, where the first stage is solved optimally as an MIP model. The generated feasibility and optimality cuts from the second stage are incorporated using the callback function to reintroduce them to the first stage model.

3.6.2 Master Problem

When some of the projected variables are integers inside the Benders' framework, standard duality theory is not applicable for deriving classical Benders cuts. Instead, a different theoretical framework is required to effectively handle the integer variables, since the dual information cannot be used to generate Benders cuts in these cases.

The *integer L-shaped method* originally introduced by Laporte and Louveaux [136] is an exact algorithm based on the stochastic integer programming problem with complete recourse. Several related works address this challenge by focusing on integer master or subproblems [77, 106, 154, 204, 205]. Our Benders framework structure is particularly similar to the L-shaped method proposed in [136], with the distinction that our MP model only involves integer decision variables, while the subproblem consists of continuous variables.

To limit the domain of our first-stage decision variables x and t, we have added the following *valid inequalities* to the master problem to shrink the feasibility region:

$$t_i \geq \min_{\omega \in \Omega, j \in V} \{\tau^{\omega}_{0,j}\} \qquad \forall i \in [\![1, n]\!], \tag{3.15a}$$

$$|t_i - t_j| \geq \min\{\tau^{\omega}_{ij}, \tau^{\omega}_{ji}\} + \min\{s^{\omega}_i, s^{\omega}_j\} \qquad \forall (i, j) \in A. \tag{3.15b}$$

where constraint (3.15a) requires that the appointment time for each customer cannot precede the completion of the team's journey from the depot in any scenario. Notice that the triangular inequality does not hold due to the stochasticity in travel time. Constraint (3.15b) states that the intermediate appointment time for any pair of customer nodes must be equal to or greater than the combined duration of travel and service times needed for either direction of visit.

Lifted Formulation for Subproblem

We further consider a lifted formulation to strengthen the MP. We start with the following lifted valid inequality. For simplicity, we do not explicitly show the scenario index for all second-stage variables.

$$a_j \geq \sum_{i \in [\![1,n]\!]} (a_i + h_i + s_i + \tau_i) x_{ij} \tag{3.16}$$

The inequality above is derived after considering any two consecutive customers i and j in the subproblem solution. To see this, first observe that

$$a_j x_{ij} = (a_i + h_i + s_i + \tau_{ij})x_{ij} \tag{3.17}$$

holds for all arc (i, j). Indeed, if $x_{ij} = 0$, this is trivial, and thus let $x_{ij} = 1$. Combining the (3.6b) and (3.6c) gives a linear constraint with equality in the form $a_i+h_i+s_i+\tau_{ij} = a_j$, which is exactly (3.17). We then accumulate (3.17) over all start nodes $i \in [\![1, n]\!]$ with the same end node j. Since $\sum_{i\in[\![1,n]\!]} x_{ij} \leq 1$ holds for all nodes j, multiplying it by a_j to obtain $a_j \geq \sum_{i\in[\![1,n]\!]} x_{ij}a_j$. This allows us to replace the left hand side in the sum by a_j which results in (3.16). However, equation (3.16) exhibits a quadratic form as a result of the product involving the variables $a_i x_{ij}$ and $h_i x_{ij}$. To convert this equation into its linear equivalent, we use McCormick envelopes [162] through the introduction of auxiliary variables ς and ϱ. These auxiliary variables are characterised by the two sets of inequalities below:

$$\varsigma_{ij} \leq x_{ij}, \quad \varsigma_{ij} \leq a_j + (L + \theta)(1 - x_{ij}), \quad \varsigma_{ij} \geq 0, \quad \varsigma_{ij} \geq a_j - (L + \theta)(1 - x_{ij}) \tag{3.18a}$$

$$\varrho ij \leq x_{ij}, \quad \varrho ij \leq h_j + (L + \theta)(1 - x_{ij}), \quad \varrho ij \geq 0, \quad \varrho ij \geq h_j - (L + \theta)(1 - x_{ij}). \tag{3.18b}$$

It is worth noting that $L + \theta$ represents the maximum workload allowed, which also serves as an upper bound for a_j and h_j. The set of inequalities in (3.18) ensures that $\varsigma_{ij} = a_j x_{ij}$ and $\varrho_{ij} = h_j x_{ij}$ when $x_{ij} \in \{0, 1\}$. Throughout the paper, we will employ the lifted formulation by incorporating (3.18) and substituting $a_j x_{ij}$ and $h_j x_{ij}$ with ς_{ij} and ϱ_{ij}, respectively.

Warm-Start

To warm-start the MIP solver, we implement an ALNS-based improvement heuristic (see Section 3.7.3) to install a feasible initial solution of reasonably good quality for the master problem. To commence with the warm start algorithm, we once again apply the root node method described in Section 3.4.2 to select a feasible fleet size while addressing a trade-off between fixed vehicle costs and routing costs. This way, we find the fleet size $\hat{m}$ as an initial solution for the RMP that is optimal for the specific travel and service times realisations. As a result, the warm-start heuristic produces an initial solution with $\hat{m}$ service tours for the RMP. We use the expected travel and service times from all scenarios during the warm-start to generate such tours.

3.6.3 Closed-form Primal Solution for Subproblem

Once the first-stage routing plan is established, the second-stage becomes a linear programming problem, for each scenario $\omega \in \Omega$, resembling a network flow problem. In fact, as the routing variables x are fixed for this stage, constraints (3.6b) and (3.6c) can be simplified, significantly reducing the size of the model. More explicitly, and dropping the scenario index, the following reduced second stage model is obtained:

(Reduced 2nd Stage)

$$Q_{\mathsf{R}}(x, t, \omega) = \min_{w,h,g} \quad c_{\text{wait}} \sum_{i\in[\![1,n]\!]} w_i + c_{\text{idle}} \sum_{i\in[\![1,n]\!]} h_i + c_{\text{over}} \sum_{i\in[\![1,n]\!]} g_i \tag{3.19a}$$

subject to

$$a_i + h_i + s_i + \tau_{ij} = a_j \qquad \forall (i,j) \in A : x_{ij} = 1, i, j \neq n+1, \tag{3.19b}$$
$$a_i + h_i + s_i + \tau_{i,n+1} \leq g_i + L \qquad \forall i \in [\![1,n]\!] : \; x_{i,n+1} = 1, \tag{3.19c}$$
$$h_i \geq (t_i - W) - a_i \qquad \forall i \in [\![1,n]\!], \tag{3.19d}$$
$$w_i \geq a_i - (t_i + W) \qquad \forall i \in [\![1,n]\!], \tag{3.19e}$$
$$0 \leq g_i \leq \theta \qquad \forall i \in [\![1,n]\!] : \; x_{i,n+1} = 1, \tag{3.19f}$$
$$a_i, h_i, w_i \geq 0 \qquad \forall i \in [\![1,n]\!], \tag{3.19g}$$
$$a_0 = h_0 = 0. \tag{3.19h}$$

Instead of feeding the above formulation to a solver, we can easily determine its optimal solution by inspection, if it exists. Algorithm 3 details the necessary steps and the following result proves its correctness. If, however, the subproblem is infeasible, then we need to cut off the master solution.

Proposition 1. *Algorithm 3 constructs an optimal solution for the reduced second-stage problem if it exists and otherwise returns a (valid) feasibility cut.*

Proof. The general approach works as follows: First, let K be the number of active teams (tours) determined by the traversal variables; i.e., $K \coloneqq \sum_{i \in [\![1,n]\!]} x_{0,i}$. Second, the set of first nodes visited in all tours after leaving the depot are collected in the set $I \coloneqq \{i : x_{0,i} = 1\}$. Similarly, the set of last visited nodes before going back to the depot are collected in $J \coloneqq \{j : x_{j,n+1} = 1\}$. Observe that there is a unique path constructed by the first-stage solution from only one node in I to a corresponding node in J, given by the k-th route. The sequence of arcs form the k-th route, a complete tour starting and ending at the depot, is denoted by $X[k]$; namely

$$X[k] \coloneqq \big((i,j) : \text{is part of the } k\text{-th route from } I \text{ to } J\big), \tag{3.20}$$

for all $k \in [\![1, K]\!]$. Now, a simple propagation of initial values of arrival time variables a in (3.19b) allows us to build a solution. For instance, consider a fixed route $k \in [\![1, K]\!]$ given by the path $X[k] = \big((0,i),(i,j),\ldots,(\ell,n+1)\big)$. As $a_0 = h_0 = s_0 = 0$, we have $a_i = \tau_{0,i}$ for the first visited node i in path $X[k]$. This initial value can be then replaced in the inequalities for the idling and waiting times at the node, such that these are satisfied; i.e., $h_i = \max\{0, (t_i - W) - a_i\}$ and $w_i = \max\{0, a_i - (t_i + W)\}$. Once this is done, the next consecutive node j in same the path is considered. In this case, the value of a_j is again set such that (3.19b) is satisfied. This can be done recursively for all the remaining nodes in the path and for all paths until the terminal node is reached, at which point overtime is computed using the maximum value of the arrival times at the duplicate depot $n+1$. By construction, the solution attained is not only feasible but optimal, as any increment in a_i to reduce h_i or w_i in the initial nodes $i \in I$ propagates into higher objective costs and any decrement in any variable results in an infeasible solution. □

Observe that the reduced second-stage model can have multiple solutions as there is no effect of splitting waiting and idle times in a sequence of visits.

3.6.4 Benders Cuts

In what follows, we present two specialised Benders cuts for determining optimality and feasibility certificates provided by the subproblems.

Algorithm 3 Closed-form solution for reduced 2nd stage variables a, g, h, w for given x and t

Let $I = \{i : x_{0,i} = 1\}$, $J = \{j : x_{j,n+1} = 1\}$, and $K = |I|$.
Retrieve $X[k]$ as in (3.20) for all $k \in [\![1, K]\!]$.
for $i \in I$ **do**
    $a_i = \tau_{0,i}$
    $h_i = \max\{0, t_i - W - a_i\}$
    $w_i = \max\{0, a_i - t_i + W\}$
for $k \in [\![1, K]\!]$ **do**
    for $(i, j) \in X[k]$ with $i \neq 0$ **do**
        if $j = n + 1$ **then**
            $g_i = \max\{0,\ a_i + s_i + \tau_{i,n+1} - L\}$
            if $g_i > \theta$ **then**
                Return feasibility cut: $\sum_{(s,r)\in X[k]} x_{sr} \leq \big|X[k]\big| - 1$
        else
            $a_j = a_i + h_i + s_i + \tau_{ij}$
            $h_j = \max\{0, t_j - W - a_j\}$
            $w_j = \max\{0, a_j - t_j + W\}$

If any subproblem under a specific scenario is feasible, a Benders optimality cut $u(b - Ax) \leq \varphi$ is generated using the dual multipliers from the subproblem; i.e., the vector u. For each subproblem under a particular scenario, infeasibility only occurs when the duration of any generated tour incurs an overtime. That is, the total travel, service, and idling times for a service team visiting a sequence of customers exceeds the maximum time given, plus the penalised overtime allowance. Therefore, any feasibility cut is equivalent to a subtour elimination cut with respect to the overloaded customer visiting sequence $B : 0 \to i_1 \to \ldots \to i_\ell \to n + 1$ for $\{0, i_1, \ldots, i_\ell, n + 1\} \in S'$, $\ell \in \mathbb{N}_{\geq 1}$. The following *subtour elimination cut* is generated to forbid the formation of such an overtime tour:

$$\sum_{i,j\in S':(i,j)\in B} x_{ij} \leq |S'| - 1, \tag{3.21}$$

which can be further strengthened into the following feasibility cut, removing the two traversal arcs from depot to the first customer and from the last customer back to the depot. In this way, we are searching for an overloaded chain here:

$$\sum_{i,j\in\overline{S}} x_{i,j} \leq |\overline{S}| - 1, \qquad \overline{S} \subset S', |\overline{S}| \geq 2; \tag{3.22}$$

where the left-hand side of the constraint showcases an overloaded customer chain $c_i \to \ldots \to c_j$ that can occur in multiple additional extended tours. Once we identify $\overline{S} = \{c_i, \ldots, c_j\}$ as an infeasible customer set that creates an overtime tour, any longer tours containing this exact chain must also yield infeasibility. Therefore, for any overtime customer chain, constraint (3.22) has a broader applicability than (3.21) in terms of forbidding time-infeasible tours, for the latter can only forbid a single overtime tour formed by precisely one set of customers including the depot. Here, $|\overline{S}| \geq 2$ is because the value settings ensure that a tour must be capable of serving at least one customer. The following result proves that the strengthened version (3.22) is stronger and can

dominate (3.21).

Proposition 2. *Given a customer chain $i_1 \to \ldots \to i_\ell$ that exceeds the overtime limit, any extended tour formed by inserting additional customers before and/or after the chain cannot restore the tour to feasibility.*

Proof. For the strengthened constraint (3.22) to hold, any longer tours involving additional customers before or after the customer chain cannot result in earlier arrival times as the tour gets back to the depot. Without loss of generality, we assume any customer j_* appears before the customer chain and k_* appears afterwards. We can then formulate the extended tour as $j_1 \to \ldots \to j_a \to (i_1 \to \ldots \to i_\ell) \to k_1 \to \ldots \to k_b$ containing the given overtime customer chain. Since all customer service times are strictly positive, any two chronologically and consecutively visited customers i and j should satisfy the appointment time inequality $t_i \leq t_j$.

We start by including a single customer on the extended tour. When $a = 1$ and $b = n + 1$, one additional customer is visited after the depot and ahead of the first customer in the chain. Depending on the values of the team idling time h_{i_1} and customer waiting time w_{i_1} at the chain's first stop i_1, we have the following three cases: (a) When $h_{i_1} = w_{i_1} = 0$, the service at i_1 begins immediately after team arrival. The total time required to include j_1 can be written as $\tau_{d,j_1} + h_{j_1} + s_{j_1} + \tau_{j_1,i_1} > \tau_{d,j_1} + \tau_{j_1,i_1} \geq \tau_{d,i_1}$ because of $h_{j_1} \geq 0$, $s_{j_1} > 0$ and the triangular inequality. Hence, the extended tour with inserted customer j is also overtime. (b) When $h_{i_1} > 0$ and $w_{i_1} = 0$, the team arrives prior to i_1's earliest availability, causing the team to wait. In this case, it is possible for the respective team to serve at least one more customer beforehand and still encounters idling time at i_1. This can be written as $\tau_{d,j_1} + h_{j_1} + s_{j_1} + \tau_{j_1,i_1} \leq t_{i_1} - h_{i_1}$. The service starting time however will not be any earlier than what was initially scheduled for the chain. Hence, the duration of the extended tour must be as long as the original overtime chain tour. (c) When $h_{i_1} = 0$ and $w_{i_1} > 0$, the team arrives late, thus causing the customer to wait until the start of service. The extended tour is strictly longer than the overtime chain tour by reusing the inequality under case (a). When $a = 0, b = 1$, one additional visit occurs after the chain of customers is served. Since $s_{k_1} > 0$, the extended tour must be longer than the chain tour.

Any tour with $a \geq 1$ or $b \geq 1$ can be treated as a further extension by considering the additional customers one by one using the above cases. This means that the extended tour's completion time cannot be any earlier than the original customer chain tour. In conclusion, any insertion of additional customers before or after an overtime chain tour cannot create an earlier finishing time than before and so must be overtime. □

Notice that our Benders model does not have complete recourse, for the subproblems in our case are not always feasible. To handle this, feasibility cuts are generated to tackle any infeasible subproblem. Furthermore, the master level subtour elimination constraints in (3.5) carry a similar format. During the computational experimentation, it was observed that a large proportion of RMP solutions contained both feasible and infeasible tours. To enhance the master problem algorithm's efficiency, we replace (3.5) with a tighter version as in (3.22). The focus of this substitution is to prevent a group of customers that previously has formed illegal customer cycles to be allocated to the same tour at the master level, without entailing considerations regarding overtime from as in the subproblem level.

3.6.5 Maximum Feasible Subsystem

The Benders feasibility cuts are generally considered unpreferable, for they bring no improvement to the lower bound in the RMP. Slow convergence of the Benders' lower bound is most often due to the creation of numerous infeasible solutions that yield many feasibility cuts before a feasible solution occurs and produces an optimality cut. In order to generate optimality cuts earlier on to accelerate the convergence of the Benders' lower bound, we adapt the idea of a Maximum Feasible Subsystem (MFS) from the work of Saharidis and Ierapetritou [203] so that each time a feasibility cut is produced, we generate alongside an additional MFS cut to return to the current RMP.

The MFS cut is formed by relaxing a minimum number of SP constraints to restore the feasibility of the subproblem. Since any infeasible subproblem in our case can only be caused by overtime, we identify the only violated constraint in the subproblem as (3.6g). By relaxing it, we can derive an MFS containing (3.6b)–(3.6f), (3.6h), and the following constraint to restore that particular scenario to feasibility:

$$g_i \leq \theta + M', \qquad \forall i \in [\![1, n]\!]; \tag{3.23}$$

where M' is a sufficiently large number to make sure that any overtime tour can now fit into the service team time capacity. We can then generate an MFS cut in the same way as with an optimality cut via computing the dual multipliers of the relaxed primal subproblem. The dual multipliers associated with the relaxed constraint (3.23) can be fixed to zero due to complementary slackness. As a result, an infeasible scenario can generate one feasibility cut and one MFS cut, which shares similar structure with an optimality cut.

Even though Rahmaniani et al. [194] suggests the strategy can be non-competitive due to the additional computational cost for finding the MFS compensating the gain in fewer cut iterations. This weakness however does not apply to our implementation because we know precisely which constraint will be infeasible. Therefore, there is no need to iterate through all the subsystems to identify the maximal one. Besides, our closed-form subproblem solution ensures that our computational time remains linear with respect to the node size.

3.6.6 Minimum Infeasible Subsystem

Another issue associated with slow convergence of the Benders bound is the poor quality of cuts generated in each iteration, especially at the beginning of the Benders algorithm. For the combinatorial Benders feasibility cut (3.22), the effect can be relatively weak, especially when the size of the set S' is large with a large excess in overtime. Therefore, we strengthen the feasibility cuts based on the Minimum Infeasible Subsystem (MIS) from the work of Codato and Fischetti [40], which is a subset of customers that cannot be served together by a single service team without overtime but the removal of any node results in a feasible single tour.

Enumerating through every existing MIS for a particular customer set would be highly computationally inefficient. To derive stronger MIS feasibility cuts, we start with the tours formulated in the master problem. We use a local-search-based heuristic for each tour within the solution to find the set of minimum infeasible subsets that are slightly overtime, yielding stronger feasibility cuts. We adapt a local search procedure based on the current status of the tour, to iteratively remove or insert customer nodes to create a MIS and return relevant cuts.

Algorithm 4 Finding MIS

```
Let tour be infeasible
while tour is infeasible do
    MIS_tour ← tour
    candidate ← ∅
    for node ∈ tour do
        aux_tour ← tour.Remove(node)
        if aux_tour is infeasible then
            gain ← cost(tour) − cost(aux_tour)
            candidate ← candidate.Append(node, gain)
    node ← node with maximum gain from candidate
    tour ← tour.Remove(node)
Return MIS_tour
```

3.7 Heuristic Algorithm

We have observed from Table 3.1 and Table 3.2 that even with efficient acceleration approaches, solving a large-scale H-SARA problem jointly to optimality is still not practical due to the time-consuming nature of exact methods. On top of that, the problem involves a range of uncertainties in real-life traversal times, service duration, and customer presence rates, all of which require a flexible solution method that focuses more on adapting to fast-changing information and a large number of scenarios, meanwhile achieving in-time solutions with good quality.

Realising the difficulty of tackling the problem as a whole, we aim to develop a simple and flexible heuristic as an alternative to explore the solution space and find a reasonably good solution for larger instances within a manageable timescale. Specifically, we decompose the problem into its fleet-sizing, districting, routing, and scheduling components and developed a problem-tailored two-stage heuristic, which dynamically tackles one or multiple decisions without leaving the others out of sight. For our tailored two-stage heuristic, we have first decomposed the problem into different stages with an embedded chronological structure, allowing us to make dynamical decisions at each stage with an increased level of information.

3.7.1 Description of the Heuristic Approach

Our two-stage heuristic resembles a typical home service rundown: previous-day initial plannings (in Section 3.7.3), service day tour refinements (in Section 3.7.4), and post-service performance evaluation (in Section 3.7.5). The computational efficiency positions our heuristic as an effective alternative for mitigating the impact brought by customer cancellation (Section 3.7.2), leveraging available information to generate high-quality decisions during the decision-making process. The heuristic showcases an "inter-feedback process" that the previously-made decisions can be re-optimised and updated at a later stage with an increased level of information. Figure 3.5 displays an example for the two-stage heuristic outputs. Specific descriptions of each stage are given in the remaining part of this section.

Initial Planning Stage

This stage represents the real-time circumstances when the decision-makers make pre-arrangements to work diligently to ensure a smooth rundown on the actual service day with incomplete information. The available information at this stage includes the customers' geographical locations, the expected probability of cancellations, and estimated travel and service times (from known distributions or historical data). Before the heuristic begins, we aim to determine a fleet of available teams m on a strategic level based on methods such as the root node approach or other estimations. At the initial stage of the heuristic, we aim to determine a workable fleet size $\hat{m}$ to use and construct a set of *a priori* routes for the service teams. We also aim to notify the customers of an initial appointment slot for the service. The decision-makers are free to select a specific fleet size within the feasibility interval according to their preference or risk-averseness, and the fleet size will then be optimised by the heuristic (see Figure 3.5 for an example).

Tour Refinement Stage

This stage resembles the actual service day. With all customer cancellations obtained before a given cut-off time, the second stage provides a refinement of the *a priori* solution from the first stage. After cancelled customer nodes are removed from the tours, the cluster and the visiting sequences are re-optimised for more balanced workload among the service teams. All non-cancelled customers will be notified of an updated and narrower appointment time window at the beginning of the service day. Specifically, we aim to design the routes such that all the second-stage updated appointment time windows lie within their associated initial time windows derived in the previous stage (see Figure 3.5 for an example). Additional scenarios are generated to evaluate the robustness of the second-stage solutions.

Post-service Evaluation Stage

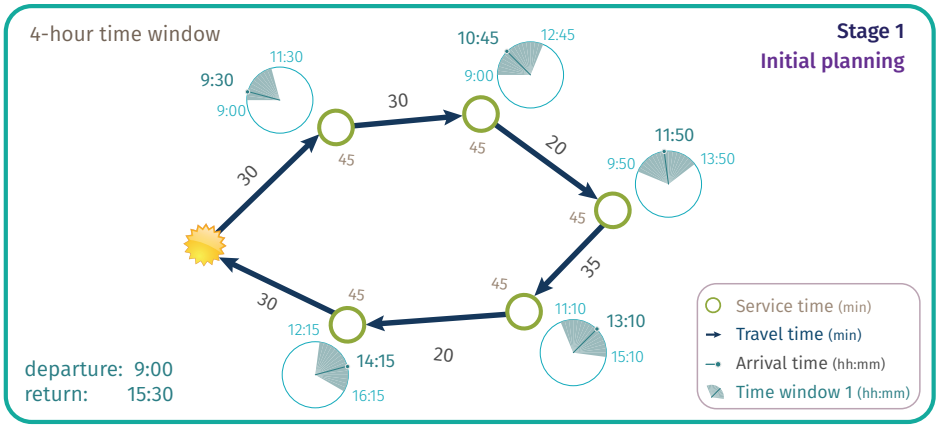
This stage possesses complete information with specific travel and service durations revealed after the actual service. This is a post-service reflection period for the service providers to evaluate the service teams' performance.

3.7.2 Uncertainties Inside the Model

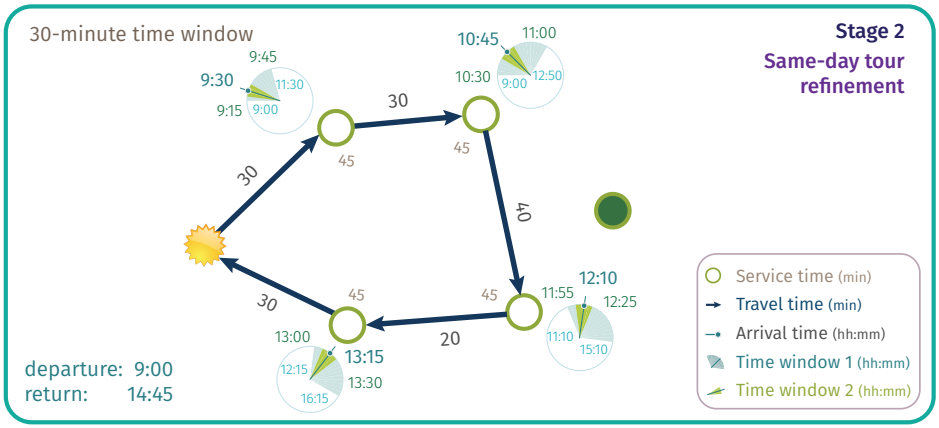
For the heuristic, while the in-time customer cancellations can be directly addressed during the tour refinement stage, the heuristic aims to deliver sound and up-to-date decisions to ultimately mitigate the impact of any other type of customer cancellations occurring after the cut-off time, as well as the uncertainties related to travel and service times. We come up with three customer cancellation scenarios on the service day and their associated policies as below:

1. *In-Time Cancellation*: Any cancellation that occurs before a fixed cut-off time or the departure of the assigned team from the depot, whichever is earlier. Any such customer is skipped by the team, which will travel from its previous stop directly to the next scheduled customer, whose appointment time is adjusted;

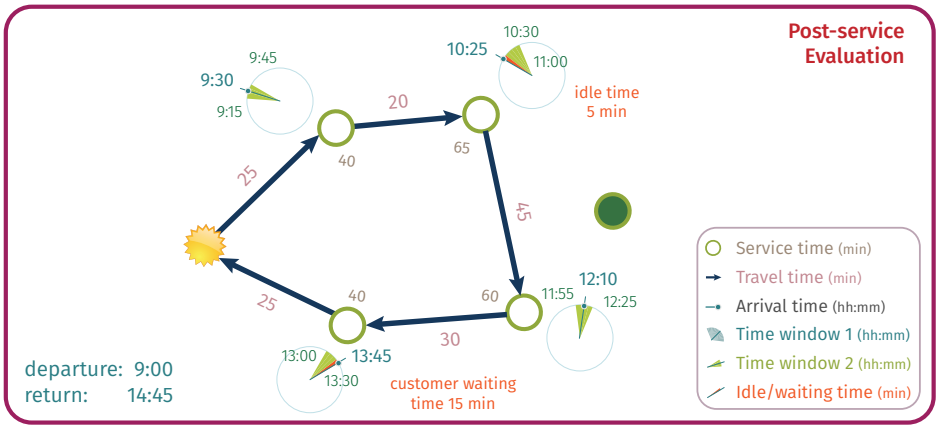
2. *Last-Minute Cancellation*: Any cancellation occurs when the team is at any service stage but before its departure for the cancelled customer from the previous node. The additional travel to the node cannot be avoided due to short notice, but the team spends zero time (service duration = 0) at the node;



- For a specific team, plan initial visiting sequence and arrival times for all assigned customers based on estimated travel and service times
- Compute initial appointment time windows using arrival times



Remove cancelled customers and update time windows (green) based on estimated travel and service times



Receive travel and service times realisations, compute idle/wait/overtime penalties, and evaluate service team's perfomance and solution robustness

Figure 3.5: Heuristic framework: initial planning, tour refinement, and post-service performance evaluation stages

3. *No-Presence*: Considered as a "no-show" or "irresponsible customer" who does not give any notifications for the cancellation. The team arrives at the customer assuming regular service and waits for a short period (less than service time) before realising the situation and thus chooses to leave for the next stop. Additional idling time occurs.

A cancellation policy analysis for the above three types is given in Section 3.8.5.

3.7.3 Initial Planning Stage

The initial planning stage consists of four consecutive steps: customer districts construction, district-based routes construction, workload balance improvement, and initial appointment scheduling. The graphs in the left-hand-side column of Figure 3.6 demonstrates the first-stage decision planning process. For the decision-makers, the initial solution provides the number of service teams to use, the sequence of customers for each service team, and an initial 2-hour appointment slot assigned to all existing customers. The solutions are generated based on estimated traversal and service times (see the next section for details) and only considers in-time cancellations.

Estimate service and travel times with probabilistic customer presence

We first provide an estimation on the activity measure, which is the expected amount of time required to include a specific customer in a tour. This helps us to determine the size of a tour servable by an individual team. In our application, the customer cancellation rate is known probabilistically, which means that the actual sequencing of customers or the computation of route lengths can be inaccurate without knowing the actual cancellation list. Notwithstanding, we can estimate the travel and service times for probabilistic customers without explicit routing as proposed by Bard and Jarrah [18]. According to their work, the estimated total time required for a group of customers can be divided into (i) *stem time*: estimated travel time from the depot to the nearest customer inside the group; (ii) *intermediate transit*: estimated travel time between customers of the same group; and (iii) *service time*: estimated stopping time at each customer. Parts (i) and (iii) are self-explanatory and can be estimated by the distributions associated to travel and service times. For (ii), we can estimate e_i, which is the expected travel time from customer i to any same-group customer j with probabilistic customer presence rate, using the following formula given in [18]:

$$e_i = \sum_{j=1}^{b_i} p_{ij}^{(*)} \cdot \frac{d_{ij}}{v_{ij}} \quad = \sum_{j=1}^{b_i} \frac{(1-p_j)(b_i - R_{ij} + 1)}{\sum_{\ell=1}^{b_i}(1-p_\ell)(b_i - R_{i,\ell} + 1)} \cdot \frac{d_{ij}}{v_{ij}} \tag{3.24}$$

where p_j is customer j's probabilistic cancellation rate, b_i is the fixed number of closest customers to customer i, R_{ij} is the rank of the j-th closest customer with respect to i, with $j \in [\![1, b_i]\!]$. $p_{ij}^{(*)}$ can be interpreted as the likelihood of customer j following i on a route. d_{ij} is the Euclidean metric and v_{ij} is the travel velocity from node i to j. As a result, the activity measure, AM_i, for customer i can be estimated by the expected service time $\hat{s}_i$ plus the estimated travel time from i to the district centre j using (3.24). Here we use the expected travel velocity $\hat{v}$. We estimate the number of nearest customers to be the average number of customers inside a district $b_i = \lceil \frac{n}{m} \rceil$.

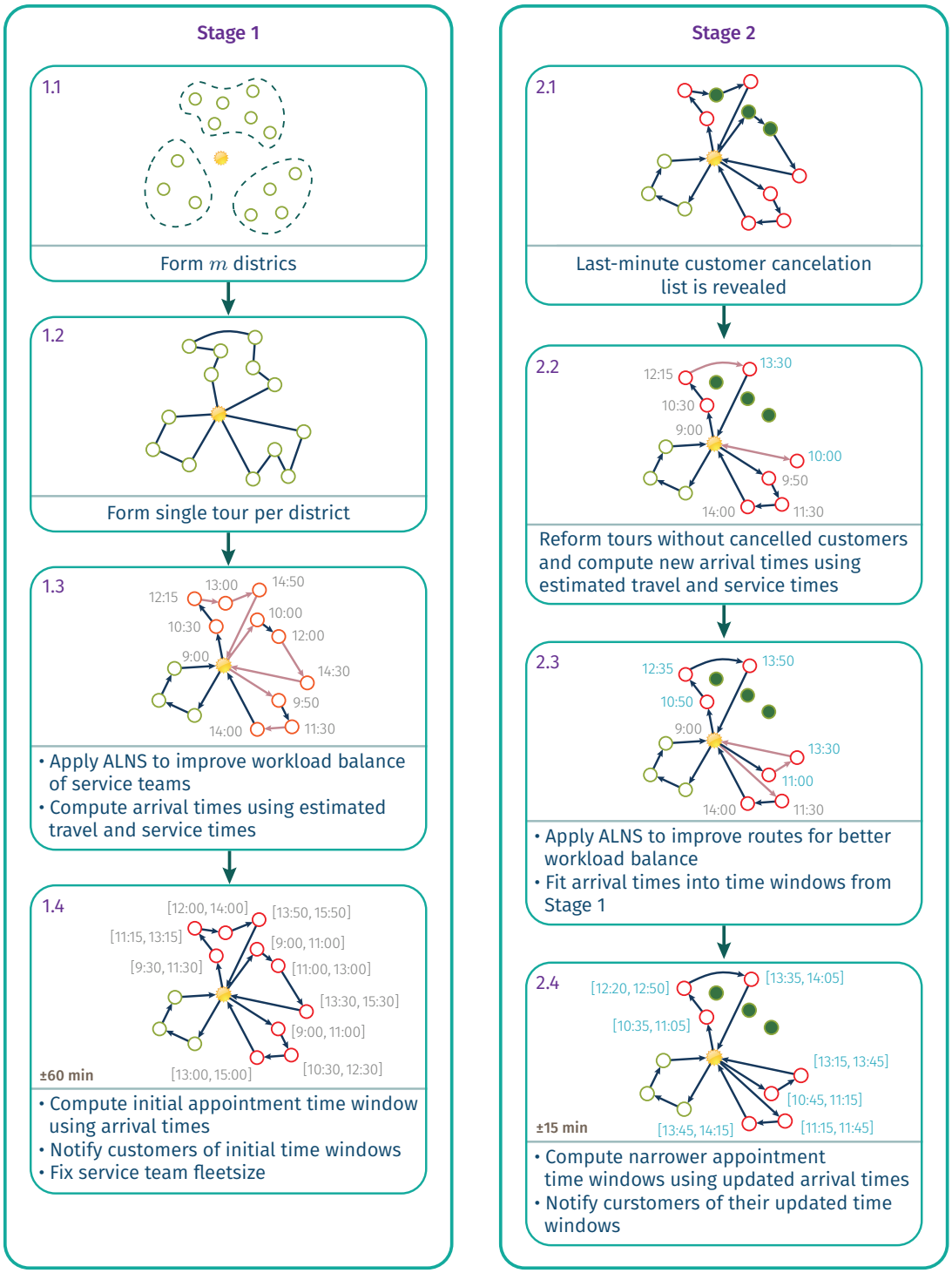


Figure 3.6: Example of the Initial Planning Stage and Tour Refinement Stage

This way we have that for a specific customer i:

$$\mathsf{AM}_i = \hat{s}_i + e_i = \hat{s}_i + \sum_{i=1}^{b_i} p_{ij}^{(*)} \cdot \frac{d_{i,j}}{\hat{v}_{ij}} \tag{3.25}$$

Here, we use the average number of nodes within a district. However, we observe that b_i is sensitive to the cancellation rate, for a high cancellation indicates that the selection of the next customer can exhibit a significantly random pattern. In contrast, a low cancellation makes the next node to be visited almost certain and establishes an inverse relationship with the distance. In this way, we are able to provide a rough estimate of the time required for serving all customers allocated to each service team.

Stage 1.1. Customer-Service Team Assignment

At the beginning of the initial planning stage, we apply a cluster-first-route-second construction heuristic to come up with an initial set of routes. We adapt the districting formulation proposed by Hess et al. [103] and solve the MIP model to optimality to derive the initial customer-driver assignments. Let $[\![1, n]\!]$, be the set of customers and $\{0\}$ be the depot as before. Let $\mathsf{AM}_i \in \mathbb{R}^+$ be the activity measure associated with customer i. The number of districts to be formed is the same as the pre-defined number of vehicles m. The average activity measure per district is defined then as $\mu \coloneqq \frac{1}{m}\sum_{i\in[\![1,n]\!]} \mathsf{AM}_i$. We denote $\mathsf{AM}_{\min} \leq 100$ and $\mathsf{AM}_{\max} \geq 100$ as the minimum and maximum percentage of activity measures in a district, respectively. Finally, the decision variable y_{ij} is equal to one if customer i is assigned to the district centred at customer j, and it is zero otherwise. Here y_{jj} takes the value of one if customer j is selected to be the district centre. The customer districting model is defined as below:

$$\min \sum_{j\in[\![1,n]\!]} \sum_{i\in[\![1,n]\!]} \mathsf{AM}_i \, d_{ij}^2 y_{ij} \tag{3.26a}$$

subject to

$$\sum_{j\in[\![1,n]\!]} y_{ij} = 1 \qquad \forall i \in [\![1, n]\!], \tag{3.26b}$$

$$\sum_{j\in[\![1,n]\!]} y_{j,j} = m \tag{3.26c}$$

$$y_{ij} \leq y_{jj} \qquad \forall j \in [\![1, n]\!], \tag{3.26d}$$

$$\sum_{i\in[\![1,n]\!]} \mathsf{AM}_i \, y_{ij} \geq \frac{\mathsf{AM}_{\min}}{100} \mu \cdot y_{j,j} \qquad \forall j \in [\![1, n]\!], \tag{3.26e}$$

$$\sum_{i\in[\![1,n]\!]} \mathsf{AM}_i \, y_{ij} + 2d_{0j} \leq L \qquad \forall j \in [\![1, n]\!], \tag{3.26f}$$

$$y_{ij} \in \{0, 1\} \qquad \forall i, j \in [\![1, n]\!]. \tag{3.26g}$$

Constraints (3.26b) require every customer to be assigned to a district. Constraint (3.26c) requires exactly m districts to be formed. Constraints (3.26d) state that each formed district must have a center. Constraints (3.26e) define the minimal workload of any district. Constraints (3.26f) stress that the workload (activity measure) within each district, together with the pendulum tour to and from the depot, cannot exceed the total time allowance.

Stage 1.2. Routing within individual districts

After clustering the customers, we form a single cycle inside each district containing all its customers and the depot. For each group of customers that forms a single district, we can adapt the TSP model proposed by Dantzig et al. [51] to generate the initial visiting sequence within each district:

$$\min \sum_{i=1}^{n} \sum_{j \neq i, j=1}^{n} d_{ij} x_{ij} \tag{3.27a}$$

subject to

$$\sum_{\substack{i \in [\![1,n]\!] \\ i \neq j}} x_{ij} = 1 \qquad j \in C, \tag{3.27b}$$

$$\sum_{\substack{j \in [\![1,n]\!] \\ i \neq j}} x_{ij} = 1 \qquad i \in C, \tag{3.27c}$$

$$\sum_{\substack{i,j \in Q \\ i \neq j}} x_{ij} \leq |Q| - 1 \qquad Q \subsetneq C, |Q| \geq 2, \tag{3.27d}$$

where x_{ij} is a binary variable indicating that arc (i, j) is traversed. Constraints (3.27b) and (3.27c) guarantee the flow conservation, and (3.27d) eliminate sub-cycles containing only a subset of nodes. A comprehensive review on the TSP heuristics methodologies and implementations can be found in Rego et al. [196]. However, considering the size of our problem, an exact solution can be obtained using existing solvers.

Stage 1.3. Improving the initial routes using ALNS

So far we have constructed the initial set of routes with the cluster-first-routing-second strategy as described in stages 1.1 and 1.2. To improve upon these routes, we employ the ALNS metaheuristic described in Section 2.4 with specified adjustments. The ALNS algorithm is tailored to fit the H-SARA problem framework in the following ways.

To start with, a list of destroy and repair operators removes a pre-defined number of customer nodes from the solution together with their linking arcs before adding them back iteratively, with the hope that the newly formed solution yields a smaller objective value consisting of the hiring, travelling and penalty costs. We introduce the following destroy operators tailored to the problem setting. The first three destroy operators are at the customer node level, while the latter two are at the routing level. We set by default the choice of $q = 5$ from our experimental results.

1. *Random Removal:* a group of q randomly selected customers are removed from their existing routes and placed inside the customer pool.

2. *Worse Removal:* a group of q customers with the highest removal gain are selected and removed from their existing routes. The removal gain refers to the difference of cost of having or not a customer inside a given allocated tour.

3. *Related Removal:* a single customer is randomly selected and moved together with the $(q - 1)$ nearest customers from their tours to the customer pool.

4. *Tour Removal:* randomly removes a single tour and moves all the allocated customers from this single tour to the customer pool.

5. *Longest Tour Break into Half:* breaks the longest tour found into two smaller tours of the same length (or with one containing one more customer). Link the starting and ending customers of the smaller tours to the depot.

After the destruction process, all customers inside the customer pool will be re-inserted by a repair operator selected from below:

1. *Greedy Insertion:* Randomly select a customer from the customer pool, insert it into the position that increases the total expected costs by the least. The insertion can be between two consecutive customers or between the depot and a linking customer.

2. *Greedy Insertion Perturbation:* The same mechanism as *Greedy Insertion*. However, the insertion cost of the selected customer at each specific position is influenced by a perturbation factor in the interval [0.8, 1.2].

3. *Greedy Insertion Forbidden:* Once again, the same mechanism as *Greedy Insertion*, only that a customer node cannot be re-inserted to the same position it has just been removed from.

For the H-SARA problem framework, the classical local search methods *move*, *swap*, and *2-opt* described in Figure 2.1 are applied after each destroy-repair iteration to further improve the repaired solutions. However, since local search is usually computationally expensive, we only wish to apply it to promising candidates whose objective values after the repair stage are within a limit of the best-found incumbent (default 30% from initial experiments).

Based on the customer layout, we propose a new local search method *overlap breaker* that merges two overlapping tours, breaking it in the middle and forming two smaller service tours that individually starts and ends at the depot. This further enhancement largely improves the solution quality from our preliminary results. A graphic description of the method is given in Figure 3.7. Since the destruction and reconstruction process allow us to modify the number of existing tours, it is therefore possible to re-optimise the fleet size during the ALNS improvement process. Consequently, ALNS allows our first-stage heuristic to be less affected by a fixed service team number m.

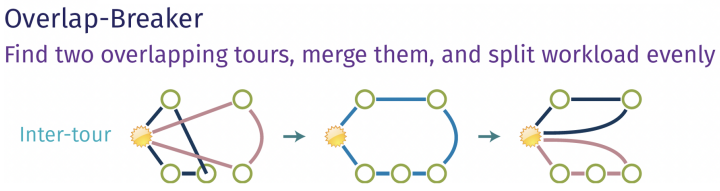


Figure 3.7: A graphical description of the overlap-breaker methods

We follow the classical ALNS framework in Section 2.4 with the embedded Roulette Wheel operator selector that dynamically adjusts the selection probability for each operator based on its historical performance, and the Simulated Annealing algorithm as the solution acceptance criterion to determine whether or not a newly-formed solution

should replace the incumbent for the search. It allows the algorithm to accept worsening moves, which is essential to diversification during the solution process. For the stopping criteria, we force the search to terminate after either a certain amount of time or a prescribed number of non-improving iterations is reached.

Stage 1.4. First-stage appointment scheduling

The last step of the first-stage heuristic is to notify all customers of their initial appointment time windows. Based on the set of ALNS-refined routes, we derive each individual appointment time for a customer from the associated team's arrival time at the customer using the expected travel and service times. To cope with potential customer cancellations and unpunctuality, we expand each individual appointment time into an appointment time window with fixed length T_1 and quote this individual-tailored appointment time window to every registered customer. For example, assuming $T_1 = 4$ hours and a customer's estimated appointment time is at 11:30 am, the first-stage appointment time window for this customer will be [9:30, 13:30] am.

Further Improvements

To further improve on the real-life practicality of our routes and schedules derived after the districting-first-routing-second construction heuristic and the ALNS improvement heuristic, we consider the following improvements for our first-stage solution: (i) team workload balance, (ii) multiple tours overlapping minimisation, and (iii) twisted tour elimination.

Each service team's assigned workload is bounded by (3.26e) and (3.26f), which means a team could still be assigned a much higher or lower workload compared to the rest of the teams if the bounds are loose. To further balance the workload amongst the teams, we include a soft workload balance penalty $P \cdot \max\{|\frac{\sum_{i \in k} \mathsf{AM}_i - \mu}{\mu}| - \alpha, 0\}$ in the ALNS objective function to penalise the extra units of workload above or below a certain threshold α for any service team (district k) and an average workload μ amongst all districts. We have chosen $\alpha = 0.3$ based on experimental results.

Occasional multiple tours overlapping is unavoidable, especially with a tight number of available service teams. Service durations have a larger scale than the inter-customer travel times, leading to customer assignments prioritising a good fit of customer service times into the remaining workload over the geographical adjacency. The randomness of customer geographical location can result in an unevenly high concentration of customers. The randomness then challenges the algorithm to form disjoint, compact, and contiguous driver zones within a reasonable computing time. However, the application of *overlap-breaker* (cf. Figure 3.7) and *2-opt* (cf. Figure 2.1) can remove the majority of overlaps and eliminate twisted tours that self-intersect.

3.7.4 Tour Refinements Stage

We make initial routing decisions during the first-stage heuristic without knowing which customers will remain in the system. At the beginning of the second stage an updated list of cancelled customers $\mathbb{I}$ becomes known. So we re-optimise the initial tours to fit the up-to-date customer information. The graphs in the right-hand-side column of Figure 3.6 shows the decision process for our second-stage tour refinement: we remove cancelled customers from the first-stage tours, compute the estimated arrival times for all non-cancelled customers, improve service teams workload balance, and notify all non-cancelled customers of a narrower appointment time window.

Stage 2.3. Second-stage Routing Improvements

At the beginning of the tour refinement stage, the list of newly cancelled customers is revealed and removed from the first-stage tours. From the right-hand-side column of the diagram in Figure 3.6, the second-stage stages 2.1 and 2.2 are relatively straightforward. For Stage 2.3, we re-apply the ALNS improvement heuristic to refine the routes and customer schedules. The differences in implementing ALNS at this stage are twofold:

First of all, in contrast to the expected travel and service times from known distributions used inside the first stage computation, we use a scenario-generated set of travel and service times at the second stage. Secondly, the objective function in the previous stage consists of total costs from team hiring, travelling, and working overtime. In contrast, our second-stage objective function consists of the averaged total costs from travelling, team hiring, and overtime plus penalties for workload imbalances and scheduling the new arrival times outside the first-stage appointment time windows. The last term encourages the algorithm to create a nested structure for the updated appointment times to lie within the previously determined time windows, which is essential to service quality because abrupt modifications to service starting times are highly unfavourable with customers.

Stage 2.4. Second-stage Appointment Scheduling

The service teams' arrival times to customers and the depot are random variables since they depend on travel and service times which are by definition random variables. This lead to our decision of quoting yet again a time to every non-cancelled customer, albeit a narrower one. Having the second-stage time window nested within the first-stage time window is essential for successful service and thus customer satisfaction. Specifically, we assume we have a first-stage time window $[T_1^{\text{start}}, T_1^{\text{end}}]$ and a second-stage estimated arrival time a_i at a non-cancelled customer i. We add to the ALNS objective term $P' \times \max\{T_1^{\text{start}} - a_i, a_i - T_1^{\text{end}}, 0\}$, which penalises any arrival time not nested within the first-stage time window. Similar to the first-stage appointment scheduling, we create a narrowed second-stage time window of length $T_2 = 30$ min. Moreover, the time windows are not necessarily centred at their arrival times. This is determined by a linear adjustment $[a_i - T_i^{\text{start}}] \cdot c_{\text{idle}} = [T_i^{\text{end}} - a_i] \cdot c_{\text{wait}}$ that forces the center forward in time to cope with more expensive waiting costs, or backward with more expensive idling costs.

3.7.5 Out-of-sample Performance Evaluation

We notice that the issue of data over-fitting might occur in our two-stage heuristic framework, since we only rely on in-sample objective values computed using a discretised set of scenarios n_e clustered from random samples. To avoid the solution being over-fitted to the in-sample population, we use a new set of benchmark scenarios to evaluate the additional out-of-sample performance of our second-stage solutions. This gives a fairer indication of how good the proposed level of service (in terms of cost and penalties) obtained from the heuristic is on an unobserved set of data. If the out-of-sample level of service deviates significantly from the in-sample level of service, then this indicates that the scenarios generated for the tour refinement stage are not a suitable representation of the uncertainties and, thus, the planning should be repeated with a newly generated set of scenarios.

3.8 Experiments

3.8.1 Experimental Setting

Since our problem setting is motivated by an application in home healthcare, we consider the parameter settings that were given in the AIMMS-MOPTA competition guidelines [213]. Specifically, we assume n customers are uniformly located over a 50×50 km geometric grid with the depot located at the origin $(0, 0)$. We set the fixed individual team hiring cost $f_m = 100$, hourly team idling time cost $c_{\text{idle}} = 2.5$, hourly overtime cost $c_{\text{over}} = 5$, and hourly customer waiting cost $c_{\text{wait}} = 4$. We also define the standard daily workload $L = 8$ hours for an individual team, the first-stage time window length $T_1 = 2$ hours, and second-stage time window length $T_2 = 30$ minutes. We assume the travel times between any two nodes are identically distributed with a log-normal distribution. This distribution has been applied in many existing studies for its recognition of the skewed distributions in modelling travel times [88]. For the customer service time, we select the gamma distribution that is not strictly symmetric in order to avoid generating a negative service time. We assume the expected service time $\hat{s} = \mu_s = 45$ min with its standard deviation set to $\mu_s/2$, the expected travel speed $\hat{v} = 1$ km/min (equivalently, expected travel time $\tau_{ij} = 1$ min/km). The traversing time on arc (i, j) can be computed by the travel time per unit distance multiplying the Euclidean distance. Moreover, we assume all customers share the cancellation probability defined at a fixed rate 5%. For the embedded ALNS metaheuristic, we adapt the list of parameter values listed in Table 3.3 based on our initial computational experimentation.

Table 3.3: ALNS Parameter values summary

Symbols	Parameter Names	Values
q	destroy scale	5%
λ	local search threshold	30%
δ_1	global best solution score	4
δ_2	local best solution score	2
δ_3	accepted worse solution score	0
l	weight segment length	15
α_{RW}	reaction factor	0.7
c_τ	temperature cooling coefficient	0.8
T	initial temperature	6
$iter$	non-changing iteration stop criteria	25

We implemented the exact methods and our two-stage heuristics framework in Python. For a unified measurement, we use CPLEX 20.1.0 as the optimisation solver for both exact methods and heuristics. The computations are performed on a machine with Intel Gold 6234 CPU and 512GB RAM installed. A single-core setting is used to run run a total of 10 test instances.

A Sampling-Based Objective Function

Since the travel and service durations as well as the customer cancellation list are random, we come up with a sampling-based objective function computed from a number of n_e randomly generated scenarios to guide the second-stage solution process, inspired

by the work of Sörensen and Sevaux [225]:

$$f^*(x,t) = \frac{1}{n_e} \sum_{i \in [\![1,n_e]\!]} f\big(x,t,S_i(\tau,s)\big) \tag{3.28}$$

where $f^*(x,t)$ is the expected total cost computed from a number of n_e randomly generated scenarios, x is the set of second-stage routes with the cancelled customers removed, and $S_i(\tau,s)$ represents the i^{th} scenario with stochastic travel and service times realisation. $f\big(x,t,S_i(\tau,s)\big)$ represents the total costs of the i^{th} scenario applied to x, and finally n_e is the total number of scenarios. The sampling-based objective function introduced above in (3.28) is used to compute the first-stage objective in (3.4a) for the experiments.

Monte-Carlo Simulation and Scenario Generation

We introduce a scenario-generating procedure to include a more diverse set of scenarios, and meanwhile, we wish to maintain a reasonable computational complexity by limiting the number of scenarios.

First, we assume a number of n_e scenarios and apply Monte-Carlo simulation to randomly generate n_s samples, each with a different pair of travel and service times realisations. If $n_s = n_e$, then each sample is an individual scenario. However, a large number of samples is usually required to leverage a probability distribution and project outcomes more accurately. Therefore, we generate a sufficiently large set of samples ($n_s \gg 30$) because of the *law of large numbers*, which states that the average experimental results converge to the expected value with a large number of trials. Mathematically, we can treat each generated sample as a vector consisting of the travel time realisation for every traversed arc and the service time realisation for every non-cancelled customer node. For example, for a second-stage solution with two teams travelling 8 arcs to serve 6 customers, the sample will be a vector of 14 values (8 travel times + 6 service times). Each value has been individually and independently drawn from the associated known distributions.

We then partition these n_s samples into n_e clusters using a k-means clustering algorithm. The sample at the center of each cluster is then taken as the representative scenario. The probability q^ω of each scenario ω is computed as the number of samples clustered together divided by the total number of generated samples. In this way, we are able to capture extreme values using a moderate number of scenarios.

3.8.2 Comparison of Exact Methods

We implemented the CPLEX stochastic full model algorithm (**MIP-full**), the CPLEX built-in Benders algorithm (**Benders-C**), and our customised Benders framework (**Benders**). MIP-full tackles the MIP model as a whole until an optimal solution is found or when the computational time limit is reached. Benders-C uses CPLEX's built-in Benders algorithm with the same decomposition framework but works as a black box. Our customised Benders algorithm, based on the framework showcased in the flowchart Figure 3.4, begins with a relaxed master problem that is iteratively checked by subproblems and has its feasibility region reshaped by optimality and feasibility Benders cuts until the master problem converges.

We compare the performance of the three algorithms and report the results in Table 3.4. The following Table 3.5 demonstrates the in-sample and out-of-sample analysis

for the algorithm performance. The maximal computational time for all three algorithms is fixed to be 1 hour. Due to the variations in service time, travel time, and cancellations between different generated scenarios, we run each customer $|C|$ and scenario $|S|$ combination for 10 times and return the averaged performance data. We apply the ALNS metaheuristic warm-start to produce an initial feasible solution from which all the algorithms begin. The generated Benders optimality and feasibility cuts and the set of valid inequalities in our algorithm are added to the master problem via the *LazyConstraintCallback*. We test the algorithms using customer numbers from $[10, 20, 30, 40]$. For the stochastic instances, we generate sets of $[20, 40, 60, 80, 100]$ scenarios for travel and service times from 200 random samples using the proposed scenario generation procedure. For each customer-scenario combination, we report the average computational time in seconds, "UB" as the best integer primal solution found in the branch-and-bound process, "LB" as the best lower bound dual solution found, and "Gap" as the percentage difference between "UB" and "LB", computed as $\text{Gap} = 100 \times (\text{UB}-\text{LB})/\text{UB}\,\%$. Moreover, we also report the number of Branch-and-Bound nodes, as well as the min, max, standard deviation, and mean for the in-sample and out-of-sample analyses.

In Table 3.4, our Benders algorithm exhibits superior performance, particularly evident in smaller instances comprising up to 30 customers. It achieves the smallest UB-LB gap across various customer and scenario sizes compared to the two alternative methods. However, as the customer size increases to 40 and coupled with more than 20 scenarios, Benders-C begins to outperform its counterparts, showcasing the most optimal gap with the most searched nodes amongst the three algorithms. In general, it is observed that as the size of customers and scenarios increases, the Benders and Benders-C algorithms exhibit a more significant outperformance compared to the MIP-full algorithm in terms of gap.

While the results depicted in Table 3.4 for the exact methods may not show significant differences among the Benders and Benders-C algorithms, the implementation of our customised Benders confers advantages in terms of strengthening the optimality gap, especially evident across instances with 10, 20 and 30 customers. It also yields a tighter upper bound usually lower than that provided by Benders-C, thus signifying a superior quality of the best known feasible solution derived from the master problem. Furthermore, our Benders demonstrates competitive results compared to Benders-C in generating the lower bound through solving the set of subproblems to determine feasibility.

The computational time for solving master-level instances averages around 1 second, and 0.02 second for the subproblem instances. Despite these fast processing times, the sheer number of possible customer combinations to form tours of any size remains exponential, rendering the verification of a master-level route plan a time-consuming task as there exist exponential number of tours of the same size which only slightly differs in the total costs. The existence of many tour results in the huge number of cuts generated, as the results indicated an average of 2 Benders' cuts per iterations with a total number of more than 50000 master iterations for the 40-customer instance. This observation is evident in the experiment, where we observed that the lower bound typically closes in fast, while the upper bound progresses slowly. Such behaviour suggests that the algorithm invests considerable time generating optimality cuts. Even when starting with the optimal fleet size, the algorithm still necessitates evaluating all possible tours before confirming optimality of fleet size.

We also compared in Table 3.4 the Benders framework and our customised ALNS-based heuristic against the built-in heuristics inside CPLEX, denoted as "CPLEX-heur". The algorithm setting is adjusted by turning on the relevant parameters for

heuristics inside the CPLEX solver, turning off the cutting planes, and solving only the root node before branching begins. We observe a significant difference between the CPLEX built-in heuristics and other methods, indicating the need of a more robust heuristic framework tailored to the complex problem. Since the ALNS heuristic framework can only tackle the H-SARA problem in its entirety, we have included only the ALNS objective value in the "Value" column of the table, which is equivalent to the primal solution, along with the corresponding computation time. It is evident from the table that the ALNS demonstrates competitive performance compared to the built-in CPLEX heuristics and exhibits time efficiency comparable to that of the CPLEX built-in heuristics. Consequently, the results demonstrate the benefits of using ALNS as a heuristic approach.

We name instances with a certain number of customers as a critical point, where the fleet size will increase by one with any additional customers. Since the team hiring costs dominate the travel costs and other penalty costs, increasing the fleet size by one leads to a significant difference in the total costs. During the solving process, an update in fleet size can also cause a sudden jump in the LB when specific scenarios contain large service and travel times that make the master-level routes infeasible. The occurrence of a critical point is the reason behind a sudden decrease in the gap from $|S| = 20$ to 60 for the 20-customer instances. This is because more scenarios can appear that require the creation of another service tour to accommodate the unusually high travel and service times. A larger scenario size means that more extreme scenarios like this can be included. The warm-start initial solution is more likely to reflect this by building an initial set of tours with a larger fleet size, and hence influence the evolving process of the UB and LB.

In Table 3.5, the observed trend in the out-of-sample compared to the in-sample results among the three algorithms remains consistent. Despite the the slightly larger standard deviation indicating greater variation in performance for our Benders, both the in-sample and out-of-sample average objective values are comparable to the alternative methods.

Table 3.4: Comparison of CPLEX stochastic whole model, CPLEX built-in Benders, our implemented Benders algorithm, and our customised ALNS-based heuristic.

		Benders-C					**Benders**					**MIP-full**					**CPLEX-heur**				**ALNS**	
$\|C\|$	$\|S\|$	**Time**	Gap	LB	UB	Nodes	**Time**	Gap	LB	UB	Nodes	**Time**	Gap	LB	UB	Nodes	**Time**	Gap	LB	UB	Time	Value
10	**20**	480.1	6.81%	189.50	203.32	200	370.4	3.59%	195.89	203.15	270	480.4	6.84%	189.59	203.49	51	0.27	13.43%	189.54	223.51	1.07	203.41
10	**40**	0.3	0.5%	202.89	203.62	0	10.3	0.5%	202.70	203.38	0	1.7	0.5%	202.94	203.78	0	0.37	8.69%	202.95	229.17	0.90	203.76
10	**60**	0.5	0.5%	202.94	203.70	0	16.2	0.5%	202.78	203.36	0	1.8	0.5%	202.99	203.78	0	0.66	0.37%	203.01	203.76	0.88	203.81
10	**80**	1029.0	14.3%	174.55	203.63	253	482.9	0.5%	202.49	203.20	1079	1032.9	14.42%	174.35	203.66	17	1.31	19.33%	174.28	218.50	1.18	203.67
20	**20**	3323.1	30.52%	211.71	304.71	2975	3325.0	30.47%	211.66	304.38	3416	3323.4	30.57%	211.57	304.73	148	1.33	35.47%	211.42.	337.31	4.55	305.21
20	**40**	3600+	33.12%	204.13	305.24	1308	3600+	32.97%	204.17	304.61	6179	3600+	33.21%	203.97	305.37	61	2.17	44.53%	203.84	373.93	5.75	338.88
20	**60**	3600+	33.22%	204.03	305.53	522	3600+	32.95%	204.12	304.43	2019	3600+	33.29%	203.90	305.63	18	4.29	54.99%	203.72	458.56	4.08	356.04
20	**80**	3168.4	21.02%	241.71	306.02	384	2285.5	20.74%	241.65	304.84	3048	3600+	21.09%	241.59	306.16	8	4.54	46.00%	254.01	475.64	6.73	372.80
30	**20**	3600+	24.97%	304.80	406.22	2255	3600+	24.86%	304.76	405.59	2102	3600+	25.07%	304.67	406.60	41	4.99	36.63%	304.57	501.45	10.19	406.52
30	**40**	3600+	25.08%	304.72	406.72	1305	3600+	24.88%	304.70	405.63	3315	3600+	25.3%	304.56	407.70	7	5.90	51.88%	304.42	642.33	17.68	423.66
30	**60**	3600+	29.45%	304.62	436.03	804	3600+	27.1%	304.58	420.30	1346	3600+	29.68%	304.45	437.10	2	11.68	57.37%	304.26	713.82	8.92	540.53
30	**80**	3600+	34.13%	304.78	468.11	406	3600+	30.91%	304.75	446.34	1262	3600+	37.12%	304.64	488.18	1	8.76	62.80%	304.38	818.14	15.33	608.28
40	**20**	3600+	20.18%	405.58	508.11	1440	3600+	20.02%	405.47	506.99	308	3600+	20.51%	405.43	510.03	11	8.81	50.95%	405.28	865.68	21.41	508.51
40	**40**	3600+	23.58%	405.47	533.78	918	3600+	25.02%	405.41	544.92	674	3600+	28.72%	405.33	572.79	1	14.37	56.83%	405.20	949.21	36.90	542.36
40	**60**	3600+	29.81%	405.34	581.29	583	3600+	31.49%	405.25	593.90	82	3600+	31.86%	405.20	596.80	0	16.84	55.93%	404.45	917.82	57.10	609.58
40	**80**	3600+	33.6%	405.42	610.58	517	3600+	34.39%	405.35	619.22	236	3600+	36.5%	405.23	644.02	0	24.37	60.08%	405.42	1015.59	38.86	659.20

64

Table 3.5: In-Sample and Out-of-Sample performance Comparison of CPLEX stochastic whole model, CPLEX built-in Benders and our implemented Benders algorithm

		Benders-C				**Benders**				**MIP-full**			
$\|C\|$	$\|S\|$	IS-Mean	IS-Std.	OS-Mean	OS-Std.	IS-Mean	IS-Std.	OS-Mean	OS-Std.	IS-Mean	IS-Std.	OS-Mean	OS-Std.
10	**20**	203.36	0.56	203.59	1.05	203.36	0.56	203.59	1.05	203.60	0.94	203.76	1.33
10	**40**	203.69	1.20	203.89	2.18	203.88	1.52	204.11	2.54	203.87	1.81	203.98	2.18
10	**60**	203.89	2.10	203.98	2.51	204.03	2.40	204.17	2.67	204.04	2.48	203.98	2.18
10	**80**	203.89	2.31	204.24	3.36	203.98	2.24	204.42	3.73	203.99	2.35	203.95	2.25
10	**100**	204.02	2.12	204.31	2.99	204.02	2.12	204.31	2.99	204.13	2.52	204.40	3.31
20	**20**	304.75	0.61	305.02	1.37	306.65	2.53	306.82	2.72	304.77	0.63	304.96	1.12
20	**40**	305.37	1.64	306.33	4.59	306.23	3.78	306.62	5.89	305.50	2.11	305.99	3.70
20	**60**	306.07	4.58	306.78	5.31	306.52	5.60	306.91	5.60	306.11	3.30	307.00	5.07
20	**80**	306.77	4.74	306.74	5.05	307.39	6.16	307.01	5.82	307.09	5.72	306.92	5.54
20	**100**	326.32	3.77	326.67	4.83	307.23	6.33	306.83	6.10	306.64	5.29	306.66	4.92
30	**20**	406.29	0.78	406.87	2.49	408.57	2.68	409.14	3.79	406.67	1.19	406.96	2.18
30	**40**	407.07	2.56	408.15	5.23	408.26	4.21	408.81	4.87	408.12	3.71	409.01	5.78
30	**60**	436.97	5.33	437.64	6.82	423.47	6.28	423.87	7.28	437.81	4.98	438.40	6.65
30	**80**	469.29	5.89	469.52	6.20	449.88	7.02	449.91	8.71	489.35	6.51	489.73	7.20
30	**100**	465.50	4.92	466.51	6.97	466.26	5.99	467.29	8.35	494.84	5.32	495.38	6.35
40	**20**	508.18	1.29	508.65	2.06	512.36	2.82	512.42	2.90	510.19	2.23	510.44	2.59
40	**40**	534.49	4.20	534.96	4.48	548.50	5.28	548.76	6.00	573.38	4.43	573.64	6.03
40	**60**	582.24	5.26	583.35	6.91	596.07	4.73	597.04	6.31	597.84	6.31	598.01	6.66
40	**80**	611.57	5.05	612.79	7.22	621.80	5.42	622.92	7.59	645.37	6.74	646.13	8.54
40	**100**	610.46	5.00	611.68	7.96	627.80	5.89	628.74	8.09	662.95	7.72	663.00	7.53

In Table 3.6, we present the average Value of the Stochastic Solution (VSS) and the average Expected Value of Perfect Information (EVPI) for 10 customers across various scenarios from $[20, 40, 60, 80, 100]$. VSS computes the expected gain from solving a stochastic model than its deterministic counterpart, and EVPI measures the price of gaining perfect information. To generate both the VSS and EVPI, we must solve the deterministic version of our problem to optimality, which means that data pertaining to these aspects are only available for smaller instance sizes. Nevertheless, we have calculated the relevant statistics for 10 customers across different scenarios.

Table 3.6: EVPI and VSS performance Comparison of CPLEX stochastic whole model, CPLEX built-in Benders and our implemented Benders algorithm

		Benders-C		**Benders**		**MIP-full**	
$\|C\|$	$\|S\|$	EVPI	VSS	EVPI	VSS	EVPI	VSS
10	**20**	3.08	0.34	3.08	0.34	3.32	0.16
10	**40**	3.54	0.69	3.73	0.63	3.72	0.55
10	**60**	3.79	0.55	3.93	0.41	3.94	0.45
10	**80**	3.80	1.23	3.88	1.15	3.90	1.12
10	**100**	3.92	0.93	3.92	0.93	4.03	0.83

3.8.3 Analysis of Lifted Constraints

In Table 3.7 and Table 3.8, we report the results produced by the MIP-full algorithm with and without the lifted formulation proposed in Section 3.6.2. For the experiment below, we abbreviate "MIP-full-L" for the CPLEX stochastic full model with the lifted formulation and "MIP-full-N" for the non-lifted alternative. "UB", "LB" and "Gap" as in the previous experiment are the best integer solution from branch-and-bound, the best bound, and the percentage difference, respectively. The "Gap-diff" refers to the value difference between the lifted and non-lifted versions, computed as:

$$\frac{(\text{GAP}_{\text{MIP-full-N}} - \text{GAP}_{\text{MIP-full-L}})}{\text{GAP}_{\text{MIP-full-L}}} \tag{3.29}$$

We set the computational time limit to be 1 hour and each row of the table is an average computed from 5 tests. We also enabled the ALNS metaheuristic warm-start for both algorithms with and without the lifted formulation.

From Table 3.7, we observe a better performance for MIP-full-L, reflected by a smaller UB-LB gap within the same computational time limit of an hour. MIP-full-L outperforms its non-lifted counterpart for almost all the instances apart from the $|C| = 40$, $|S| = 20$ case. For instances with a fixed customer size, MIP-full-L shows better effectiveness in finding stronger LB. This difference is the largest for the instance with $|C| = 20$ and $|S| = 80$. Nevertheless, we observe that a large number of lifted constraints is added to the master problem that, on the one hand helps with locating the integer solution with higher quality in each iteration, whereas on the other hand, it can slow down the whole iterative convergence process by burdening the solving process with a more extensive set of constraints.

To cope with this, Table 3.8 gives another set of experimental results, for which we fix the number of branch-and-bound nodes to be visited for both MIP-full-L and

Table 3.7: Comparison of MIP-full-L and MIP-full-N algorithms

		MIP-full-L			MIP-full-N			
$\|C\|$	$\|S\|$	UB	LB	Gap	UB	LB	Gap	Gap-diff
20	20	305.33	203.52	33.34%	305.33	203.47	33.36%	0.05%
20	40	305.41	203.41	33.40%	305.41	203.33	33.42%	0.08%
20	60	306.10	237.21	22.51%	306.09	236.78	22.64%	0.61%
20	80	306.57	303.45	1.02%	306.60	303.34	1.06%	4.64%
20	100	307.32	303.38	1.28%	307.32	303.35	1.29%	0.76%
30	20	407.12	303.85	25.37%	407.10	303.67	25.41%	0.16%
30	40	408.80	303.60	25.73%	408.88	303.55	25.76%	0.10%
30	60	458.46	303.42	33.82%	458.46	303.24	33.86%	0.12%
30	80	508.16	303.22	40.33%	508.16	303.15	40.34%	0.03%
30	100	509.04	303.17	40.44%	509.04	303.13	40.45%	0.02%
40	20	509.56	405.40	20.44%	509.56	405.41	20.44%	-0.01%
40	40	576.44	405.21	29.71%	576.44	405.15	29.71%	0.03%
40	60	609.41	405.11	33.52%	609.41	405.09	33.53%	0.01%
40	80	609.73	404.98	33.58%	609.73	404.98	33.58%	0.00%
40	100	610.51	404.72	33.71%	610.51	404.70	33.71%	0.00%

Table 3.8: Comparison of MIP-full-L and MIP-full-N algorithms (with fixed number of integer nodes)

			MIP-full-L			MIP-full-N			
$\|C\|$	$\|S\|$	**nodes**	UB	LB	Gap	UB	LB	Gap	Gap-diff
20	20	100	305.22	203.97	33.17%	305.38	203.92	33.23%	0.16%
20	40	100	306.45	203.94	33.45%	306.43	203.74	33.51%	0.19%
20	60	100	306.89	203.88	33.56%	306.89	203.62	33.65%	0.26%
20	80	40	307.30	203.77	33.69%	307.30	203.53	33.77%	0.23%
20	100	15	307.69	203.10	33.99%	307.69	203.02	34.02%	0.08%
30	20	100	407.66	304.87	25.21%	407.66	304.45	25.32%	0.41%
30	40	100	408.99	304.62	25.52%	408.99	304.28	25.60%	0.33%
30	60	40	467.84	304.37	34.94%	467.84	304.19	34.98%	0.11%
30	80	10	508.94	304.10	40.25%	508.94	304.01	40.27%	0.04%
30	100	5	509.75	303.48	40.47%	509.75	303.41	40.48%	0.03%
40	20	10	509.97	405.42	20.50%	509.97	405.32	20.52%	0.10%
40	40	10	586.48	405.04	30.94%	586.48	404.95	30.95%	0.05%
40	60	10	609.76	404.94	33.59%	609.76	404.89	33.60%	0.03%
40	80	5	609.97	404.50	33.68%	609.97	404.47	33.69%	0.01%
40	100	3	610.85	404.27	33.82%	610.85	404.25	33.82%	0.01%

MIP-full-N algorithms and compare the performances between them. The number of integer nodes is determined from the previous experiments so that both versions of the algorithm can explore the same number of branch-and-bound nodes within the time limit of an hour. On reaching the node limit, the algorithm will return the UB and LB. We observe a more extensive performance difference between MIP-full-L and MIP-full-N, especially for instances with many customers and scenarios. This shows our lifted formulation contributes to a faster convergence for the stochastic full model.

3.8.4 Two-stage Heuristic Approach

In this section, we report the performance of our two-stage heuristic. For comparison, we include the following algorithms: **SVRP** is our stochastic H-SARA-2 model solved with the proposed time-saving root node approach that pre-fixes the fleet size m. **2-Stage Heur** is our two-stage heuristic method with the local search operators enabled in the ALNS improvement process. **2-Stage ALNS** is our proposed two-stage heuristic method without the tour-overlap-breaker local search operator in the ALNS improvement process. This model applies the classical ALNS and is regarded as the benchmark model in comparison to the two-stage heuristic. **"1-stage" Heur** is solved as a comparison to our two-stage heuristic, assuming the full customer cancellation list being available before the initial planning stage. Thus the second-stage re-routing and re-scheduling are excluded from the solution process. We solve this by not removing any customers at the second stage. This comparison tells how much customer cancellations cost the business apart from other uncertainties.

The experiment results are given in Table 3.9, where all the results (Score and Time) are averaged from 10 experiments. Specifically, Score is the expected objective function value averaged from 10 experiments on 100 test instances generated out of 10,000 samples by k-means. Time is measured in seconds, and t^* refers to the upper limit of computing time which is 1800 seconds. Scenarios used in methods are generated randomly and independently from test instances. We have observed the following points: To begin with, our two-stage heuristic can tackle larger customer sizes within a reasonable time. It takes no more than 2 minutes on our computer to compute a solution for a 40-customer instance, whereas the deterministic model requires 19 minutes on average, and the stochastic model cannot even terminate within 30 minutes. If we further increase the model's size to 100 customers, none of the exact MIP approaches can terminate in hours, but our two-stage heuristic can still obtain results within 5 minutes, and within 10 minutes for 150 customers.

For the solution quality, our two-stage heuristic provides competitive solutions compared to CPLEX solutions. By comparing same-scenario columns between the exact methods and the two-stage heuristic, we observe that within the given time limit, our two-stage heuristic is able to find solutions within 4% of the solutions computed by CPLEX. Even though all exact and heuristic methods columns are non-optimal (since the global optimum is extremely difficult to compute), we want to showcase the fact that our two-stage heuristic is able to provide same-quality solutions and within less amount of time compared to CPLEX. Besides, the two-stage heuristic is more robust in real-life applications and can provide up-to-date decisions at different service preparation stages based on different levels of available information.

Hypothetically, if we obtain the complete customer cancellation information in the first place, we can simply merge the two heuristic stages and deal with only stochastic travel and service times. To determine the additional cost of making multi-stage decisions, we run a parallel experiment "1-stage Heur", assuming complete information

Table 3.9: Computational results from different solution methods

Scenarios	**1**								**10**				**20**				**50**				**100**			
Method	SVRP		2-stage Heur		2-stage ALNS		"1-stage" Heur		SVRP		2-stage Heur		SVRP		2-stage Heur		SVRP		2-stage Heur		SVRP		2-stage Heur	
Customers n	Score	Time	Score	Time	Score	Time	Score	Time	Score	Time	Score	Time	Score	Time	Score	Time	Score	Time	Score	Time	Score	Time	Score	Time
10	209.23	1.638	169.42	1.28	168.79	1.27	169.07	0.81	211.04	1.93	172.54	2.49	209.10	2.29	164.03	3.84	208.19	3.47	161.28	7.84	205.94	13.20	162.74	14.08
20	236.65	34.55	235.51	15.75	238.43	15.83	234.56	8.16	237.35	192.46	237.71	19.96	-	t*	240.46	24.64	-	t*	237.15	36.56	-	t*	235.76	57.73
30	315.68	364.54	318.77	19.87	318.96	19.87	315.64	15.32	-	t*	318.83	27.68	-	t*	319.55	35.77	-	t*	318.08	50.12	-	t*	318.95	91.42
40	410.30	1115.24	418.28	35.21	418.72	35.48	409.82	23.52	-	t*	417.54	47.04	-	t*	424.14	58.98	-	t*	417.62	95.54	-	t*	417.62	157.54
50	-	t*	521.91	72.87	522.50	73.98	512.59	50.90	-	t*	521.37	88.80	-	t*	521.64	105.65	-	t*	521.82	154.27	-	t*	521.22	239.22
100	-	t*	1012.28	198.84	1016.24	199.51	1020.43	173.96	-	t*	1018.09	256.23	-	t*	1021.63	310.54	-	t*	1004.83	459.90	-	t*	998.48	790.11
150	-	t*	1459.51	609.81	1465.49	612.87	1504.49	586.06	-	t*	1460.89	716.40	-	t*	1460.82	827.44	-	t*	1461.24	1148.10	-	t*	1460.78	1645.79

∗ Notice that "2-stage" is used as an abbreviation to represent "two-stage" in the heuristic experiment for space saving.

for cancelled customers. It achieves lower objective costs than the two-stage heuristic "2-stage Heur", which receives no customer cancellation list but only cancellation probability during the initial planning stage. Yet, our two-stage heuristic is not worse-off in terms of average objective values and computing time from the results. For experiment sets with 100 and 150 customers, "2-stage Heur" outperforms "1-stage Heur" in the expected objective function value although with slightly longer computing time on average. We recognise two potential reasons behind this phenomenon: local search-based heuristics cannot guarantee the global optimum in general, and the solutions computed by "1-stage Heur" being over-fitted to the single scenario than the benchmark instances/scenarios from the evaluation stage. For more than 10 customers, the "2-stage Heur" and "2-stage ALNS" columns show that our Overlap-breaker operator further improves the convergence speed and solution quality comparing to a classical ALNS heuristic.

Thus, we are able to include customer cancellations into our solution process and make initial decisions based on probabilistic customer cancellations, all at a reasonable additional cost. The additional cost is mainly due to our requirement to nest the second-stage narrower appointment time window within the first stage's, thus limiting the freedom to optimise the best routes and leading to slightly worse-off solutions. However, no perfect information exists in reality. The differences between one-stage and two-stage solutions can be treated as the costs of "imperfect information" (a priori decisions and previous-day customer notifications without getting the complete picture).

3.8.5 Analysis of Cancellation Policy

In this section, we design a second set of experiments to analyse the different cancellation types and their effect on the customers, service teams, and the decision-making process. We assume that the in-time cancellation policy allows sufficient time for rescheduling the service to prevent visits to the cancelled customers. In contrast, under the last-minute cancellation and no-presence policies, the service teams have to visit the cancelled customers, such that the arcs linking the cancelled nodes will be traversed and the service time at each cancelled node will be adjusted to 0 and 15 minutes, respectively. We fix a cancellation rate of 0.1 across all the following experiments.

From Figure 3.8, last-minute and no-presence cancellations result in less efficient schedules and higher idling time and overtime for the service teams. Since last-minute and no-presence cancellations occur only after the tour refinement stage in our two-stage decision process, the second-stage routes and schedules fail to satisfactorily adjust to the sudden change, which results in an additional idling time (at the consecutive customer nodes) of 25 minutes per service team if all customers cancel their service last minute. The no-presence scenario brings an additional 14 minutes of idling time to each service team on average, which is a 78% increase compared to the in-time cancellation scenario.

For instances with more than 30 customers, the last-minute and no-presence cancellations lead to a more prolonged service team overtime. Nevertheless, a longer idling time due to last-minute and no-presence cancellations can bring down the average waiting time for customers, since service teams arrive earlier than scheduled at the following customers. If all customers follow last-minute cancellation or no presence, the average overtime for each team will be 18 minutes more, which is a 40% increase, but the average customer waiting time will be 11 minutes less over all the experiments.

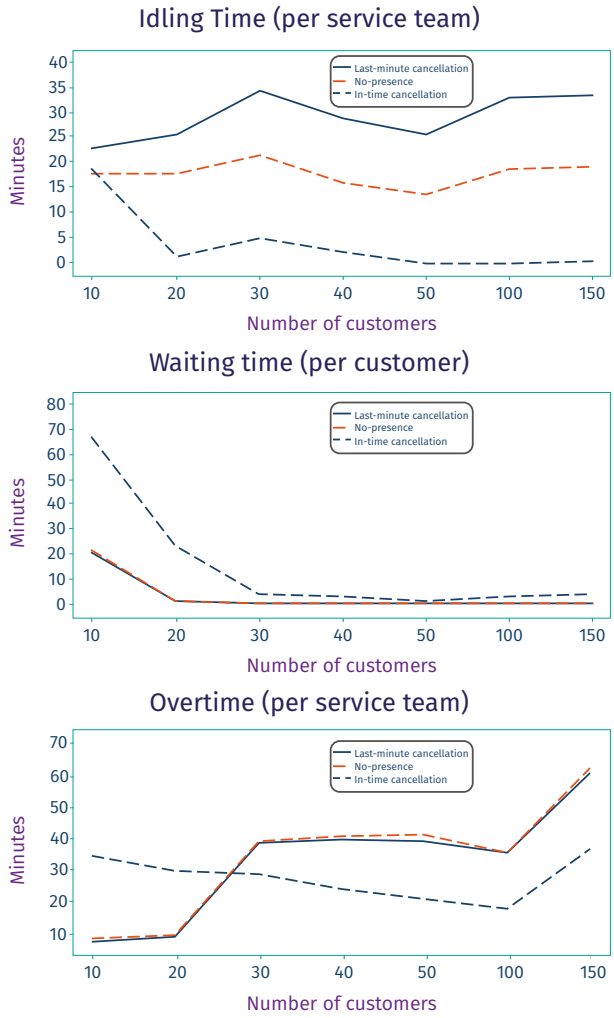


Figure 3.8: The idling time, waiting time and overtime from different cancellation policies

3.9 Conclusions

This paper studied a problem that integrates fleet-sizing, assignment, routing, and scheduling problems. Deterministic and stochastic MIP models have been proposed and solved using both CPLEX built-in methods and a customised Benders' decomposition algorithm. Acceleration approaches have been proposed based on the natural partition of our two-stage stochastic model to tackle small-sized instances. Firstly, we proposed a set of valid inequalities for the master problem that proved to be beneficial for reducing the number of feasibility cuts needed to be added and overall speeding up the convergence of the algorithm. A master-level lifted formulation was beneficial for reducing the number of feasibility cuts and accelerating the convergence of the algorithm. Besides, we introduced a closed-form solution tailored to the problem structure has been derived for the primal subproblem to allow the computational time to increase linearly with the instance size. We also developed an adaptive large neighbourhood search metaheuristic-based warm-start process to accelerate the convergence process.

For larger instances, we developed a tailored two-stage heuristic solution method with an embedded ALNS improvement heuristic to support a real-life decision-making process taking the evolution of information into account. Our proposed heuristic shows good computational time and solution quality performance. It also demonstrates flexibility and robustness in adapting to multiple scenarios with different travel times, service times, and customer cancellation rates. Using our decision support framework, we can provide high-quality fleet sizing, districting, routing, and scheduling decisions with low idling, waiting, and overtime costs, as well as two sets of customer appointment time windows, and a balanced service team workload within geographically clear service zones. The time and quality-wise good results showcase the strength of ALNS metaheuristic. Good adaptability to highly complicated problems containing multi-aspects and uncertainties, and meanwhile, it is relatively easy to implement and tailor to give good result. Potential avenues for future research could include the integration of a three-indexed traversal variable, which increases the number of constraints but can offer a stronger formulation that facilitates faster convergence within the Benders framework. Another potential direction is to introduce stronger optimality cuts and strengthen the current ones, since numerous optimality cuts are generated during the current solving process, but they prove to be ineffective due to the frequent iterations of including new cuts.

Chapter 4

Application of ALNS in Food Distribution Problem

4.1 Introduction

4.1.1 Motivation

Nowadays, many local restaurants in urban areas rely on food delivery services from outside the urban area to replenish their food supplies. These restaurants benefit from specialised food-preservation techniques and more affordable shipping costs due to the economies of scale resulting from large-scale operations in the supply chain, as opposed to individual manufacturers making direct shipments to restaurants on smaller scales. As the size of a city and the complexity of its supply chain grow, the introduction of intermediate transport points breaks down the entire network into smaller and more manageable parts, allowing for specialisation in the tasks performed by vehicles and service teams at different levels. On a larger scale, this approach avoids heavy trucks travelling directly to city centres and thus reduces environmental impact by alleviating traffic congestion and pollution.

Coordination of operations in such transport points through the consolidation of multiple shipments, as described by Crainic et al. [45], facilitates the movement of a large number of food products through the network more efficiently and at a lower cost. Consolidation centres (CCs) and local cross-docks/depots (LCs) serve as transfer points that help to disaggregate, sort, and repack decompressed freight into individual orders. Nowadays, domestic retailers and foreign manufacturers dispatch freight in the food supply chain that is shipped to CCs on the outskirts of the city, where food products on pallets are sorted into customer-based roll cages. The commodities then go through a cross-docking process at the LCs inside the city, in non-central areas, where shipments from heavy trucks are loaded onto lighter, city-centre-friendly vehicles to comply with municipal regulations and achieve better mobility for final deliveries.

As food perishability profoundly influences the food industry supply chain, proper locations for the CCs and LCs facilities, allocations of customers to facilities and drivers, and end-to-end routing are needed to optimise the intensive shipments of massive multi-commodity food products through the network. However, the dynamic nature of the customer list and uncertain demand patterns pose challenges to future route planning and the design of drivers and LC-based service territories. Furthermore, the variability in the quantities of ambient, frozen and chilled commodities necessitates the acquisition of heterogeneous delivery vehicles that with different proportions of chilling and freezing

capacity.

Due to the complexity of the network design and the numerous required model features to consider, the problem is difficult to solve using existing models based on individual customers. In this chapter, we propose a MIP model and a heuristic solution approach to efficiently plan customer deliveries in densely populated urban areas. Customer-based aggregations are often employed to break down the whole problem into individual catchment areas. In our algorithm, the whole coverage area can be divided into single driver-servable cells, which are then aggregated into facility-based districts, providing a basic template for the routing and location decisions. This approach significantly reduces the computational complexity of the problem and can be easily extended with a periodic improvement stage to maintain driver workload balance and customer service consistency on a driver and facility basis.

4.1.2 Model Features

The focus of this research is a food distribution problem that can be considered as a generalisation of the two-echelon Location Routing Problem (LRP) with multiple periods, multiple commodities, and heterogeneous vehicles, as illustrated in Figure 4.1. The first echelon of the transportation process includes split deliveries, which enable multiple centralised CCs to replenish a single local LC, while the second echelon of the chain involves single vehicle deliveries to each customer in a round trip. The first echelon utilises homogeneous vehicles for replenishment purposes, whereas the secondary echelon employs heterogeneous vehicles that are equipped with different capacities for room-temperature ambient food and food that requires cooling or freezing. The cooling capacity of any vehicle is semi-permeable, allowing the cooling proportion to be adapted to store ambient food, whereas any remaining ambient capacity cannot accommodate food stored in the frozen or chilled section. Every customer has a fluctuating demand for ambient and frozen/chilled supplies in all periods throughout the planning horizon. The demand of each customer for each type of product must be satisfied during each time period. Aggregated heterogeneous demand from a driver-based district must fit into the truck capacity since no split delivery is allowed for the second echelon. If exceed the largest vehicle capacity, the customer has to move to another driver zone and served by a new vehicle.

In this context, the decisions to be made include determining the locations of the CC and LC facilities, allocating customers to their corresponding drivers and LCs, devising replenishment routes from CCs to LCs, and establishing delivery routes from LCs to customers. The second echelon deliveries are constructed based on driver-based and facility-based districts. The facility-based districts are updated tactically, aiming to ensure LC-to-customer supply consistency. The driver-based districts are updated more frequently, aiming at driver-to-customer service consistency and workload balance amongst drivers.

There are several interdependent subproblems embedded within the distribution model: a strategic Facility Location Problem (FLP) carrying a long-term effect on urban construction, a tactical Districting Problem (DP) creating delivery zones by allocating individual customers to drivers (forming driver-based zones) and drivers to LCs (forming LC-based zones), and an operational Vehicle Routing Problem (VRP) computing daily delivery routes. Treating these subproblems separately leads to easier-targeted yet sub-optimal solutions, which stresses the necessity to properly combine all three levels of decisions to build a single model and an integrated algorithm for the system. In our heuristic approach, we develop different stages that are linked

and feedback to each other: we have a master stage that determines the CC and LC locations and LC-based districting, and a mid-term planning stage with tactical driver-based districting and routing within each driver's territory, adjusted based on fluctuating customer demand. The two stages are interdependent and linked by an interacting loop.

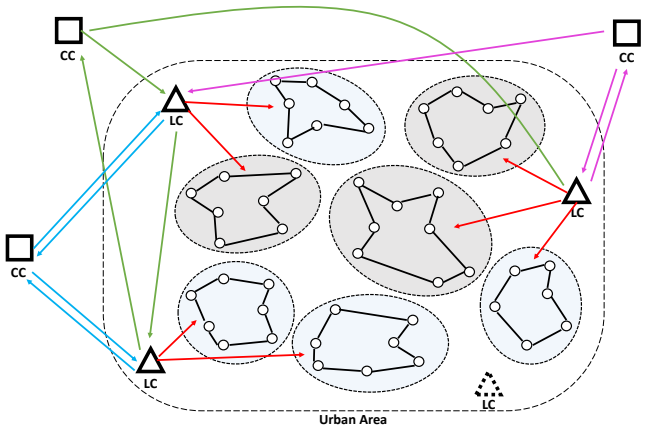


Figure 4.1: Supply Chain Design: the first echelon is from squares to triangles, and the second is from triangles to circles. Driver districts with the same colour hire identical secondary vehicles. Each LC's catchment area consists of driver districts allocated to the same facility.

As depicted in Figure 4.1, this study is founded on a multi-echelon model encompassing three levels of nodes: CCs positioned on the outskirts of the city, LCs located within the city, and Customers located at the city center. The first echelon encompasses nodes CCs and LCs, along with the interconnected arcs linking any pair of first-echelon nodes. The second echelon consists of LCs and Customers together with their linking arcs. Specific constraints are imposed on the functionalities of the facilities: CCs function as centers for freight repacking process, while LCs serve as load-on-load-off vehicle transfer hubs with no storage availability. The operation involves the transportation of multiple types of food products, including ambient and frozen/chilled items, originating from one or multiple CCs to one or multiple LCs. Due to food perishability considerations, storage at any specific level of the supply chain is inherently temporary. Therefore, the current study does not consider long-term storage arrangements at the CC and LC level.

4.2 Literature Review

The Location-Routing-Districting (LoRD) problem relates to a set of widely studied problems in both academia and industry. The following sections summarise the different models found in the literature relating to this research.

4.2.1 VRP

The first relevant problem is the VRP, a generalisation of the well-known Travelling Salesman Problem (TSP), which seeks a Hamiltonian cycle of minimum length. The earliest mathematical work on the VRP dates back to Dantzig and Ramser [50] on modelling a fleet of capacitated homogeneous trucks to replenish a group of gas stations

departing from a central hub. For a typical VRP, the main aim is to determine a set of minimum-distance tours visiting all the locations and starting and ending at a depot, meanwhile satisfying a list of general limitations including space and time capacity, time windows, and traversal distance, etc. With numerous applications in logistics, transportation and general distribution management, the VRP has been exclusively studied in the literature and has been extended with various variants and applications, the readers are referred to Laporte [135].

Multi-echelon networks have been commonly considered in the VRP variants. The earliest paper about Two-echelon VRP (2E-VRP) was proposed by Jacobsen and Madsen [112] on a Two-echelon Routing-Location problem, which emerged from the problem of designing the number and locations of transfer points inside a newspaper distribution system. The general 2E-VRP was first modelled and evaluated by Crainic et al. [45] as a short-term scheduling and resource management problem for an integrated two-tiered distribution system. Later, Perboli et al. [182] introduced the family of the Two-Echelon Capacitated VRPs (2E-CVRPs) with variants. The authors proposed an Integer Programming (IP) formulation with a set of valid inequalities and presented two math-based heuristics to deal with the addressed problem. Baldacci et al. [17] formulated a 2E-CVRP model and derived valid lower bounds using an exact algorithm. Santos et al. [207] addressed the same problem and proposed a set-partitioning reformulation as well as two branch-and-price algorithms. In their following work [208], the authors further introduced a branch-cut-and-price algorithm to tackle the problem.

A number of papers below have also investigated VRP variants and proposed various solution approaches. Grangier et al. [84] addressed a two-echelon multiple-trip routing problem with a time window, multi-trips and synchronisation constraints and proposed an ALNS metaheuristic to solve this problem. Li et al. [144] introduced a time-constrained 2E-VRP in line haul-delivery systems at a short-term tactical level and tackled the problem with a two-stage heuristic algorithm. More recently, Dellaert et al. [54] proposed two path-based mathematical formulations and respective branch-and-price–based algorithms for the 2E-VRP with time windows. Anderluh et al. [14] addressed 2E-CVRP with a focus on deliveries to the urban 'grey zone' between the inner-city centre and city outskirt. A metaheuristic with an embedded large neighbourhood search was proposed. Breunig et al. [30] studied the electric 2E-VRP and proposed an exact mathematical programming algorithm as well as a large neighbourhood search metaheuristic to solve the problem. Li et al. [145] studied the 2E-VRP with time windows and mobile satellites that combines unmanned aerial vehicles (UAVs) with the traditional human-driven vans for parcel deliveries. The authors developed a vehicle-flow formulation and proposed an ALNS heuristic. Mühlbauer and Fontaine [168] studied 2E-VRP with swap containers for cargo-bicycles and developed a parallelised large neighbourhood search heuristic. Kitjacharoenchai et al. [128] considered drone delivery in an 2E-VRP model for the last mile delivery. The authors presented a MIP model and two heuristic algorithms.

4.2.2 LRP

Since we consider both strategic network design of facility location and operational routing, our LoRD model mostly resembles a variant of the famous LRP, namely the Two-echelon LRP (2E-LRP). As suggested by Cattaruzza et al. [33], the 2E-VRP can be treated as a special case of 2E-LRP, where all the upper and lower facilities locations are given *a priori*. According to Lopes et al. [152], an LRP determines the number, size and location of facilities, the allocation of the demands to the facilities, and the

simultaneous design of the vehicle routes. A typical LRP integrates the strategic and operational decision levels since it minimises the fixed and operational costs incurred by opening facilities and planning vehicle routing. LRP is an NP-hard problem as it encompasses two NP-hard combinatorial optimisation problems, the Facility Location and the Vehicle Routing problem. We refer the readers to the existing LRP survey papers by Lopes et al. [152], who provided a taxonomy on structural characteristics and algorithmic approaches of LRP. Besides, we refer the readers to Drexl and Schneider [62], Mara et al. [158] and Prodhon and Prins [188] for reviews on LRP. Some early surveys can be found in the work of Min [165] as well as in Nagy and Salhi [171].

The popularity of LRP research has been increasing over the last decade. In the early years, Albareda-Sambola et al. [8], Laporte and Nobert [137], Laporte et al. [139], Liu and Lee [150], Melechovský et al. [164], Prins et al. [185], Tuzun and Burke [233], Wu et al. [246] provided studies on the LRP, and Lin and Lei [148] on the 2E-LRP. Recent studies focus more commonly on solving LRP with additional features such as depot and vehicle capacities, multi-echelon, multi-time period, and heterogeneous fleet, etc. To begin with, Baldacci et al. [16] presented an exact algorithm for solving the Capacitated LRP (CLRP) based on a set-partitioning formulation and provided the lower bounds using different bounding procedures. Belenguer et al. [20] presented a branch-and-cut algorithm for the CLRP and introduced a number of valid inequalities to strengthen their LP relaxation. Their work has been extended by Contardo et al. [43], who introduced three new arc-variable based formulations and provided several new families of valid inequalities to strengthen the LP relaxations. The same authors have also presented a follow-up work, including an exact cut-and-column generation algorithm and valid inequalities for the CLRP in Contardo et al. [42], and a three-phase heuristic to tackle the same problem in Contardo et al. [44], where their proposed algorithm employs Greedy Randomised Adaptive Search Procedure (GRASP) and heuristic column generation in the construction and improvement stages. Fazel Zarandi et al. [70] studied a Multi-depot CLRP with a time window, together with uncertainties in customer demand and travelling time. A fuzzy chance-constrained programming model and a simulated annealing algorithm were presented in their work. Yakıcı [250] studied LRP arises in the location and routing of small UAVs. An integer linear program formulation and an ant colony algorithm were proposed.

The following three paragraphs summarised the existing LRP literature with multi-echelon, multi-period, and clustering aspects, respectively.

Multi-echelon

The Multi-echelon CLRP (ME-CLRP) variant has attracted growing attention for better resembling real-life and has been studied in a number of papers. To begin with, a three-echelon location-allocation problem in the rubber industry was first studied by Nambiar et al. [173], and a three-echelon distribution network design problem in the food industry was later studied by Ambrosino and Grazia-Scutellà [11]. Both have proposed MIP formulations without applying any exact solver. The exact approach for the Two-echelon CLRP (2E-CLRP) was first introduced in Boccia et al. [27] in the context of city logistics. Contardo et al. [41] and Hemmelmayr et al. [101] both addressed 2E-CLRP and introduced a branch-and-cut algorithm and an ALNS metaheuristic for solving the problem. Nguyen et al. [176] studied the same problem and presented four constructive heuristics and a hybrid metaheuristic with an embedded GRASP complemented by a learning process and path relinking. Govindan et al. [83] developed a multi-objective 2E-LRP with a time window and sustainability requirements for per-

ishable food in supply chain network optimisation. The authors introduced a hybrid heuristic approach that combines multi-objective Particle Swarm Optimisation and ALNS to optimise the transportation quantity of products at each echelon. Winkenbach et al. [245] presented a large-scale static and deterministic MIP model for the 2E-CLRP and developed a closed-form approximation formula for the optimal routing cost as well as an optimisation heuristic. Vidović et al. [238] investigated the 2E-CLRP in non-hazardous recyclables collection and presented a MIP formulation with the reversed flow and a hierarchical location-first routing-second heuristic based on known second-level routes. Wu et al. [247] studied a three-echelon LRP with tight time windows and time budget constraints in the catering services for high-speed railways. The authors obtained lower bounds by relaxing the time-window constraints and introduced a hybrid cross-entropy algorithm for solving the problem. More recently, Dai et al. [48] studied the general multi-echelon LRP arising in city logistics and developed a two-phase method based on improved Clarke and Wright (CW) savings algorithm to deal with the problem.

Multi-period

A Multi-period LRP (MP-LRP) can be obtained by bringing a multi-period horizon to a classical LRP [188]. Alternatively, it can be treated as the Periodic VRP (PVRP) with an additional location aspect, with the goal to schedule a set of possible combinations of delivery days associated with each customer. The Periodic LRP (PLRP) was first introduced by Prodhon and Prins [187], who proposed an Evolutionary Memetic algorithm with population management. Later, Prodhon [186] studied the same problem and developed a hybrid evolutionary algorithm that operates through an evolutionary local search hybridized with a randomized extended Clarke and Wright heuristic to solve large-sized PLRP instances. Albareda-Sambola et al. [9] studied the MP-LRP and approximated the original routing component using spanning trees due to the high computation complexity related to the VRP. The authors proposed two heuristics and decoupled time scales to capture the different scopes between strategic location and operational routing decisions, which forces long-term location decisions to only be modified in certain periods of the planning horizon. Hemmelmayr [100] proposed sequential and parallel variants of an LNS algorithm to tackle the PLRP. In another work, Hemmelmayr et al. [99] studied the PLRP arising in collaborative recycling and proposed a MIP model for a new PLRP variant with various types of operational flexibility and an ALNS metaheuristic to solve the problem. Tunalıoğlu et al. [231] worked on the MP-LRP for the treatment of olive oil mill wastewater and tackled the problem with an ALNS metaheuristic. Koç [132] studied three variants of the PLRP with considerations of heterogeneous fleet and time windows. The authors designed a unified ALNS metaheuristic to solve the problem. More recently, Vahdani et al. [234] developed a multi-objective, multi-period and multi-commodity LRP model for distribution centres location and emergency roadway repair operations in damaged areas caused by earthquakes. Rabbani et al. [193] studied the multi-objective MP-LRP in the context of industrial hazardous waste management and proposed an integrated heuristic approach. Ben Mohamed et al. [22] studied the stochastic multi-period distribution network design problem with demand uncertainty and solved the proposed model using Benders' decomposition together with the sample average approximation approach.

LRP with a clustering aspect

Districting is the problem of clustering small geographic areas into larger geographic districts, such that the formed territories are balanced, contiguous, and compact [116]. According to Kovacs et al. [131], it is essential in territory design to guarantee multi-period driver workload balance, and customer satisfaction as a result of consistent service, which can be interpreted as the driver-customer service consistency as well as facility-customer supply consistency.

Many LRP papers contain a clustering aspect in the solution construction stage to determine an initial feasible solution concerning the capacities of their facilities and vehicles. After achieving an initial solution, further improvement techniques are applied to search for a near-optimal solution within a reasonable computation time. For clustering-based heuristic approaches, Ambrosino and Sciomachen [12], Ambrosino et al. [13], Chao et al. [34], Wang et al. [241], Liu et al. [151] applied facility-first vehicle-second clustering in their algorithms that first decompose the original problem into several facility-based VRPs, before minimising the number of vehicles regarding capacity limit. Fazel Zarandi et al. [70], Mehrjerdi and Nadizadeh [163], Rabbani et al. [192] applied a vehicle-first facility-second clustering approach in their algorithms that first forms driver-based clusters considering vehicle capacity before allocating them to the facilities. Teymourifar et al. [228] applied a sub-sectorization approach that breaks a whole tour into several sectors each manageable by an individual driver. Even though the clustering-based approach is one of the most frequently-applied heuristics approaches for solving LRPs [171], the clustering aspect is rarely studied as a model feature nor a determinant model component.

For papers that consider the customer clustering aspect in their model, Ambrosino et al. [13] studied a regional-LRP that considers partitioning customers into facility-based regions so that decisions are related to the regional clusters. Teymourifar et al. [228] formed depot-based sectors after allocating customers to facilities. However, none of the above work considered extending the facility and vehicle clusters into multiple periods. There is a gap in the territory design literature of forming practical facility-based and driver-based districts that provide good driver workload balance as well as consistency in customer-driver and customer-facility service assignment in throughout the planning horizon. The benefit of maintaining consistency between customers and drivers lies in fostering strong relationships, while ensuring that customer consistently corresponds to a facility enables effective traceability for freights. The work of Huang et al. [110] is the only paper we found that considered multi-period facility-based and driver-based districts that are part of the territory design methodology to accommodate daily variations in delivery. Nevertheless, their model does not include the facility location decision.

Moreover, most LRP papers consider distance and capacity as their clustering criteria. Some also specify customer demand [12], the waste type [192], or product preferences and geographical coordinates [241]. Only the work of Bender et al. [23] and Huang et al. [110] considered districting-based workload balance and service consistency for their clustering criteria. In our work, we consider workload balance, and service consistency for both driver-based and facility-based districts, as well as tactical-level adaptation for the districts. We scanned the relevant literature on (location-)routing problems with districting, and preferably multi-period aspects inside the models or solution approaches, and the comparisons are given in Table 4.1, from which we notice that no existing paper has included location, routing and districting decisions in a multi-period setting as in our work.

Table 4.1: Comparison of LRP-related papers with a clustering aspect

Papers	Decisions		Clustering Aspect			Model Features			
	Location	Routing	FT	DT	DA	Multi-echelon	Multi-period	Multi-commodity	Vehicle Type
Ambrosino and Sciomachen	*	*	*					*	S
Ambrosino et al.	*	*	*			*		*	M
Fazel Zarandi et al.	*	*	(*)	(*)					S
Mehrjerdi and Nadizadeh	*	*	(*)	(*)					S
Rabbani et al.	*	*	(*)	(*)				*	M
Huang et al.		*	*	*	*	*	*		S
Liu et al.	*	*	(*)			*			S
Wang et al.	*	*	(*)			*		*	S
Chao et al.	*	*	(*)			*			S
Bender et al.		*		*	*		*		M
Teymourifar et al.	*	*	(*)						S
Our work	*	*	*	*	*	*	*	*	S&M

(Clustering Aspect): \(*)" for those papers to indicate that clustering only appears as part of the algorithm but not as part of the model. **FT**=facility-based territories with respect to customer-supply consistency, **DT**=driver-based territories with respect to vehicle capacity, **DA**=district-based adaptations, **(Vehicle Type)**: **M**=multi-type (heterogeneous) vehicles, **S**=single-type (homogeneous) vehicles

4.2.3 Novelties

According to Nagy and Salhi [171], the location and routing decisions are interrelated decisions that cast influence on one other. However, despite the decisions' interdependency, the strategical location decision is valid for a more extended period than its routing counterpart, therefore casting a longer-lasting effect on the network design. According to Mara et al. [158], dealing with decisions of two different planning horizons has been an essential issue in LRPs to be explored. In our model, we try to include this challenging property by developing a two-stage decision approach as shown in Figure 4.2.

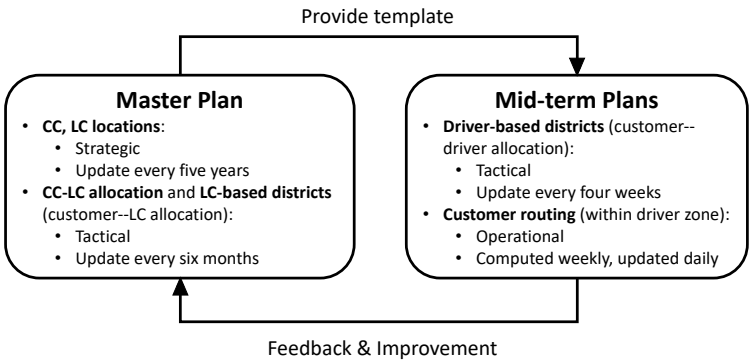


Figure 4.2: Master-Operational Planning Framework

We first consider the strategic level and work with a long-term plan (master plan) that is the basis for the tactical/operational level plannings (mid-term plans). The location of CCs and LCs, as well as the LC-based districts, are decided at a strategic level for a longer time length. Periodical adaptations of the driver districts and visiting routes are made to cope with workload imbalance due to customer demand variation.

The novelties of our work are summarised below.

- We study a new variant of the LRP model with multi-echelon, multi-period, multi-commodity, driver and facility-based districts, split replenishment tours in the first echelon and heterogeneous fleet delivery in the second echelon.
- We design a cluster-based heuristic approach and extend it to contain districting features, including workload balance amongst drivers and service consistency

within each LC-based and driver-based district.

- We propose an interdependent two-level planning approach in a multi-period context, using the master planning (location, facility-based districting) as a template for the tactical-level mid-term planning (driver-based districting, routing), which feedbacks backwards to the master plan.
- We aim at good solution quality in terms of minimal total costs, good workload balance and service consistency over time using the ALNS improvement heuristic.

4.3 Problem Statement

4.3.1 Notation

The whole problem can be formed using a graph, can be written as $G = (V, E, C)$, where V consists of all the nodes, including the potential locations for CCs as $H = \{1, \dots n_1\}$, LCs as $I = \{n_1 + 1, \dots, n_1 + n_2\}$, and customers as $J = \{n_1 + n_2 + 1, \dots, N\}$, where $n_1 + n_2 + n_3 = N$. We use $V^1 = H \cup I$ for all nodes in the first echelon reachable by a primary homogeneous vehicle, and $V^2 = I \cup J$ for all nodes in the second echelon reachable by a secondary heterogeneous vehicle. The notation E^1 represents all the arcs in the first echelon of the supply chain, including all arcs $\{h, i\}$ linking a CC to an LC and arcs $\{i_1, i_2\}$ linking two different LCs. E^2 represents all edges inside the second echelon, including all edges $\{i, j\}$ linking an LC to a customer and edges $\{j_1, j_2\}$ linking two customers. All edges in E^1 and E^2 are the shortest paths and each edge is associated with a traversal cost and time. The first-echelon vehicles can only visit the first echelon nodes V^1 and arcs E^1, and the same requirement applies to the second-echelon vehicles.

4.3.2 Model Assumptions

To denote the functional characteristic of tours at different levels, first-level replenishment connections between CCs and LCs are called *primary tours* or *replenishment tours*. Similarly, second-level city-centre delivery connections between LCs and Customers are referred to as *secondary tours* or *delivery tours*. Both first and second-level tours are required to initiate and terminate at the same facility. We prohibit the occurrence of cross-echelon deliveries to streamline the process at different levels of the supply chain. However, split deliveries are allowed in the first echelon, thereby enabling multi-source resource allocation with multiple visits to the same LC by tours departing from different CCs.

We assume that all customers have variability in their demands across all time periods and must be included in a tour and served during each time period within the planning horizon. The demand of any customer must be exclusively served by a single LC, with no allowance for multiple LCs engagements. Additionally, each customer should be visited by a single vehicle that performs either an individual tour or as part of a multi-stop round tour. Each driver is assigned to a single tour that commences and terminates at an LC.

We consider large-scale homogeneous transportation trucks for the primary replenishment tours and small-scale heterogeneous vehicles for the secondary delivery tours. Moreover, the utilisation of bulkheads allows concurrently loading multi-type freight items together onto any primary or secondary carrier. The cumulative weight of the transported load on each carrier cannot surpass its prescribed capacity bound.

In the context of transportation from each CC to LC and subsequently from each LC to customers, the cumulative amount of freight transferred from any upstream facility to all its downstream facilities should not exceed the capacity of the upstream facility. The secondary secondary delivery vehicles must cover at least one customer's demand so that customer split delivery does not exist.

4.4 MIP Formulation

4.4.1 Base MIP Model

In this section, we present a base MIP formulation for the LoRD model that incorporates two echelons, homogeneous primary trucks, heterogeneous secondary vehicles, a single period, a single commodity, and estimated customer demands obtained by averaging demand of individual customer over the planning horizon. We do not yet consider LC-based and driver-based districts and the multi-period adaptations. However, in the subsequent Section 4.4.2, we will elaborate on the extension of this base model to address multi-period and multi-product types for the distribution system. The list of sets, parameters, and decision variables with their descriptions can be found in Table 4.2.

In order to obtain feasible solutions for the a classical LRP model, Laporte et al. [139] suggested the necessity of three distinct types of constraints. The first type is referred to as the *degree constraint* or *flow conservation constraint*, which stresses the preservation of flow balance within the distribution network. Specifically, it ensures that the amount of freight arriving at a particular node is equal to the sum of freight leaving the node and the demand of that node. The second type is known as the Subtour Elimination Constraint (SEC), which serves the purpose of eliminating any illegal vehicle tour without the inclusion of a facility node. The third and final type is known as the *bar-chaining constraint* that eliminates the formation of any vehicle route that links one facility to a different facility of the same type. By imposing this constraint, we prohibit a vehicle to end at another facility that differs from where it originated (i.e. from CC_1 to CC_2, or from LC_1 to LC_2). This type of bar-chaining cut proposed by Laporte et al. [139] has been strengthened by Belenguer et al. [20] into the Path Elimination Constraint (PEC), which we will adapt for our model. Apart from the three mandatory types of feasibility constraints given above, we will also apply constraints for the depot capacity, depot status and tour type that are presented in [20].

In the domain of multi-echelon food distribution modelling, one of the pioneering works was put forth by Ambrosino and Grazia-Scutellà [11], who proposed a three-index formulation based on the waste collection VRP model presented by Perl and Daskin [183]. However, it has been observed that the efficiency of this model is limited due to the inclusion of an additional vehicle index within the decision variables. In more recent advancements, Belenguer et al. [20] has introduced a refined approach incorporating an undirected flow formulation to reduce the computational complexity, while Contardo et al. [42] presented a set partitioning formulation, both of which have been suggested as more efficient modelling alternatives by location science researchers (see [141, Chapter 15]).

Our proposed MIP model differs from the one proposed by Belenguer et al. [20] in several aspects. Firstly, we introduce a two-echelon LRP model, whereas their model is a single-echelon LRP. Second, we modify the arc traversal decision variable to incorporate a direction to facilitate the tracking of freight, meaning that u_{hi} does not equal to u_{ih} as they are of opposite directions. Lastly, we introduce an additional dimension

Table 4.2: Table of symbols for LoRD model

Symbol	Definition
(Sets)	
$J = \{j_1, \ldots, j_{n_2}\}$	Set of customers, $j \in J$
$I = \{i_1, \ldots i_{n_1}\}$	Set of potential Local Centres/Depots (LCs), $i \in I$
$H = \{h_1, \ldots, h_{n_3}\}$	Set of potential Consolidation Centres (CCs), $h \in H$
$K_{type} = \{1, 2 \ldots k_t\}$	Set of second-level vehicle types of different capacities
$L = \{1, 2 \ldots v_1\}$	Set of first-level vehicles, $l \in L$ and equipped with the same vehicle capacity
(Parameters)	
d_j	Demand of customer $j \in J$
W_h^C	Capacity of consolidation centre $h \in H$
W_i^L	Capacity of local depot $i \in I$
Q^1	Primary vehicle capacity, assuming a homogeneous fleet
Q_k^2	Secondary vehicle capacity, assuming a heterogeneous fleet of different vehicle types $k \in K$
M^1	Maximum primary vehicle number
M_k^2	Maximum secondary type-based vehicle number
M_{ik}^2	Maximum secondary type-based vehicle number at depot i
O_i	Opening cost for local depot $i \in I$
O_h	Opening cost for consolidation centre $h \in H$
F^1	Fixed hiring cost for primary vehicles in the first echelon
F_k^2	Fixed hiring cost for secondary vehicles in the second echelon
OP^C	Operational cost per unit throughput at the consolidation centre level. We assume all consolidation centres have the same operational cost
OP^L	The whole price for operational cost per unit of throughput at the local depot level. We assume all LCs have the same operational cost
C_{ij}	Total cost of transshipment between nodes i and j for an individual vehicle, $\forall i, j \in V$. Cross-echelon links are not included (e.g. consolidation centre-customer)
$D^2(S_J)$	Total demands for a group of selected customers $S_J \subseteq J$, so that $D(S_J) = \sum_{j \in S_J} d_j$
$D^1(S_I)$	Total "demand" for a group of selected depots $S_I \subseteq I$, so that $D(S_I) = \sum_{i \in S_I} d_i$, where d_i is the total demand of customers allocated to the depot i. This parameter is unknown until the allocations of customers to depots are known
$b_k^2(S_J) = \lceil \frac{D^2(S_J)}{Q_k^2} \rceil$	A lower bound of the minimum number of secondary vehicles needed to serve the aggregated demand of customers in set $S_J \subseteq J$. Here as the secondary vehicle capacity is dependent on vehicle type, the lower bound should also be type-dependent.
$b^1(S_I) = \lceil \frac{D^1(S_I)}{Q^1} \rceil$	A lower bound on the minimum number of primary vehicles needed to serve the aggregated demand of depots in the set $S_I \subseteq I$. From Nguyen et al. [176], this can be further approximated by $\frac{D^2(S_J) \cdot \sum_{j \in S_J} \sum_{i \in S_I} x_{ij}}{Q^1}$ without the ceiling function to avoid nonlinearity.
(Decision Variables)	
y_i	A binary variable equals 1 if and only if the selected local centre i is opened.
v_h	A binary variable equals 1 if and only if the selected consolidation centre h is opened.
z_{ijk}	A binary variable equals 1 if and only if a type k secondary vehicle traverses from node i to j, both of which must be from the second echelon, i.e. $i, j \in V^2$ or $\{i, j\} \in E^2$.
u_{hi}^l	A binary variable equals 1 if and only if a primary vehicle $l \in L$ traverses a first-echelon edge $\{h, i\} \in E^1$ from node h to i.
x_{ij}	A binary variable equals 1 if and only if customer $j \in J$ is allocated to depot $i \in I$, so that the total demand of this customer is fully satisfied by the assigned depot.
f_{hi}^l	A non-negative real number which tracks the freight flow from first-echelon node h to node i carried by a primary vehicle $l \in L$, where $\{h, i\} \in E^1$.

to the second-echelon arc traversal variable z_{ijk} to indicate if two nodes have been traversed consecutively by a specific second-level vehicle type or not. For the second echelon we keep only the vehicle type instead of individual vehicles and in the first echelon as a compromise due to the multi-sourcing requirement and a relatively small fleetsize, we adapt the vehicle index.

The MIP formulation for the base LoRD model is presented below:

$$\min_{x,z,u,v} \quad \sum_{\substack{i,j \in E^2, \\ i \neq j}} \sum_{k \in K} C_{ij} \cdot z_{ijk} + \sum_{\substack{h,i \in E^1, \\ h \neq i}} \sum_{l \in L} C_{hi} \cdot u_{hi}^l + \sum_{i \in I} \sum_{j \in J} \sum_{k \in K} F_k^2 \cdot z_{ijk} + \sum_{h \in H} \sum_{i \in I} F^1 \cdot u_{hi}^l$$
$$+ \sum_{i \in I} O_i y_i + \sum_{h \in H} O_h v_h + \sum_{i \in I} \sum_{j \in J} \left(OP^L \cdot d_j x_{ij} \right) + \sum_{h \in H} \sum_{i \in I} \left(OP^C \cdot f_{hi}^l \right) \tag{4.1a}$$

subject to

$$\sum_{j_1 \in I \cup J \setminus \{j_2\}} z_{j_1 j_2 k} = \sum_{j_1 \in I \cup J \setminus \{j_2\}} z_{j_2 j_1 k}, \qquad \forall j_2 \in J, \forall k \in K \tag{4.1b}$$

$$\sum_{\substack{j_1, j_2 \in S_J \\ j_1 < j_2}} \left(z_{j_1 j_2 k} + z_{j_2 j_1 k} \right) \leq |S_J| - b_k^2(S_J), \qquad \forall S_J \subseteq J, \forall k \in K \tag{4.1c}$$

$$\sum_{j_1 \in S_J \cup \{s,t\}} \sum_{\substack{j_2 \in S_J \cup \{s,t\} \\ j_2 < j_1}} (z_{j_1 j_2 k} + z_{j_2 j_1 k}) + \sum_{i \in I'} (z_{sik} + z_{isk}) + \sum_{i \in I \setminus I'} (z_{tik} + z_{itk}) \leq |S_J| + 2$$
$$\forall S_J \subseteq J, \forall I' \subseteq I, \forall s, t \in J \setminus S_J, \forall k \in K \tag{4.1d}$$

$$\sum_{k \in K} \left(z_{i_1 j k} + z_{j i_2 k} \right) \leq 1, \qquad \forall j \in J, \forall i_1, i_2 \in I, i_1 \neq i_2 \tag{4.1e}$$

$$\sum_{i \in I \cup J \setminus \{j\}} \sum_{k \in K} z_{ijk} = 1 \qquad \forall j \in J \tag{4.1f}$$

$$\sum_{\substack{i_1, i_2 \in S_I, \\ i_1 < i_2}} \left(u_{i_1 i_2}^l + u_{i_2 i_1}^l \right) \leq 2 b^1(S_I) \qquad \forall S_I \subseteq I, \forall l \in L \tag{4.1g}$$

$$\sum_{i_1 \in S_I \cup \{s,t\}} \sum_{\substack{i_2 \in S_I \cup \{s,t\} \\ i_2 < i_1}} (u_{i_1 i_2}^l + u_{i_2 i_1}^l) + \sum_{h \in H'} \left(u_{sh}^l + u_{hs}^l \right) + \sum_{h \in H \setminus H'} \left(u_{th}^l + u_{ht}^l \right) \leq |S_I| + 2$$
$$\forall S_I \subseteq I, \forall H' \subseteq H, \forall l \in L, \forall s, t \in I \setminus S_I, s \neq t \tag{4.1h}$$

$$u_{h_1 i}^l + u_{i h_2}^l \leq 1 \qquad \forall h_1, h_2 \in H, h_1 \neq h_2, \forall i \in I, \forall l \in L \tag{4.1i}$$

$$\sum_{t \in H \cup I \setminus \{s\}} u_{ts}^l = \sum_{t \in H \cup I \setminus \{s\}} u_{st}^l, \qquad \forall s \in I, \forall l \in L \tag{4.1j}$$

$$\sum_{i \in I} \sum_{l \in L} f_{hi}^l \leq W_h^C \qquad \forall h \in H \tag{4.1k}$$

$$\sum_{j \in J} d_j x_{ij} \leq \sum_{s \in H \cup I \setminus \{i\}} \sum_{l \in L} (f_{si}^l - f_{is}^l) \leq W_i^L \qquad \forall i \in I \tag{4.1l}$$

$$u_{hi}^l \leq v_h \qquad \forall h \in H, \forall i \in I, \forall l \in L \tag{4.1m}$$

$$\sum_{l \in L} f_{hi}^l \leq Q^1 \sum_{l \in L} u_{st}^l \qquad \forall h, i \in H \cup I, h \neq i \tag{4.1n}$$

$$x_{i,j_2} + 1 \geq x_{i,j_1} + \sum_{k \in K} (z_{j_1 j_2 k} + z_{j_2 j_1 k}) \qquad \forall i \in I, \forall j_1, j_2 \in J, j_1 < j_2 \tag{4.1o}$$

$$\sum_{k \in K} z_{ijk} \leq x_{ij} \qquad \forall i \in I, j \in J \tag{4.1p}$$

$$\sum_{k \in K} z_{jik} \leq x_{ij} \qquad \forall i \in I, j \in J \tag{4.1q}$$

$$\sum_{i \in I} x_{ij} = 1 \qquad \forall j \in J \tag{4.1r}$$

$$x_{ij} \leq y_i \qquad \forall i \in I, \forall j \in J \tag{4.1s}$$

$$y_i, v_h, x_{ij}, z_{ijk}, u_{ij}^l \in \{0,1\}, f_{st}^l \in R^+ \qquad \forall i \in I, j \in J, h \in H, k \in K \tag{4.1t}$$

The objective function (4.1a) minimises the sum of traversal and fixed hiring costs of the secondary and primary vehicles, and the opening costs and operational costs from the CCs and LCs. The first two terms are the secondary and primary vehicles traversal costs. The third and fourth terms are the secondary and primary vehicles fixed hiring costs. The fifth is the opening costs of all LCs with at least one customer

allocated. The sixth is the opening costs of all CCs having nonzero freight outflow. The seventh term is the total operational costs at the LC-level related to the amount of freight flow going through all the LCs. Here we use the outflow to compute the LC-level operational costs but we can also use inbound flow to compute the operational cost, by $\sum_{i\in I}\sum_{s\in E^1,s\neq i}\sum_{l\in L}(f^l_{si}-f^1_{is})$. The last term is the total operational costs at the CC level.

Constraints (4.1b), (4.1c) and (4.1d) are the three mandatory constraints suggested in the work of Belenguer et al. [20]. Specifically, (4.1b) are the secondary degree constraints stating that flow conservation must hold for each customer node as "what goes in must go out". (4.1c) are the secondary subtour elimination and vehicle capacity constraints, which indicate that the total set of traversed edges of a subgroup of customers S_J, where $1 < |S_J| \leq |J|$, cannot form a subtour, i.e. without having the linkage to an LC. Here $b^2_k(S_J) = \lceil \frac{D^2(S_J)}{Q^2_k} \rceil$ indicates the minimum number of secondary vehicles of a particular type required in order to deliver to a subgroup of customers S_J, where $D^2(S_J)$ is the aggregated demand for the subgroup. Secondary path elimination constraints (4.1d) forbid any illegal chain linking two different LCs. The left-hand side of the inequality is the sum of the number of edges that are linking any two customers inside group $S_J \cup \{s,t\}$, one customer s to any depot i, and the other customer t to any other depot i'. The constraints require that the sum cannot exceed the number of selected customers plus two, and therefore always has one edge less to form an illegal tour. Since (4.1d) only consider illegal paths consisting of at least two customers, we include constraints (4.1e) to specifically forbid a single customer from chaining two LCs. Since any direct arc linking two LCs is not defined in our model, no path elimination constraint is required for this case. Constraints (4.1f) state that each customer must be served. Together with (4.1b), we derive that each customer node should be visited by exactly one secondary vehicle of a particular type.

Since we allow split deliveries from multiple CCs to a single LC using multiple primary vehicles, the original subtour elimination constraints by Belenguer et al. [20] are no longer valid for our first echelon model. Therefore, we adapt the subtour elimination and vehicle capacity constraints proposed by Nguyen et al. [176] with the estimated transition freight amount. Moreover, we index each primary vehicle to allow one LC to be visited by multiple primary vehicles sent from different CCs, since it would be challenging to track freight without identifying individual vehicles. Constraints (4.1g) are the first level subtour elimination constraints. According to Nguyen et al. [176], $b^1(S_I)$ can be estimated by $\frac{\sum_{i\in S_i}\sum_{j\in J} d_j x_{ij}}{Q^1}$. Similar to the secondary path elimination constraints (4.1d), constraints (4.1h) forbid an illegal path formed by a primary vehicle replenishing two or more LCs and returning to a CC different from the one where it started. Constraints (4.1i) expressly forbid one LC chaining two CCs together. The first level degree constraints (4.1j) force flow conservation such that any primary vehicle l visiting a depot must leave from the node afterwards.

The CC capacity constraints (4.1k) require that the sum of the total freight departing from any CC cannot exceed its physical capacity. The functionalities of (4.1l) are two-fold: They limit each LC's throughput capacity such that the total amount of first level freight distributed to this LC cannot exceed its physical capacity. The constraints also give an upper bound on the total customer demands allocated to this LC. Constraints (4.1m) require that a CC must be opened if a primary vehicle l departs from or returns to it. Constraints (4.1n) force that any primary arc with nonzero freight flow must be traversed by a primary vehicle. Constraints (4.1o), (4.1p) and (4.1q) create linkages between arc traversal and customer allocation variables, requiring that each

starting customer must be allocated to an LC. Constraints (4.1r) state that every customer must be allocated to an LC. Constraints (4.1s) state that customers can only be allocated to an open LC. Lastly, constraints (4.1t) give the decision variables domains.

4.4.2 Model Extension

Based on the single-period, single-commodity MIP formulation given in (4.1), the following extension can be further applied with additional variables indices and constraints.

Multi-Period

If we extend the current formulation into the multi-period setting, we need to add an additional index (dimension) t to every decision variable to indicate the period this decision was made. If we define a single period to be each day, the customer allocation variables x_{ijt} and the freight tracking variables f^l_{hit} related to facility-based districting are tactical, meaning that they can only be updated every few numbers of periods ($t \in T^{\alpha_1} = \{\alpha_1 t \mid \alpha_1 \in \mathbb{Z}^+, t \in T\}$). The arc traversal variables z_{ijkt} are related to the driver districts that are also tactical, and therefore are changeable at every few number of periods ($t \in T^{\alpha_2} = \{\alpha_2 t \mid \alpha_2 \in \mathbb{Z}^+, t \in T\}$). We should define α_1 to be greater than α_2 to showcase the longer service consistency between customers and facilities versus customers and drivers. The location decisions y_{it} and v_{ht} are strategic that should be determined for a much more extended scale. In our work, this is done in a similar fashion as the authors did in Albareda-Sambola et al. [9].

Facility-based and Driver-based Districting

In the multi-period setting, we are able to include the facility-based and driver-based districting for our model. The facility-based district can be considered as the set of customers $\{j \mid x^t_{ij} = 1\}$ for any LC with index i in time period t. For the driver-based districts, we need to identify the individual routes. To achieve that, we let K^* to be the set of drivers of the secondary vehicles and re-define the arc traversal binary variable z^t_{ijk} to be one if an arc $\{i, j\}$ is traversed by an individual driver (rather than a particular secondary vehicle type as before) in period t for $k \in K^*$. We then define the time-expanded binary decision variable $e^t_{j,k}$ that equals 1 if customer j is assigned to district served by driver $k \in K^*$ in time period t, and zero otherwise. Moreover, let $d^t_j \in R^+$ be the freight demand (ambient and frozen/chilled) associated with customer j for period t. Each driver district can be treated as the set of customers from the same route departing and terminating at a particular LC. As each driver $k \in K^*$ is individually equipped with a secondary vehicle of a specific capacity Q^2_k, each district's total allocated demand should not exceed the respective vehicle capacity. Besides, we define $s_{\min}$ and $s_{\max}$ as the minimum and maximum percentage of occupied vehicle capacity in any driver district, respectively.

Several districting features are discussed in this section. First of all, each district's compactness can be optimised by minimising the distance between all customers from the same district. This coincides with the multi-period version of the first objective function term in (4.1a) that minimises the total distance between all customers from explicit routes. For each driver district, we also consider workload balance and service consistency by including the following constraints to the multi-period version of our MIP formulation (4.1).

$$2 \cdot z^t_{j_1,j_2,k} \leq e^t_{j_1,k} + e^t_{j_2,k} \qquad \forall j_1, j_2 \in J, j_1 \neq j_2, \forall k \in K^*, \forall t \in T \quad (4.2a)$$

$$\sum_{j \in J} d^t_j \; e^t_{j,k} \geq \frac{s_{\min}}{100} Q^2_k \qquad \forall k \in K^*, \forall t \in T \quad (4.2b)$$

$$\sum_{j \in J} d^t_j \; e^t_{j,k} \leq Q^2_k \qquad \forall k \in K^*, \forall t \in T \quad (4.2c)$$

$$\sum_{j \in J} \sum_{k \in K^*} \left| e^t_{j,k} - e^{t+1}_{j,k} \right| \leq 2 \cdot \frac{s^1_{\text{change}}}{100} \cdot |J| \qquad \forall t \in \{1, ..., |T| - 1\} \quad (4.2d)$$

$$\sum_{i \in I} \sum_{j \in J} \left| x^t_{i,j} - x^{t+1}_{i,j} \right| \leq 2 \cdot \frac{s^2_{\text{change}}}{100} \cdot |J| \qquad \forall t \in \{1, ..., |T| - 1\} \quad (4.2e)$$

$$e^t_{j,k} \in \{0, 1\} \qquad \forall j \in J, \forall k \in K^*, \forall t \in T \quad (4.2f)$$

Since we have the explicit traversal sequence of individual routes (hence the driver-based districts) from our base model (4.1a)-(4.1t), we can already guarantee all customers to be compact without using the compactness constraints from the districting model proposed by Hess et al. [103]. Instead, we have constraints (4.2a) to create a linkage between the secondary arc traversal and customer-to-driver-district allocation, forcing any pair of directly traversed customers to be allocated to the same driver district, but not the other way around since two non-consecutive customers can still belong to the same route.

To satisfy the workload balance among drivers under the same vehicle type, we have constraints (4.2b) to define the minimal vehicle load for any driver district, and constraints (4.2c) to yield that the vehicle capacity cannot be violated. In this way, we force any existing tour with a total workload smaller or larger than the vehicle capacity to be invalid.

To maintain the service consistency between any consecutive periods, we have (4.2d) for the driver-customer allocation consistency, and (4.2e) for the facility-customer allocation consistency. Constraints (4.2d) enforce that at most $s^1_{\text{change}}\%$ of customers can be allocated to another driver-based district in any different period, and (4.2e) enforce that at most $s^2_{\text{change}}\%$ of customers can be allocated to another facility-based district in any different period. Finally, (4.2f) state the binary decision variable.

Multi-Commodity

For the multi-commodities feature, we need to specify the secondary vehicles to have different weights and different proportions of cooling volumes $\left[\omega^k_{min}, \omega^k_{max}\right]$, for a certain vehicle k. We therefore define a new decision variable ω^k as the cooling area fraction for the secondary vehicle indexed k. For the MIP formulation (4.1), we need to change the secondary vehicle capacity constraints (4.1c), specifically, the computation of $b^2_k(S_J)$, to make sure every vehicle k has to satisfy the ambient and cooling capacity demands, respectively. We also need to adapt the first-echelon constraints (4.1g), (4.1k) and (4.1l) to include this multi-commodities aspect.

4.5 Exact Solution Method

While solving a smaller-scaled MIP model with an exact solver may be computationally manageable, it becomes impractical to find feasible solutions for large or moderate instances. This challenge mainly arises from four sets of exponential-size constraints, namely the subtour elimination constraints for both first echelon (4.1c) and second echelon (4.1g), and the path elimination constraints for first echelon (4.1d) and second echelon (4.1h). Moreover, in order to construct facility-based and driver-based districts, it becomes necessary to track individual vehicle indices. We therefore need to shift the secondary vehicle index k from vehicle types to individual drivers, leading to a significant increase in computational complexity for our multi-period, multi-commodity MIP model with facility and driver districts.

Consequently, we propose two acceleration approaches, namely a Constraint Relaxation approach introduced in Section 4.5.1 and the root node approach given in Section 3.4.2 to increase the efficiency of the LoRD model. Nevertheless, we only employ the exact method as a benchmarking tool to generate optimal solutions for the performance evaluation of our proposed heuristic in Section 4.6.

4.5.1 Constraint Relaxation Approach

To address the exponential-size constraints within our MIP model, we adopt the Constraint Relaxation approach (CRA) proposed in Laporte and Nobert [137] that aims to exclude the whole set of exponential-size constraints and iteratively reintroduce the violated ones in a cutting-plane fashion until feasibility occurs. In this approach, we first solve a relaxation X_s of the original problem without the inclusion of the subtour elimination constraints (4.1c) and (4.1g) and the path elimination constraints (4.1d) and (4.1h) for both echelons, which results in a fractional initial solution that may not be optimal or feasible. We recursively identify any original subtour elimination or path elimination constraints the incumbent violates and reintroduce them to the MIP model to cut off the fractional solution from the feasible region. This process is continued until no illegal subtour or path exists in the solution and all the vehicle capacity restrictions are satisfied.

We execute the following two algorithms for each of the above iterations to detect any violated constraint. The first function presented in Algorithm 5 focuses on identifying and reintroducing any violated subtour elimination constraint. This function operates by enumerating through any given vehicle type k and each customer node j that the algorithm has not previously reached and recorded. Each valid tour can be treated as a sequence of nodes that begins and terminates with a depot. For each j', we branch downwards and record its successors until encountering either an LC, indicating the discovery of part of a valid tour is found, or the node j itself, meaning that an illegal subtour is detected. During this exploration, the vehicle capacity for type k is continuously verified to ensure that any subset of nodes with total demands exceeding the capacity restriction will result in a vehicle capacity violation and hence a corresponding cut.

The second function outlined in Algorithm 6 detects and reintroduces the violated path elimination constraints. In this function, each vehicle type k and LC is examined to determine if the facility has been traversed in the solution and remains open. We then branch downwards from each opened LC to find the successors until returning to the original LC or reaching another LC. In the latter case, we identify an illegal path chaining two different facilities and introduce a corresponding path elimination

constraint to remove this path.

Empirical findings reveal that the first-echelon subtour elimination constraints (4.1g) and path elimination constraints (4.1h), which entail relatively small-sized nodes and arcs, have a negligible effect on the computational efficiency of the MIP formulation. As a result, we exclusively employ the CRA for the second-echelon customer-based constraints (4.1c) and (4.1d). These constraints govern the vehicle capacity and flow of goods within the second echelon, which reveal a more complex level of the network involving a larger number of customer nodes and arcs.

Algorithm 5 CRA: Subtour Elimination

```
let SubtourExist = False, VehCapExceed = False
for k in VehicleType do
    define N as an empty set
    for j in Customer do
        if j is in N then
            continue
        let Branching = True, StartNode = j
        define Q as an empty set
        while Branching = True do
            add StartNode to set Q
            find i the successor of StartNode
            if i appears in set Q or i is a facility then
                Branching = False
            let StartNode = i
        if total demand in Q ≥ vehicle type k capacity then
            add subtour elimination constraint (Q, k)
            VehCapExceed = True
            continue
        if StartNode is not a facility then
            if set Q contains more than 1 node then
                add subtour elimination constraint (Q, k)
                SubtourExist = True
    add all nodes in set Q to set N
```

Algorithm 6 CRA: Path Elimination

```
PathExist = False
for k in VehicleType do
    for i in Facility do
        let Branching = True, StartNode = i
        define Q as an empty set
        while Branching=True do
            add StartNode to set Q
            if successor of StartNode exists then
                name the successor node j
                if j is a facility then
                    Branching = False
                let StartNode = j
            else
                break
        if StartNode ≠ i then
            add path elimination constraint (i, StartNode, Q, k)
            PathExist = True
        else
            PathExist = False
            next
```

4.5.2 Bounding the number of districts

This section presents the upper and lower bounds of a feasible number of secondary vehicles to hire, or equivalently, a feasible number of driver-based districts to form. Since we consider heterogeneous vehicles with different capacities and proportions of space for ambient and frozen/chilled products, we develop the following equations to

derive the maximum and minimum number of hired secondary vehicles, taking both commodity types into account.

We assume that only frozen/chilled commodities require additional equipment, i.e. remote fridge, during the delivery process. Therefore, frozen/chilled food can only take up a certain proportion of a truck due to technical restrictions. For notation simplicity, a type k vehicle with capacity Q_k can allocate a positive percentage between $\left[\omega^k_{min}, \omega^k_{max}\right]$ for the cooling area, where $0 \leq \omega^k_{min} \leq \omega^k_{max} \leq 1$. Here w^k_{max} can be treated as the maximum capacity for cooling area but the volume can be filled with some ambient food. The remaining proportion of space is only available for ambient food, which is between $\left[1-\omega^k_{max}, 1-\omega^k_{min}\right]$. The customer demand of ambient and frozen/chilled food involved in the following models are the expected values averaged over all periods represented by $\hat{d}^A_j$ and $\hat{d}^F_j$ for each customer, where $\hat{d}^A_j + \hat{d}^F_j = \hat{d}_j$. We can find the upper and lower bounds on the number of secondary vehicles required to satisfy the aggregated demand from all customers by solving the following equation:

$$UB = \max\left\{\frac{\sum_j \hat{d}^A_j}{\min_k Q^k\left(1-\omega^k_{max}\right)}, \frac{\sum_j \hat{d}^F_j}{\min_k Q^k \omega^k_{min}}\right\} \tag{4.3a}$$

$$LB = \max\left\{\frac{\sum_j \hat{d}^A_j}{\max_k Q^k\left(1-\omega^k_{min}\right)}, \frac{\sum_j \hat{d}^F_j}{\max_k Q^k \omega^k_{max}}\right\} \tag{4.3b}$$

where UB is the maximal number of trucks required to satisfy all customer demands for either type of commodity, given that only the smallest volume truck is hired. Likewise, LB is the minimal number of trucks required to distribute the orders if every customer is visited by the maximum volume truck. Equation (4.3b) guarantees that the selected fleetsize is always feasible without exceeding the capacity. To find a good estimation for the initial value of the fleetsize m during the solution construction process, we first limit the fleet size value between the upper and lower bounds given in Section 4.5.2. We then apply the root-node solution method introduced in Section 3.4.2 to identify the best number of secondary vehicles for our initial solution. To implement the root-node approach, we iterate over a valid range of hired vehicles, encompassing the upper and lower bounds derived earlier. We pre-define the fleet size m and introduce the following constraint to the MIP model (4.1):

$$\sum_{i\in I, j\in J, k\in K} z_{ijk} = m \tag{4.4}$$

We also remove the third term of the objective function (4.1a) so that the solver is no longer required to optimise the second-echelon fleetsize but treats it as an input parameter.

4.6 Heuristic Algorithm

This section focuses on the multiple-period LoRD model and proposes a two-stage heuristic for the location, allocation, and routing decisions. We introduce the driver-based and facility-based districting as extensions of the current allocation decisions in order to include all the districting-related characteristics, such as the equitable size and the geographical compactness of each district. We aim to form compact, round-shaped and undistorted driver and LC territories, which are beneficial to avoid unnecessary

travels across the city, making it easier for neighbourhood-based deliveries and rebalancing workload among drivers. Moreover, compact driver and facility districts form the basis for facilitating the maintenance of both good drivers' workload balance and service consistency, resulting in a good customer satisfaction rate (Kovacs et al. [131]).

The two-stage heuristic begins with a long-term master template which contains the long-term decisions. The mid-term tactical plan contains the tactical decisions, which can be updated based on weekly/monthly operation results. We require the decisions regarding facility location, driver and facility-based districting to be made only in certain selected time periods of the planning horizon, and the decisions cannot be modified in between those periods. The solution procedure to derive the Master template is given below, with the graphical description of the Master stage depicted in Figure 4.3.

- (step 0) **Estimate customer demand**: relax the multiple commodities and secondary vehicle types requirements, compute the estimated average demand for each customer over a number of time periods.
- (step 1) **Form driver-based districts**: use the root node approach introduced in Section 3.4.2 to select the best number of customer districts such that no district is overloaded. Solve a districting problem to form driver-based districts. Designate a central customer within each cluster as a representative node responsible for carrying the total demand of the associated driver district.
- (step 2) **Form LC-based districts**: solve a relaxed version of the MIP formulation (4.1) using the representative nodes so that the driver districts are closely packed around the LC. Form the LC districts considering capacitated LCs and uncapacitated secondary vehicles.
- (step 3) **Solve TSP within each driver district**: solve a TSP within each driver district to form a single tour with the allocated LC node included. Reintroduce commodity types (ambient, frozen/chilled) and secondary vehicles capacities. Identify any infeasible driver-based tours with exceeded capacity for either ambient or frozen/chilled carriage. Choose the smallest secondary vehicle capable of serving each group.
- (step 4) **ALNS improvement of the initial solution** minimising:
  1. the number of first and second echelon vehicles (hiring costs)
  2. the total traversal costs for all vehicles
  3. the secondary vehicle capacity violation for ambient or frozen/chilled sector
  4. the workload imbalance among all drivers (penalty for workload outside a certain geographic region). This means that the further away a district is from its assigned LC, the fewer customers it should contain on average, due to the longer stem distance between the LC and the customers.

  In this step, after finalising the second-echelon decisions during the ALNS improvement, we can then optimise the the first-echelon decisions for facility location and routing by solving a MIP formulation. Step 4 can be also applied to derive the tactical weekly/monthly variations using the master template as the initial solution.

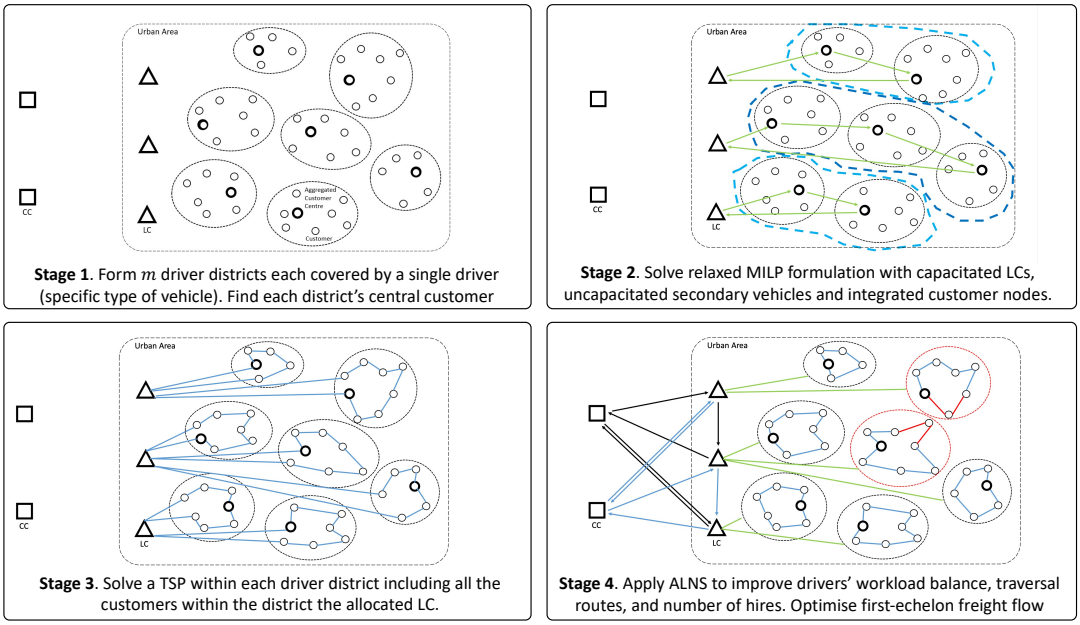


Figure 4.3: Illustrating master-level decisions involving the formulation of plans encompassing location, routing, and decisions related to customers and facility districts.

The following Section 4.6.1, Section 4.6.2, Section 4.6.3 and Section 4.6.4 describe the four sequential stages of the master-level planning in details.

4.6.1 Forming driver-based districts

At the beginning of the master planning, we apply a districting model to form an initial set of driver territories in the second echelon. To quickly obtain an initial solution while considering the *NP-hard* nature of the districting problem, we chose to address a simplified version by relaxing the multi-commodity and heterogeneous secondary vehicles restrictions. Furthermore, we opted to assign all the secondary vehicles to have the largest capacity, as adjustments to the number and type for vehicles can be made in subsequent steps.

The motivation for addressing such an *NP-hard* districting problem stems from the need to optimise three interrelated factors: (1) the capacity requirements for each LC, (2) the geographical compactness of driver-based districts that are assigned to the same LC, and (3) the shortest distance between each driver-based district and its designated LC. While a simplistic approach, such as a simple greedy allocation or k-means algorithm may address one or two of these factors, it is incapable of balancing the interplay between all three, as demonstrated in Figure 4.4. Specifically, we seek to minimise the inter-travel distance between driver districts and the distance from LC to a driver district while taking into account the facility capacity constraints.

Mathematically, we first aggregate customers into m compact and balanced districts that are each manageable by an individual driver, before developing explicit routes within each district at a later stage. A feasible fleetsize m can be pre-determined using the best root node solution within the upper and lower bound, as in Section 3.4.2.

We use the same districting formulation as in Section 3.7.3 using notations under the LoRD setting. Let J be the set of customers, and I be the set of LCs as before. Let $\omega_j \in R^+$ be the aggregated freight demand (ambient and frozen/chilled) associated with customer j. The average aggregated freight demand per district is defined as

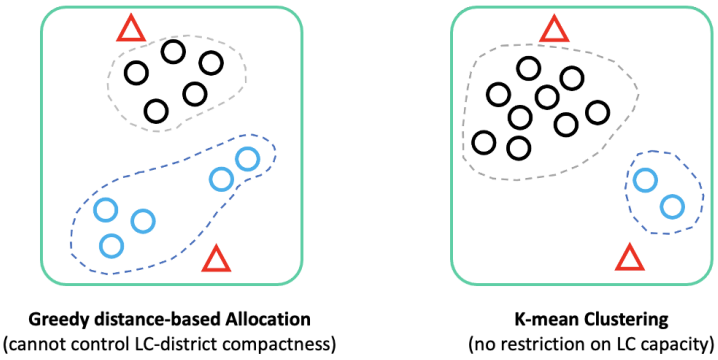


Figure 4.4: The drawbacks for greedy allocation and k-mean algorithms

$\mu = \frac{1}{m}\sum_{j\in J}\omega_j$. We denote $s_{\min}$ and $s_{\max}$ as the minimum and maximum percentage of freight demand in any district, respectively. Finally, we set the decision variable e_{j,j^*} to be one if customer j is assigned to the district centred at customer j^*, and it is zero otherwise. Here e_{j^*,j^*} takes the value of one if customer j^* is selected as the district centre. The compactness can be optimised by minimising the distance from customers to district centre. The districting MIP model can be defined as below:

$$\min \sum_{j\in J}\sum_{j^*\in J} \omega_j \; c^2_{j,j^*} \; e_{j,j^*} \tag{4.5a}$$

$$\sum_{j^*\in J} e_{j,j^*} = 1 \qquad \forall j \in J \tag{4.5b}$$

$$\sum_{j\in J} e_{j,j} = m \tag{4.5c}$$

$$e_{j,j^*} \leq e_{j^*,j^*} \qquad \forall j^* \in J \tag{4.5d}$$

$$\sum_{j\in J} \omega_j \; e_{j,j^*} \geq \frac{s_{\min}}{100}\mu \cdot e_{j^*,j^*} \qquad \forall j^* \in J \tag{4.5e}$$

$$\sum_{j\in J} \omega_j \; e_{j,j^*} \leq \frac{s_{\max}}{100}\mu \cdot e_{j^*,j^*} \qquad \forall j^* \in J \tag{4.5f}$$

$$e_{j,j^*} \in \{0,1\} \qquad \forall j, j^* \in J \tag{4.5g}$$

Constraints (4.5b) require every customer to be assigned to a driver district. Constraint (4.5c) requires exactly m driver districts to be formed. Constraints (4.5d) state that each formed driver district must have a central customer. Constraints (4.5e) and (4.5f) together define the minimal and maximal workload of any driver district. Finally, constraints (4.5g) state the binary decision variable. Since the size of our current instance is rather small, we can ignore the constraints that help to form compact districts to preserve continuity for large-sized instances. Different contiguity criterion as part of a mathematical programming model can be found in Laporte et al. [141, Chapter 23].

4.6.2 Forming LC-based districts

In the second stage of master planning, we determine the secondary facility location and customer-facility allocation decisions. The decisions that appear in the upper echelon will be considered at a later stage.

To reduce the problem size, we aggregate customers within a driver district into an integrated node, which carries the total demand of the group. We use the central customers $\{j^* \mid e_{jj*} = 1, j, j^* \in J\}$ formed in the previous stage as the integrated nodes, each representing a driver district. Afterwards, we solve the following simplified MIP model (4.6) using the CRA given in Section 4.5.1. The solution contains the LC opening status and the traversed arcs. Each LC-based district can be treated as the set of central customers linked to the same LC, together with the remaining customers from the corresponding driver districts. In this way, we assign driver districts to each LC and form facility districts. At this stage, we relax the secondary vehicle capacity requirement since all tours are auxiliary using integrated customer nodes in order to determine the facility districts.

We solve the simplified MIP model below with only the second-echelon-related constraints, preserving the LC capacity while relaxing the vehicle capacity requirement. We update constraints (4.1c) by (4.6b) for the relaxation of secondary vehicles capacity $Q^2 = \max\{Q_k^2\}$, and update (4.1l) by (4.6c) to keep the LC capacity.

$$\begin{aligned} min \sum_{\substack{i,j \in E^2, \\ i \neq j}} \sum_{k \in K} C_{ij} \cdot z_{ijk} + \sum_{i \in I} \sum_{j \in J} \sum_{k \in K} F_k^2 \cdot z_{ijk} \\ + \sum_{i \in I} O_i y_i + \sum_{i \in I} \sum_{j \in J} \left(OP^L \cdot d_j x_{ij} \right) \end{aligned} \tag{4.6a}$$

subject to

$$(4.1b), (4.1d), (4.1e), (4.1f), (4.1o), (4.1p), (4.1q), (4.1r), (4.1s)$$

$$\sum_{\substack{j_1, j_2 \in S_J, \\ j_1 < j_2}} \left(z_{j_1 j_2 k} + z_{j_2 j_1 k} \right) \leq |S_J| - 1 \qquad \forall S_J \subseteq J, \forall k \in K \tag{4.6b}$$

$$\sum_{j \in J} d_j x_{ij} \leq W_i^L \qquad \forall i \in I \tag{4.6c}$$

$$y_i, x_{ij}, z_{ijk} \in \{0, 1\} \qquad \forall i \in I, j \in J, h \in H, k \in K \tag{4.6d}$$

The generated solution provides the list of LCs that are opened together with their assigned central customers, which are the integrated nodes corresponding to the driver districts. By the assignment of the driver districts to the facilities, we can derive the facility-based districts.

4.6.3 Routing within each driver district

After clustering the customers into driver-based and facility-based regions, we form a single cycle inside each driver district to link all the included customers as well as the LC node this district is allocated to. Since we only care about the minimal route length at this stage and not the vehicle capacity, the carrying load nor the freight type, this is equivalent to solving the homogeneous TSP for m times, where m is the number of driver districts. We apply the same TSP formulation in Section 3.7.3 using notations under the LoRD setting.

$$\min \sum_{j_1 \in J'} \sum_{\substack{j_2 \in J', \\ j_1 \neq j_2}} C_{j_1, j_2} \, z_{j_1, j_2} \tag{4.7a}$$

$$\sum_{j_1 \in J'} z_{j_1,j_2} = 1 \qquad \forall j_2 \in J', j_1 \neq j_2 \tag{4.7b}$$

$$\sum_{j_2 \in J'} z_{j_1,j_2} = 1 \qquad \forall j_1 \in J', j_1 \neq j_2 \tag{4.7c}$$

$$\sum_{j_1 \in Q} \sum_{\substack{j_2 \in Q, \\ j_1 \neq j_2}} z_{j_1,j_2} \leq |Q| - 1 \qquad Q \subsetneq J', |Q| \geq 2, \tag{4.7d}$$

where z_{j_1,j_2} is a simplified version of the binary lower-echelon traversal variable indicating that arc (j_1, j_2) is visited by a secondary vehicle with homogeneous capacity. Constraints (4.7b) and (4.7c) guarantee the flow conservation that each customer node should has exactly one vehicle arriving and leaving. Constraints (4.7d) eliminate subcycles containing only a subset of nodes if there are more than two customers inside the group. Taking into account the size of our problem, an exact solution can be obtained using existing solvers.

4.6.4 ALNS improvement metaheuristic

In the previous sections, we have presented the cluster-first-route-second strategy to construct an initial solution for the second-echelon routes. To further enhance the solution quality and balance the workload of the drivers on both strategical and tactical levels, we apply the ALNS metaheuristic in Section 2.4 to improve the facility-based and driver-based districts. To initiate this improvement process, we reintroduce the consideration of a heterogeneous fleet and the different types of commodity demands, which were excluded during the initial route construction. We assign the smallest secondary vehicle capable of serving the estimated district-based customer demands to each driver district. This step ensures that we allocate the smallest feasible vehicle capacity for any type of food while preventing any potential issues related to capacity exceeding. After the improvement heuristic, we then generate the upper echelon freight flow and CC location decisions.

The reason behind the implementation of this metaheuristic is due to its renowned performance observed on various complicated VRP variants with large instance sizes in the literature [201, 101]. The list of ALNS parameter values is given in Table 4.3 based on our computational experimentation and existing values from the literature.

Table 4.3: ALNS Parameter values summary

Symbols	**Parameter Names**	**Values**
q	destroy scale	10%
λ	local search threshold	20%
δ_1	global best solution score	33
δ_2	local best solution score	31
δ_3	accepted worse solution score	19
l	weight segment length	50
α_{RW}	reaction factor	0.5
c_τ	temperature cooling coefficient	0.99
T	initial temperature	60
$iter$	non-changing iteration stop criteria	500

Destroy and Repair Operators.

The ALNS algorithm removes a pre-defined number of nodes from the solution together with their linking arcs before adding them back iteratively, with the hope that the newly formed solution yields a smaller objective value. The removed q customer nodes are temporarily placed in a "customer pool". We introduce the whole list of destroy operators below:

1. *Random Removal:* a group of q randomly selected customers are removed from their existing routes and placed inside the customer pool.
2. *Worse Removal:* removes the q customers with the highest removal gain, which is the difference in cost when this customer is inside an allocated tour, and when the customer is not.
3. *Related Removal:* a single customer is randomly selected and moved together with the $(q-1)$ nearest customers from their tours to the customer pool.
4. *Tour Removal:* randomly removes a single tour, then moves all the allocated customers from this single tour to the customer pool.
5. *Facility Removal:* randomly selects an open facility to shut down and moves all its allocated customers to the customer pool.
6. *Facility Opening:* randomly selects a closed facility to open and select the q geographically nearest customers from other facilities and places inside the customer pool.
7. *Facility Swap:* randomly selects an open facility to shut down and selects the geographically nearest close facility to open.
8. *Worst Balance tour Removal (smallest):* selects the vehicle tour with the smallest cumulated customer demand from the average tour load.
9. *Worst Balance tour Removal (largest):* selects the vehicle tour with the largest cumulated customer demand from the average tour load.
10. *Routing Redistribution:* select from each open facility a random number k ($1 \leq k \leq 3$ of routes to be removed.

The first three destroy operators are at the customer node level, and the latter three are at the routing level. Customers inside the customer pool will be re-inserted by a repair operator selected from the below:

1. *Greedy Insertion:* Randomly selects a customer from the customer pool, inserts it into the position that increases the total expected costs by the least. The insertion can be between two consecutive customers or between the depot and a linking customer.
2. *Greedy Insertion Perturbation:* The same mechanism as *Greedy Insertion*. However, the insertion cost of the selected customer at each specific position is influenced by a perturbation factor d between [0.8, 1.2].
3. *Greedy Insertion Forbidden:* The same mechanism as *Greedy Insertion*, only that a customer node cannot be re-inserted to the same position removed from.

4. *k-Greedy Insertion Regret:* ranks the nodes inside customer pool from the highest to lowest regret values computed as the insertion cost difference for the first and $k-th$ cheapest position.

We apply the ALNS framework with the embedded Roulette Wheel operator selection mechanism and Simulated Annealing solution acceptance mechanism to intensify and diversify the search. We adapt the *move*, *swap*, and *2-opt* local search methods shown in Figure 2.1 to further strengthen incumbents within the best-found solution threshold. The ALNS improvement objective involves a linear combination of four objective terms, consisting of the second-echelon traversal and vehicle hiring costs, together with facility opening and operational costs.

First-echelon freight flow decision

After optimising the second-echelon deliveries and LC locations via the ALNS metaheuristic, we compute the demand of each LC using its throughput and solve the following MIP formulation to optimise the first-echelon freight flow and CC location decisions.

$$min \sum_{\substack{h,i \in E^1, \\ h \neq i}} \sum_{l \in L} C_{hi} \cdot u_{hi}^{l} + \sum_{h \in H} \sum_{i \in I} F^1 \cdot u_{hi}^{l} + \sum_{h \in H} O_h v_h + \sum_{h \in H} \sum_{i \in I} \left(OP^C \cdot f_{hi}^{l} \right) \tag{4.8a}$$

subject to

$$\begin{aligned} &(4.1g), (4.1h), (4.1i), (4.1j), (4.1k), (4.1m), (4.1n) \\ &v_h, u_{ij}^{l} \in \{0,1\}, f_{st}^{l} \in R^{+} \end{aligned} \tag{4.8b}$$

4.6.5 Tactical Planning

Tactical plans accommodate customer demand variations and provide feedback to the existing master template to fit the up-to-date customer data. We apply the ALNS metaheuristic to update the routing, vehicle type and capacity decisions on a more frequent basis. During the optimisation process, upgrading or downgrading the vehicle's total or cooling area capacity is allowed. However, vehicle overweight is prohibited and will incur a penalty equal to the cost of hiring an additional vehicle capable of holding the exceeding freight.

For each period, the system simulates a route within each driver district based on individual type-specific customer demands on that particular day. Although we assume that all customers require daily visits, the vehicle capacity might limit the driver's ability to visit all assigned customers in a single tour, resulting in an extra tour or a recourse visit by another driver. This is the reason for introducing the tactical planning stage.

Unlike the ALNS objective during the master template planning, we aim to balance two conflicting goals: balancing the traversal load amongst drivers while maintaining a good service consistency. That is, we wish to keep customers within their initial driver district for as many periods as possible unless the workload of that district becomes significantly overwhelming (or underwhelming). Nevertheless, if transferring a customer to another driver district is unavoidable, we wish the move to be within the facility district to maintain the freight supply consistency.

We adjust the driver districting decision for each period (i.e. week) based on the performance data computed from the previous period. For instance, if a driver district

constantly has the carrying weight or driver workload being outside the deviation zone, the driver district should be re-optimised so that driver-based and facility-based service consistency are ultimately maintained. The tactical plans will be feedback to the master template to keep it up-to-date.

4.7 Experiments

4.7.1 Parameter Setting

The Python API for IBM CPLEX is used on a Window 10 computer with 64-bit Operating System, CPU 1.70-1.90GHz and 16 GB RAM memory.

For data generation, the customer demand for each type (ambient and frozen/chilled products) is randomly generated between 1 and 10 units following a uniform distribution for all customers. For the capacitated instances, the capacity for all CCs is set to 800 units and 200 units for all LCs. The capacity for the primary vehicles is set to 200 units, and the two types of secondary vehicles are set to 100 and 60 units. For the geographical locations, all LCs and customers coordinates are generated within a 100 by 100 rectangle (each distance unit is 0.1 mile), shown in Figure 4.5 in turquoise, representing the city area. All CCs coordinates are randomly generated outside the coloured zone but within the outer 150 by 150 rectangle in white, representing the rural area around a city. The transportation costs in this model are independent of the weight carried by the vehicles. For any vehicle type in any echelon, the transportation cost between two nodes (X_1, Y_1) and (X_2, Y_2) is simplified as the integer part of the Euclidean distance $\sqrt{(X_1 - X_2)^2 + (Y_1 - Y_2)^2}$ divided by 10. The primary vehicle fixed cost is 200 per vehicle, and 150 and 100 per vehicle for each secondary vehicle types. The opening cost for CC is 2000 per site and 1000 per site for LC. The operational cost for a CC is 20 per unit freight and 15 per unit for an LC. An example of the optimal design of an instance consisting of 2 CCs, 3 LCs and 7 customers is shown in Figure 4.6 below.

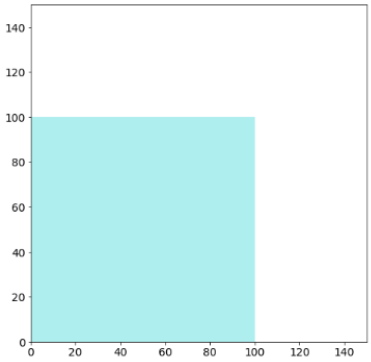


Figure 4.5: The geographical design for urban layout and the generation of CCs, LCs and customers nodes

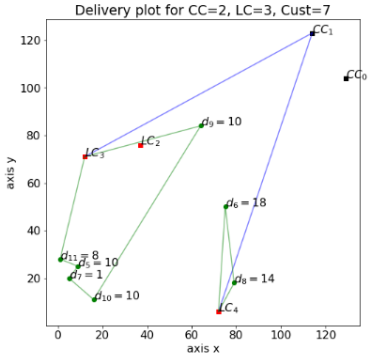


Figure 4.6: Sample traversal plot for 2 CCs, 3 LCs and 8 customers (capacitated facilities and vehicles)

4.7.2 Solution Method Comparisons

We run experiments for the single-period LoRD base model using the exact method. In the following tables, the first columns list the facility and vehicle capacity setting,

so a "cap - upcap" means the primary and secondary facilities are set with capacities, whereas the primary and secondary vehicles are set without. In contrast, an "uncap - cap" means uncapacitated facilities and capacitated vehicles for both levels. The following three numerical values represent the number of CCs, LCs and customers that are applied in the computation, for instance, a "2-3-5" represents an instance consisting of 2 CCs, 3 LCs and 5 customers, respectively. Two computational approaches are used in the first set of experiments: "CPLEX" is the single-period model solved with all the constraints given in (4.1) and the districting constraints (4.2). "CRA" is the single-period model solved with the CRA given in Section 4.5.1 with the exponential-sized subtour elimination and path elimination constraints constraints added in a cutting-plane fashion.

From Table 4.4, CPLEX reaches optimality faster than CRA under the "CC" and "UC" settings for all instance sizes. CRA suppresses CPLEX in its performance with an increasing customer size under the "CU" setting. In "UU", CRA reaches optimality faster than CPLEX. The observation is related to the mechanism of the two solution methods. The CRA algorithm starts from a subgroup of constraints that gives a lower bound to the minimisation problem. Whenever a violated constraint is detected under the CRA approach, a corresponding cut is added to the model for re-optimisation. The method reaches feasibility by iteratively cutting off the infeasible incumbent and thus increasing the solution's lower bound. Therefore, uncapacitated vehicles lead to a much lower chance of any vehicle-related constraint being reintroduced and having the model re-optimised. The CPLEX method, on the other hand, starts from a feasible upper bound and tries to reach optimality/global minimum.

Even though the vehicle capacity is more commonly required in real-life planning, meaning that we should opt for "CC" or "UC" settings if possible, both exact solution methods fail to reach the optimal solution for small-scale instances up to 10 customers within the time scale of an hour. Particularly, for the "UC-2,3,10" instance, although CPLEX API has achieved optimality, it consumed 85.3% of the time optimising on the last 0.03% to reach optimality.

In comparison, the proposed ALNS-based heuristic demonstrates more efficient performance in achieving high solution quality within a short computational time. In Table 4.4, the heuristic secures near-optimality for small-scale instances with no more than 2% difference within a fraction of time compared to its exact method counterparts. For comparison, we also tested the CPLEX built-in heuristic to evaluate its performance against the proposed ALNS-based heuristic. This additional set of experiments are under "CPLEX-H". The algorithm setting is adjusted by turning on the relevant parameters for heuristics inside the CPLEX solver, turning off the cutting planes, and solving only the root node before branching begins. The CPLEX built-in heuristic demonstrates notable computational efficiency, successfully attaining optimal solutions across all UU instances and half of the remaining instances. Nonetheless, concerning the CC instances and all 8 to 10-customer scenarios, our ALNS-based heuristic surpasses the built-in heuristic by yielding superior-quality solutions.

Table 4.4: Experiment 1. Small scale computation

Setting	CPLEX			CRA			ALNS			CPLEX-H		
Status-Size	CPU (sec)	Solution	Gap	CPU (sec)	Solution	Difference	CPU (sec)	Solution	Difference	CPU (sec)	Solution	Difference
CC-2,3,5	0.6	4514	0	0.6	4514	0.00%	6.8	4514	0.00%	0.6	4514	0.00%
CC-2,3,6	1.7	4638	0	4.9	4638	0.00%	8.2	4638	0.00%	0.9	4638	0.00%
CC-2,3,7	9.1	4862	0	45.6	4862	0.00%	9.7	4930	1.40%	3.2.	4914	1.07%
CC-2,3,8	99.9	5726	0	840.4	5726	0.00%	18.5	5730	0.07%	10.5	5748	0.38%
CC-2,3,9,	1103.5	5771	0	3600+	nf	-	14.9	5859	1.52%	8.4	5892	2.10%
CC-2,3,10	3600+	5793	2.87%	3600+	nf	-	16.6	5856	1.09%	12.7	5936	2.47%
UC-2,3,5	0.8	4474	0	0.8	4474	0.00%	6.9	4474	0.00%	0.9	4474	0.00%
UC-2,3,6	1.4	4628	0	5.7	4628	0.00%	7.2	4628	0.00%	1.1	4628	0.00%
UC-2,3,7	7.2	4640	0	40.5	4640	0.00%	8.6	4640	0.00%	2.6	4640	0.00%
UC-2,3,8	43.8	4685	0	80.8	4685	0.00%	8.8	4685	0.00%	6.5	4685	0.00%
UC-2,3,9,	73.6	4728	0	3600+	nf	-	9.4	4748	0.42%	9.6	4761	0.70%
UC-2,3,10	1368.8	4738	0	3600+	nf	-	9.7	4738	0.00%	10.4	4833	2.01%
CU-2,3,5	0.7	4474	0	0.8	4474	0.00%	7.2	4474	0.00%	0.5	4474	0.00%
CU-2,3,6	1.9	4504	0	2.4	4504	0.00%	8.6	4504	0.00%	0.6	4504	0.00%
CU-2,3,7	7.9	4730	0	6.8	4730	0.00%	9.5	4730	0.00%	4.1	4730	0.00%
CU-2,3,8	107.8	5628	0	189.2	5628	0.00%	10.4	5654	0.46%	2.8	5682	0.96%
CU-2,3,9	3600+	5683	0	406.2	5683	0.00%	9.6	5683	0.00%	8.9	5697	0.25%
CU-2,3,10	3600+	5693	2.88%	2146.3	5693	0.00%	12.8	5697	0.07%	4.5	5716	0.41%
UU-2,3,5	0.9	4474	0	0.5	4474	0.00%	7.2	4474	0.00%	0.8	4474	0.00%
UU-2,3,6	2.1	4504	0	1.9	4504	0.00%	7.5	4504	0.00%	0.2	4504	0.00%
UU-2,3,7	5.5	4506	0	4.1	4506	0.00%	8.9	4506	0.00%	2.2	4506	0.00%
UU-2,3,8	44.8	4539	0	21.7	4539	0.00%	8.8	4539	0.00%	4.3	4539	0.00%
UU-2,3,9,	152	4540	0	98.4	4540	0.00%	9.2	4540	0.00%	4.4	4540	0.00%
UU-2,3,10	483.5	4546	0	390.2	4546	0.00%	9.8	4546	0.00%	4.6	4546	0.00%

Status: the capacity status for facilities and vehicles in both echelons.
Size: the number of CC, LC and customers defined in the experiment.
nf: unfinished. Computation time limited to 3600 seconds.

100

Table 4.5: Experiment 2. large scale computation using ALNS

Number of customers	**CPU (sec)**	**Solution improve**	**Opening costs improve**	**Vehicle fixed costs improve**	**Number of hires (prior)**	**Number of hires (post)**	**Travel distance improve**	**Vehicles utilities improve**	**Workload balance improve**
40	42.9	29.65%	-55.56%	0.00%	9	9	5.31%	1.37%	99.60%
60	53.6	52.22%	8.42%	-5.71%	12	13	-18.96%	4.64%	88.21%
80	63.5	48.31%	0.00%	9.09%	16	15	-4.77%	9.68%	99.06%
100	71.2	44.03%	0.00%	11.11%	20	18	-16.50%	9.22%	90.80%
120	75.6	42.00%	0.00%	6.35%	23	22	-5.89%	6.49%	95.09%
160	87.7	48.34%	0.00%	2.33%	30	29	-2.94%	1.95%	92.68%
200	107.4	39.75%	0.00%	5.77%	36	35	-16.50%	5.52%	64.35%
avg.	71.7	43.47%	-6.73%	4.13%			-4.39%	5.55%	89.97%

For larger instances shown in Table 4.5, the proposed ALNS-based heuristic can solve instances up to 200 nodes within 2 minutes of computational time. For most instances, the heuristic can reduce the number of customer tours within the solution and bring down the total hiring and travelling costs. Except for the 60-customer instance, the heuristic can reduce the fleet size by at least one for each instance size. Nevertheless, for the case of 60 customers, the increased fleet size is due to the algorithm exchanging a low-utility large vehicle with two smaller ones to ensure a more balanced workload and higher utility. Notice that since the vehicle utility and the driver workload balance are considered inside the objective value, the total traversal distance becomes slightly worse since some drivers with lower utility than average may need to travel additional distance to cover customers initially assigned to drivers with workloads much higher or vehicles with a high utility rate.

4.7.3 Heuristic Parameters Analysis

In this section, a series of parameter analyses akin to those conducted in Section 2.5 are performed to evaluate the performance of the proposed ALNS-based heuristic under various parameter configurations. Unless specified, the parameters adhere to the values specified in Table 4.3 and undergo testing across a dataset comprising 50 generated instances, each encompassing 200 customer nodes.

In addition to the parameters whose values conform to those reported in the literature that demonstrate good performance, we have tested the values of the following parameters, including the ALNS non-changing iterations $iter$, destroy scale q, and local search threshold λ, focusing on their impact on the best-found solution quality as well as computational efficiency. In each experiment, we record the computational time consumed by executing each round of the ALNS heuristic, in contrast to the computational iterations recorded in Section 2.5. This approach is motivated by the emphasis on the computational time relevant to real-world applications, rather than iterations, as a metric to evaluate the efficiency of heuristic designs. Furthermore, three additional grid searches were performed to optimise additional parameters including the reaction factor and weight segment length for Simulated Annealing, the initial temperature and cooling factor, and one for the Roulette Wheel scores δ_1, δ_2, and δ_3.

To begin with, analysis of the data presented in Figure 4.7 reveals that increasing the parameters for non-changing iterations leads to a decrease in the average objective value in general. However, this adjustment is concurrent with an increase in computational time. A balance point is observed at approximately 500 for this parameter.

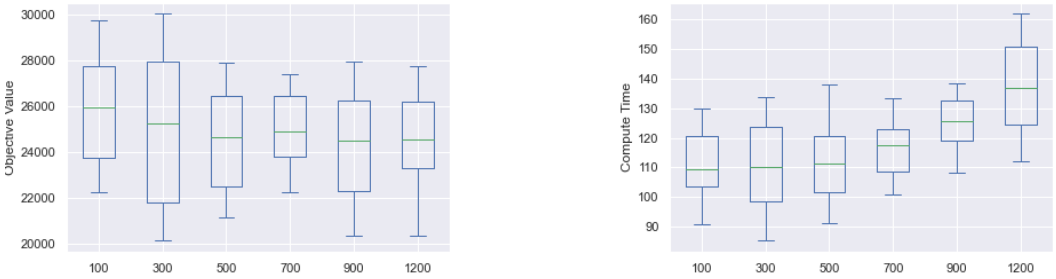


Figure 4.7: Non-changing iterations for ALNS heuristic. Each box plot contains 50 repetitions. Left: best-found objective values. Right: computational time.

The observation depicted in Figure 4.8 presents a complexity, wherein a small scale

of destruction may prove inadequate for sufficiently exploring deeper regions of the neighbourhood, where potentially superior solutions exist, without incurring excessive computational cost in identifying local optima. Moreover, given the multifaceted nature of the LoRD problem, it is evident that employing a larger scale of destruction may entail the risk of destabilising the existing solution structure. Indeed, prevailing conventions in the literature often advocate for a modest destruction scale ranging between 10% to 20%.

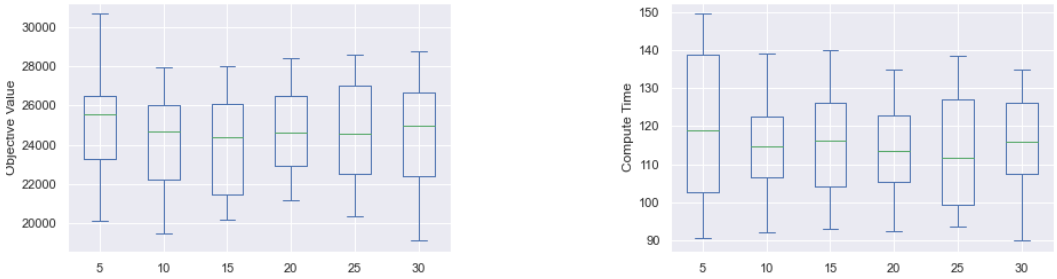


Figure 4.8: ALNS operator destroy scale analysis. Each box plot contains 50 repetitions. Left: best-found objective values. Right: computational time.

From the findings depicted in Figure 4.9, no obvious pattern is evident. However, there is a consistent rise in the average computation time required as the local search threshold increases. This phenomenon can again be attributed to the inherent complexity of the LoRD problem, where the exploration of the local neighbourhood may exhibit diminished efficiency.

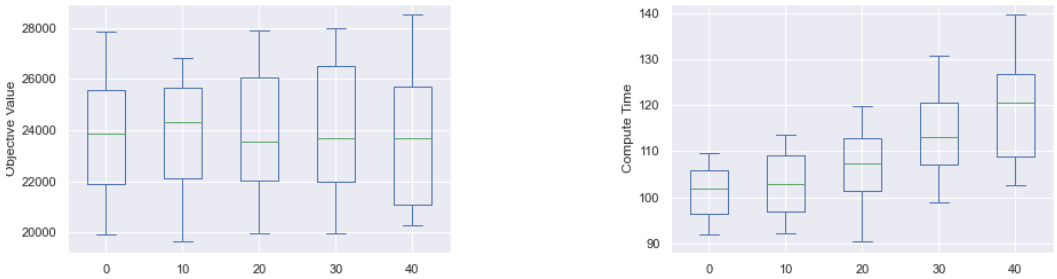


Figure 4.9: ALNS local search threshold value analysis. Each box plot contains 50 repetitions. Left: best-found objective values. Right: computational time.

In examining Figure 4.10, it becomes evident that superior performance is attained when the weight segment length is configured for fewer than 100 operator pairs. This observation suggests that accumulating a higher selection probability for a specific operator might yield less favourable outcomes compared to employing a diverse range of operators within the search trajectory.

From Figure 4.11, we discern numerous instances of locally optimal combinations for the scores δ_1, δ_2 and δ_3. This observation suggests that the established literature setting of $33 : 31 : 19$ (the scale is approximately equivalent to $100 : 90 : 60$) may benefit from refinement through fine-tuning or a more comprehensive factorial analysis tailored to individual instances.

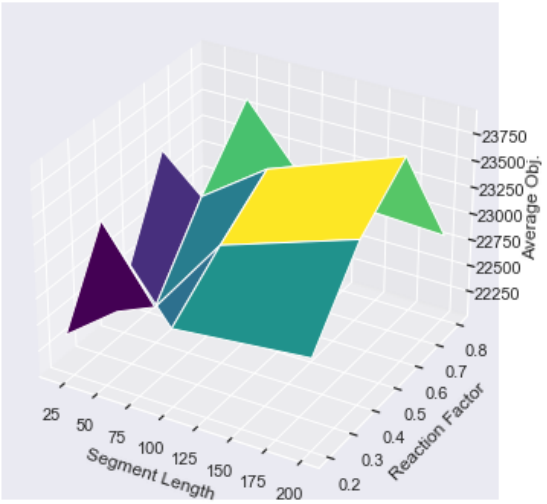


Figure 4.10: RW segment length-reaction factor combination grid search. Each grid is averaged from 50 repetitions of the minimum best-found solution.

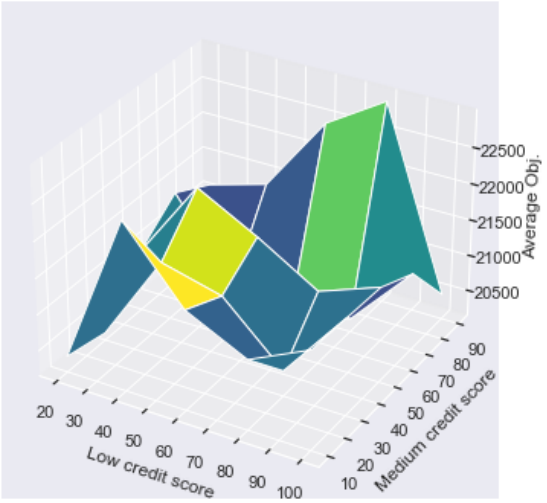


Figure 4.11: RW credit score grid search. Each grid is averaged from 50 repetitions of the minimum best-found solution.

Lastly, it is observed from Figure 4.12 that a reduced temperature descent rate increases the probability of the search algorithm reaching a higher-quality solution.

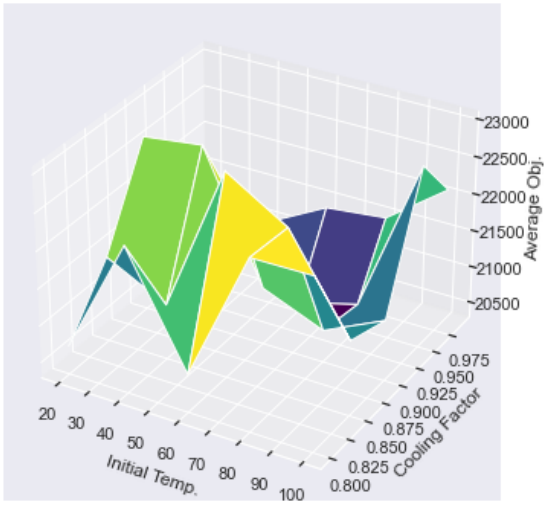


Figure 4.12: SA initial temperature-cooling factor combination grid search. Each grid is averaged from 50 repetitions of the minimum best-found solution.

4.8 Conclusions

This research proposes a Location-Routing variant with a districting aspect (LoRD model). We expanded the classical customer allocation decision into delivery zone partitioning, where customer-driver and customer-facility allocations are updated periodically based on daily demand variations. We developed MIP formulations of the LoRD model with multiple periods as well as facility-based and driver-based catchment areas. The complexity of solving a small-instance problem results in the development of a tailored two-stage heuristic approach based on the ALNS metaheuristic. The first stage of the heuristic is to come up with a master template with minimised total costs and drivers' workload balance. The second stage is to update the template on a tactical level to fit the up-to-date customer demand meanwhile maintaining reasonable workload balance and service consistency rates.

For future research, a multi-period model with workload balance and service consistency can be further evaluated, with the potential to update the multi-period model to include period-dependent customer demand and develop a feedback system corresponding to the change. The current daily customer demand can be enriched with an ordering pattern to resemble a real-life situation, where each customer is visited only on certain days of the week. Besides, CCs in reality, may only supply certain types of commodities (i.e. only ambient or frozen/chilled food), meaning that a more type-specific network optimisation approach should be developed for the first echelon. Lastly, a more comprehensive factorial analysis on the heuristic parameters to examine their combinatorial improvement on the heuristic performance is beneficial.

Chapter 5

Improvement of ALNS using Reinforcement Learning

5.1 Introduction

The Adaptive Large Neighbourhood Search (ALNS) metaheuristic was first introduced by Ropke and Pisinger [201] as an extension of LNS to solve combinatorial optimisation problems (COPs). The method involves a "relax-and-reoptimise" perturbation process that iteratively deconstructs and reconstructs part of the solution in order to explore unvisited parts of the search space towards promising directions.

The ALNS metaheuristic is renowned for its effectiveness in generating good-quality solutions within reasonable computational time. However, despite its extensive application for solving a variety of COPs, the specific contributions of each ALNS component towards its general performance are not well-explored or understood. In a recent state-of-the-art review of ALNS by Mara et al. [159], it was highlighted that only a meager 2 out of 252 papers have gone beyond the straightforward implementation and delved into in-depth component-based analysis, including Santini et al. [206] who focuses on the selection of the ALNS acceptance criterion, and Turkeš et al. [232] who investigates the effectiveness of the ALNS adaptive layer for operator selection. These studies, while shedding light on key aspects of ALNS, nevertheless indicate the need for further research and in-depth analysis of its individual components to enhance its overall performance and advance our understanding of the metaheuristic.

We summarise three main deficiencies that exist in the current ALNS framework. Firstly, several studies have indicated that the adaptive layer within the ALNS framework has limited capability to dynamically select the best operators, despite being intentionally engineered for this purpose. The recent meta-analysis on ALNS by Turkeš et al. [232] reported a mere 0.14% average improvement brought by the adaptive layer from the analysis of 25 ALNS implementations in the literature, emphasising the need for a more efficient operator selection mechanism that reflects the contribution of individual operators accurately. Guided by preliminary experiments, a few authors including Lehuédé et al. [142], Grangier et al. [85] and Tellez et al. [227] went as far as to replace the adaptive layer altogether with a uniform random distribution or decided on a total omission due to its heavy computation cost. Moreover, the frequent attempt of hybridisation with an additional local search procedure employed at an additional computation cost to intensify the search process [159] indicates that the incumbent created by the selected operator pairs may not have sufficiently good performance after all. This poor adaptability may arise from certain groups of operators being more effective

at specific stages of the search process and not useful at another, which can fail to be detected by an adaptive layer that relies on periodic updates of each operator's selection probability. Consequently, if the operator selection process in the classic ALNS framework is not carefully fine-tuned, it can easily result in suboptimal performance. To address this issue, it is crucial to develop a more automatic and refined strategy to update the operator selection probability and ensure capturing a more accurate reflection of the pinpoint fitness of each operator, therefore optimising the effective selection process and enhancing the metaheuristic's efficiency of finding high-quality solutions.

Secondly, operator portfolio design for a particular problem can require considerable manual evaluation [159]. Optimising the designated operator portfolio for ALNS remains largely handcrafted based on expert knowledge, as there is no clear problem-oriented guideline of which configuration provides the best direction in improving the objective value for a specific problem. Naturally, the process of identifying an efficient set of operators from the existing literature entails a significant amount of unavoidable manual testing, which is compounded by the burden of an insufficient adaptive layer. It also necessitates manual efforts in designing problem-specific operators as well as adapting existing ones to fit the problem features. Evaluating an operator's efficiency by its average performance can be subjective. As suggested by Kallestad et al. [117], while handpicking individual operators based on their average performance may seem reasonable in operator portfolio design, it can result in the neglect of good performance at specific moments on the search trajectory, thus hindering the overall power of an operator set. In addition, the complexity of operator selection increases when the integrated performance of an operator set surpasses the individual performance of its components, whereas some combination of operators can have a detrimental impact on the overall performance. Therefore, sequentially including or eliminating operators until attaining a certain set may lead to a suboptimal outcome. Finding the optimal combination of operators that gives the best solution will result in a subset selection problem that is NP-hard. Alternatively, we require a costly factorial analysis to systematically explore the best integrated performance out of all operator combinations for any particular problem. Overall, selecting operators to form an ideal portfolio can be an expensive handcrafted tailoring process requiring immense labour to handcraft rules and repetitive trial-and-errors experiments - the common drawbacks of classic heuristic design (i.e. see Khalil et al. [125], Wu et al. [249] and Garmendia et al. [74]). The process itself is also highly problem-specific, which leads to increasing manual and computational costs that eventually decreases the adaptability of the ALNS metaheuristic to generalised problems. To address this issue, we need an agile operator selection system with online adaptation to ensure a continuous refinement of each operator's selection probability.

Lastly, the choice of portfolio size is also delicate: too few operators might not enable the search to visit unexplored neighbourhoods, but a plethora of operators can introduce noise to the adaptive layer, especially when some of the operators may not be helpful. Thomas and Schaus [229] proposed a blackbox ALNS equipped with a portfolio of 30 destroy operators and 36 repair operators grouped explicitly into several categories to fit specific neighbourhood structures. Despite achieving the best performance out of 25 ALNS implementations in the meta-analysis by Turkeš et al. [232], the authors actually showed that only a handful of operators could be particularly beneficial for a given COP. They even indicated that the non-adaptive version of ALNS with a single best operator pair could outperform ALNS with many if selected appropriately. As a result, the task of refining an appropriate portfolio as well as enhancing the selection accuracy of each operator at the right time have become increasingly crucial in the

pursuit of better heuristical performance, especially when tackling a broader spectrum of optimisation problems characterized by numerous problem-specific neighbourhoods. Still, with a relatively lower probability of making poor choices given a larger sized operator portfolio, there exists the typical game of using quantity to make up for quality and hiring an "army" of operators to hit the target by chance. However, the way of computing operator selection probability within the current ALNS framework means that this method can backfire. Our studies in Section 5.6.2 illustrate the varying performance with different operator portfolio sizes, demonstrating that the potential of search can be limited by either an insufficient or oversized operator portfolio. Consequently, an automatic way of selecting operators without considering the population probability is needed.

In the literature, the primal foci of researchers have been the horse-racing game: whether an ALNS implementation can produce better speed and solution quality than another exact method or heuristic, as well as applying ALNS to new problem variants. Comparatively less attention has been dedicated to the performance of ALNS components and how to improve them individually. After 16 years of intensive research into ALNS, we still require a more adaptive online mechanism that tells which operators to use and when to use them, both of which received limited attention in the existing ALNS literature. In contrast, the core goal of this Chapter is to design an effective yet principled operator selection mechanism.

We achieve this through the use of Machine Learning (ML), an increasingly important tool for learning to optimise the optimisation process [25]. Advancements in this area have the potential to not only optimise the search process but also enhance the quality and speed of finding high-quality solutions for various combinatorial optimisation problems. In this study, we opt for Reinforcement Learning (RL), a type of ML technique that provides a framework for learning how to optimise a given reward function by trial-and-error. In the ALNS context, an agent is rewarded for selecting sequences of operators that lead to the largest improvements in the objective function value of the solution. This helps to develop a learning-based operator selection mechanism that enhances the accuracy on determining the appropriate selection of operators to improve the overall solution quality. Experimental results demonstrate that our learning-based adaptive layer exhibits enhanced sensitivity and robustness compared to the classical RW. In our work, we make the following contributions:

- We formulate the choice of a sequence of destroy and repair operators as a standalone Markov Decision Process (MDP), in which an agent receives a reward proportional to the improvement in the solution. We draw a correspondence between value-based RL methods used to solve MDPs, such as Q-learning, and the classic RW update used in ALNS. A key insight is that RL estimates are conditioned on the current solution, while RW updates are independent of it, which indicates the potential to learn a stronger operator selector through the RL framework;

- We propose a practical approach based on Deep RL for learning to select operators. Furthermore, we highlight the potential of Graph Neural Networks (GNNs) for state representation, which allows for end-to-end learning of rich features while generalising to problem instances of different sizes;

- We carry out an extensive evaluation that includes a large selection of representative operators from the literature. Our results demonstrate that the proposed

approach performs significantly better than the RW mechanism. We also analyse the impact of important practical considerations such as portfolio sizes and destroy operator scales on the quality of the solutions.

- Our proposed operator selection processes empowered by a Deep Q-Network and a learning-based Roulette Wheel mitigate the drawbacks arising from inadequate manual design of the portfolio by enabling the decision-making agent to intelligently select beneficial operators while avoiding detrimental ones from the portfolio. Our approach minimises the limitations associated with manual design of ALNS operators.

5.2 Literature Review

In recent years, the use of ML methods to solve highly complex COPs has become increasingly prominent [25], especially for the Vehicle Routing Problem (VRP) and its variants [15]. ML encompasses multiple learning techniques, including supervised learning, unsupervised learning, and reinforcement learning [26]. One significant advancement involves training RL models to directly provide solutions based on the input instances of discrete optimisation problems. Once trained, these models can be applied to solve a broad range of instances with minimal time requirements. This capability places RL as a promising alternative to classical solution methods for solving highly complex COPs.

5.2.1 ML as construction algorithm

Several pioneering studies applied RL to directly construct solutions for routing-related problems. Bello et al. [21] were the first to apply policy gradient algorithms to tackle the Travelling Salesperson Problem (TSP). They developed a neural combinatorial optimisation framework and trained it with end-to-end RL to learn approximate solutions using the tour length as a reward signal. Their method is capable of obtaining close-to-optimal results on TSP graphs. However, their tests were only up to 100 customer nodes. Khalil et al. [125] addressed three graph optimisation problems including the minimum vertex cover, maximum cut and TSP. They used Deep Q-learning to train a greedy policy parameterised by a GNN to incrementally construct solutions, with the set of actions determined by the solution state. Their proposed model was trained with up to 100 nodes, while its generalisation ability were evaluated on instances comprising up to 1200 nodes. Demonstrating superiority over several competitive popular heuristics, the model attains near-optimal results, with margins of $0.2\%, 2.2\%$ and 4.8% on average, respectively, compared to the optimal solutions for the corresponding optimisation problems. Nazari et al. [175] proposed an end-to-end framework with attention mechanism that outputs solutions directly from the routing-based problem instances. Their method can efficiently solve various routing problem instances of size up to 100 nodes, and outperforms many classic VRP heuristics. Kool et al. [130] proposed a construction heuristic that consists of an attention-based decoder trained with Reinforce policy gradient estimator to auto-regressively build solutions for TSP, orienteering problem, and prize-collecting TSP. Their model demonstrated competitive performance with efficient computing time, but again was tested for up to 100 nodes when compared to several state-of-the-art learning baselines and heuristics. Kim et al. [126] proposed a collaborative scheme combining two iterative Deep RL policies that construct and improve candidate solutions to various routing problems including TSP, prize-collecting

TSP, and CVRP. Other variants of using RL to tackle routing problems can be found in the work of Deudon et al. [57], Qiu et al. [189] and Zhang et al. [257]. More broadly in the combinatorial optimisation context, Emami and Ranka [65] designed a novel policy gradient method to learn policies on permutations, which is a fundamental class of COPs.

5.2.2 ML as enhancement algorithm

ML techniques can also be applied in many cases to enhance the performance of heuristics and metaheuristics [226], the latter group are problem-independent optimisation guidelines applicable to a broad range of problems. The reader is referred to the work of Karimi-Mamaghan et al. [120] for a comprehensive review on the integration of ML and metaheuristics to tackle COPs. In recent years, RL has been studied and applied to a wide variety of COPs [161] and with a particular focus on routing problems [195]. There are a number of RL applications on existing metaheuristics, including local search [261], iterated local search [118, 119], simulated annealing [167], variable neighbourhood search [60, 115], memetic search [93], genetic algorithm [3], artificial bee colony algorithm [64, 240], and other optimisation processes [259].

Traditional improvement heuristics are typically guided by handcrafted algorithms that require extensive domain knowledge and a considerable investment of manual designing effort and time. Yet, despite being carefully handcrafted by domain experts, classical heuristics often struggle to achieve satisfactory performance due to the intrinsic difficulties of the problems they tackle and the frequencies those heuristics need to be updated to fit different problems, no matter how slight the difference is. Often, recurring designing afresh for each new problem and numerous trial-and-error experiments are unavoidable before securing good heuristic performance [125]. To overcome these limitations, incorporating ML techniques with the classic solution framework offers a viable alternative to alleviate the complexity associated with manually designing algorithms. Lu et al. [155] presented a learn-to-improve framework for CVRP with up to 100 nodes to iteratively refine an initial solution with customised improvement operators chosen by an RL-based controller. Chen and Tian [35] proposed a Deep RL-based algorithm to iteratively select and rewrite the local components of an existing solution until convergence, which is used to tackle multiple problem domains including VRP and scheduling problems. Wu et al. [249] applied Deep RL to learn the improvement heuristics to iteratively employ local search-based operators for routing-based COPs including TSP and CVRP. Karimi-Mamaghan et al. [121] integrated Q-learning into the iterated greedy metaheuristic as an efficient operator selection mechanism for the permutation flowshop scheduling problem up to 800 jobs and 60 machines.

To our knowledge, the majority of the referenced literature on end-to-end learning primarily focuses on relatively small problem instances. Consequently, the applicability of these approaches on a larger scale, frequently accompanied by resource-intensive off-line training processes, is notably constrained. This limitation renders them less competitive when compared to state-of-the-art classical combinatorial optimisation approaches tailored for addressing such challenges.

5.2.3 Learning Representation & ML-COP Integration

Another orthogonal yet important aspect when applying ML to combinatorial optimisation is the learning representation that is used, i.e., the way the inputs are provided to and processed by the model. A classic learning architecture is the Multi-Layer Per-

ceptron (MLP) neural network, which consists of layers of linear multiplications with learnable weight matrices, followed by applications of a non-linear activation function. While the MLP is demonstrably able to represent any continuous real function with sufficient parameters [46], it may not necessarily exploit problem structure effectively. Akin to the convolutional neural networks successful in the computer vision domain, Graph Neural Networks (GNNs) [209, 222] are learning architectures explicitly designed to operate on graphs and encode suitable inductive biases. As improvement of the Pointer Networks [239], GNNs provide us with an effective architecture to represent the problem features without the need to consider any specific order of the input sequence, which is beneficial since many optimisation problems have shown an inherent graph structure or have the data itself being represented by a graph [237].

In recent years, the graph ML community has explored the development of methods that provide different trade-offs in terms of expressibility and scalability (e.g., [47, 91, 235]). Advantages of GNNs that are worth mentioning in this context are their end-to-end learning of relevant features without requiring manual design, their permutation invariance, as well as their ability to operate on problem instances of different sizes (unlike the MLP). Apart from message passing-based methods that typically occurs in structured graphical models, GNNs encompass various other types, such as the spectral-based methods. The reader is encouraged to consult a recent paper by Zhou et al. [260] for a comprehensive reivew on GNN. Let us briefly refer to other works that apply RL and GNNs to combinatorial optimisation. Beyond the already-discussed pioneering work of Khalil et al. [125], similar methodologies have found applications in accelerating the column generation algorithm for solving linear programs [37], finding maximum independent sets in graphs [4], and learning to construct graphs representing infrastructure networks [52] or molecules [252].

Furthermore, in recent years, a key advancement is the development of neural combinatorial optimisation [74], the idea of learning good heuristics by the means of neural network models and deep (reinforcement) learning, to provide solutions to difficult NP-hard COPs. Besides a few earlier works by Khalil et al. [125] and Kool et al. [130] on this topic, Drori et al. [63] proposed a unified framework for model-free RL using a GNN representation trained on a line graph generated by converting edges into nodes. Their framework can be generalised to train on small-sized graphs while being capable of evaluating larger graphs, as well as addressing a number of polynomial and NP-hard COPs such as Minimum Spanning Tree, Single-source Shortest Paths and TSP by minor adjustments in the reward function. Song et al. [220] applied an end-to-end Deep RL approach composed of a heterogeneous graph representation to learn high-quality dispatching rules to assign jobs to machines in the Job-shop Scheduling Problem. Their GNN-based neural architecture effectively captures complex relationships within the job-to-machine assignment with high efficiency, resulting in better performance than traditional manually-designed heuristics of similar nature. Garmendia et al. [74] proposed an end-to-end RL algorithm utilising a GNN encoder and an attention mechanism as the decoder to solve the Linear Ordering Problem. Their learning-based optimisation algorithm achieves competitive performance compared to those generated from the exact solver, a constructive heuristic, and two state-of-the-art metaheuristics. Zhang et al. [258] proposed a Deep RL-based manager-worker framework to tackle multiple TSPs with time window and rejections features. The framework incorporates a management agent trained on GNN responsible for customer assignment, alongside a worker agent for routing based on a self-attention encoder-decoder policy network. Watkins et al. [243] employed DeepQ-learning with a GNN for feature extraction to learn a greedy construction heuristic for the NP-Complete graph colouring problem to

determine the colouring order for the vertices. By incorporating state parameterisation of the graph, their proposed approach demonstrated enhanced performance when compared to existing greedy algorithms.

5.2.4 Incorporating ML with LNS

Several recent studies focused on the integration of ML techniques with the classic LNS. As the first paper on this topic, Hottung and Tierney [109] proposed two generalised random-based destroy operators and a single repair operator with automated learning based on a deep neural network with an attention mechanism. Their work was the first to consider the application of RL to LNS for solving a VRP, and achieved solutions of better quality than classic optimisation approaches. Nevertheless, their proposed learning mechanism only focuses on repairing incomplete solutions during the repair phase. In another work, Falkner et al. [69] integrated a pre-trained neural construction heuristic as the repair operator in the LNS framework to solve the VRP with time windows. The destroy procedures remain handcrafted and are classified into two groups without any learning involved. Oberweger et al. [179] enhanced the LNS framework with an ML-guided destroy operator to solve a staff rerostering problem. For the reconstruction phase, the authors developed a mixed-integer linear program as a repair method. Lagos and Pereira [133] proposed a multi-armed bandit algorithm for learning the online parameters that determine the selection of the next operator in the ALNS framework. Hendel [102] presented a structured heuristic framework encompassing ALNS for MIP, incorporating eight embedded LNS heuristics alongside a selection strategy guided by a multi-armed bandit algorithm that determines which embedded heuristic to employ based on both successful calls and failures. Syed et al. [224] proposed a neural network in an LNS setting to solve a vehicle ride-hailing problem. However, it uses Supervised Learning (SL), which requires a substantial training dataset and, furthermore, can only perform as well as the algorithm used for its generation. Bongiovanni et al. [28] also proposed a hybrid data-driven optimisation framework that contains an SL-guided operator classifier to predict the destroy-repair pair to improve an electric autonomous dial-a-ride problem. Although their framework can largely outperform the classic LNS and ALNS, the embedded SL technique requires training offline on a large labelled dataset to learn to select destroy-repair operator pairs, rendering it impractical when scalability of the method is a concern. Moreover, there is another set of works [1, 172, 219, 248] combining learning with LNS. However, this line of research focuses on learning the neighbourhood selection policies for generalised (mixed-) integer programming instead of metaheuristic hybridisation or solving a specific COP heuristically.

5.2.5 Comparison with Concurrent Works

Two very recent works by Kallestad et al. [117] and Reijnen et al. [197] both applied Deep RL to improve ALNS operator selection, using approaches that are similar to each other. The key differences with respect to them are given below.

Firstly, Kallestad et al. [117] and Reijnen et al. [197] both considered a state space that only uses high-level features related to the search status (e.g., the search step, the temperature, the performance gap from the best-found solution), while neglecting information linked to the current solution itself (e.g., the structure of the tours that make up the solution in a routing problem). In contrast, our method uses a substantially more expressive representation that focuses on the structure of the current solution itself

to decide the choice of an operator.

Furthermore, we formulate the decision-making problem and train the operator selection policy independently of the ALNS process. This allows us to isolate the problem of selecting an optimal sequence of operators without the noise introduced by the search process (i.e., the occasional acceptance of a worse solution) and to benchmark different methods accurately. It also implies that our approach is not limited to the specific variant of ALNS that is considered, and can be applied transparently with other modifications and enhancements to the higher-level metaheuristic. Our approach also features a GNN as a learning representation, as opposed to the other works that opt for standard MLP architectures. The GNN enables our method to use granular information about the current solution while being applicable to instances of different sizes. This also allows the learned model to successfully generalise to unseen problem sizes, including larger instances.

Finally, an important aspect that differentiates our work is the scope and size of the the considered operator portfolio, which contains 28 destroy and 7 repair operators. In comparison, Reijnen et al. [197] used 4 destroy and 3 repair operators and Kallestad et al. [117] opted for several destroy operators that are identical apart from the difference in destroy scales. Beyond the substantially larger portfolio size itself, a contribution of our work lies in the systematic review and integration of the destroy and repair heuristics that are found in the routing literature. As such, the experimental results we present are representative of the current state-of-the-art in this literature, and reflect the possible choices that a practitioner might face when deciding the operator portfolio for a given problem in the routing space.

We have designed comparisons and analysis of our method versus the learning framework presented by Reijnen et al. [197] and Kallestad et al. [117] in Section 6.3.6.

5.3 Neural Network

5.3.1 Artificial Neural Network

Artificial neural networks are a prominent ML technique that emulates the operation of brains in living organisms to identify and process underlying structures in a given dataset. Neural networks have a broad spectrum of applications in different domains, such as function approximation, classification, and data processing. A neural network comprises interconnected nodes, or *neurons*, that function as an information processing mechanism. Each neuron transmits activation signals to connected neurons after receiving such signals from adjacent neurons. In a biological nervous system, a nerve cell receives sensory input from preceding cells via dendrites and transmits impulses to consecutive cells through its axon. Similarly, an artificial neuron is a one-way algorithm that simulates a nerve cell's operation to accumulate, transform and relay signals to subsequent neurons, creating a domino effect. The activation value represents the numerical expression of the electrical signal associated with each neuron, with higher values indicating greater activation. This numerical information is transmitted throughout the network.

In an artificial neural network, neurons are densely connected to a large set of other neurons. Therefore, each of them accumulates signals fired from multiple input arcs and propagates signals forward through the network. As shown in Figure 5.1, an individual neuron is characterised by its *weight*, *bias*, and *activation function*. The connecting link between each pair of neurons carries a weight, determining how actively each neuron reacts to the input signals. Mathematically, the weight is a positive number

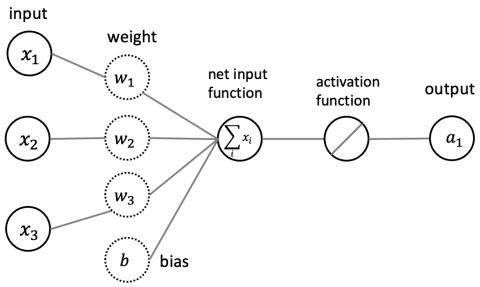


Figure 5.1: A graph of a single neuron with 3 input arcs linking to preceding neurons. This is the simplest neural network called a perceptron.

if a neuron excites its consecutive neuron or negative if it suppresses. A higher weight value indicates a stronger signal transmit to the other nodes. The activation value of a neuron is computed from the weighted sum of all activation values from the preceding neurons and with an additional bias term attached to shift the output range. The weighted sum prior to propagation is subjected to a predefined activation function that determines whether or not, or how much a neuron reacts to the cumulated signal from the preceding neurons. The output or activation of a neuron can be computed as:

$$a = g(w_1x_1 + w_2x_2 + w_3x_3 + \cdots + w_nx_n + b) = g(WX + b) \tag{5.1}$$

where $W = [w_1, w_2, ..., w_n]$ is the weight vector for all the input arcs, x is the input feature vector to the network, b is the added bias, and g is referred to as the activation function.

Many existing activation functions are dependent on the functionality of the neural network. A binary step function $g(x) = \begin{cases} 1 & \text{if } x < 1 \\ 0 & \text{otherwise} \end{cases}$ is commonly selected for a classification neuron that decides whether to be activated by comparing the input feed to a threshold value. A linear function, also called "no activation" $g(x) = x$ is selected for a neuron to perform linear regression. For nonlinear regression and classification problems that are not linearly separable, a nonlinear activation function adds nonlinear mappings between the neural network's inputs and outputs. Some popular choices include the Sigmoid (Logistic) function $g(x) = \frac{1}{(1+e^{-x})}$ with real valued inputs and re-scaled outputs between 0 and 1 that serves as input to the next neurons, the Rectified Linear Unit (ReLU) function $g(x) = \max\{0, x\}$ with deactivation for negative inputs and linear activation for all other cases, and Softmax function $g(x_i) = \frac{exp(x_i)}{\sum_i exp(x_i)}, \forall i \in I$ that transforms the set of I multiple real numbers into a probability distribution representing potential outcomes.

An artificial neural network propagates the computed values between the input and output neurons, utilising the weights as intermediate parameters. Training a neural network therefore is equivalent to letting the computer figure out a valid set of values for the weights and biases so that any output generated by the network closely approximates the real outcome.

5.3.2 Neural Network Architecture

The architecture of a simple neural network is formed by multiple layers of neurons, where artificial neurons belonging to the same layer are not connected among themselves

but to all neurons from the adjacent layers. Such a fully connected multi-layer neural network is called a Multi-layer Perceptron (MLP), which contains three types of layers: the *input layer* that takes in the information from the outside world, the *hidden layer* that analyses and processes the signals from the previous layer and passes them on to the next, and the *output layer* that outputs the processed result by the neural network.

For the input layer, the elements of the input vector are called *input features*. The number of features matches the number of neurons in the input layer of the network. The features contain a range of attributes that serve as predictors in the network. The output layer can contain a single or multiple nodes depending on the classification category. A single neuron is presented for regression tasks, and multiple neurons for classification tasks with the same number of classes as neurons, whose activation indicates the coordinating of the input vector to a categorised class within the output vector. Neural networks can possess multiple hidden layers to process data but have only a single input layer and output layer. An MLP is a typical example of a feedforward neural network that processes data in a one-way direction from the input layer to the output layer.

In a neural network, the output of each neuron can be interpreted as a probability distribution over all the possible classes, with the predicted class being the one with the highest probability score. Thus, the value of each neuron serves as a measure of the network's confidence in its prediction for that specific class. By evaluating the probability score from all neurons in the output layer, the network can distinguish the predicted class with the highest score that is deemed most probable.

5.3.3 Deep Neural Network

A neural network with several interconnected hidden layers is qualified as a Deep Neural Network (DNN) or Deep Learning Network. DNNs are compositions of functions computed at individual neurons. They operate in the feedforward way where the activations in one layer determine the activations in the subsequent layer. For a layer m consisting of n nodes with corresponding functions $g_1(\cdot), g_2(\cdot), \ldots, g_n(\cdot)$, a composition function in the next layer $m+1$ using the layer-m input can be represented as $f(g_1(\cdot), g_2(\cdot), \ldots, g_n(\cdot))$. If all $g_i(\cdot)$ are fixed as identify activation function, the network can be simplified to linear regression. In contrast, using nonlinear activation functions invokes the power of multi-layer networks, for they can theoretically map any size of input vectors to their corresponding output vectors. Hornik et al. [108] indicates that a single-hidden-layer feedforward network with nonlinear units and a single linear output layer is already capable of approximating most of the measurable functions to any desired degree of accuracy. Therefore, DNNs are fundamentally more powerful than their building blocks in jointly approximating the mapping function from inputs to outputs in the target domain. For this reason, DNNs are commonly utilised as *function approximators* for estimating the underlying mapping function, which is often referred to as the *target function* for being the target of the training process.

An iterative feedback process is applied to train DNNs to improve their predictions over time, i.e. fine-tuning to reach the ideal weight and bias realisations based on the learned representations. The training procedure consists of forward and backward propagations that enable the neural network to improve its accuracy over time on data prediction. *Forward propagation* is the process of feeding input data through the layers of neurons in a forward direction to generate output predictions, during which the input data is successively transformed at each layer by the corresponding weighted sums and activation functions. *Backward propagation*, also known as *backpropagation*,

is the process of improving prediction accuracy by updating the network weights and biases based on the loss function value, which measures the network performance by quantifying the error observed between the output prediction of the network and the true target values. The backpropagation process involves computing the gradient of the loss function with respect to network weights and biases, propagating through the network from the output layer back to the input layer. This loss function is then used in optimisation algorithms (e.g. gradient descent) to adjust the network weights in the next iterations and minimise future prediction discrepancy. As a result, the neural network performance is iteratively refined for a number of epochs using forward and backward propagations.

DNNs are renowned for providing powerful mappings from any input to the output type. Yet their complex architectures require extensive training compared to a simpler network or alternative ML techniques. One efficient approach for handling large training data is to employ mini-batch training, which divides the data into smaller samples called batches that are sequentially used to fit and refine the weights during the network training process. Applying mini-batches improves the gradient computation efficiency, preventing data overfitting caused by processing the entire dataset in a single round and thus accelerating convergence.

5.3.4 Graph Neural Network

Graph Neural Networks (GNNs) are a type of convolution neural network architecture that are explicitly designed to operate on data with graph-structured features. At a high level, given a vector of input features $\mathbf{x}^i$ for each node in the graph, GNNs compute an embedding vector μ^i that encode individual node features and its structural relationship to other nodes within the graph. This is achieved by applying several message passing layers that aggregate the features of neighbouring nodes and apply a non-linear activation function. From the node-wise embeddings, representations of the entire graph can be obtained, for example, by a sum aggregation: $\sum_i \mu^i$. Through this aggregation process, a GNN captures and encodes both local and global node-based information and transform the entire graph into higher-dimensional format called *graph representations* that can be more efficiently processed by algorithms and models.

GNNs are proven to possess good extrapolation ability for dynamic programming problems with graph-structured data [235], which allows a generalisation of learned graph representations to unseen examples based on learned training data. This leads to the desirable characteristics of many GNN architectures that parametrisation θ (i.e., the union of all neural network parameters in all the layers) is independent of the number of nodes in the graph. This implies that GNNs enable learning an approximation of the state-action value function on small instances and applying it directly on large instances, which is an appealing approach for COPs [25].

Graph Attention Networks (GAT) architecture [235] is a type of GNN that incorporates attention mechanisms for each node in the graph to selectively assign varying importance to different nodes based on their relevance. GAT performs a flexibly-weighted neighbourhood aggregation via the attention mechanism, thus enabling the network to capture the dependencies between nodes of the graph-structured data.

5.4 Deep Reinforcement Learning

5.4.1 Reinforcement Learning

RL is a subfield of ML that is concerned with teaching machines to make a sequence of decisions based on rewards and punishments [223]. One of the advantages of RL algorithms is their capability to learn the optimal policy from previous actions and rewards without relying on external supervision or pre-programmed rules. This makes it particularly suitable for a wide range of complex problems where the optimal solution is not immediately apparent, including natural language processing, robotics and autonomous driving.

RL encompasses a diverse set of algorithms that can be classified into *model-free* and *model-based* methods. The central concept behind the model-free RL is "trial-and-error", whereby an agent is trained to learn an optimal policy through interaction with an uncertain environment that provides reward signals. The learned *policy* guides the agent to decide which action to choose in order to react to a given situation to maximise its expected future rewards. The agent uses a feedback loop to learn from its experiences, refining its behaviour to improve the performance over time and achieve the best possible outcome. On the contrary, model-based RL focuses on constructing a model of the environment based on an agent's interactions, which enable the agent to efficiently predict the reward for an action before knowing the consequences that action lead to. As a result, good performance from any model-free method heavily rely on the accuracy of the learned model. Model-free methods, in contrast to the model-based ones, consist of acquiring knowledge about the environment's dynamics, and utilises the learned model to make decisions. In this way, model-free methods eschew dependence on an explicit model of the environment dynamics, rendering them more adaptable and capable of handling complex environments with unknown dynamics.

Model-free RL can be further classified into value-based (e.g., Q-learning, SARSA, and Deep Q-Networks) and policy-based approaches (e.g., Actor-Critic Algorithms, Policy Gradient, and Proximal Policy Optimization). The value-based algorithms focus on estimating the value function of actions or states to guide the decision-making process, whereas policy-based algorithms focus on directly parameterising the policy that maps states to actions. Particularly, the Deep Q-Networks [166] has been one of the most popular approaches amongst all value-based methods that combines the strength of supervised and reinforcement learning to tackle complicated problems. RL methods can also be categorised into actor-based and critic-based [129], or based on other ways, such as whether they operate in an online or offline manner.

5.4.2 Markov Decision Process

Mathematically, RL is a learning method mapping states to actions to maximise the cumulative reward signals. To apply the RL techniques, we need to reformulate the COP as a Markov Decision Process (MDP) to model decision-making with discrete time steps over a finite time horizon.

An MDP is a tuple $(\mathcal{S}, \mathcal{A}, P, R, \gamma)$ containing the state space, action space, transition function, reward function and discount factor. A state s_t is a representation of a specific time step t within the COP environment. In each state s_t of the state space $\mathcal{S}$, the decision-maker (agent) selects an action a_t from the set of valid actions $\mathcal{A}(s_t)$ and receives a reward signal r_t according to a reward function $R(s_t, a_t)$ as a consequence of its selected action a_t in the given state s_t. Afterwards, the agent transitions to a new state s_{t+1} that depends on the state-transition probability function $P(s_{t+1}|s_t, a_t)$,

which governs the environment dynamics and is dependent only on the preceding state and action, for all $s_t, s_{t+1} \in S$, $r_t \in R$, $a_t \in A(s_t)$. Finally, γ is the discount factor determining the agent's focus on distant future versus immediate rewards.

The agent interacts with the environment in episodes, each of which is a finite sequence formed of $s_t \rightarrow a_t \rightarrow r_t \rightarrow s_{t+1}$ until a terminal state s^+ is reached. The evolution of an episode forms a *trajectory*, during which the agent is trained through trial and error by trying out different action combinations and observing the corresponding reward. In this way, the agent learns a *policy* $\pi\left(a|s\right)$ that completely specifies its behaviour. The return, or cumulative reward, of a trajectory is measured by the discounted sum of rewards $r_{t+1} + \gamma r_{t+2} + \gamma^2 r_{t+3} + \cdots = \sum_{k=0}^{\infty} \gamma^k r_{t+k+1}$. The *state-action value function*, known as *Q-values* in short, is a measure of the expected state-action reward computed by Equation (5.2), from which the agent can derive a policy π by selecting actions greedily based on their associated Q-values at a particular state. Specifically, $Q_\pi(s_t, a_t)$ is the expected reward the agent receives by picking action a_t at a given state s_t, then following π.

$$Q_\pi(s, a) = \mathbb{E}_\pi\left[\sum_{k=0}^{\infty} \gamma^k r_{t+k+1} | s_t = s, a_t = a\right] \tag{5.2}$$

The optimal policy π^* is defined as $\pi^* = \arg\max_\pi Q_\pi(s, a)$, according to which the agent is able to select the best action at a particular state that maximises the total amount of future rewards.

5.4.3 Q-Learning

Q-learning [242] is one of the most widely employed RL algorithms to learn an optimal policy by solving a MDP. It is a model-free approach that relies on simulated samples of agent-environment iterations to estimate the state-action value function $Q(s, a)$, also known as Q-values. For any non-terminal time step t and its consecutive step t', any particular $Q_\pi(s, a)$ reflects the expected cumulative rewards obtained by the agent when starting in state s, taking action a and following policy π that defines the agent's behaviour. A policy π can be derived by selecting actions greedily with respect to their associated Q-values at a particular state. The agent's interactions with the environment generates tuples of the form (s, a, r, s') and the Q-value estimates are iteratively updated according to the following rule:

$$Q(s, a) \leftarrow (1 - \alpha_{\text{RL}}) \cdot Q(s, a) + \alpha_{\text{RL}} \cdot \left(r + \gamma \cdot \max_{a' \in \mathcal{A}(s')} Q(s', a')\right) \tag{5.3}$$

where $Q(s, a)$ is the Q-value for a particular state-action pair (s, a) for the current state s, and $\max Q(s', a')$ represents the maximum Q-value amongst all the available actions in the subsequent state s'. For the parameters, α_{RL} denotes the learning rate that balances the information from the current and future states, r denotes the immediate reward feedback from the environment after taking action a in state s, and γ is a discount factor that trades immediate versus long-term rewards. This update rule allows Q-learning to iteratively refine its Q-value estimations based on the existing state-action pairs and converge towards an optimal value. For every move the agent takes, $r + Q(s', a')$ becomes the immediate target that the agent aims to maximise based on $Q(s, a)$. Eventually, the agent learns a policy throughout the MDP that guides its behaviour towards achieving higher future rewards. The repeated application of

this rule is known to lead to the optimal action-value function and optimal policy at convergence [242].

To track the Q-learning process, a Q-table is a tabular representation to store the learned Q-values for every possible combination of state and action generated by the Q-learning algorithm (5.3). Throughout the training process, the agent explores the environment and takes actions guided by the Q-table, which in turn undergoes subsequent updates and converges to the expected future rewards for each combination of state and action. However, a notable issue associated with Q-table is its high memory and computation demands, which compel the need for alternative techniques that approximate the optimal Q-values using DNNs instead of storing them explicitly.

5.4.4 Deep Q-Learning

Many realistic problems exhibit very large state spaces, which makes the application of standard tabular Q-learning intractable. This is often the case for COPs, in which it is impossible to enumerate all possible solutions and store them in memory beyond problems of a trivial size. For such problems, DNNs are commonly employed as *function approximation* in order to estimate the $Q(s, a)$ function. It differs from classic Q-learning in the way that it can handle complex, high-dimensional state and action spaces while avoiding the memory issue of maintaining a separate Q-value for each state-action pair. Let us denote this estimate by $\hat{Q}(s, a; \theta)$, where θ is the set of learnable parameters of the neural network. Function approximation is a powerful technique to generalise across states of the environment that, while not identical, share common characteristics and will lead to similar expected rewards. By leveraging this capability, we are able to combine the strength of supervised and reinforcement learning to tackle complex problems effectively.

The DeepQ-network (DQN) algorithm proposed by Mnih et al. [166] uses this state generalisation capability together with replay buffers and target networks, and has been successfully applied for a variety of decision-making problems with large state spaces. The DQN algorithm takes the state of the environment as input and outputs a vector of Q-values corresponding to the set of available actions. The algorithm requires the training of two networks: the *main network* and the *target network*. The main network utilises the replay memory mechanism. The use of replay buffers means that (s, a, r, s') tuples encountered by the agent are stored in a memory bank, and updates to the parameters θ are performed periodically by sampling mini-batches of tuples from this buffer, instead of online. The target network mechanism stores a separate copy of the parameters θ', which are updated periodically by copying the main network parameters θ, and kept fixed at every intermediate step. The weights are gradually updated through stochastic gradient descent to minimise the discrepancy, measured by mean squared error, between the predicted $\hat{Q}(s, a; \theta)$ and the learning target $r + \gamma \max_{a'} \hat{Q}(s', a'; \theta')$. Eventually, the training concludes when the two networks converge. The main reason for keeping two separate networks is to avoid calculating the current and target values using the same network, therefore increasing the stability of the algorithm by mitigating the "moving target" problem, which is when the main network tries to predict a target network that is constantly changing.

5.5 Methodology

5.5.1 Blueprint of our Approach

Our learning-based approach to improve the operator selection in ALNS consists, at a high level, of the following two steps. Firstly, we isolate the operator choice from the considerations of the SA process in ALNS, which introduces additional noise for navigating the solution space that may obscure the operators' contributions. To achieve this, we formulate operator selection for the COP as a standalone MDP shown in Section 5.5.2, in which an agent is given a limited budget of operators, and must learn to take actions by selecting those that lead to the best solutions. Specifically, the agents are trained based on two neural network architectures, as given in Section 5.5.4. Secondly, the trained model is integrated into the ALNS framework and used to select operators in the SA process, as given in Section 5.5.5. Moreover, we discuss the comparison of action reward values between the classic RW algorithm and Q-learning in Section 5.5.3, highlighting that the Q-learning approach captures more information regarding the reward of each individual state than the classic RW approach.

The above two steps form the learning-based operator selection policy that replaces the classic RW mechanism within the ALNS framework to more accurately capture the operator selection process. This is depicted in Figure 5.4.

5.5.2 Operator Selection as a Markov Decision Process

For the MDP formulation, we are given an undirected graph $G = (V, E)$ defined by the given COP and a feature matrix $\mathbf{X}$ in which each row corresponds to a node embedding such as the coordinates of nodes, node demand, and distance between nodes. For the time horizon, each episode consists of a certain number of action tuples predefined by the budget B. We formulate the MDP as a tuple $(\mathcal{S}, \mathcal{A}, P, R, \gamma)$ with details below.

- *States* $\mathcal{S}$: each state S_t is a tuple $(G, \mathbf{X}, J_t, C_t, \varphi_t, b_t)$, wherein the graph G and feature matrix $\mathbf{X}$ remain static. J_t is the set of *tours* that start and end at the depot, forming the solution at time t. The removal list $C_t = V \setminus J_t$ holds all the q nodes temporarily removed from the solution. The destroy scale q is a non-negative integer not exceeding $|V|$. φ_t indicates the *phase*: whether the set of destroy or repair operators is eligible to be applied at step t. Finally, b_t indicates the operator pair budget available to the agent.

- *Actions* $\mathcal{A}$ involve the selection of an operator $o_t^{+/-}$, with those available defined as $\mathcal{D}$ if $\varphi_t = -$ during the destroy phase, and $\mathcal{R}$ if $\varphi_t = +$ during the repair phase.

- *Transitions* P apply the selected operator $o_t^{+/-}$ to the current solution. Applying a destroy operator removes q nodes from the tours J_t and places them in the removal list C_t. Using a repair operator reinserts the nodes from C_t into J_t, leaving C_t empty and the solution J_t complete, and decreases the operator pair budget by 1. Transitions are stochastic due to the inherent randomness of the operators.

- *Rewards* R are provided once the operator budget is exhausted (the last action tuple accomplishes) and the improvement in solution quality can be assessed via an objective function F. Concretely, $R_t = F(S_t) - F(S_0)$ if $b_t = 0$, and equals to 0 otherwise.

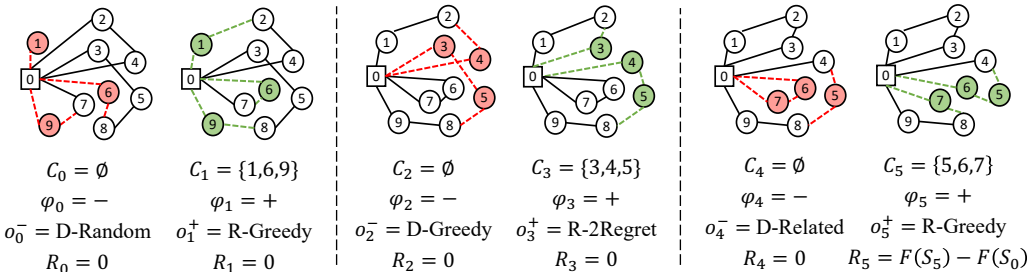


Figure 5.2: Illustration of an MDP episode with budget $b = 3$ and destroy scale $q = 3$ with 3 selected action pairs by the agent before reaching the termination state.

On a generalised level without going into the detailed feature of a particular VRP variant, we use Figure 5.2 to showcase a visualisation of an episode, demonstrating the sequence of actions involved in the ALNS perturbation process. The perturbation process can be treated as an episodic task in the context of RL due to its sub-sequences that comprise the agent-environment interaction. This natural episodic structure allows the agent to select valid MDP action tuples (destroy, repair) as a trajectory until reaching the terminal state.

Assume an episode with an action budget $B = 3$ and a destroy scale $q = 3$, where the agent iteratively selects three action tuples before budget exhaustion. The iterative process begins with the deconstructing phase $\varphi_t = -$, in which the agent assesses the feasible collection of destroy operators at the current state and decides which one to choose. In the subsequent phase $\varphi_{t+1} = +$, the agent selects a repair operator that is feasible to use for the current state. Depending on the specific phase of the operator, the agent performs either the removal or insertion of 3 nodes into the graph. We assume that nodes within a tour are sequentially linked based on their positions so that all tours start and end at the depot. The action spaces contain 3 destroy operators $\mathcal{D} = \{\text{Random, Greedy, Related}\}$ and 2 repairs $\mathcal{R} = \{\text{Greedy, 2Regret}\}$.

The graph showcases the decision-making process starting from the left-most initial graph, where the agent begins at state S_0 with customer pool $C_0 = \emptyset$ and routes $J_0 = \{[1], [2, 4], [3, 5, 8, 6], [7, 9]\}$. In the first graph, the agent selects the destroy operator o_0^- = Random at the deconstructing phase with $\varphi_0 = -$. At this state, nodes 1, 6 and 9 together with their linking arcs are removed (marked as red) and placed into the temporary customer pool. Subsequently, the agent selects the repair operator o_1^+ = Greedy at the reconstructing phase $\varphi_1 = +$ by reinserting the removed nodes back into the tours (marked in green) using the instruction specified by o_1^+ and fixing the linking arcs to form a complete tour, thus transiting to a new state S_2. By completing one destroy-repair iteration, the number of tours goes from 4 to 3. The episode continues with the agent selecting a second action tuple (o_2^- = Greedy, o_3^+ = Regret) and third action tuple (o_4^- = Related, o_5^+ = Greedy) until the budget is exhausted and the terminal state S_5 with routes $J_5 = \{[1, 2, 3], [4, 5, 6], [7, 8, 9]\}$ is reached. Finally, it receives a reward $R_5 = F(S_5) - F(S_0)$ proportional to the improvement in solution quality, where all the non-terminal rewards $R_0 = \cdots = R_4 = 0$.

5.5.3 Comparing Q-learning to Roulette Wheel update

Q-learning estimates the state-action value function $Q(s, a)$, from which a policy π can be derived by acting greedily with respect to it. The agent's interactions with the environment generate (s, a, r, s') tuples, and its estimates are updated according to

the one-step Q-learning update rule (Equation 5.3). By comparing it to the classic RW update rule (Equation 2.1), we notice that they both utilise a weighted factor to balance two terms representing the historical and current estimates of algorithm performance.

In the classic RW framework, the weight of each operator can be interpreted as the operator's ability to improve the solution quality for the current segment. A segment refers to a consecutive number of destroy-repair iterations during the solution process. These weights are periodically updated using a weighted balance of the operator's historical and in-time performance at the end of each segment. As the segment progresses, each operator's selection probability is directly proportional to its weight, leading to the more frequent selection of operators with a higher weight. As a result, the ALNS adaptive layer reflects each operator's capability on a "macro-level" by sampling operators according to the fix probabilities within a segment, as mentioned in [117].

Upon comparing the two updating formulas, we observe that the Q-learning approach is conditioned on each individual state. This implies that during a single destroy or repair iteration, the algorithm captures more specific information regarding the efficiency of the applied operator on that given state. In contrast, the RW approach simply averages the gains obtained by the operators within the segment, and can be interpreted as a rough approximation of the Q-learning update, tracking the average reward from all the possible trajectories where an operator is included regardless of the state-based performance. Consequently, the Q-learning update formula is in a better position to accurately guide the future action selection process by tracking each operator's performance at the individual state level. The RW update formula can be interpreted as a rough approximation of the Q-learning update, providing the average reward from all the possible trajectories where an operator is included regardless of the state-based performance. From an intuitive standpoint, the RW algorithm's utilisation of state-based information enables us to obtain operator selection policies that perform at least as well as its Q-learning counterpart. Hence we can consider the RW formula as an approximation of the Q-learning update. This nevertheless implies that Q-learning demands higher sample complexity. However, we found this was not a significant concern in practice, as a relatively small number of training steps suffices to achieve a good policy.

5.5.4 Learning an Operator Selection Policy

In this work, we consider two possible neural network architectures. Firstly, we use a MLP formed of layers that apply a linear transformation of the inputs followed by a non-linear activation function as described in Section 5.3.1. Despite the simplicity of such a model, MLPs have gained prominence in the field of ML to be known as universal function approximators, meaning that they are able to approximate any continuous function to an arbitrary degree of accuracy, given that an adequate number of hidden layers and suitable activation functions are provided.

Another architecture we consider is GNN, which are means of performing graph representation learning that is explicitly designed to operate on graph-structured data. Figure 5.3 visualises this approximation process. We start by extracting the list of nodes and structural feature values associated with each node in the graph. The features are then encoded to create an initial graph representation (embedding) that can be processed by the neural network. In our case, the graph representation takes the form of a feature matrix formulated by compressing the feature values from all states within the same batch, such that each state is represented by a row in the matrix. Each layer within the neural network propagates information by performing message

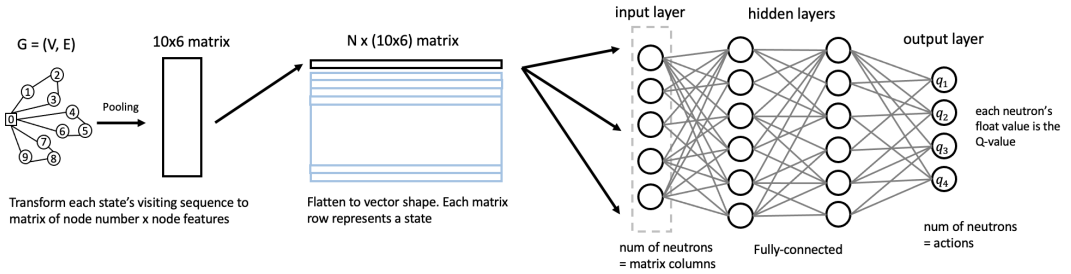


Figure 5.3: Convert visiting sequence graph into node features that match the observation space; Neural network maps discrete states to (action, Q-value) pairs.

passing, which updates the graph representation by iteratively aggregating node information using the neighbour nodes, so that each node embedding contains structural and feature-based information of the whole graph. The graph representation formed by each layer is passed to the consecutive layer to perform the same propagation process. This allows the network to identify higher-level dependencies in the graph and thus to be able to capture complex patterns and structures that enables the model to more accurately predict on unseen graphs. In practice, this advantage allows training on smaller instances and extrapolation to graphs of larger sizes, rendering it an attractive feature of GNN even with a heavier computational overhead compared to MLP. For applications and real-life scenarios, evaluating the GNN for a solution may take considerably more time than the rather simple operator choice of the classical ALNS. Nevertheless, the effort can still be negligible, as offline re-training is required sporadically only when data distribution changes significantly, and meanwhile the trained model can be applied to multiple rounds of online operations.

5.5.5 Integration of RL with ALNS

As mentioned in our algorithmic blueprint, during the first stage of our learning-based approach, we isolate the operator selection process from the ALNS framework and convert it into a MDP for the training of the agent. The second stage involves integrating the fully-trained model into the ALNS algorithm to search for good-quality solutions. An overview of changes in the operator selection compared to the one inside the clas-

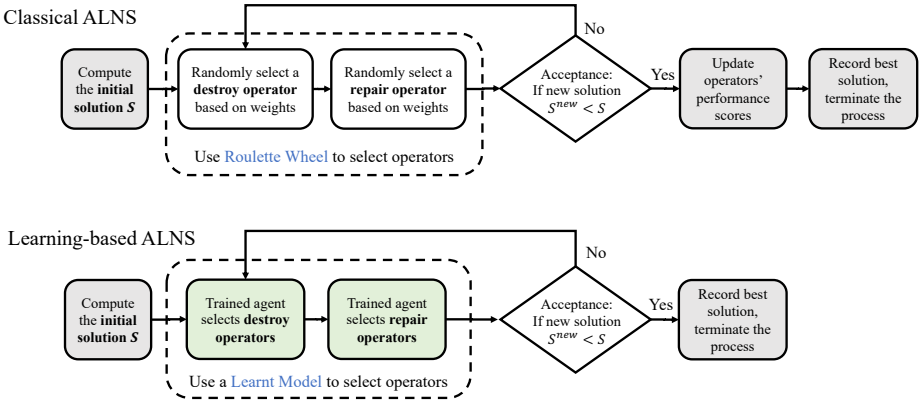


Figure 5.4: Comparison of the operator selection update between classic ALNS and hybrid RL-ALNS frameworks.

sical ALNS are given in Figure 5.4, where we have replaced the classical perturbation process with a learning-based decision-making model. That is, instead of using the classical RW approach to iteratively select an operator pair while taking track of their performance scores, we employ an RL agent to make decisions on the choice of an operator pair based on a learned policy formed during training.

In line with the previous discussion, the resulting learned policy acts greedily with respect to the learned state-action value function, always choosing the action with the highest expected cumulative reward. However, integrating this greedy-based policy directly into the classic ALNS framework might prove problematic, as its inherent greediness could potentially cause the search to get trapped in suboptimal solutions.

To address this concern, we introduce a mechanism to obtain a *probabilistic* policy. This is achieved by using a softmax function as shown in Equation (5.4), in which the temperature τ allows adjusting the level of greediness exhibited by the policy. Specifically, the probabilities assigned to different actions become more uniform when $\tau \to \infty$, resulting in a more exploratory behaviour. Conversely, the action with the highest expected reward dominates the probability distribution with its probability approaching 1 when $\tau \to 0$, allowing for greater exploitation of a certain action.

$$\pi_\tau(a|s) = \frac{\exp(Q(s,a)/\tau)}{\sum_{a' \in \mathcal{A}(s)} \exp(Q(s,a')/\tau)} \tag{5.4}$$

5.5.6 ALNS Operator Classification

In the literature, operators are carefully tailored to fit different problem structures and features. Despite the large variety of operator designs, the mechanisms behind them are surprisingly similar to the first version of ALNS proposed by Ropke and Pisinger [201]. We conducted a thorough analysis of operators in the literature and have identified the following 3 classes: *random-based* destroy that randomly removes q nodes according to specific availability criteria, *greedy-based* destroy that removes the top-ranking q nodes with respect to a particular measure, and *related-based* destroy as an extension of Shaw's destroy [212] that removes the q most similar nodes according to a certain proximity value. Variations can include perturbations or using problem-specific features including distance, time, cost, workload, demand level, inventory level, removal gain, historical information, etc.

Barring a few *random-based* operators, almost all current repair operator designs are related to *greedy-based* mechanisms that insert each node at the position with the smallest cost. Variations can include a pre-sorting that changes the order of node insertions according to certain criteria, including global minimum insertion or smallest regret value. Others can have a noise factor that perturbs the insertion cost values, or use restrictions based on historical information.

For the portfolio design, we identified 12 popular destroy operators from the ALNS literature that span the representative categories, including: 2 random-based variations *random node destroy*, and *random route destroy*, 3 greedy-based variations *worst distance destroy*, *greedy route longest destroy*, and *greedy route shortest destroy*, and 7 related-based variations *proximity distance destroy*, *node neighbourhood destroy*, *route neighbourhood destroy*, *cluster destroy*, *zone destroy*, *pair destroy*, and *historical pair removal*. For the repair operator portfolio, we include the *greedy repair*, and *regret-2 repair*. A summary of the destroy and repair operators is given in Table A.3.

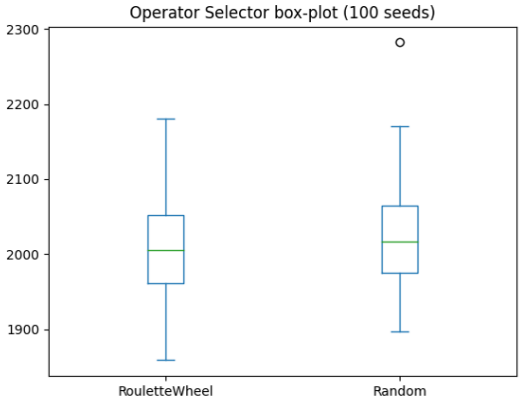


Figure 5.5: Performance comparison between the ALNS with and without an adaptive layer.

5.6 Classic ALNS Experiments

5.6.1 Efficiency of ALNS Adaptive Layer

To analyse how the adaptive layer within the RW operator selection mechanism benefits the whole ALNS framework, we have designed two versions of the ALNS algorithm with and without an adaptive layer. In the latter case, we substituted the adaptive layer with a uniform random sampling selection method for all operators to carry the same selection probability throughout the search process. Both algorithm versions implement 6 destroy and 2 repair operators and are assessed on the Solomon C instances for the CVRP. The parameter values follow Table 5.1.

The two box plots in Figure 5.5 illustrate the performance difference between the two ALNS algorithms, where the box plot on the left is generated using the classic ALNS with an adaptive RW that updates the selection probability of each operator based on historical performance and the right has the adaptive layer replaced by a uniform random sampling process with all operators sharing the same selection probability. A comparison of the two box plot distributions displays a similar performance level between the two operator selection methods, indicating that the ALNS adaptive layer does not significantly improve the algorithmic performance compared to a random sampling selector. This resonates with the conclusion by Turkeš et al. [232] about the limited efficiency of the embedded adaptive layer.

5.6.2 Manual Design of Operator Portfolio

We use the following experiments to demonstrate how the manual design of an operator configuration can cause a performance difference within the classic ALNS framework. We highlight the significance and necessity of conducting thorough examinations of the number and type of operators to include in the configuration, which can be time-consuming yet essential for optimising the algorithmic efficiency. We apply the classic ALNS with RW based on the 12 destroy operators from the existing literature further explained in Section 5.5.6.

Primarily, we demonstrate that the efficacy of the algorithm can significantly vary due to the involvement of different operators when the number of operators within the configuration remains consistent. Figure 5.6 exhibits the best solution obtained in

100 rounds, where each group utilised a specific operator configuration and a destroy scale of $q = 5$. Each configuration consists of a total of 4 destroy operators, comprising the 3 generic destroy operators *random node destroy*, *worst distance destroy* and *proximity distance destroy*, and one additional operator selected from a range of 9 options, including *node neighbourhood destroy*, *route neighbourhood destroy*, *zone destroy*, *pair destroy*, *cluster destroy*, *greedy route longest destroy*, *greedy route shortest removal*, *random route destroy*, and *historical pair destroy* in the exact order.

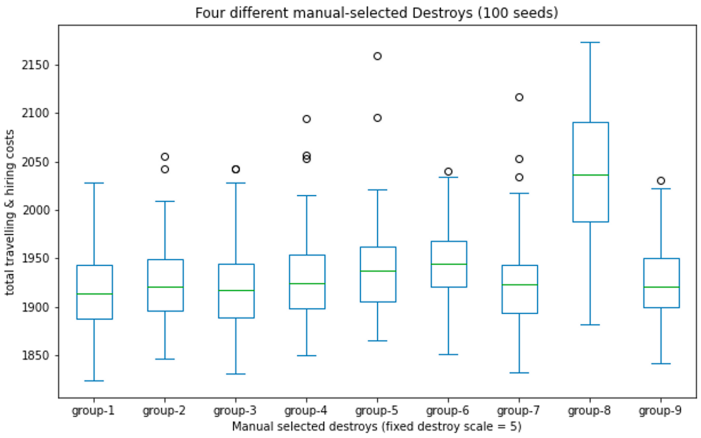


Figure 5.6: Destroy operator configuration analysis: manually selecting different operator groups to form operator portfolio

Upon comparing the distribution of the 9 box plots, we observe that *node neighbourhood destroy* (group 1) exhibits the lowest average objective value hence the highest level of performance, whereas *random route destroy* (group 8) has the poorest average performance with high variation. This observation may lead to the development of an iterative inclusion (or exclusion) strategy that iteratively includes or excludes each individual operator based on its performance, which has been applied in a few works [104, 180, 244]. To determine a final operator portfolio using this strategy, one way is to iteratively increase the configuration size while incorporating the next unapplied destroy operator from the sequence that contributes to the best joint performance until identifying the combination of operators that any additional operator will only bring down the performance. However, handcrafting a configuration with such a greedy bin-packing strategy may not necessarily help to identify the optimal configuration during the manual design process since the initial sequence in which operators are ordered can significantly influence the outcome.

We employ a secondary set of experiments to demonstrate the aforementioned point that suboptimal configuration can be formed easily. Figure 5.7 is generated using the same experiment setting with the only difference being the component of operator configuration, each of which contains 6 operators that are partially different. In this experiment, all configurations again contain the 3 generic destroy operators (*random node destroy*, *worst distance destroy* and *proximity distance destroy*), together with 3 manually selected destroy operators from the 12 mentioned previously. Specifically, the first configuration (group 1) contains *node neighbourhood destroy*, *cluster destroy* and *greedy route longest destroy*, the second configuration (group 2) includes *node neighbourhood destroy*, *cluster destroy* and *route neighbourhood destroy*, the third configuration (group 3) includes *route neighbourhood destroy*, *cluster destroy* and *greedy route longest*

destroy, and the final configuration (group 4) contains *route neighbourhood destroy*, *random route destroy* and *greedy route longest destroy*.

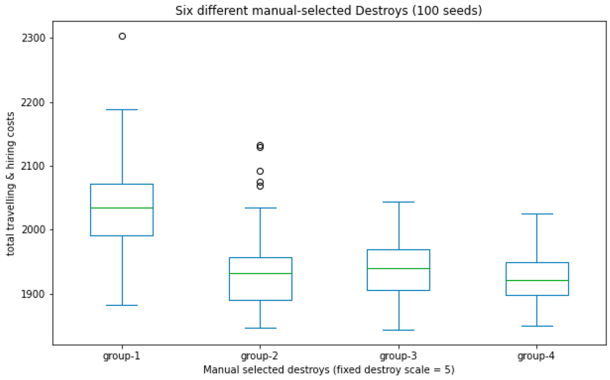


Figure 5.7: Destroy operator configuration analysis: manually selecting different operator groups to compare configuration performance. Lower the better.

By comparing the box plot distribution between the first and second configurations, we may immediately assume that *greedy route longest destroy* is the inadequate destroy operator that brings down the overall performance of the first configuration. Consequently, it becomes imperative to exclude this operator from the forthcoming configuration design process. Nevertheless, the identical "underperforming" *greedy route longest destroy* is also present in the third and fourth configurations, both of which exemplify a considerably more satisfactory performance than the first one regarding average objective values and confidence interval width. Conversely, in both cases, the absence of the *node neighbourhood destroy* operator implies that its joint performance with the *greedy route longest destroy* operator may be unsuitable for the given instance. But this conclusion is not trivial as we were initially led astray by the superior performance given by *node neighbourhood destroy* in Figure 5.6. Furthermore, while *random route destroy* displays the least favourable outcome in Figure 5.6, its inclusion within the final configuration in Figure 5.7 contributes to the further enhancement of collaborative performance. The above observations undoubtedly cast a challenge on the manual design process since it necessitates the importance of strong domain knowledge on the operators as well as the ALNS framework and yet, most often, it still requires considerable amount of trial-and-error and development time. Therefore, the best configuration can remain unclear unless a thorough examination of the operator combinations is conducted in the form of a factorial analysis.

We use the following experiment presented in Figure 5.8 to highlight the significant manual effort required to fine-tune the operator configuration in order to achieve the desired level of ALNS algorithm performance. We conducted a complete factorial analysis of all possible combinations of operators given a specific budget ranging from 1 to 12. Each box plot is conducted by the best-performing configuration selected from a total of $\binom{12}{k}$ possible combinations that share a budget of k operators. The destroy operators portfolio for each configuration size is given in Table A.4. The overall result displays a "U-shape" performance curve, signifying the importance of conducting a thorough analysis to determine the optimal number of operators and the best combinations in order to guarantee a good ALNS performance. This finding resonates with the indications presented in Turkeš et al. [232] and Kallestad et al. [117], that a huge amount of

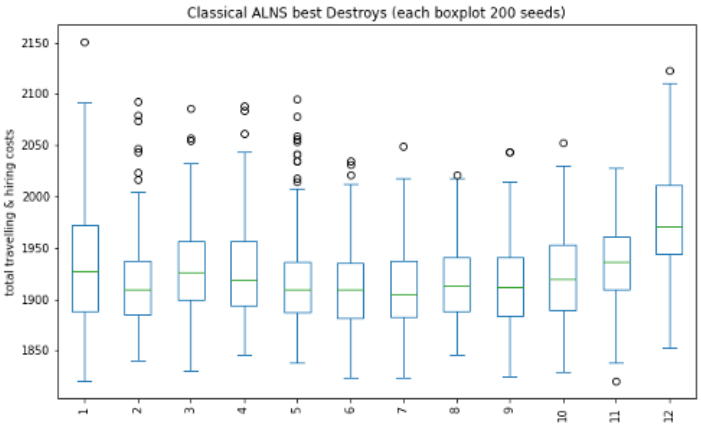


Figure 5.8: Destroy operator configuration analysis with 200 sample points each as the best-found solution from an ALNS round. The best configuration is selected from $\binom{12}{k}$ possible combinations. Lower is better.

preliminary experiments is unavoidable during the metaheuristic design process.

5.7 Preliminary Settings

To mitigate the classic ALNS drawbacks demonstrated by the sets of experiments in the previous section, we introduce the learning-based ALNS and design experiments to illustrate its robustness and efficiency.

5.7.1 Experimental Setup

Problem Settings

In this work, we consider the Capacitated Vehicle Routing Problem (CVRP) with a single depot, a group of customer nodes and a number of homogeneous vehicles each visiting an individual group of customer nodes. The capacity restriction applies to the total carrying load of vehicles. Each customer node can only be visited once. We use the R, C and RC instances (random, clustered, and mixed random-clustered nodes) of the classic Solomon dataset [218], each containing a depot and up to 100 customer nodes. We assign the vehicle capacity to be 200, and adjust it proportionally if scaling down the instance to fewer customers. The objective function value is computed as the sum of travelling and vehicle hiring costs from the tours. The initial tours are randomly generated with length proportional to the defined vehicle capacity (therefore the instance size). This is so that the performance gaps between the different operator selector strategies are emphasised in the evaluation. Nevertheless, the methodology is applicable to any other means of initial solution generation. We adapt the ALNS parameter values given in Table 5.1.

Operator Selection Approaches

The proposed DQN agent is compared to the following approaches. As a baseline, we consider a uniform Random sampling (RAN) of operators. We also compare against the classic RW (CRW), which can only be used within ALNS since it requires information about the SA outcomes and search progress. To make the RW mechanism applicable

Table 5.1: ALNS parameter values summary

Symbols	Parameter Names	Values
q	destroy scale	10%
λ	local search threshold	0%
δ_1	global best solution score	4
δ_2	local best solution score	2
δ_3	accepted worse solution score	0
l_K	segment length	200
α_{RW}	reaction factor	0.5
c_τ	temperature cooling coefficient	0.998
T	initial temperature	100
$iter$	non-changing iteration stop criteria	1000

in the MDP setting, we make the adaptations described in Section 5.7.2 to obtain a method we call Learned RW (LRW).

Training and Evaluation Methodology

For each instance, we generate 3 distinct sets $\mathcal{J}^{\text{train}}, \mathcal{J}^{\text{validate}}, \mathcal{J}^{\text{test}}$ of 128 randomly initialised tours each. $\mathcal{J}^{\text{train}}$ is used by DQN and LRW for model training. $\mathcal{J}^{\text{validate}}$ is used for hyperparameter tuning and model selection. Finally, $\mathcal{J}^{\text{test}}$ is used to perform the final evaluation and obtain the reported results. There are two evaluation "modes": MDP-compatible agents can be evaluated in a standalone fashion given an operator budget (CRW is excluded), while all operators (including CRW) can be evaluated in the ALNS framework. Training and evaluation is repeated across 10 random seeds for all agents, which are used to compute confidence intervals.

DQN Architectures and Inputs

For the DQN, we consider MLP and GNN representations. The MLP has 256 units in the first hidden layer, with the subsequent layers having half the size. As a GNN, we opt for the GAT, which allows for flexible aggregation of neighbour features. We use 3 layers and a dimension of node embeddings equal to 32. Both use a learning rate of $\alpha_{\text{RL}} = 0.0005$ and are trained for $15 \cdot 10^3$ and $25 \cdot 10^3$ steps, respectively. To obtain the inputs, we construct vectors $\tilde{\mathbf{x}}_t^i$ that concatenate the static instance-specific features $\mathbf{x}^i$ with time-dependant relevant information such as whether the node i is routed in a tour and the number of tours in J_t. For the MLP, we stack the vectors in a matrix $\widetilde{\mathbf{X}}_t$ as inputs, while for the GNN the node features are provided directly. Unless otherwise stated, we use a softmax temperature $\tau = 0.01$.

5.7.2 Converting Roulette Wheel into Learning-based

Even though both the CRW and learning-based DQN models are operator selection approaches, their different underlying frameworks forbid any direct comparison between them. This distinction arises from the fact that DQN undergoes both the processes of MDP training and ALNS evaluation, whereas CRW only goes through the latter without the need to be pre-trained. We therefore implement an intermediate learning-based RW (LRW) model that undergoes both stages that is first trained within the MDP framework while using the RW updating rule (2.1) before evaluating within the

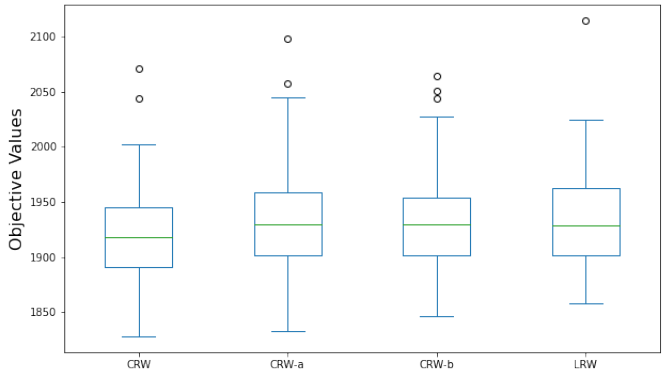


Figure 5.9: Converting CRW to LRW to fit into the MDP framework necessities two alterations: (*a*) re-definition of reward term, and (*b*) adjustment of reward frequency.

ALNS framework. In this way, we are able to draw a direct performance comparison between the learning-based models with DQN and RW updates.

The functional-wise transition from CRW to LRW requires a twofold alteration: Firstly, in Equation (2.1), we replace the manually-defined operator scores ψ computed from the discretised δ with the continuous objective value F. Secondly, we adjust the reward feedback frequency from every operator pair in CRW to every episode in the MDP framework for LRW, where each episode consists of multiple operator pairs. This frequency of feedback is adapted in the MDP framework in the way that the reward is given to the whole sequence of operators applied in the episode, necessitating their collaborative performance on the incumbent quality.

To demonstrate that the two changes will not place LRW at a disadvantageous position, we conduct Figure 5.9 to showcase the performance difference between CRW and LRW in the classic ALNS framework using 150 seeds for each box plot.

When implementing only the first alternation with the operator score, its immediate impact on the operator selection performance quality is not discernible. This observation is substantiated by the distributions of CRW and CRW-a groups, which exhibit a statistical significance of $p = 0.082$, consequently supporting the null hypothesis that the two groups follow the same distribution. The second alternation of reducing the reward feedback frequency, intuitively, can exert a considerable negative influence on the performance as a substantial amount of information used to adapt operator selection probability is removed from the system. Indeed, the comparison between CRW and LRW groups has a statistical significance of $p = 0.013$, resulting in the rejection of the null hypothesis. Nevertheless, the comparison of CRW-a versus CRW-b gives $p = 0.709$ that fails to reject the same null hypothesis. Besides, preliminary experimental results suggested that with both alternations included, the average performance difference between the LRW and CRW groups is within 0.6% when applied in ALNS without any prior training. This indicates that the LRW performance, even though being slightly worse off, remains approximately on the same level as CRW.

5.7.3 Feature Representation

We first construct the graph representation by defining the node and arc sets and associate them with features. Based on the visiting tours in each solution, the following

Table 5.2: Node features descriptions for solutions generated using CVRP instances

Feature	Type	Dynamicity	Description
x, y	node	static	Geographical coordination for each customer node
demand	node	static	Demand quantity associated with each customer node
tour_id	node	dynamic	The index of each tour represented in integer number
pos_id	node	dynamic	The sequential index of each node within each tour as an integer
adj_dis	edge	dynamic	The Euclidean distance from each node to its subsequent neighbour in the tour
inst_size	global	static	The number of customer nodes in the current instance

6 node features have been selected to represent the graph: the coordinate x and y for each customer node, the individual customer demand, the tour and position index of each node, the Euclidean distance from the subsequent neighbour node, and the instance size.

As a proof-of-concept, we adopt a supervised learning experiment below that utilises the list of node features in Table 5.2 to predict the corresponding objective value for any incumbents during the search process. This is to ensure that the input feature matrix is sufficient in representing the state-based solution quality to train the neural network. Due to the variations observed in the best-found solutions across multiple ALNS rounds, the objective values generated during the search process are normalised between $[0, 1]$ to better demonstrate the magnitude. For the supervised learning parameters, we employ 5 neural network layers, with the first layer containing 1028 neurons and each consecutive layer having exactly half the number of neurons compared to the previous. The batch size as a hyperparameter defines the number of training examples utilised in one learning iteration. Therefore at each training step, the model sees a random sample batch of solutions with its size defined by the hyperparameter. For the experiment, we set the batch size to be 16, the learning rate to be 0.001, and use *MaxAbsScaler* as our data pre-processing scaler. A total of 100 samples have been used to train and evaluate the supervised learning model, where the training and evaluation sets are split into 80 and 20 correspondingly with no data overlapping.

Since the length of the search process can vary, we partition the search process into percentiles representing the time step of the search process, so that "50%" returns the best-found solution at exactly half of the search before the termination. This conversion is due to the search trajectory of each round based on different instances taking slightly different length from each other. Moreover, since the best-found solution for each ALNS round can vary, we set the initial solution's objective value to be 1 and use it to normalise the remaining best-found solutions objective values.

For each solution at a certain percentile, we store the current best-found solution represented by a set of visiting sequences and compute its normalised objective value. Afterwards, we extract the feature matrix from the solution as the neural network input and let it predict the corresponding solution's normalised objective value. Using a total 100 rounds as an example, we first train the neural network model using 80 ALNS rounds. Afterwards, we run the remaining 20 ALNS rounds for evaluation, recording the solutions at specific percentiles and letting the trained model predict their objective values based on the feature matrices. The predicted normalised objective value of each certain percentile is averaged from the 20 ALNS rounds.

Table 5.3 demonstrates that different batch sizes can influence the prediction accu-

Table 5.3: Predicting 25% percentiles of solution quality

Batch Size	**25%**	**50%**	**75%**	**100%**
actual	0.933	0.867	0.491	0.298
8	0.836	0.862	0.463	0.331
16	0.856	0.828	0.472	0.334
32	0.854	0.831	0.449	0.338

racy. It showcases how an increase of batch sizes influences the sensitivity and highlights that a batch size of 16 gives accurate prediction in most of the times to the actual normalised objective value.

Table 5.4: Comparison of different encodings in the model's prediction power

Index Type	**25%**	**50%**	**75%**	**100%**
actual	0.933	0.867	0.491	0.298
one-hot	0.848	0.699	0.464	0.346
duplication	0.859	0.831	0.476	0.323

We have also explored alternative node feature encoding techniques and investigated their effects on prediction accuracy. One-hot index encoding converts the categorical variables, such as the node's tour and position index into binary values. The advantage of such encryption is to avoid uneven scales that oscillate the learning process and allow the model to learn more effectively. Conversely, it significantly increases the size of the node feature matrix, which can burden the network training process.

Another encoding technique we tested is feature duplication. Since the different categorical variables are of different scales, we apply the feature duplication encoding that replicates those features with small numerical value multiple times inside the feature matrix. The motivation is to amplify the prominence of those low numerical values by increasing their appearance frequencies. In this way, the under-represented features due to low numerical magnitudes can still take up an adequate portion of the network input, so that we can enhance the network's ability to efficiently capture any critical information and meaningful patterns otherwise neglected during the training process.

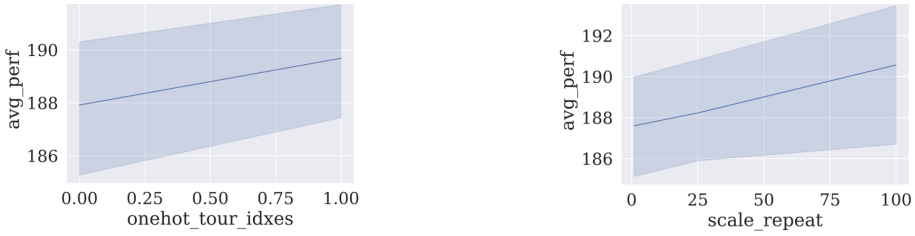


Figure 5.10: Comparison of different encoding techniques during the search

We assume that the general model settings remain the same as in the previous experiment, and the only difference is in the feature encoding technique. The results presented in Table 5.4 indicate that feature duplication encoding yields prediction that more closely aligned with actual values, thus demonstrating superior predictive capability compared to one-hot encoding. Moreover, when both methods are examined within the MDP training context from the same initial solution, feature duplication encoding

exhibits greater capacity to guide the agent towards accumulating a higher amount of rewards during training. Figure 5.10 illustrates that feature duplication encoding provides a better performance than one-hot encoding in terms of cumulative rewards throughout the search. As a result, we will adapt feature duplication encoding for the following experiments.

Finally, we present the average performance achieved by the supervised learning model employing feature duplication encoding. Figure 5.11 shows the normalised objective values predicted by the supervised learning model trained using node features from 80 rounds of individual ALNS search. The actual normalised objective values are computed using the travelling and vehicle hiring costs and are shown as the orange line. The model prediction of the best-found solutions with feature duplication encoding is depicted by the blue line, which aligns closely with the actual objective value in orange. This suggests that the given graph-based node features' prediction power can accurately predict the actual objective value and thus can sufficiently represent the quality of any unknown solution at a particular ALNS iteration. There are slightly inaccurate predictions for bad-quality solutions at the beginning of the search due to the quick improvement of solution quality at the beginning, which causes inaccuracy in network prediction. However, this can be easily overcome by increasing the training budget on solutions with similar objective values.

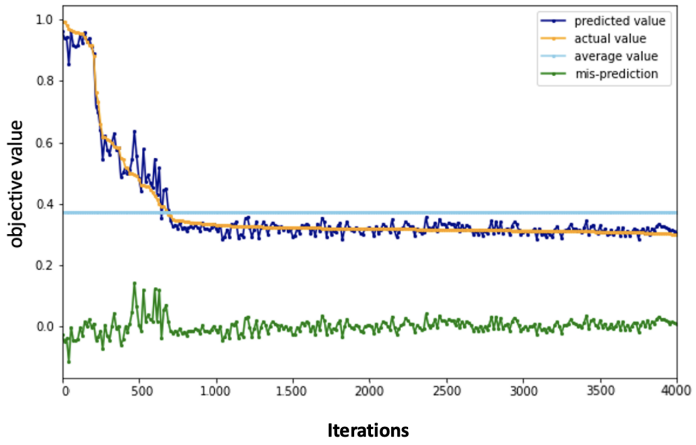


Figure 5.11: Sample supervised learning progress curve from 80 ALNS rounds on predicting the real objective value during the search process based on node features, a batch size of 16 and feature duplication encoding.

5.8 Experiments

5.8.1 Evaluating Agents within the MDP Framework

In this experiment, we compare the cumulative rewards gained by the DQN with an MLP representation, LRW and RAN agents on the test set $\mathcal{J}^{\text{test}}$ after undergoing training. To make the training and evaluation processes less computationally intensive, we use the first 20 customer nodes and the depot from the Solomon R, C and RC instances [218]. We define operator portfolios of different sizes ranging from 2 to 12 by incrementally incorporating the destroy operators based on the combinations previously tested in Section 5.6.2, together with the 2 repair operators. The operator portfolios

ranging from size 2 to 12 in this comparison are the same as those identified as the best ones in the factorial analysis within the classic ALNS framework in Table A.4. The destroy scale is fixed as $q = 4$ and the operator pair budget is $b = 10$, yielding MDP episodes of length 20.

The following set of experiments aims to compare the performances of different operator selection agents DQN, LRW and RAN for the training process given a predefined number of operators. Table 5.5 shows that the DQN agent is able to outperform competing methods as the size of $\mathcal{D}$ grows. When the destroy portfolio is smaller than 3, the DQN agent performs slightly worse than the LRW and RAN agents due to the limited action space in which the impact of the selected actions is difficult to distinguish from chance. The DQN agent begins to outperform the others when the operator size grows larger than 3, which is a threshold since individual operator's selection probability decreases and choosing good operators by chance becomes much more difficult. The DQN agent also yields smaller confidence intervals and hence a steadier performance. As expected, the RAN agent fails to show a clear increase in rewards as the portfolio size grows. The LRW agent, although showing a certain improvement, performs significantly worse than the DQN.

Table 5.5: MDP evaluation results: cumulative rewards gained by the DQN, RAN and LRW agents with destroy portfolios $\mathcal{D}$ of different sizes. Higher is better.

	C-instance			R-instance			RC-instance		
$\lvert\mathcal{D}\rvert$	**DQN**	**RAN**	**LRW**	**DQN**	**RAN**	**LRW**	**DQN**	**RAN**	**LRW**
2	232.2±2.9	**252.2±3.6**	252.1±4.5	216.3±5	**222.9±3.7**	222.8±4.4	240.2±7.5	**259.4±3.4**	258.9±5.6
3	228.6±9.0	221.7±3.5	**245.9±6.0**	212.9±5.6	208.4±4.7	**215.1±3.8**	**236.1±6.1**	224.0±3.4	230.8±7.6
4	232.5±5.1	221.2±6.4	**240.3±5.8**	**220.2±3.1**	206.9±6.0	216.2±4.1	**241.2±5.8**	222.4±5.1	239.5±4.8
5	**328.8±2.4**	258.5±5.5	293.3±8.0	**330.9±4.2**	246.8±3.7	273.9±6.1	**331.1±2.9**	260.9±4.1	272.5±7.6
6	**329.9±4.2**	231.7±5.8	284.9±8.9	**329.5±3.3**	217.5±5.5	272.5±14.9	**329.5±2.6**	239.5±5.4	261.6±5.1
7	**328.5±2.8**	246.7±4.4	282.4±11.0	**330.5±3.8**	236.1±4.8	253.7±8.1	**331.7±2.9**	247.6±3.8	264.6±6.4
8	**329.5±3.9**	235.6±5.2	281.6±10.6	**330.9±3.6**	220.4±2.0	264.5±7.0	**333.7±3.3**	243.7±4.5	254.7±4.8
9	**330.7±3.1**	225.8±5.7	274.6±12.2	**328.9±4.6**	212.6±3.8	260.3±5.7	**332.4±3.7**	226.7±7.0	250.3±8.4
10	**330.2±4.5**	224.4±4.7	276.2±9.3	**330.3±3.3**	206.7±4.5	258.6±7.0	**331.0±2.6**	224.3±5.8	252.6±15.3
11	**330.3±2.9**	222.0±4.8	275.9±6.4	**327.2±8.0**	210.4±4.6	259.1±9.0	**326.1±15.2**	223.3±6.6	245.7±8.3
12	**361.8±0.2**	246.9±5.5	323.7±7.0	**404.2±0.6**	246.8±7.0	313.1±17.7	**354.9±4.1**	258.5±4.1	283.9±13.0
mean	**305.7±3.7**	235.2±5.0	275.5±8.2	**305.6±4.6**	221.4±4.6	255.4±8.0	**308.0±5.2**	239.1±4.8	255.9±7.9

Two performance jumps in the DQN and LRW agents were observed: from size 4 to 5 and 11 to 12 for all 3 instance types, the reason for which is the inclusion of a more efficient operator in the portfolio that suits the behaviour of a greedy-based agent. Moreover, the combination of several destroy operators performs better than their single counterpart in isolation. For instance, when increasing from size 4 to 5, the operator portfolio incorporates *route neighbourhood destroy* and *random route destroy* while excluding *node neighbourhood destroy*. As evidenced in Figure 5.6, this updated portfolio with *route neighbourhood destroy* and *random route destroy* substantiates its ability to attain promising performances.

In conclusion, the training results demonstrate that even with an optimal operator configuration design, the DQN agent can improve further and attain additional performance gains compared to the LRW method. This is due to its better utilisation of state-based information, which allows the agent to more accurately analyse the influence of individual operators on each solution.

5.8.2 Evaluating Agents within the ALNS Framework

Using the same experimental setup as above, we now apply the operator selection approaches within ALNS with a fixed number of iterations. For the setting, 10 initial graphs, each generated by a different seed, is used to evaluate the trained agents'

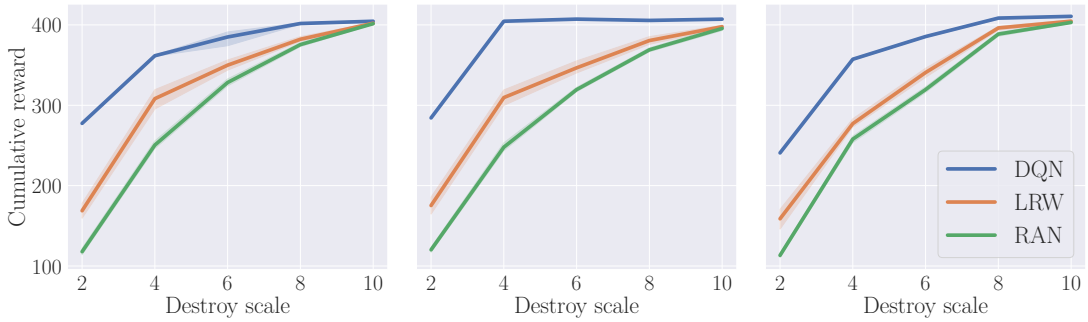


Figure 5.12: Performance as a function of destroy scales for the DQN (blue), LRW (orange) and RAN (green) agents. Higher is better.

performance. Each trajectory/episode contains a budget of 10 action tuples (or 20 selected operator actions). We generate randomly 128 initial states for the training/validation/test dataset, each of which associates with an initial solution generated by clustering customers into tours of 4 customers per tour. These are the same across all model seeds.

As shown in Table 5.6, the DQN agent yields the lowest objective values (best results) when used to perform operator selection in ALNS for portfolios containing than 5 destroy operators. Interestingly, LRW is able to perform substantially better than CRW due to having undergone training on a different dataset of solutions prior to being applied in evaluation. As already observed, the performance of CRW is indistinguishable from RAN in the setting across all the 3 instances.

Observe that a limited budget is established for each operator selection approach to assess its capability in selecting a fixed number of operators that maximally enhances the solution quality. Consequently, we are evaluating the efficiency in solution improvement amongst operator selection approaches given limited search length, rather than comparing the best-found solutions while termination criteria are satisfied. This is also the primary distinction between the CRW in this experiment and CRW in the previous factorial analysis in Section 5.6.2.

5.8.3 Impact of Destroy Scale

Furthermore, we analyse the impact of the destroy scale on the agents' performances during the MDP training, with a smaller scale implying the removal and reinsertion of a smaller proportion of nodes. We vary the destroy scale $q \in [2, 4, 6, 8, 10]$ with destroy portfolio $|\mathcal{D}| = 12$ on a total of 20 nodes. Results are shown in Figure 5.12.

The gap between the DQN and other methods is largest for the smallest scale, suggesting that a careful selection of operators to remove the most expensive nodes contributes significantly to better solution quality. In contrast, a larger destroy scale requires building up the solution from the ground in the repair step, stressing the operators' reconstruction ability rather than the destroy operator selection policy. When increasing the destroy scale, the cumulative rewards gained by different agents all converge to a similar level.

After training the models, we evaluate the impact of the destroy scale on the model performances within the ALNS framework. The results are given in Table 5.7.

Table 5.6: Evaluating operator selection approaches in ALNS with destroy portfolios $\mathcal{D}$ of different sizes. Values represent the average and best objective values found within a fixed number of iterations, using each approach to select operators. Lower is better.

C-inst	$\lvert\mathcal{D}\rvert$	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**	**mean**
DQN	*avg*	316.28	314.96	311.18	**216.33**	**213.03**	**216.34**	**215.12**	**215.75**	**214.19**	**213.09**	**188.11**	**239.49**
	min	244.64	245.28	240.32	**154.06**	**149.29**	**153.05**	**151.87**	**153.04**	**150.7**	**149.99**	**146.71**	**176.27**
LRW	*avg*	**293.84**	**299.56**	**302.66**	248.31	259.06	261.17	266.59	264.65	265.62	264.78	224.49	268.25
	min	**209.8**	**211.62**	**207.6**	172.97	180.96	178.07	187.71	183.88	182.07	185.3	160.95	187.36
RAN	*avg*	293.5	313.98	315.05	284.31	299.86	294.57	299.64	303.58	312	305.74	286.08	300.76
	min	209.95	219.79	212.33	194.6	206.28	199.23	203.63	208.35	212.2	213.38	197.54	207.03
CRW	*avg*	293.94	314.56	314.28	285.53	299.51	295.77	302.67	307.64	311.32	308.53	286.75	301.86
	min	210.27	222.46	213.42	197.82	208.25	198.37	206.6	210.7	213.69	213.38	194.63	208.14
R-inst	$\lvert\mathcal{D}\rvert$	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**	**mean**
DQN	*avg*	330.37	334.09	331.24	**217.25**	**215.27**	**215.41**	**215.89**	**216.17**	**215.91**	**221.71**	**159.29**	**242.96**
	min	273.81	266.66	272.16	**144.97**	**143.56**	**144.11**	**144.19**	**144.58**	**144.49**	**153.64**	**108.07**	**176.39**
LRW	*avg*	**322.22**	**329.14**	**327.18**	269.97	265.26	286.74	281.67	286.63	284.94	274.17	238.46	287.85
	min	**246.19**	**254.83**	**243.38**	187.33	183.51	199.16	196.51	201.83	196.82	189.54	157.33	205.13
RAN	*avg*	323.25	337.03	337.68	298.76	317.84	310.1	321.11	321.76	327.19	321.3	294.45	319.13
	min	253.64	257.11	250.05	211.7	235.07	218.61	234.45	231.79	234.31	234.38	206.54	233.42
CRW	*avg*	322.71	333.59	336.81	298.38	320.5	308.7	318.42	321.34	327.17	324.96	296.05	318.97
	min	252.39	255.69	250.28	211.47	235.15	215.55	230.06	232.49	235.87	233.28	208.18	232.76
RC-inst	$\lvert\mathcal{D}\rvert$	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**	**mean**
DQN	*avg*	306.06	311.02	313.78	**216.84**	**218.75**	**218.16**	**217.66**	**217.48**	**218.12**	**224.56**	**201.81**	**242.2**
	min	228.16	235.04	240.93	**151.89**	**154.74**	**153.71**	**152.85**	**152.06**	**153.79**	**161.32**	**159.09**	**176.69**
LRW	*avg*	**294.94**	**314.93**	**310.02**	278.18	282.01	286.05	292.6	293.97	293.44	292.15	261.08	290.85
	min	**205.05**	**216.86**	**209.63**	187.78	189.92	190.45	197.61	196.75	197.62	196.21	172.7	196.42
RAN	*avg*	297.2	319.87	317.7	288.95	298.41	294.26	300.61	308.9	315.42	305.53	283.71	302.78
	min	207.17	222.85	209.95	193.42	198.33	195.88	200.04	207.6	210.9	204.25	185.29	203.24
CRW	*avg*	**294.94**	319.07	320.76	289.51	299.99	296.09	299.98	310.67	315.4	309.69	285.73	303.8
	min	**205.05**	221.27	213.01	195.39	200.23	196.03	202.8	207.05	211.47	205.59	186.24	204.01

We notice similar trends, consistent with the previous ALNS evaluation findings, can be observed for the mean and min performance gaps. DQN demonstrates the overall best performance, particularly in scenarios with more challenging requirements, in this case, the perturbation of a smaller fraction of the graph to enhance solution quality. Even though DQN is occasionally outperformed by other models, the performance gap is narrow, which again emphasises on the operators' reconstruction ability while neglecting the operator selection in contributing to the algorithm's performance.

5.8.4 Scaling to Larger Instances with GNN

In this experiment, we train the DQN with a GNN representation and the LRW on instances of size 20, then evaluate them in an MDP setting on instances of size up to 100. The operator budget is kept the same while the destroy scale is increased proportionally to the size, i.e. $q = n/5$. We use the largest destroy portfolio with $|\mathcal{D}| = 12$, which is the complete destroy operator portfolio. As shown in Figure 5.13, the DQN+GNN agent outperforms the other methods, suggesting the strong generalization of the learned operator selection policies. A larger confidence interval is observed for the C instances, due to one model seed that generalises poorly on $\mathcal{J}^{\text{test}}$ despite good performance on $\mathcal{J}^{\text{validate}}$.

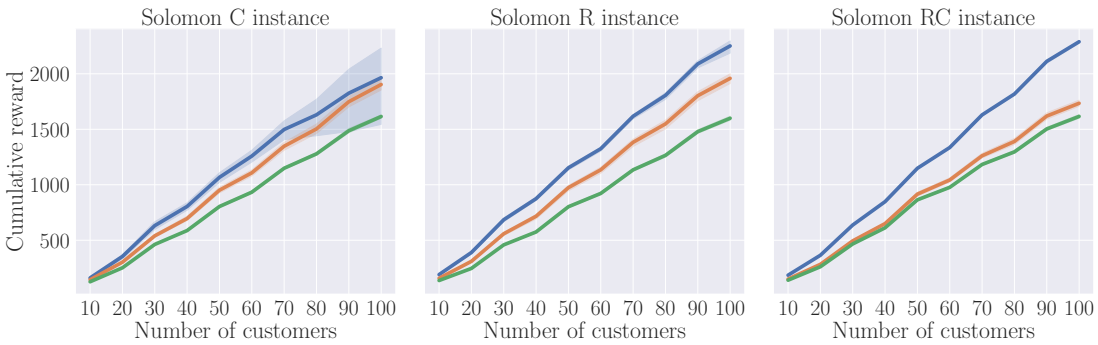


Figure 5.13: Cumulative rewards for the DQN (blue), LRW (orange) and RAN (green) agents with GNN representation. Higher is better.

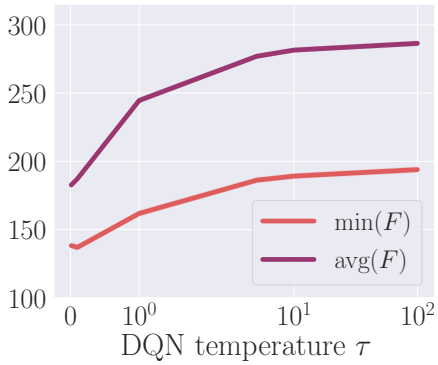


Figure 5.14: Values of F when varying DQN temperature in ALNS. Lower is better.

Table 5.7: Evaluating the impact of the destroy scale on the model performances within the ALNS framework. Values represent the average and best objective values found within a fixed number of iterations, using each approach to select operators. Lower is better.

C-inst	$\lvert\mathcal{D}\rvert$	**2**	**4**	**6**	**8**	**10**	**mean**
DQN	*avg*	**272.35**	**188.11**	**158.57**	**126.50**	**131.14**	**175.34**
	min	**199.25**	**146.71**	127.19	**108.17**	102.30	**136.72**
LRW	*avg*	343.12	224.49	206.47	170.37	146.28	218.15
	min	246.64	160.95	**125.74**	110.37	102.21	149.18
RAN	*avg*	393.45	287.99	231.14	185.82	152.23	250.12
	min	292.70	200.44	145.74	115.86	**101.98**	171.34
CRW	*avg*	395.45	286.75	230.09	184.11	150.75	249.43
	min	295.55	194.63	145.50	116.25	102.52	170.89
R-inst	$\lvert\mathcal{D}\rvert$	**2**	**4**	**6**	**8**	**10**	**mean**
DQN	*avg*	**267.72**	**159.29**	**136.65**	**123.27**	**121.33**	**161.65**
	min	**191.30**	**108.07**	**104.61**	**104.54**	**104.45**	**122.59**
LRW	*avg*	355.37	238.46	209.79	172.81	152.99	225.89
	min	262.40	157.33	126.79	109.21	105.35	152.21
RAN	*avg*	396.62	294.45	231.89	186.35	156.72	253.21
	min	300.31	206.54	143.80	115.42	105.43	174.30
CRW	*avg*	395.22	296.05	233.14	187.41	156.65	253.69
	min	299.92	208.18	148.82	116.67	104.96	175.71
RC-inst	$\lvert\mathcal{D}\rvert$	**2**	**4**	**6**	**8**	**10**	**mean**
DQN	*avg*	**310.64**	**201.81**	**159.19**	**129.72**	**128.26**	**185.92**
	min	**258.48**	**159.09**	**128.00**	**104.02**	104.69	**150.85**
LRW	*avg*	374.77	261.08	226.70	162.47	144.55	233.91
	min	293.47	172.70	165.42	104.17	**103.38**	167.83
RAN	*avg*	413.46	283.71	241.43	173.23	152.12	252.79
	min	334.30	185.29	173.91	105.59	103.69	180.56
CRW	*avg*	412.71	285.73	240.95	172.42	153.50	253.06
	min	333.23	186.24	173.20	105.43	103.69	180.36

5.8.5 Impact of the DQN Temperature

As discussed in Section 5.5.5, the temperature parameter τ controls the greediness of the resulting policy. Figure 5.14 shows the minimum and mean F obtained with ALNS as a function of $\tau \in \{10^{-2}, 10^{-1}, 10^{0}, 10^{1}, 10^{2}\}$, averaged over the 3 instance sets. Even though a probabilistic policy may be desirable in some ALNS scenarios, we find that performance generally degrades as the temperature increases. This suggests that, in the settings tested, the inherent stochasticity of the operators is sufficient to explore the search space without the need to combine different choices.

5.9 Conclusions and Future Research

In this chapter, we have proposed an operator selection mechanism based on Deep Reinforcement Learning to enhance the performance of the ALNS metaheuristic on the Capacitated Vehicle Routing Problem. A key insight and contribution is the proposal of an operator selector that is conditioned on the decision space characteristics of the current solution. We have demonstrated its ability to outperform the classic Roulette Wheel and random operator selection, as well as the potential of using Graph Neural Networks to scale the model to large problem instances. Our results also highlight the impact of the operator portfolio size and the destroy scale on performance. Plans for future extension of the work involve the consideration of computation efficiency as part of the operator quality evaluation to cope with different operators having very different computation times, and broader applications to other combinatorial optimisation problems.

Chapter 6

Generalisation of RL-ALNS to VRP Variants

6.1 Framework Extension

6.1.1 Overview

Building upon the learning-based ALNS framework outlined in Chapter 5 to formulate the operator selection process as a MDP (see Section 5.5.2), learning an operator selection policy using neural network architectures (see Section 5.5.4), and integrating the DQN technique with the classic ALNS methodology (see Section 5.5.5), we have demonstrated that DQN algorithm [166] as a value function based method is applicable for learning the operator selection policy that the agent uses to select operators.

In this chapter, we aim to further demonstrate the wide applicability of our learning-based operator selection approaches to multiple problem variants. We also continue utilising GNNs for state representation for generalising to larger problem instances than seen during training. Compared to the previous model, we generalise the methodology, carry out an extensive evaluation on 5 routing problems, and substantially expanded the considered operator portfolio from 12 destroy and 2 repair operators to 28 and 7 operators respectively, to include a representative selection of operators from the literature. The reason for choosing routing-related COPs is due to their graph-structure data, which is ideal for implementing GNN architecture to achieve generalisation in their premeditation and extrapolate to larger instance sizes. Furthermore, we have implemented and integrated an RL method based on concurrent work by Kallestad et al. [117] and Reijnen et al. [197], both applied Deep RL to improve ALNS operator selection, to which our proposals are compared. Finally, our approaches are successfully compared to a wide selection of experiments and baselines, including practical aspects such as the search and computational budgets, means of constructing initial solutions, and scalability to large instances with up to 1000 nodes are considered.

6.1.2 Operator Selection Approaches

As introduced in Chapter 5, we continue utilising the three learning-based models: the `DQN` model with a greedy policy, the `LRW` model with a learned RW updating algorithm, and the `RAN` model with a uniform-random policy. Following the algorithmic blueprint in Section 5.5, the proposed set of learning-based approaches all contain two stages, where we first isolate the operator selection process from the ALNS framework and convert it into a MDP for training the agent, then integrating the fully-trained model

into the ALNS algorithm to search for good-quality solutions. This results in employing learning-based agents to make decisions on the choice of an operator pair based on a learned policy formed during training. Since GNN architecture is applied to all models in this chapter, we name the method comprising the MDP formalisation and the DQN+GNN learning mechanism as Graph Reinforcement Learning for Operator Selection (`GRLOS`) in order to differentiate from the previous chapter's DQN model. We also present the benchmarking `CRW` model from the classic ALNS framework with no prior training required. Moreover, we collectively refer to the approaches proposed by Kallestad et al. [117] and Reijnen et al. [197] as Reinforcement Learning Baseline (`RLB`). The differences of our approach comparing to theirs are analysed in Section 5.2.5.

Beyond the specific GRLOS framework, a broader methodological conundrum is worth noting regarding the integration of RL and ALNS, which applies more widely to the use of any ML approach with this metaheuristic. Namely, after a certain point in the ALNS process, the model will be asked to select operators for solutions that are dissimilar to those seen during training. This is by definition since, if we knew how to select operators optimally at every step, the optimal solution reached at the end of the process would be part of the training set and we would not need to use ALNS to begin with. This phenomenon, known as *distribution shift*, greatly impacts the performance of ML models [191]. Consequently, we distinguish between the *training operator budget* B_{MDP} and the *ALNS operator budget* B_{ALNS}, assuming that $B_{\text{ALNS}} > B_{\text{MDP}}$. In this context, we refer to the states that can be encountered when applying operators within the defined training operator budget B_{MDP} as *in-distribution*. Further application of operators beyond this point until B_{ALNS} may cause the model to suffer in performance since it will encounter states dissimilar to those it was trained on. We refer to this setting as *out-of-distribution*.

To mitigate the issue of distribution shift, we propose the following partial mitigation that applies in an out-of-distribution setting beyond the training operator budget of the GRLOS algorithm, where its predictions of future rewards become inaccurate due to distribution shift. We compute the frequencies with which the operators were applied on the training set when the model is within the training horizon. Afterwards, operators are sampled proportionally to their frequencies beyond the training horizon. We denote this mitigated approach as `GRLOS-M`, a two-phase policy that applies greedy operator selection policy for as long as the training horizon, then switches to a probabilistic policy to match those observed during the training horizon until the search is completed. Similarly, for the RLB baseline, we denote this variant as `RLB-M`. As we will demonstrate in our experimental evaluation, this mitigation greatly improves the performance of RL models when used in ALNS.

Comparing to GRLOS, the LRW model tracks simple statistics, owing to the utilisation of much sparser information compared to the state-based data-intensive DQN-based policy, and hence is substantially simpler and not subject to this caveat. Accordingly, we also consider a `Hybrid` policy in our evaluation, which combines the strengths of GRLOS and LRW. This method uses the GRLOS policy for the horizon of operators used during training and switches to the LRW policy once the RL model is out-of-distribution.

6.2 Preliminary Settings

In this section, we introduce the state representation for each specific routing-based problem variant, the list of actions (operators) associated, and the preliminary design and dataset for the experiments.

6.2.1 VRP Variants

Tackling NP-hard COPs by classic heuristic techniques most often requires the repeated handcrafting of rules, even when the same problem structures are maintained and the differences occur within the instances. Our proposed learning-based method exploits this recurring effect by automating the process, making the same approach transparently applicable to different problem variants and dataset. To demonstrate this wide applicability, we expand from CVRP applied in the previous chapter to the following four routing problems:

1. *Travelling Salesman Problem (TSP)*: A single vehicle serving all customer nodes within a single grand tour with no vehicle capacity considered. The objectives are to minimise the total traversal cost.

2. *Heterogeneous Vehicle Routing Problem (HVRP)*: Forming multiple tours from a single depot served by a fleet of heterogeneous vehicles, each type associated with a given fixed capacity and a finite number of available vehicles. The objectives are to minimise the total traversal and vehicle hiring costs.

3. *Vehicle Routing Problem with Time Windows (VRPTW)*: Establishing multiple tours from a single depot by a homogeneous fleet of vehicles, where the customer node arrival time must be within the a priori customer time window to avoid additional penalty. The objectives are to minimise the total traversal, vehicle hiring, and out-of-time-window penalty costs.

4. *Location-Routing Problem (LRP)*: Given a set of potential depot locations, jointly consider the opening of a selected subset of depots, the allocation of each customer node to an open depot, and forming tours using a finite set of homogeneous vehicles to serve the customers from the decided depot. The objectives are to minimise the total traversal, vehicle hiring, and depot opening costs.

One reason for choosing these problems is that they are some of the most studied COPs, enabling us to leverage a large body of literature on heuristic operator design. Furthermore, solutions to these problems can naturally be represented as graphs: the tours making up the solution are a collection of cyclic paths on the fully-connected graph. More broadly speaking, the proposed methodology applies to any COP that can be naturally modelled as a graph, a mathematical structure that is pervasive in combinatorial optimisation.

6.2.2 Operator Variants

All operators considered in this study already exist in the literature. Their mechanisms are coded with minor adaptations aiming only to fit the specific COP environment, without fine-tuning the operator design to boost performance. In this way, the list of operators can yield varying performances in the given model, some of which may even bring down the solution quality in general. However, encompassing a spectrum of qualities for the action is important for the model training process, an effective agent should be capable of avoiding the operators that harm the solution improvement process without human intervention.

The sets of destroy operators applied to each problem variant, summarised in Table 6.1, vary due to differences in individual neighbourhood structures. We select 27 popular destroy operators from the ALNS literature spanning the representative categories and include an additional no-action operator that leaves the solution unchanged.

Table 6.1: Destroy operators used for the different VRP variants, as indicated by ✓.

Features	Random	Greedy	Related	TSP	CVRP	HVRP	VRPTW	LRP
Distance-based	Random Node, Random Route	Worse Distance, Route Deviation, Greedy Route Longest, Greedy Route Shortest	Proximity Distance, Shaw, Cluster, Pair, Zone, Node Neighbourhood, Route Neighbourhood	✓	✓	✓	✓	✓
Historical-based	Forbidden Random,	Historical Knowledge,	Historical Pair	✓	✓	✓	✓	✓
Historical-based		Largest Demand Deviation, Smallest Demand, Largest Demand	Proximity Demand		✓	✓	✓	✓
Time Window-based		Worst Time	Proximity Time, Proximity Service				✓	
Facility-based	Close-an-open-Facility, Open-a-closed-Facility	Worst Facility Node,	Facility Swap					✓

The repair operators remain the same for all problem variants. Here we use random repair, greedy repair, greedy perturbation repair, deep greedy repair, and k-regret repair ($k = 2, 3, 4$). A summary of the destroy and repair operators utilised in Chapter 5 and Chapter 6 can be found in Table A.3.

6.2.3 Feature Representation

This section discusses the features that are used by GRLOS and RLB to represent the state, as well as details of the other components used for training.

For all the variants, the environment is based on a complete, undirected graph $G = (V, E)$, where the set of vertices $|V|$ consists of the depot and customer nodes, and the set of arcs follows a symmetric distance matrix $D \in R^{n*n}$. The node-based feature vector is denoted as F. We distinguish between three types of features: *node* features correspond to each node in the solution, *edge* features are attached to arcs, and *global* features characterise the current solution at a high level beyond any particular node or arc. Furthermore, features can be *static*, meaning they do not change with modifications to the solution (e.g., for the geographical coordinates of the depot in routing problems), or *dynamic* otherwise.

The list of features utilised by GRLOS are shown in Table 6.2, which is an extension to Table 5.2. Since each VRP variant is characterised by different constraints, the feature representation differs slightly between the variants. Notably, we exclude the *tour_id* for TSP and include an additional set of time-related node features *tw_start*, *tw_end*, and *service_time* for VRPTW. The node features for all the potential facilities are also included inside the feature matrix for LRP. The remaining variants follow the same node features.

Table 6.2: Features used by GRLOS to represent state.

Feature	Type	Dynamicity	Description
x, y	node	static	Geographical coordinates for each customer node.
demand	node	static	Physical demand associated with each customer node.
tour_id	node	dynamic	The index of the tour this node belongs to, represented as an integer (excluded for TSP).
pos_id	node	dynamic	The sequential index of each node within each tour as an integer.
tw_start	node	static	The start time of the customer time window (VRPTW only).
tw_end	node	static	The end time of the customer time window (VRPTW only).
service_time	node	static	The service time at each customer node (VRPTW only).
adj_dis	edge	dynamic	The Euclidean distance from each node to its subsequent neighbour in the tour.
inst_size	global	static	The number of customer nodes in the current instance.

In contrast, the features used by the RLB baseline are shown in Table 6.3, and are in essence a union of the features used by Kallestad et al. [117] and Reijnen et al. [197], minus the few features that are undefined outside of the ALNS process (such as

Table 6.3: Features used by the competing RLB baseline [117, 197] to represent state.

Feature	Type	Dynamicity	Description
reduced_cost	global	dynamic	Cost reduction between previous and current solution.
cost_difference_best	global	dynamic	Difference between objective value of current and best solution.
cost_of_best	global	dynamic	Cost of best encountered solution.
is_current_best	global	dynamic	Whether the current solution is the best solution (0 or 1).
cost	global	dynamic	Cost of the current solution.
iteration	global	dynamic	Current iteration (i.e., the time step).
stagnation_count	global	dynamic	Number of iterations without improving the best found solution.
unseen	global	dynamic	1 if solution was not previously encountered, and 0 otherwise.
was_changed	global	dynamic	1 if solution was changed from the previous one, and 0 otherwise.
last_action_sign	global	dynamic	1 if the last action improved the solution, and 0 otherwise.
last_action	global	dynamic	One-hot encoding of the last taken action.

the temperature and cooling schedule). We note that all features used by this baseline are global and dynamic. An essential observation from these features is that Markov assumption, which underlies the MDP formalism that states the future state of the system is solely dependent on the preceding state and action, is not satisfied given the dependencies of transitions and rewards are not only on the current state but also past states implied by these features. Consequently, this may harm performance given the reliance of standard RL algorithms on this assumption.

For GRLOS, we use the node and edge features as input to the message-passing procedure of the GNN, and use sum aggregation to compute an embedding for the entire graph. This is then concatenated with the global features to obtain the state representation. As previously stated in Chapter 5, we again opt for the GAT [235], which allows for flexible aggregation of neighbour features. We use 3 message-passing layers and a dimension of node embeddings equal to 32. Both GRLOS and RLB use an MLP architecture consisting of hidden layers and ReLU activations, with the final layer containing one output unit for each valid destroy or repair operator. The MLP has 64 units in the first hidden layer, with the subsequent layers having half the size.

6.2.4 Dataset and Initial Solutions

For the VRP variants, we use the R, C and RC instances (random, clustered, and mixed random-clustered nodes) of the classic Solomon dataset [218] each containing a depot and up to 100 customers. Up to 3 depots are provided for the LRP variant with 35 customers per depot. We assign the vehicle capacity to be 200, and similar as before, adjust it proportionally if scaling down the instance to fewer customers. Furthermore, we also consider the larger-scale Gehring and Homberger (GH) instances [75] with up to 1000 nodes. The GH dataset follows a similar structure to Solomon dataset with the same R, C, RC categories but with different node locations, demands and instance sizes.

In terms of initial solutions that are provided to the ALNS process, we primarily consider randomly generated initial tours whose length is proportional to the defined vehicle capacity (therefore the instance size). As a comparison, we also generate initial tours using a self-implemented Clarke-Wright saving algorithm with noise, denoted as the "noisy Clarke-Wright", to construct relatively good-quality initial solutions. The noisy Clarke-Wright algorithm adheres to the procedure given in Clarke and Wright [39], with the actual traveling distance between each pair of nodes multiplied by a noise factor randomly selected from $[0.8, 1.2]$. The inclusion of noise ensures that the initial solutions are not identical, which is critical for a meaningful evaluation of learning-based approaches, while still starting the search process within a good neighbourhood.

The width of such an interval is determined in Section 6.3.4 to satisfy that the solution redundancy ratio for any instance size does not exceed a specific value. We note that this noise factor is only multiplied with the distance matrix used to formulate the initial solution, and not to any solution formed during the training or evaluation process.

6.2.5 Training and Parameter Setting

For each instance, we generate 3 distinct sets of initial solutions $\mathcal{J}^{\text{train}}, \mathcal{J}^{\text{validate}}, \mathcal{J}^{\text{test}}$ that are used for model training, parameter tuning and model selection, and final evaluation respectively. $\mathcal{J}^{\text{train}}$ and $\mathcal{J}^{\text{test}}$ contain 128 sets of tours between 20 and 100 customers with an increment of 10, yielding 1152 sets of tours in total. The validation set $\mathcal{J}^{\text{validate}}$ is smaller in comparison and contains 64 sets of tours with each of $[20, 30, 40, 50, 100]$ customers, yielding 320 sets of tours in total. This enables meaningful model selection while being sufficiently small to assess performance quickly during the training process.

Both GRLOS and RLB use identical parameters: a learning rate of $\alpha_{\text{RL}} = 0.0005$, stochastic gradient descent with the Adam optimiser and a batch size of 16. The exploration rate ϵ is linearly decayed from 1 to 0.1 in the first 10% of steps, then remains fixed. The replay buffer size is equal to 3% of the number of steps. For LRW, we use a weight segment length of 200 and reaction factor of 0.99. For fairness of comparison with methods that have undergone prior training, we allow a tuning of parameters for the CRW method. We consider values for the segment length in $\{20, 75, 200\}$ and reaction factor in $\{0.9, 0.95, 0.99\}$. The optimal parameter configuration out of the 9 possible ones is chosen separately according to performance on $\mathcal{J}^{\text{validate}}$ for each type of instance and problem variant. We do not carry out parameter tuning for the other methods beyond the reported values due to computational budget constraints.

The remaining ALNS parameter values are defined as below, with the sole difference being the method used for operator selection. We set the destroy scale $q = 10\%$, the scores $\delta_1, \delta_2, \delta_3$ to be $2, 1, 0$ respectively, the temperature cooling coefficient $c_\tau = 0.998$, the initial temperature $\tau = 100$, and the non-changing iteration stop criterion to be 1000. Unless otherwise stated, we execute ALNS for $B_{\text{ALNS}} = 100$ steps.

6.3 Experiments

6.3.1 Model Validation

We first test the applicability of the GRLOS, LRW and RAN models proposed in Section 5.7.1 on the five VRP variants. Under the MDP setting, a budget $B = 10$ and a fixed instance size of 20 nodes generated from the Solomon CR-instance dataset are given. Notably, a smaller budget $B = 5$ is also given to TSP due to its simplicity, which assumably requires fewer operators than the other problem variants.

From Table 6.4, we observe the consistent superior performance by the GRLOS model compared to the LRW and RAN models in almost all the scenarios. This validates that with minor model adaptations on the state infrastructure, our models can be easily adapted to operator selection training for multiple VRP variants and resume similar performance ranking amongst the trained GRLOS, LRW and RAN agents. The only inferior performance of GRLOS compared to LRW is observed for the case of TSP, which is the easiest variant amongst all the presented problems. Nevertheless, we once again observe a better performance from GRLOS when the search budget is limited to half the previous allowance, indicating that the efficiency of GRLOS in accumulating

Table 6.4: MDP evaluation for 5 VRP variants (higher the better).

	C instance			R instance			RC instance		
	GRLOS	LRW	RAN	GRLOS	LRW	RAN	GRLOS	LRW	RAN
TSP ($B = 5$)	**211.45**	209.1	150.77	**365.25**	351.93	261.82	**474.26**	467.46	324.89
TSP ($B = 10$)	220.43	**223.53**	153.64	372.66	**372.33**	267.42	489.43	**492.77**	330.53
CVRP	**288.22**	276.82	214.26	**562.7**	558.28	376.36	**424.05**	357.14	234.76
HVRP	**424.88**	416.97	248.03	**655.07**	633.15	380.05	**648.49**	637.55	391.97
VRPTW	**397.11**	387.28	284.25	**565.1**	557.08	346.98	**415.06**	363.0	233.83
LRP	**422.15**	394.94	54.32	**816.14**	769.95	262.73	**414.68**	300.71	-16.28

the rewards within a specified search length surpasses that of the LRW, and that LRW is able to achieve a comparable level of performance once the budget is sufficiently large.

6.3.2 MDP Evaluation with GNN Scale-up

After validating the applicability of the GRLOS, LRW and RAN models in the previous section, we evaluate the models in the MDP setting, in which the operator budget used for model training ($B_{\text{MDP}} = 20$) matches the evaluation horizon precisely, and the methods are benchmarked independently of the ALNS process.
From Figure 6.1, we observe that the learning-based GRLOS, LRW and RAN models are able to achieve good cumulative rewards and possess generalisation to varying instance sizes and initial solutions. All learning-based methods outperform uniform random operator selection (RAN) statistically significantly. For all routing problems, GRLOS is able to accumulate a larger sum of reward than the LRW and RLB approaches. The only exception occurs for the R instance and LRP, where LRW achieves better average results than GRLOS with instance sizes larger than 80. However, the confidence intervals for GRLOS include the LRW performance, and the best reward accumulated by GRLOS is still higher than that of LRW.

Another observation is that our proposed GNN-based GRLOS and LRW models demonstrate commendable extrapolation capabilities, enabling them to be initially trained on smaller-scale instances and subsequently extend their learned insights to generalise to more extensive instances. Specifically, the GRLOS and LRW models can scale up to 100 with better performance than the RAN baseline.

6.3.3 ALNS Evaluation

In this experiment, based on the models from the MDP training, we evaluate the performance of the models within the ALNS framework, where the evaluation horizon $B_{\text{ALNS}} = 100$ exceeds the one used for model training. We additionally include the GRLOS-M, Hybrid, and CRW methods as previously discussed. For visual compactness, the comparison with the RLB baseline is deferred to Section 6.3.6.

From Figure 6.2, we observe an overall performance hierarchy among the models that follows the pattern of GRLOS-M, LRW, and Hybrid models exhibiting better overall performance compared to the vanilla GRLOS, followed by CRW, and finally the RAN baseline. In TSP and LRP, we notice a sharp decline in the GRLOS model performance level compared to the other GRLOS variants for TSP above 70 nodes and LRP above 80. In TSP and LRP, we notice a sharp decline in the GRLOS model performance level compared to the other methods above 70 nodes. However, the other

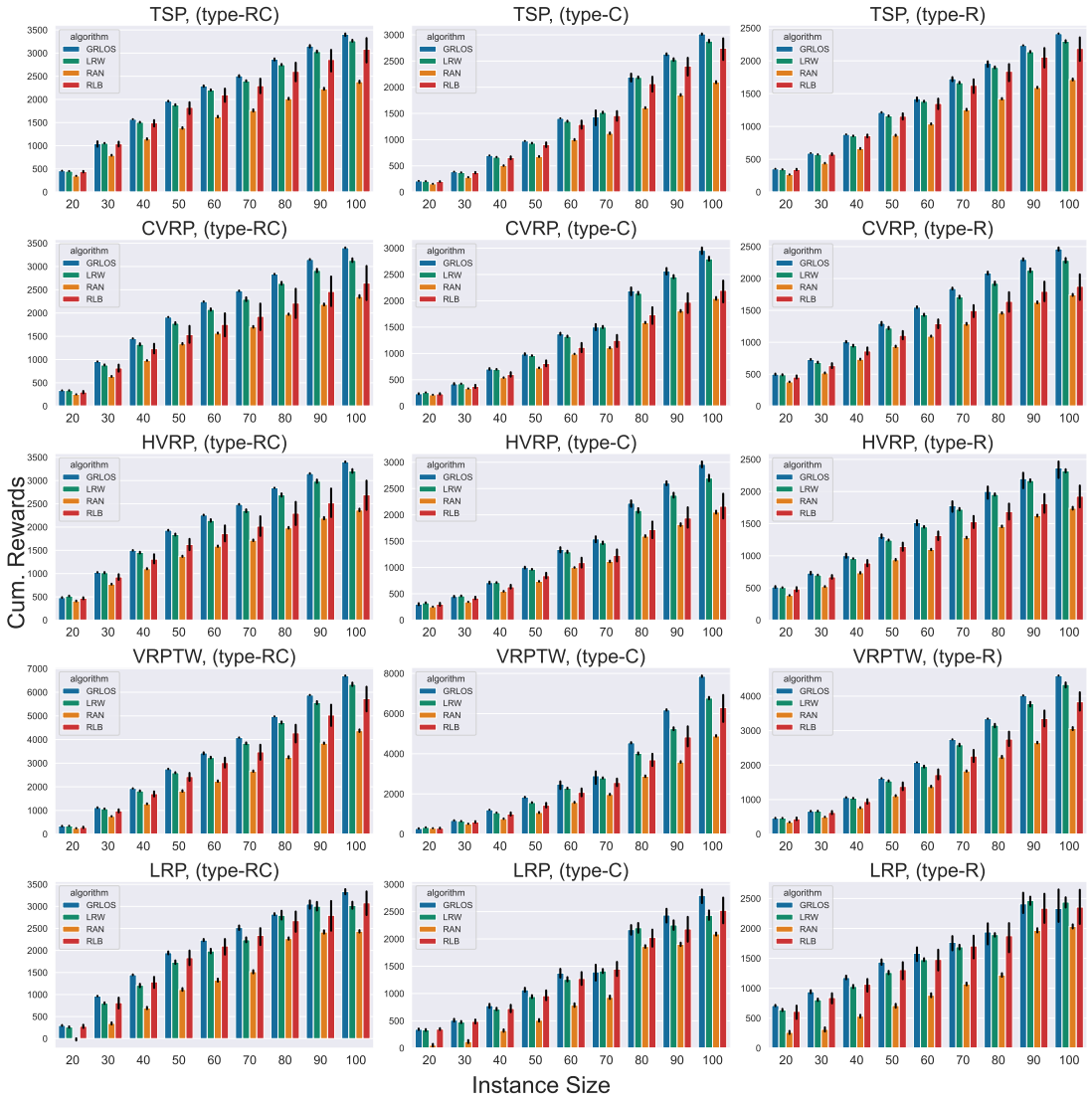


Figure 6.1: Cumulative rewards for GRLOS, LRW, RAN and RLB agents with GNN representation. Random initial tours are formed with length proportional to vehicle capacity. Higher is better.

GRLOS-based models (GRLOS-M and Hybrid) show no similar trend. The reason behind the sudden performance drop for standard GRLOS in some cases (most notably, LRP above 80 nodes) is due to the previously mentioned distribution shift limitation of ML approaches. These results, however, show that the proposed GRLOS-M mitigation is effective, as it leads to improvements in every setting tested. A jump is observed in the LRP variant for all methods, the reason for which is the inclusion of an additional depot to serve the total customer demands, which causes the objective function values of the instances to increase sharply.

The overall improvement in solution quality comparing to the RAN baseline is summarised in Table 6.5 and Table 6.6, averaged across all sizes of a problem variant and instance type respectively. In general, the Hybrid method results in the best overall improvement over uniform RAN method, with the exception of the simpler TSP problem, on which LRW performs better.

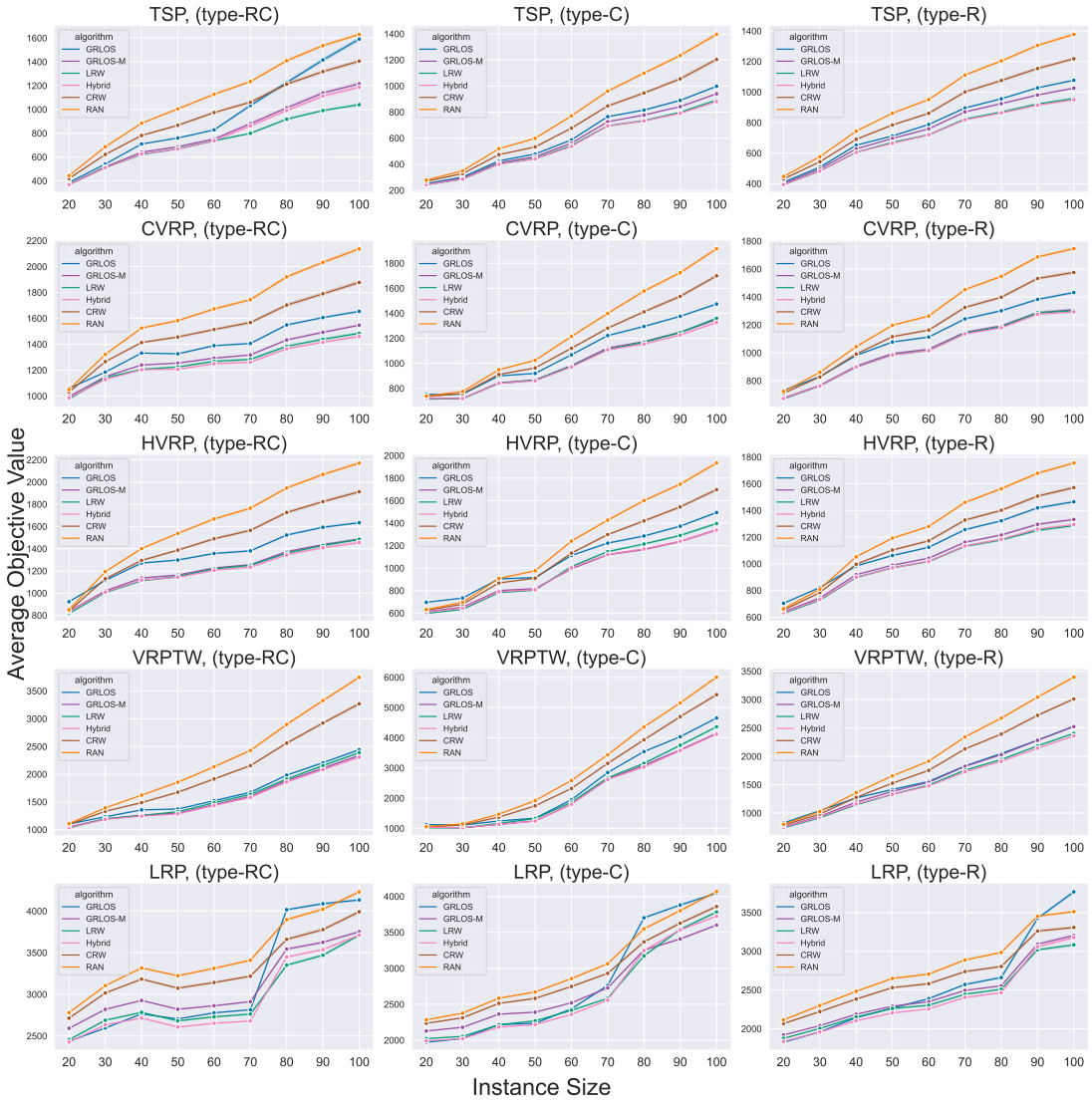


Figure 6.2: Average objective value from 128 seeds during ALNS evaluations for GRLOS, GRLOS-M, LRW, Hybrid, CRW, and RAN models on 5 VRP variants and 3 Solomon dataset categories. Lower is better.

Table 6.5: Solution quality improvement made by each model comparing to RAN, averaged over the same Solomon dataset category

Model Comparison	**RC**	**C**	**R**
GRLOS *versus* RAN	17.43%	13.21%	12.11%
GRLOS-M *versus* RAN	21.31%	18.71%	16.64%
LRW *versus* RAN	23.87%	19.39%	18.85%
Hybrid *versus* RAN	**24.57%**	**20.31%**	**19.13%**
CRW *versus* RAN	8.62%	7.21%	7.09%
RLB *versus* RAN	15.64%	10.03%	7.92%
RLB-M *versus* RAN	15.57%	9.79%	9.15%

Table 6.6: Solution quality improvement made by each model comparing to RAN, averaged over the same instance type

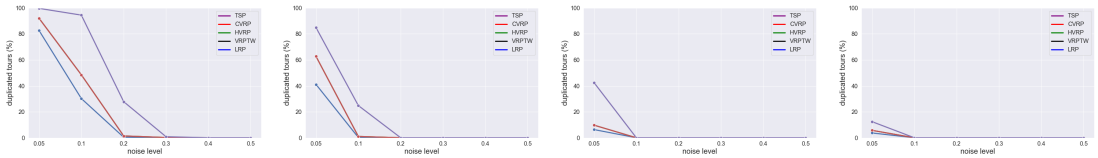
Model Comparison	TSP	CVRP	HVRP	VRPTW	LRP
GRLOS *versus* RAN	17.86%	12.46%	10.94%	18.23%	9.41%
GRLOS-M *versus* RAN	23.44%	18.55%	19.32%	22.98%	10.81%
LRW *versus* RAN	**27.03%**	19.50%	20.16%	23.18%	13.61%
Hybrid *versus* RAN	26.26%	**19.99%**	**20.40%**	**24.47%**	**14.53%**
CRW *versus* RAN	10.52%	7.44%	7.94%	8.08%	4.36%
RLB *versus* RAN	23.58%	6.11%	3.09%	11.14%	12.03%
RLB-M *versus* RAN	23.29%	7.59%	5.80%	13.67%	7.16%

6.3.4 Clarks-Wright Initial Solution Analysis

In this experiment, we retain all model and parameter settings in the experiments above, but instead generate initial solutions using the noisy Clarke-Wright algorithm as previously described.

Noise Level Analysis

Figure 6.3 demonstrates the changes in duplicated initial solution percentage with respect to the noise level applied to the distance matrix computation. The results are the maximum percentage found among the three Solomon instance types. A noise level of 0.2, or a noise factor from the range $[0.8, 1.2]$, can sufficiently limit the duplication percentage to within 1% for instances below 30 nodes and 0% for larger-scale instances with more complicated solution.



(a) 20 Nodes (b) 30 Nodes (c) 40 Nodes (d) 50 Nodes

Figure 6.3: Noise level analysis for Clarks-Wright in producing duplicated tours

Evaluation Performance with Clarke-Wright Initial Solutions

To start with, the MDP evaluation results on the GRLOS, LRW and RAN agents with Clarke-Wright initial solutions are presented in Figure 6.4. Given a good-quality initial solution to start the search, the GRLOS and LRW models can still effectively accumulate positive rewards, showcasing that they are able to make improvements to a non-optimal initial solution, despite the initial solution quality nor the length of search trajectory to reach a local optimal solution. In contrast, the RAN baseline gains negative rewards, leading the search away and creating worse-quality incumbents. For the LRP variant, the only case in which RAN gains positive rewards, the scale of the improvement is considerably smaller compared to the GRLOS and LRW models.

Figure 6.5 illustrates the ALNS evaluation for all models (excluding RLB baselines), notably, the out-of-distribution degradation in GRLOS performance does not occur with these initial solutions, which can be explained by the fact that the solutions the

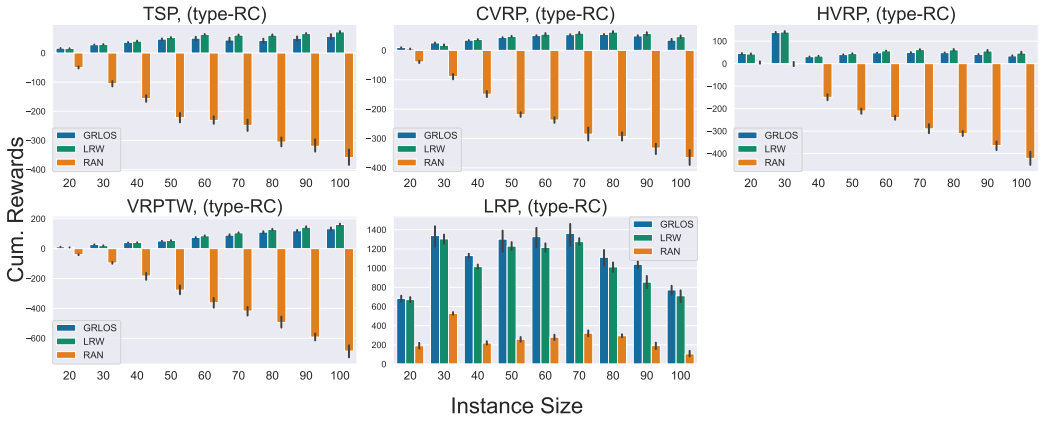


Figure 6.4: Cumulative rewards obtained by the agents with noisy Clarke-Wright initial solutions. Higher is better.

model encounters are more similar to each other, as they are closer to optimality. Akin to the ALNS evaluation with random initial solutions presented above, the reason for the sudden jump in objective values for LRP is the inclusion of an additional depot impacting the objective function values.

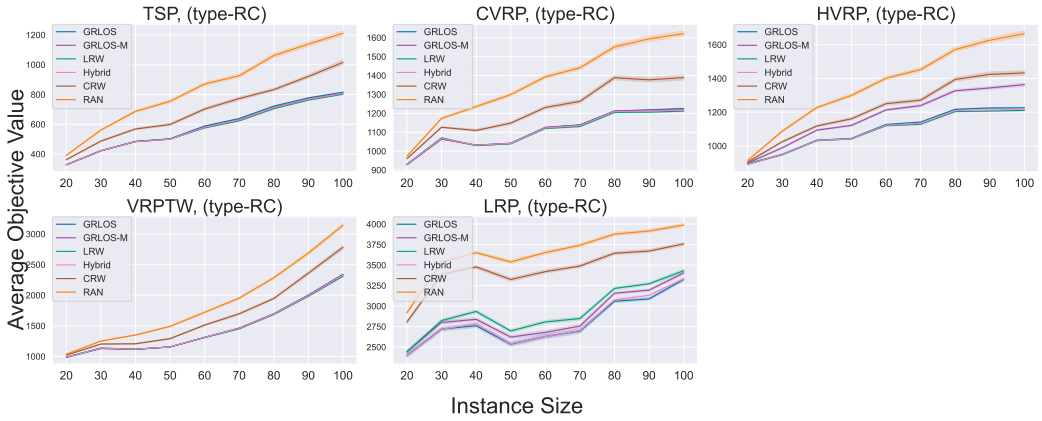


Figure 6.5: Average objective function value during ALNS evaluations for the different operator selection mechanisms on 5 routing problems and 3 Solomon dataset categories with noisy Clarke-Wright initial solutions. Lower is better.

6.3.5 Budget Size Analysis

For this experiment, we fix the instance size to 100 and analyse the performance of all agents when given a specific ALNS budget ranging from 20 (equal to the MDP training budget) to 100.

We observe from Figure 6.6 that, given a budget as large as the training budget, the GRLOS-based agents are able to more effectively reduce the objective value. Going beyond the training budget and hence out-of-distribution, the Hybrid and LRW agents begin to achieve better performance while the standard GRLOS converges to no or poor solution improvements, which is partially mitigated by GRLOS-M. In other words, GRLOS is able to construct a good solution within a similar evaluation horizon to its training budget, but does not perform as well as the other agents for pure exploration.

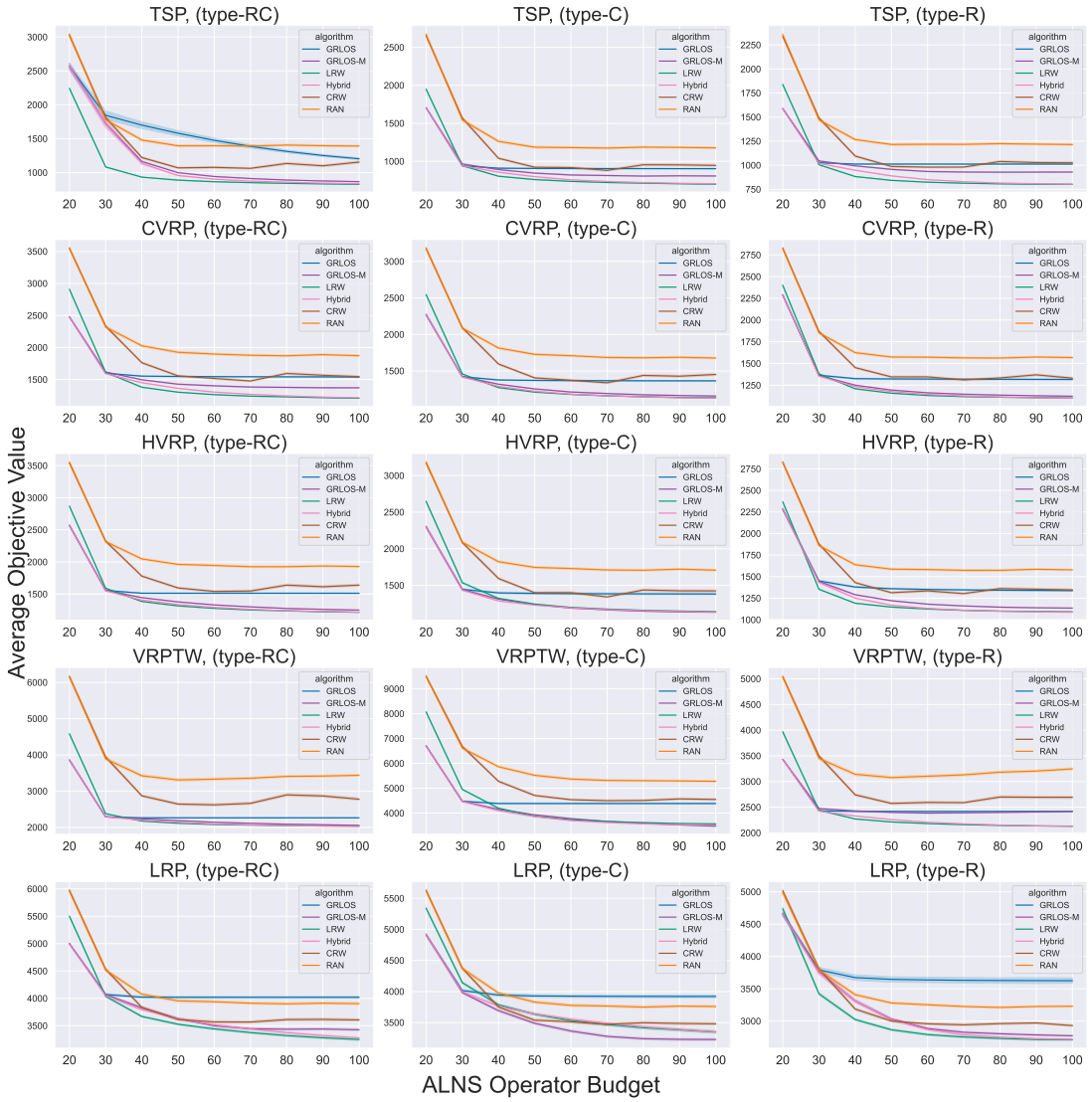


Figure 6.6: Average objective value during ALNS evaluations with certain budget size for the GRLOS-based, LRW, CRW, and RAN models on 5 VRP variants and 3 Solomon dataset categories. Lower is better.

Once again, this reflects the limitation of ML models under distribution shift, which also applies to the RLB and RLB-M methods. Nonetheless, the advantage of GRLOS is its ability to find good solutions with efficiency.

Certainly, training the model for the entire ALNS budget of 100 steps is infeasible due to the number of possible sequences of operators ($\approx 17^{50} \times 7^{50}$ for the TSP and up to $25^{50} \times 7^{50}$ for the LRP variant). Hence, one necessarily needs to "cut off" the training horizon at a certain point. Nevertheless, one may choose a larger budget and hence a longer training horizon, which would lead to an increase in performance for the ML approaches compared to the baselines, at the expense of requiring more training steps given the greater depth of the MDP.

6.3.6 Baseline Comparison

In this experiment, we compare the average ALNS evaluation performance of the GRLOS-M model to the baseline model proposed by Kallestad et al. [117] and Reijnen et al. [197], which we adapted and named RLB-M. The values in Table 6.7 are computed using the formula below and represent the percentage gaps in solution quality improvements:

$$gap = \frac{(\text{RLB-M} - \text{GRLOS-M})}{\text{RLB-M}} \times 100\% \tag{6.1}$$

Table 6.7: Average gap in objective value during ALNS evaluations between GRLOS-M and RLB-M models on 5 VRP variants and 3 Solomon instance types. Positive values denote better comparative performance of GRLOS-M.

Variant	Instance	20	30	40	50	60	70	80	90	100
TSP	RC	-0.08%	-1.08%	-0.38%	0.71%	1.69%	1.20%	0.78%	0.73%	0.28%
	C	0.77%	1.42%	1.80%	2.15%	2.61%	0.51%	0.67%	1.33%	1.64%
	R	-0.32%	-0.57%	-1.18%	-0.45%	-1.01%	-1.02%	-1.00%	-0.47%	-0.54%
CVRP	RC	0.47%	2.08%	2.22%	6.80%	8.35%	8.55%	8.49%	9.39%	9.93%
	C	1.72%	4.87%	8.13%	12.44%	15.55%	15.72%	21.36%	23.53%	25.16%
	R	4.78%	8.03%	10.33%	13.70%	15.15%	17.83%	19.93%	21.06%	21.84%
HVRP	RC	0.75%	7.21%	9.89%	14.12%	15.82%	17.47%	18.01%	18.98%	19.53%
	C	2.14%	4.57%	9.87%	14.11%	17.55%	19.02%	25.06%	27.54%	29.31%
	R	-0.02%	2.22%	6.80%	10.82%	12.25%	14.15%	15.77%	16.92%	18.04%
VRPTW	RC	1.70%	4.79%	7.90%	11.37%	12.82%	14.55%	14.59%	15.95%	16.39%
	C	-0.48%	4.96%	12.82%	19.92%	15.60%	11.26%	16.01%	17.79%	18.57%
	R	0.55%	1.58%	4.20%	7.81%	10.15%	12.35%	13.59%	14.86%	15.52%
LRP	RC	0.21%	1.79%	2.17%	2.02%	2.51%	3.43%	2.44%	3.53%	4.68%
	C	1.37%	1.90%	1.97%	3.34%	3.82%	2.08%	3.01%	4.56%	4.51%
	R	4.08%	4.70%	5.65%	7.00%	6.96%	7.06%	7.93%	6.68%	5.85%

We observe that GRLOS-M outperforms the RLB-M baseline model in 90.4% of cases. In cases where RLB-M does perform better, the gap is less than a percentage point (-0.58%) on average. Every cell in the table is tested via a Mann-Whitney U test with a 95% significant level ($\alpha = 0.05$). The few cells highlighted in grey represent tests whose outcome is not statistically significant, i.e. we cannot reject the null hypothesis that the performances of the two methods follow the same distribution. As a general trend, our model excels with larger instances and problem variants with more intricate neighbourhood structures, emphasising the crucial role of operator selection.

6.3.7 Extrapolation to Larger Instances

To further measure the extrapolation ability of the learning-based approaches, we evaluate the models trained on Solomon instances (with instance size available up to 100) on an unseen set of larger-scale GH instances from Gehring and Homberger [75] with up to 1000 nodes. The GH dataset follows similar structure to Solomon dataset with the same C, R, RC category but with different node locations and instance sizes. We consider 128 sets of tours each for instance sizes in $[200, 400, 600, 800, 1000]$ of the RC type. Out of the Deep RL methods, we only include the Hybrid method in this experiment as the results in Table 6.5 and Table 6.6 show it has the best performance.

In Figure 6.7, we observe that the average objective values of LRW are better than those of the RAN and CRW models, obtaining an average improvement of 31.09% and 22.24% respectively across all problem variants and instance sizes. This is consistent

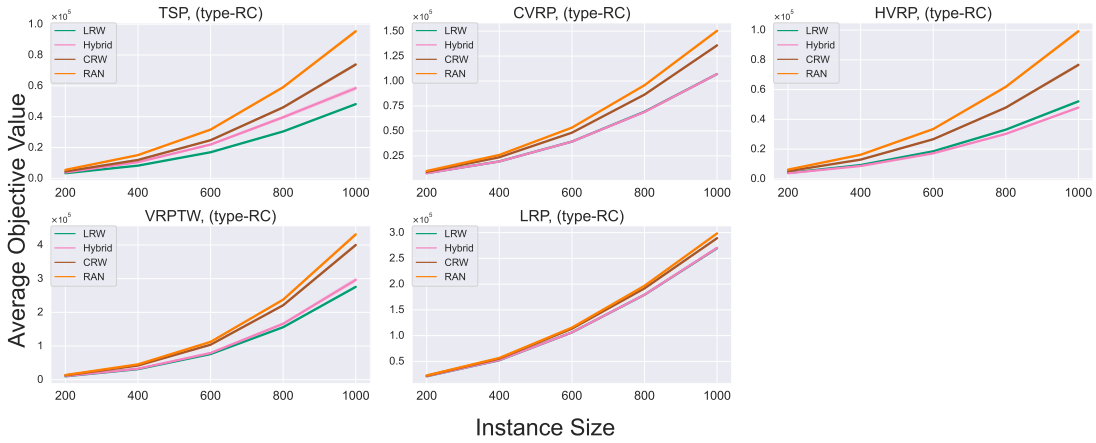


Figure 6.7: Average objective value from 128 seeds during ALNS evaluations for the LRW, CRW, and RAN models on 5 VRP variants and large-scale GH dataset RC category. Lower is better.

with the results on the Solomon dataset. Comparing Hybrid to RAN and CRW, we obtain improvements of 28.45% and 19.03%, respectively. We observe a slight degradation in performance of the Deep RL models compared to the Solomon dataset due to their limitations in out-of-distribution tasks. Moreover, aside from different initial solutions, the model is transferred to larger problem instances with different settings including a brand new set of customer locations. The Hybrid model is nonetheless able to outperform the LRW model for HVRP and VRPTW, while obtaining similar performance to LRW for CVRP and LRP.

6.3.8 Computational Time

The training process for a complete Solomon instance requires approximately 5 days using our reference pure Python implementation on a single core of a commodity CPU. Our training sets consist of small-scale instances up to 100 nodes, which entail relatively low computational demands manageable by CPU facilities. Consequently, we found that CPU is more power-efficient than GPU, whose benefits don't stand out despite the its greater computational power and parallel processing capabilities.

Concretely, we utilised Intel Xeon E5-2637 v4 processors for reporting the measurements in this section. Throughout the evaluation process, the trained agent executes tasks in an offline manner based on the policy formed during the training process and compares its performance against other learning or non-learning-based methods for operator selection. Most of the computational expense during the training process arises not from the model weight adjustment, but from the periodic validation of the model, which requires applying the operators to all solutions in the validation set to assess performance. Assumingly, the training cost can be reduced drastically with a substantially more efficient implementation of the operator selection environment in a lower-level programming language. Given training times are dominated by the validation costs, GRLOS, LRW, and RLB all require similar timeframes for training despite their different learning mechanisms.

The evaluation times within the ALNS framework are shown in Table 6.8. The computational time for a single iteration within the ALNS framework mostly takes less than 5 seconds. The maximum computational time per round is for VRPTW

Table 6.8: The average computation time in second per individual ALNS evaluation round with $B = 20$ operators and 100 customers for different problem variants and Solomon instance types.

variant	type	GRLOS	GRLOS-M	LRW	Hybrid	RAN	CRW	RLB	RLB-M
TSP	RC	0.87	0.58	0.50	0.58	0.47	0.51	0.22	0.24
	C	0.85	0.63	0.50	0.63	0.53	0.56	0.29	0.31
	R	0.86	0.58	0.35	0.53	0.52	0.56	0.24	0.26
CVRP	RC	0.75	0.52	0.57	0.63	0.51	0.56	0.23	0.25
	C	0.61	0.48	0.56	0.55	0.50	0.55	0.37	0.48
	R	0.83	0.54	0.48	0.60	0.55	0.55	0.58	0.53
HVRP	RC	1.64	1.36	1.10	1.27	1.08	1.10	0.99	0.99
	C	1.77	1.35	1.12	1.38	1.01	1.11	1.56	1.28
	R	1.61	1.40	1.26	1.52	1.11	1.19	1.32	1.10
VRPTW	RC	3.18	2.85	2.39	2.72	2.23	2.43	2.77	2.25
	C	2.58	2.35	2.40	2.47	2.00	2.29	3.06	3.23
	R	1.97	1.78	2.57	2.55	2.22	2.41	3.18	2.39
LRP	RC	0.68	0.53	0.64	0.65	0.57	0.62	0.28	0.68
	C	0.69	0.48	0.60	0.64	0.56	0.59	0.67	1.14
	R	0.92	0.74	0.55	0.71	0.58	0.61	0.21	0.37

problems, partially due to the slightly longer time required for the specific destroy and repair operators. We observe that the GRLOS-based models are slightly more time-consuming than the others, yet they involve state-based information that helps to improve the training and evaluation performance at relatively small runtime costs. GRLOS-M computational times are strictly better than GRLOS since it performs a simple sampling instead of the neural network forward pass beyond the training horizon.

We observe that while the learning-based approaches (GRLOS, LRW, Hybrid), which entail evaluating the GNN for a solution, may require considerably more training time compared to the relatively straightforward online operator choice in the classical ALNS framework (CRW). However, the learning-based models, once gone through training, can execute offline operator selection exhibit no significant variance in computational time compared to the CRW, where the difference mostly arises from discrepancies in computational time among different operators. Furthermore, previous experiments Table 6.5 and Table 6.6 demonstrate that the learning-based approaches consistently yield better solutions, thus justifying the additional training time invested.

6.4 Conclusion and Future Direction

In this chapter, we extended the learning-based operator selection framework in the ALNS metaheuristic to address various routing problems. Based on formulating the framework as a Markov Decision Process, we proposed two mechanisms for learning operator selection policies: GRLOS, which combines an RL technique with Graph Neural Network representations of the state, and LRW, a simple learning-based approach inspired by the classic Roulette Wheel update. The two approaches are complementary: GRLOS is conditioned on the decision space characteristics of the current solution, while the LRW is independent of the current solution and more lightweight as a result.

We have observed the following insights from the evaluation. First of all, GRLOS and LRW both consistently outperform the classic Roulette Wheel update and uniform random operator selection. GRLOS is able to learn strategies that are statistically significantly more performant than those produced by LRW when the distribution of solutions for evaluation matches that seen during training, but may suffer in perfor-

mance under distribution shifts. A hybrid approach combining GRLOS and LRW also yields better performance on average than either method alone. Noticeably, GRLOS is shown to outperform another RL-based method proposed by Kallestad et al. [117] and Reijnen et al. [197] in 90.4% of cases. Furthermore, the findings extend to initial solutions generated using the Clarke-Wright heuristic, and successfully generalise to very much problem instances with up to 1000 nodes in the case of LRW.

The following directions are highlighted for future work. Firstly, applications to other combinatorial optimisation problems for which ALNS is amenable can be considered. This requires the specification of action spaces $\mathcal{D}$ and $\mathcal{R}$ corresponding to problem-relevant destroy and repair operators, which may be designed or learned. Moreover, it also requires defining the node-wise feature vectors $\mathbf{x}$. Herein, as done in the classic ALNS framework, we have considered a pre-defined destroy scale for the operators. Further work may consider incorporating the choice of destroy scale into the decision-making process for the RL agent, which may yield finer-grained control. Another potential aspect is to incorporate the ALNS search history like the iteration number or formerly applied operators as additional information during the training process without the violation of the Markov Property. Finally, the robustness of RL approaches when applied to states different from those seen during training remains an open question, but an important one to address if hybrid approaches combining metaheuristics and machine learning are to be successful.

Appendix A

Supplementary Material

A.1 Supplementary Material for Chapter 3

Table A.1: H-SARA Table of symbols (in alphabetical order)

Symbol	Definition	Description
a_i	Service team's arrival time at customer i	Continuous variable
A	Arc set enclosing all edges linking any two nodes	Non-negative integer
AM_i	Activity measure for customer i	Non-negative parameter
$\mathsf{AM}_{\min}$	Minimum percentage of activity measure in a district	Non-negative parameter
$\mathsf{AM}_{\max}$	Minimum percentage of activity measure in a district	Non-negative parameter
b_i	Number of closest customers to i	Positive integer
c_{wait}	Customer waiting cost (per unit time)	Non-negative parameter
c_{idle}	Team idling cost (per unit time)	Non-negative parameter
c_{over}	Team overtime cost (per unit time)	Non-negative parameter
d_{ij}	Travel (Euclidean) distance between i and j	Non-negative parameter
e_i	Expected travel time from customer i to j (same cluster)	Positive number
f_m	Cost for hiring any service team m	Positive parameter
g_i	Overtime length returning to depot from customer i	Continuous variable
h_i	Idle time of service team at customer i	Continuous variable
$\mathbb{I}$	Set of customers with service cancellations	Non-negative integer set
I	Customers set	Non-negative integer set
i, j	Node index	Non-negative integer
ℓ_l	Lower bound on the number of teams	Non-negative integer
ℓ_u	Upper bound on the number of teams	Non-negative integer
L	Standard maximum allowed working time	Positive parameter
L_δ	Safety time at the end of each tour	Positive parameter
m	Service team index	Non-negative integer
$\hat{m}$	Upper limit for the service team	Non-negative parameter
M	Big M	Positive integer
n	Number of customers	Non-negative parameter
q_i	cancellation probability of customer i	Non-negative parameter
$p_{ij}^{(*)}$	Likelihood of customer j following i on a route	Non-negative number
q_ω	Scenario ω's associated probability	non-negative parameter
R_{ij}	Rank of the j^{th} closest customer to i,	Positive integer
s_i^ω	Service time at customer i under scenario ω	Positive parameter
$\hat{s}_i$	Expected service time at customer i	Positive number
S	Stochastic service duration vector	non-negative parameter
t_i	Appointment time of customer i	Continuous variable
τ_{ij}^ω	Traversal time from node i to j under scenario ω	Positive parameter
$\hat{\tau}_{ij}$	Expected traversal time from node i to j	Positive number
T	Stochastic traversal duration matrix	non-negative parameter
T_1	First-stage customer appointment time window (length)	Positive parameter
T_2	Second-stage customer appointment time window (length)	Positive parameter
v_{ij}	Traversal speed from node i to j	Positive number
$\hat{v}_{ij}$	Expected travelling speed from node i to j	Positive number
V	Node set enclosing all customer nodes	Non-negative integer
w_i	Waiting time for customer i	Continuous variable
W	Appointment time window half width	Continuous variable
x_{ij}	Determines if arc (i, j) is traversed by a team	Integer (binary) variable
y_{ij}	If customer i is assigned to district centred at j	Integer (binary) variable
ω	Scenario with travel and service times realisations	samples
Ω	Total set of scenarios $[\omega]$	samples set
θ	Maximum overtime length	Non negative parameter
λ	Unit travel time cost (per hour)	Positive parameter
μ	Average activity measure per district	Non-negative number

Table A.2: HHC papers with a focus on routing and scheduling and uncertainties

Entry		HHC Decisions				Uncertainty			Solution Method	
Author	Year	Fleetsize	Assignment	Routing	Scheduling	Travel Time	Service Time	Cancellation	Modelling Approach	Solution Approach
Begur	1997		*	*	*				MIP	Multiple heuristics
Cheng	1997		*	*	*				MIP	Two-phase heuristic
De Angelis	1998		*				*		Stochastic LP	NA
Blais	2003		*						MIP	Tabu Search
Bertels	2006		*	*	*				MIP	Combination of constraint programming, tabu search and simulated annealing
Eveborn	2006		*	*	*			*	IP, Set Partitioning	Repeated matching algorithm
Akjiratikarl	2007		*	*	*				IP	Particle Swarm Optimisation Meta-heuristic
Eveborn	2009		*	*	*			*	IP, Set Partitioning	Repeated matching algorithm
Hertz	2009		*						MIP	Commercial solver (CPLEX), Meta-heuristic based on Tabu Search
Bennett	2011			*	*				Combinatorial optimisation	Greedy heuristic
Trautsamwieser	2011		*	*	*			*	IP	Meta-heuristic (VNS)
Trautsamwieser	2011		*	*	*				IP	Meta-heuristic (VNS)
An	2012			*	*				MIP	Two-phase heuristic
Bennett	2012		*	*	*				MIP	Evaluations
Koeleman	2012	*	*	*	*				Markov decision process	Heuristic
Lanzarone	2012		*				*		Stochastic Programming	Structural policy
Lanzarone	2012		*				*		(Stochastic) Integer programming	Not specified - OPL 5.1 solver used
Nickel	2012		*	*	*			*	Constraint programming	two-phase method based on Constraint Programming and ALNS metaheuristic
Rasmussen	2012		*	*	*				IP (Set Partitioning formulation)	Branch-and-Price
Shao	2012		*	*	*				MIP	Greedy randomised adaptive search procedure (GRASP)
Allaoua	2013	*	*	*	*				ILP	decomposition-based heuristic
Bard	2013		*	*					MIP	Commercial solver and Heuristics
Benzarti	2013		*						MIP	Commercial solver (CPLEX)
Liu	2013		*	*	*				MIP	tabu search and genetic algorithm
Bard	2014		*	*	*				MIP	Heuristic based on a greedy randomised adaptive search procedure (GRASP)
Bard	2014		*	*	*				MIP	Branch-cut-and-price, rolling horizon heuristic
Carello	2014		*				*		Robust Optimisation	Commercial solver (CPLEX)
Kergosien	2014		*	*	*				MIP	Tabu-search + Variable neighbourhood search
Lanzarone	2014		*				*		Analytical	Analytical policy
Liu	2014		*	*	*				MIP	Tabu Search
Mankowska	2014		*	*	*				MIP	Heuristic
Trautsamwieser	2014		*	*	*				MIP	Branch-and-Price-and-Cut
Yalçındağ	2014		*	*		*			MIP, Kernel Regression	Assignment-first-routing-second and simutaneous policies
Bowers	2015		*	*	*				Combinatorial optimisation	Modified Clarke-Wright algorithm, simulation
Cappanera	2015		*	*	*				ILP	Branch and Bound
Fikar	2015		*	*	*				IP	two-stage matheuristic based on tabu search
Hiermann	2015		*	*	*				Constraint programming	Two-stage heuristic
Maya	2015		*	*	*				Bi-objective MIP	Three-stage heuristic
Nguyen	2015		*	*	*			*	Robust Optimisation	Matheuristic
Rodriguez	2015	*					*		Stochastic Proramming	Branch-and-cut
Rest	2015		*	*	*				MIP	Meta-heuristic approach based on tabu search Tabu search-based metaheuristic
Yuan	2015		*	*	*		*		Stochastic Programming	Branch-and-price
Braekers	2016		*	*	*				MIP	Meta-heuristic
Decerle	2016		*	*	*				MIP	Two-phase metaheuristic
Errarhout	2016		*				*		Stochastic Programming	Commercial solver (CPLEX)
Fikar	2016		*	*	*			*	IP	Discrete-event driven metaheuristic
Lin	2016		*						MIP	Commercial solver (Gurobi)
López	2016		*	*	*			*	Multi-agent approach	Commercial solvers (CPLEX, JADE)
Redjem	2016			*	*				MIP	Heuristic Approach

Table A.2 (continued)

Entry		HHC Decisions				Uncertainty			Solution Method	
Author	**Year**	**Fleetsize**	**Assignment**	**Routing**	**Scheduling**	**Travel Time**	**Service Time**	**Cancellation**	**Modelling Approach**	**Solution Approach**
Rest	2016		*	*	*	*			MIP	Combination of dynamic programming and meta-heuristics based on tabu search
Wirnitzer	2016		*		*				MIP	Commercial solver (Gurobi)
Yalçındağ	2016		*	*	*				ILP	Patter-based two-phase approach
Du	2017		*	*	*				MIP	Genetic algorithm
Guericke	2017		*	*	*				MIP	Meta-heuristic approach based on adaptive large neighbourhood search (ALNS)
Han	2017		*	*	*		*	*	MIP	hybrid heuristic algorithm (DP + TS)
Laesanklang	2017		*	*	*				MIP	Heuristic decomposition
Lin	2017		*						MIP	Greedy heuristic
Liu	2017		*	*	*				MIP	Branch-and-price
Shi	2017		*	*	*		*		MIP + fuzzy chance constraint	Hybrid genetic algorithm
Shi	2017		*	*	*	*	*		Stochastic Programming	Hybrid genetic algorithm
Yuan	2017			a priori	a priori			*	IP	Heurisitic
Cappanera	2018		*	*	*			*	Robust Optimisation	Metaheuristic
Carello	2018		*				*		MIP	Commercial solver (CPLEX)
Decerle	2018		*	*	*				MIP	Memetic Algorithm
Issabakhsh	2018		*	*	*	*			MIP + Robust optimisation	uncertainty level analysis
Lin	2018		*	*	*			*	MIP	Heuristic
Liu	2018		*	*	*				Bi-objective MIP	Metaheuristic
Nasir	2018	*	*	*	*				MIP	heuristic algorithm based on VNS approach
Nasir	2018	*	*	*	*				ILP	Commercial solver (CPLEX)
Shi	2018		*	*	*	*	*		Stochastic Programming	Commercial solver (CPLEX) & SA-based heuristic algorithm
Yuan	2018		*	*	*	*	*		Stochastic Programming	Branch-and-Price
Zhan	2018		*	*	*		*		Stochastic Programming, MIP	Tabu Search-based heuristic
Demirbilek	2019		*	*	*			*	MIP	Scenario Based heuristic Approach
Grenouilleau	2019		*	*	*				MIP	set partitioning heuristic
Heching	2019		*		*				MIP, constraint programming	Logic-based Benders Decomposition, Branch-and-check
Liu	2019		*	*	*	*	*		IP	Branch-and-Price
Shi	2019		*	*	*	*	*		Robust Optimisation	Gurobi Solver, Heuristics (Simulated Annealing, Tabu Search, and Variable Neighbourhood Search)
Frifita	2020		*	*	*				MIP	Variable Neighbourhood Search
Grenouilleau	2020		*		*				Two-stage formulation	Logic-based Benders Decomposition, LNS meta-heuristic
Kandakoglu	2020		*	*	*				MIP	Specific decision support system (HDSS)
Restrepo	2020	*	*		*		*		Stochastic Programming	Commercial solver (CPLEX)
Shahedah	2020	*	*	*	*		*		Robust Optimisation	decomposition-based algorithm
Zhan	2020		*	*	*		*		MIP	L-shapted Method, MTSP heuristic
Bazirha	2021		*	*	*	*	*		Stochastic Programming with Recourse	Commercial solver (CPLEX) , genetic algorithm (GA), general variable neighbourhood search (GVNS)
Carello	2021		*				*		MIP, Robust Optimisation	Commercial solver (CPLEX)
Cinar	2021		*	*	*				MIP	Adaptive Large Neighbourhood Search (ALNS) and iterative mathematical programming
Demirbilek	2021		*	*	*			*	MIP	Scenario Based heuristic Approach
Khodabandeh	2021		*	*	*				Bi-objective optimisation	Epsilon-constraint-based approach
Li	2021		*	*	*				nonlinear & convex programming	"outer-approximation method, genetic algorithm "
Liu	2021		*	*	*				MIP	Matheuristic integrating (adaptive large Neighbourhood search, ALNS) heuristic and commercial solver (Gurobi)
Liu	2021		*	*	*				MIP	Hybrid metaheuristics
Naderi	2021	*	*	*	*	*	*		MIP	Benders Decomposition (logic-based Benders branching-decomposition algorithm)
Nikzad	2021	*	*	*	*	*	*		Stochastic Programming	Multi (two)-phase matheuristic
Bushehri	2021		*	*	*	*	*		robust Optimisation	Meta-heuristic (simulated annealing, genetic algorithm, memetic algorithm)
Shiri	2021		*	*	*		*		Robust Optimisation	Hybrid three-phase procedure
Xiang	2021		*	*	*				Bi-objective MIP	Hybrid elitist nondominated sorting genetic algorithm (hybrid NSGA-II)
Yang	2021		*	*	*	*	*		Uncertain Programming	Metaheuristic
Yadav	2022		*	*	*				MIP	Metaheuristic
Ours	**2023**	*	*	*	*	*	*	*	MIP	**Benders Decomposition, two-stage heuristic**

A.2 Supplementary Material for Chapter 5

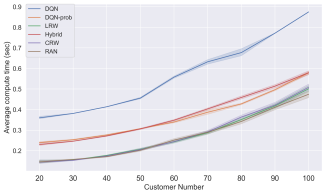
Table A.3: ALNS destroy and repair operators summary. * means the operator mechanism has been adjusted to fit specific neighbourhood type (e.g., time to demand-based) without changing the mechanism itself.

Operator Type	Detailed operator	Description	Reference
Random-based Destroy Operators	Random Node Destroy	Removes a set of q randomly-selected nodes from their existing routes and placed inside the customer pool.	Ropke and Pisinger [201]
	Random Route Destroy	Randomly and iteratively selects a route from the solution and remove a set of q nodes sequentially from the route to the customer pool based on the visiting sequence, repeat this process until the destroy budget is met.	Demir et al. [55]
	Forbidden Random Destroy	Randomly removes a set of q nodes that have not been removed from the solution for a minimum number of times in the preceding iterations.	Li et al. [143]
	Close-an-Open-Facility Destroy	Randomly closes an open facility and removes all its allocated nodes from the solution to the customer pool. Bypass the destroy budget requirement if the number of allocated customers exceeds the destroy scale.	Hemmelmayr et al. [101]
	Open-a-Closed-Facility Destroy	Randomly opens a closed facility and removes a set of q nodes from the existing routes that are located in the immediate geographical vicinity of this facility and places them inside the customer pool.	Hemmelmayr et al. [101]
Greedy-based Destroy Operators	Worst Distance Destroy	Removes a set of q nodes with the highest removal gains computed from the geographical distance to their successive nodes.	Ropke and Pisinger [201]
	Route Deviation Destroy	Removes a set of q nodes from the routes which total distances exhibit the highest deviation from the average route length.	Demir et al. [55]
	Greedy Route Longest Destroy	Removes a set of q nodes from the worst routes with the longest total travelling distance	Demir et al. [55]*
	Greedy Route Shortest Destroy	Removes a set of q nodes from the worst routes with the shortest total travelling distance	Demir et al. [55]*
	Historical Knowledge Destroy	Removes a set of q nodes with the highest travel distance discrepancies from their successive nodes based on the historical record and place them inside the customer pool.	Alinaghian and Shokouhi [10]
	Largest Demand Deviation Destroy	Removes a set of q nodes with their demands exhibiting the highest deviation from the average.	Demir et al. [55]*
	Smallest Demand Destroy	Removes a set of q nodes with the smallest demands from the population.	Demir et al. [55]
	Largest Demand Destroy	Removes a set of q nodes with the highest demands from the population.	Demir et al. [55]
	Worst Facility Node Destroy	Randomly selects and removes a set of q nodes with the largest removal gain from the same facility until the destroy budget is met.	Hemmelmayr et al. [101]*
	Worst Time Destroy	Removes a set of q nodes with the most significant deviations in service starting time from the average estimate and places them inside the customer pool.	Keskin and Çatay [124]*
Related-based Destroy Operators	Proximity Distance Destroy	Randomly selects a single node and removes it together with a set of $q-1$ nodes that are closely associated in terms of the highest geographical proximity.	Lehuédé et al. [142]
	Pair Destroy	Randomly selects a set of $q/2$ nodes to place inside the customer pool, then for each node removes its geographically nearest paired acquaintance that has not yet been removed from the route.	Mancini [157]
	Cluster Destroy	Ranks pairs of nodes according to their inter-travelling distance in ascending order and sequentially removes nodes with the smallest distance that still persist within the solution. This algorithm is based on the Kruskal's algorithm for generating minimum spanning trees, to separate two or multiple groups of nodes that are geographically far away from each other.	Pisinger and Ropke [184]
	Node Neighbourhood Destroy	Randomly selects a single node and removes it together with a set of $q-1$ nodes that lie within a rectangular boundary surrounding the selected node.	Alinaghian and Shokouhi [10]
	Zone Destroy	Randomly selects a single node and removes it together with $q-1$ nodes within a predefined region using the Cartesian coordinate system encompassing the node.	Emeç et al. [66]
	Route Neighborhood Destroy	Randomly chooses a pair of routes and iteratively removes a pair of customers from each route with the shortest inter-node travelling distance until the destroy budget is met.	Emeç et al. [66]
	Historical Pair Destroy	Compute the number of times the any two nodes x and y have been served by the same vehicle. This is used as the related weight in Shaw removal.	Tellez et al. [227]
	Shaw Destroy	Removes a random single node and the remaining $q-1$ nodes with the highest relatedness measures to the single node and places inside the customer pool. Relatedness is computed using a weighted sum of $\alpha \times$ geographical distance $+ \gamma \times$ same vehicle status $+ \delta \times$ demand difference, with an additional term $\beta \times$ start time difference if VRPTW variant is applied.	Ropke and Pisinger [201]
	Proximity Demand Destroy	Removes a random single node and a set of $q-1$ nodes with the most similar demand level and places inside the customer pool.	Demir et al. [55]
	Facility Swap Destroy	Exchange the statuses of a pair of open and closed facilities with a probability inversely proportional to their geographical travelling distance.	Hemmelmayr et al. [101]
	Proximity Time Destroy	Removes a random single node and a set of $q-1$ nodes that have the most similar service starting times and places inside the customer pool.	Demir et al. [55]
	Proximity Service Destroy	Removes a random single node and a set of $q-1$ nodes that have the most similar service lengths and places inside the customer pool.	Pisinger and Ropke [184]
	Do Nothing Operator	Make no change to the current graph simply pass on the current solution to the next state.	(Our work)
Repair Operators	Random Repair	Randomly selects a node and insert it in a random position during each iteration until the customer pool is empty.	Qu and Bard [190]
	Greedy Repair	During each iteration, randomly picks a node from the customer pool and place it in the position that minimally increases the total costs. The insertion can be between two consecutive customer nodes or between the facility and its linking customer node.	Ropke and Pisinger [201]
	Greedy Perturbation Repair	Incorporates the identical mechanism as previously described under Greedy Repair. The insertion cost at each specific position is influenced by a perturbation factor d randomly drawn between [0.8, 1.2].	Contardo et al. [41]
	Deep Greedy Repair	Ranks the nodes inside customer pool from the lowest to highest using the insertion cost, then insert the nodes in accordance with the specified order using Greedy Repair.	Laporte et al. [140]
	Regret-2 Repair	Ranks the nodes inside customer pool from the highest to lowest using the regret value, which is computed as the insertion cost difference for the first and k-th cheapest position, where k = 2.	Ribeiro and Laporte [199]
	Regret-3 Repair	Same as above with k=3	Ribeiro and Laporte [199]
	Regret-4 Repair	Same as above with k=4	Ribeiro and Laporte [199]

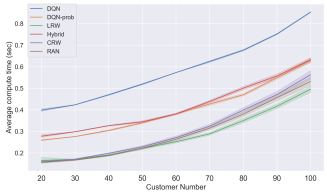
Table A.4: Factorial Analysis Operator Set Summary

Methods	1	2	3	4	5	6	7	8	9	10	11	12
Random Node Destroy			✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
Worst Distance Destroy		✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
Proximity Distance Destroy	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
Node Neighbourhood Destroy				✓			✓	✓	✓	✓	✓	✓
Zone Destroy							✓	✓	✓	✓	✓	✓
Pair Destroy											✓	✓
Route Neighbourhood Destroy					✓	✓	✓	✓	✓	✓	✓	✓
Cluster Destroy						✓			✓	✓	✓	✓
Greedy Route Longest Destroy						✓		✓	✓	✓	✓	✓
Greedy Route Shortest Destroy												✓
Historical Pair Destroy										✓	✓	✓
Random Route Destroy					✓		✓	✓	✓	✓	✓	✓

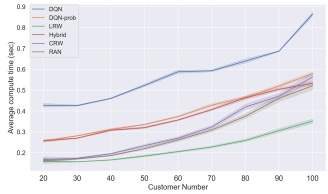
A.3 Supplementary Material for Chapter 6



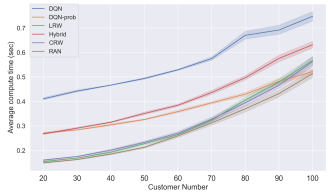
(a) TSP, Solomon RC



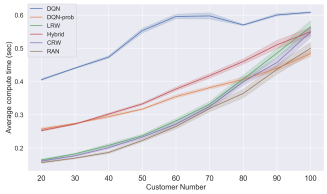
(b) TSP, Solomon C



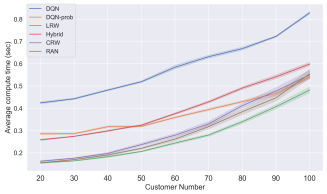
(c) TSP, Solomon R



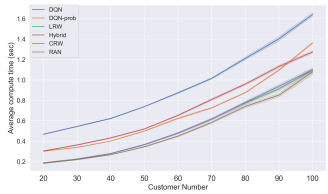
(d) CVRP, Solomon RC



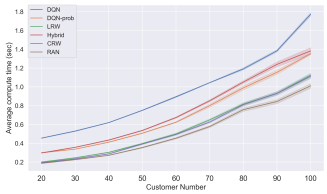
(e) CVRP, Solomon C



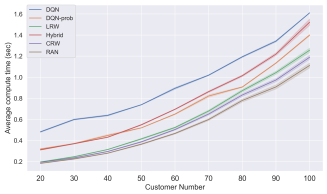
(f) CVRP, Solomon R



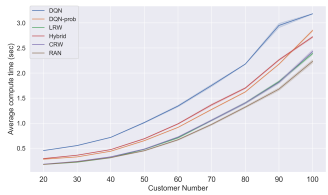
(g) HVRP, Solomon RC



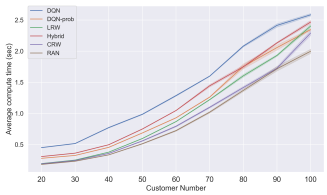
(h) HVRP, Solomon C



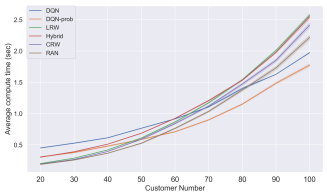
(i) HVRP, Solomon R



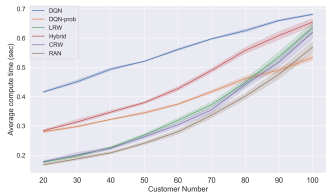
(j) VRPTW, Solomon RC



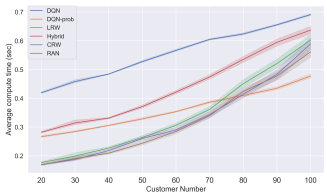
(k) VRPTW, Solomon C



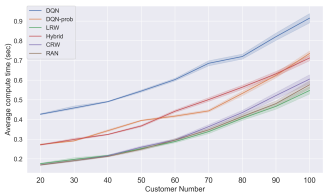
(l) VRPTW, Solomon R



(m) LRP, Solomon RC

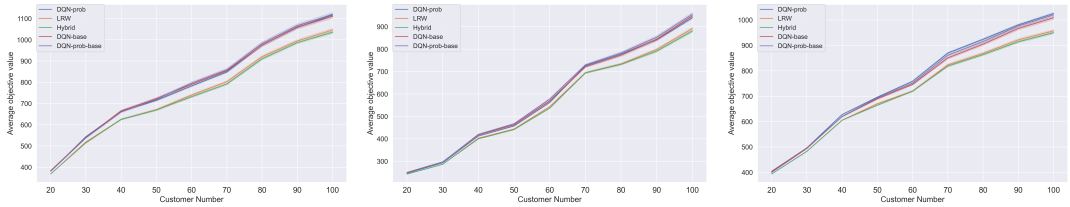


(n) LRP, Solomon C

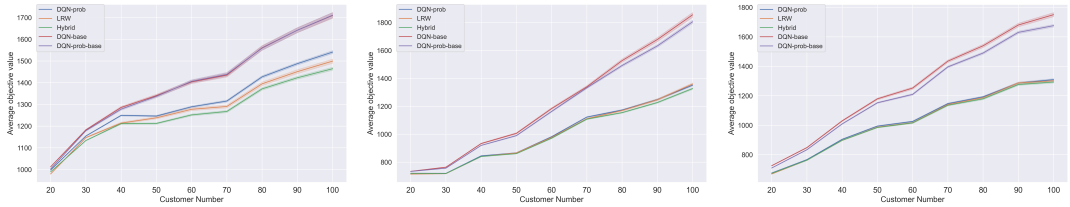


(o) LRP, Solomon R

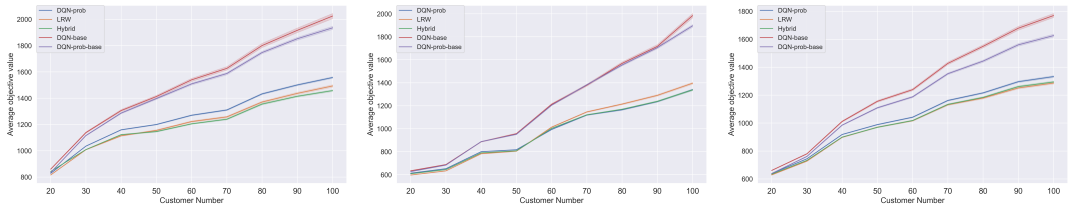
Figure A.1: Average computation time from 128 seeds during ALNS evaluations for the LRW, CRW, and RAN models on 5 VRP variants and Solomon dataset with 3 categories. Lower is better.



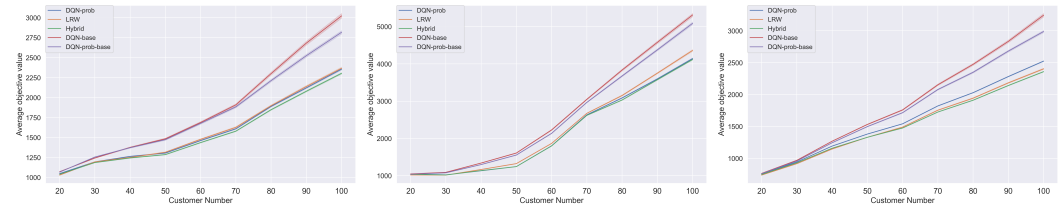
(a) TSP, Solomon RC (b) TSP, Solomon C (c) TSP, Solomon R



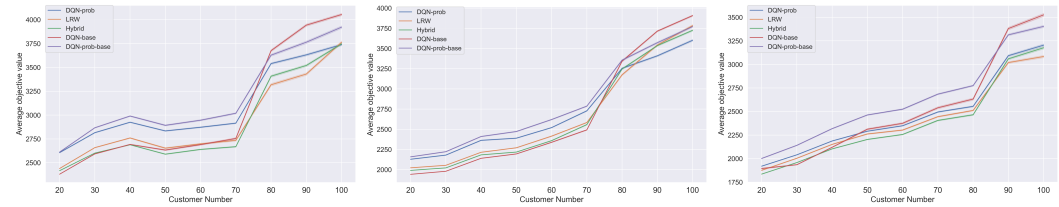
(d) CVRP, Solomon RC (e) CVRP, Solomon C (f) CVRP, Solomon R



(g) HVRP, Solomon RC (h) HVRP, Solomon C (i) HVRP, Solomon R



(j) VRPTW, Solomon RC (k) VRPTW, Solomon C (l) VRPTW, Solomon R



(m) LRP, Solomon RC (n) LRP, Solomon C (o) LRP, Solomon R

Figure A.2: Average objective value from 128 seeds during ALNS evaluations for the DQN-based, LRW, and baseline models (DQN-baseline and DQN-prob-baseline from Kallestad et al. [117]) on 5 VRP variants and 3 Solomon dataset categories. Lower is better.

Bibliography

[1] R. Addanki, V. Nair, and M. Alizadeh. Neural large neighborhood search. In *NeurIPS LMCA Workshop*, 2020.

[2] Y. Adulyasak, J.-F. Cordeau, and R. Jans. Optimization-based adaptive large neighborhood search for the production routing problem. *Transportation science*, 48(1):20–45, 2014.

[3] E. Ahmadi, B. Goldengorin, G. A. Süer, and H. Mosadegh. A hybrid method of 2-tsp and novel learning-based ga for job sequencing and tool switching problem. *Applied Soft Computing*, 65: 214–229, 2018.

[4] S. Ahn, Y. Seo, and J. Shin. Learning what to defer for maximum independent sets. In *ICML*, 2020.

[5] R. K. Ahuja, Ö. Ergun, J. B. Orlin, and A. P. Punnen. A survey of very large-scale neighborhood search techniques. *Discrete Applied Mathematics*, 123(1-3):75–102, 2002.

[6] Ö. Ş. Akpunar and Ş. Akpinar. A hybrid adaptive large neighbourhood search algorithm for the capacitated location routing problem. *Expert Systems with Applications*, 168:114304, 2021.

[7] D. Aksen, O. Kaya, F. S. Salman, and Ö. Tüncel. An adaptive large neighborhood search algorithm for a selective and periodic inventory routing problem. *European Journal of Operational Research*, 239(2):413–426, 2014.

[8] M. Albareda-Sambola, J. A. Díaz, and E. Fernández. A compact model and tight bounds for a combined location-routing problem. *Computers & Operations Research*, 32(3):407–428, 2005.

[9] M. Albareda-Sambola, E. Fernández, and S. Nickel. Multiperiod location-routing with decoupled time scales. *European Journal of Operational Research*, 217(2):248–258, 2012. ISSN 0377-2217.

[10] M. Alinaghian and N. Shokouhi. Multi-depot multi-compartment vehicle routing problem, solved by a hybrid adaptive large neighborhood search. *Omega*, 76:85–99, 2018.

[11] D. Ambrosino and M. Grazia-Scutellà. Distribution network design: New problems and related models. *European Journal of Operational Research*, 165(3):610–624, 2005.

[12] D. Ambrosino and A. Sciomachen. A food distribution network problem: a case study. *IMA Journal of Management Mathematics*, 18(1):33–53, 2007. ISSN 1471-678X.

[13] D. Ambrosino, A. Sciomachen, and M. G. Scutellà. A heuristic based on multi-exchange techniques for a regional fleet assignment location-routing problem. *Computers & Operations Research*, 36(2):442–460, 2009. ISSN 0305-0548.

[14] A. Anderluh, P. C. Nolz, V. C. Hemmelmayr, and T. G. Crainic. Multi-objective optimization of a two-echelon vehicle routing problem with vehicle synchronization and 'grey zone' customers arising in urban logistics. *European Journal of Operational Research*, 2019.

[15] R. Bai, X. Chen, et al. Analytics and machine learning in vehicle routing research. *International Journal of Production Research*, pages 1–27, 2021.

[16] R. Baldacci, A. Mingozzi, and R. Wolfler Calvo. An exact method for the capacitated location-routing problem. *Operations Research*, 59(5):1284–1296, 2011.

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

[18] J. F. Bard and A. I. Jarrah. Large-scale constrained clustering for rationalizing pickup and delivery operations. *Transportation Research. Part B: Methodological*, 43(5):542–561, 2009.

[19] E. Barrena, D. Canca, L. C. Coelho, and G. Laporte. Single-line rail rapid transit timetabling under dynamic passenger demand. *Transportation Research. Part B: Methodological*, 70:134–150, 2014.

[20] J.-M. Belenguer, E. Benavent, C. Prins, C. Prodhon, and R. Wolfler Calvo. A branch-and-cut method for the capacitated location-routing problem. *Computers & Operations Research*, 38(6): 931–941, 2011.

[21] I. Bello, H. Pham, Q. V. Le, M. Norouzi, and S. Bengio. Neural combinatorial optimization with reinforcement learning. In *ICLR Workshops*, 2016.

[22] I. Ben Mohamed, W. Klibi, and F. Vanderbeck. Designing a two-echelon distribution network under demand uncertainty. *European Journal of Operational Research*, 280(1):102–123, 2020. ISSN 0377-2217.

[23] M. Bender, J. Kalcsics, and A. Meyer. Districting for parcel delivery services–a two-stage solution approach and a real-world case study. *Omega*, 96:102283, 2020.

[24] J. F. Benders. Partitioning procedures for solving mixed-variables programming problems. *Numerische Mathematik*, 4(1):238–252, 1962.

[25] Y. Bengio, A. Lodi, and A. Prouvost. Machine learning for combinatorial optimization: a methodological tour d'horizon. *European Journal of Operational Research*, 290(2):405–421, 2021.

[26] C. M. Bishop and N. M. Nasrabadi. *Pattern recognition and machine learning*, volume 4. Springer, 2006.

[27] M. Boccia, T. G. Crainic, A. Sforza, and C. Sterle. Location-routing models for designing a two-echelon freight distribution system. *Rapport technique, CIRRELT, Université de Montréal*, 91, 2011.

[28] C. Bongiovanni, M. Kaspi, J.-F. Cordeau, and N. Geroliminis. A machine learning-driven two-phase metaheuristic for autonomous ridesharing operations. *Transportation Research. Part E: Logistics and Transportation Review*, 165:102835, 2022.

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

[30] U. Breunig, R. Baldacci, R. Hartl, and T. Vidal. The electric two-echelon vehicle routing problem. *Computers & Operations Research*, 103:198–210, 2019. ISSN 0305-0548.

[31] P. Cappanera, M. G. Scutellà, F. Nervi, and L. Galli. Demand uncertainty in robust Home Care optimization. *Omega*, 80:95–110, 2018. doi: 10.1016/j.omega.2017.08.012.

[32] G. Carello, E. Lanzarone, and S. Mattia. Trade-off between stakeholders' goals in the home care nurse-to-patient assignment problem. *Operations Research for Health Care*, 16:29–40, 2018.

[33] D. Cattaruzza, N. Absi, D. Feillet, and J. González-Feliu. Vehicle routing problems for city logistics. *EURO Journal on Transportation and Logistics*, 6(1):51–79, 2017.

[34] C. Chao, T. Zhihui, and Y. Baozhen. Optimization of two-stage location–routing–inventory problem with time-windows in food distribution network. *Annals of Operations Research*, 273 (1):111–134, 2019. ISSN 0254-5330.

[35] X. Chen and Y. Tian. Learning to perform local rewriting for combinatorial optimization. *NeurIPS*, 2019.

[36] M. Cherkesly, G. Desaulniers, and G. Laporte. A population-based metaheuristic for the pickup and delivery problem with time windows and lifo loading. *Computers & Operations Research*, 62:23–35, 2015.

[37] C. Chi, A. Aboussalah, E. Khalil, J. Wang, and Z. Sherkat-Masoumi. A deep reinforcement learning framework for column generation. In *NeurIPS*, 2022.

[38] M. Cissé, S. Yalçındağ, Y. Kergosien, E. Şahin, C. Lenté, and A. Matta. Or problems related to home health care: A review of relevant routing and scheduling problems. *Operations Research for Health Care*, 13-14:1–22, 2017. ISSN 2211-6923.

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

[40] G. Codato and M. Fischetti. Combinatorial benders' cuts for mixed-integer linear programming. *Operations Research*, 54(4):756–766, 2006.

[41] C. Contardo, V. Hemmelmayr, and T. G. Crainic. Lower and upper bounds for the two-echelon capacitated location-routing problem. *Computers & Operations Research*, 39(12):3185–3199, 2012.

[42] C. Contardo, J.-F. Cordeau, and B. Gendron. An exact algorithm based on cut-and-column generation for the capacitated location-routing problem. *INFORMS Journal on Computing*, 26 (1):88–102, 2013.

[43] C. Contardo, J.-F. Cordeau, and B. Gendron. A computational comparison of flow formulations for the capacitated location-routing problem. *Discrete Optimization*, 10(4):263–295, 2013. ISSN 1572-5286.

[44] C. Contardo, J.-F. Cordeau, and B. Gendron. A grasp+ ilp-based metaheuristic for the capacitated location-routing problem. *Journal of Heuristics*, 20(1):1–38, 2014.

[45] T. G. Crainic, N. Ricciardi, and G. Storchi. Models for evaluating and planning city logistics systems. *Transportation Science*, 43:432–454, 11 2009.

[46] G. Cybenko. Approximation by superpositions of a sigmoidal function. *Mathematics of control, signals and systems*, 2(4):303–314, 1989.

[47] H. Dai, B. Dai, and L. Song. Discriminative embeddings of latent variable models for structured data. In *ICML*, 2016.

[48] Z. Dai, F. Aqlan, K. Gao, and Y. Zhou. A two-phase method for multi-echelon location-routing problems in supply chains. *Expert Systems With Applications*, 115:618–634, 2019.

[49] G. Dantzig, R. Fulkerson, and S. Johnson. Solution of a large-scale traveling-salesman problem. *Journal of the Operations Research Society of America*, 2(4):393–410, 1954.

[50] G. B. Dantzig and J. H. Ramser. The truck dispatching problem. *Management Science*, 6(1): 80–91, 1959. ISSN 0025-1909.

[51] G. B. Dantzig, D. R. Fulkerson, and S. M. Johnson. Solution of a large-scale traveling-salesman problem. In *50 Years of Integer Programming 1958-2008*, pages 7–28. Springer Berlin Heidelberg, Berlin, Heidelberg, 2009.

[52] V.-A. Darvariu, S. Hailes, and M. Musolesi. Goal-directed graph construction using reinforcement learning. *Proceedings of the Royal Society A: Mathematical, Physical and Engineering Sciences*, 477(2254):20210168, 2021.

[53] C. Dawson. Royal mail day before delivery time notifications launched. https://tamebay.com/2019/04/royal-mail-day-before-delivery-time-notifications-launched.html, 4 2019.

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

[55] E. Demir, T. Bektaş, and G. Laporte. An adaptive large neighborhood search heuristic for the pollution-routing problem. *European Journal of Operational Research*, 223(2):346–359, 2012.

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

[57] M. Deudon, P. Cournut, A. Lacoste, Y. Adulyasak, and L.-M. Rousseau. Learning heuristics for the tsp by policy gradient. In *CPAIOR*, 2018.

[58] M. Di Mascolo, C. Martinez, and M.-L. Espinouse. Routing and scheduling in home health care: A literature survey and bibliometric analysis. *Computers & Industrial Engineering*, page 107255, 2021.

[59] M. Dorigo, M. Birattari, and T. Stutzle. Ant colony optimization. *IEEE computational intelligence magazine*, 1(4):28–39, 2006.

[60] J. P. Q. dos Santos, J. D. de Melo, A. D. D. Neto, and D. Aloise. Reactive search strategies using reinforcement learning, local search algorithms and variable neighborhood search. *Expert Systems with Applications*, 41(10):4939–4949, 2014.

[61] DPD. Guide to dpd. https://www.dpd.co.uk/pdf/dpd_sales_guide_2020_v3.pdf, 2020.

[62] M. Drexl and M. Schneider. A survey of variants and extensions of the location-routing problem. *European Journal of Operational Research*, 241(2):283–308, 2015.

[63] I. Drori, A. Kharkar, W. R. Sickinger, B. Kates, Q. Ma, S. Ge, E. Dolev, B. Dietrich, D. P. Williamson, and M. Udell. Learning to solve combinatorial optimization problems on real-world graphs in linear time. In *2020 19th IEEE International Conference on Machine Learning and Applications (ICMLA)*, pages 19–24. IEEE, 2020.

[64] R. Durgut, M. E. Aydin, and I. Atli. Adaptive operator selection with reinforcement learning. *Information Sciences*, 581:773–790, 2021.

[65] P. Emami and S. Ranka. Learning permutations with sinkhorn policy gradient. *arXiv:1805.07010*, 2018.

[66] U. Emeç, B. Çatay, and B. Bozkaya. An adaptive large neighborhood search for an e-grocery delivery routing problem. *Computers & Operations Research*, 69:109–125, 2016.

[67] P. Eveborn, P. Flisberg, and M. Rönnqvist. Laps care—an operational system for staff planning of home care. *European Journal of Operational Research*, 171(3):962–976, 2006.

[68] P. Eveborn, M. Rönnqvist, H. Einarsdóttir, M. Eklund, K. Lidén, and M. Almroth. Operations research improves quality and efficiency in home care. *Interfaces*, 39(1):18–34, 2009.

[69] J. K. Falkner, D. Thyssens, and L. Schmidt-Thieme. Large neighborhood search based on neural construction heuristics. *arXiv:2205.00772*, 2022.

[70] M. H. Fazel Zarandi, A. Hemmati, S. Davari, and I. Burhan Turksen. Capacitated location-routing problem with time windows under uncertainty. *Knowledge-Based Systems*, 37:480–489, 2013.

[71] T. A. Feo and M. G. Resende. Greedy randomized adaptive search procedures. *Journal of Global Optimization*, 6:109–133, 1995.

[72] C. Fikar and P. Hirsch. Home health care routing and scheduling: A review. *Computers & Operations Research*, 77:86–95, 2017. ISSN 0305-0548.

[73] C. Fikar, A. A. Juan, E. Martinez, and P. Hirsch. A discrete-event driven metaheuristic for dynamic home service routing with synchronised trip sharing. *European Journal of Industrial Engineering*, 10(3):323–340, 2016.

[74] A. I. Garmendia, J. Ceberio, and A. Mendiburu. Neural combinatorial optimization: a new player in the field. *arXiv preprint arXiv:2205.01356*, 2022.

[75] H. Gehring and J. Homberger. A parallel hybrid evolutionary metaheuristic for the vehicle routing problem with time windows. In *EUROGEN99*, 1999.

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

[77] M. Gendreau, G. Laporte, and R. Séguin. An exact algorithm for the vehicle routing problem with stochastic demands and customers. *Transportation science*, 29(2):143–155, 1995.

[78] M. Gendreau, G. Laporte, and R. Séguin. Stochastic vehicle routing. *European Journal of Operational Research*, 88(1):3–12, 1996.

[79] N. Genet, W. Boerma, M. Kroneman, A. Hutchinson, R. B. Saltman, W. H. Organization, et al. *Home care across Europe: current structure and future challenges*. World Health Organization. Regional Office for Europe, 2012.

[80] A. M. Geoffrion. Generalized benders decomposition. *Journal of optimization theory and applications*, 10:237–260, 1972.

[81] B. Golden, A. Assad, L. Levy, and F. Gheysens. The fleet size and mix vehicle routing problem. *Computers & Operations Research*, 11(1):49–66, 1984.

[82] M. I. Gomes and T. R. P. Ramos. Modelling and (re-) planning periodic home social care services with loyalty and non-loyalty features. *European Journal of Operational Research*, 277(1):284–299, 2019.

[83] K. Govindan, A. Jafarian, R. Khodaverdi, and K. Devika. Two-echelon multiple-vehicle location–routing problem with time windows for optimization of sustainable supply chain network of perishable food. *International Journal of Production Economics*, 152:9–28, 2014. ISSN 0925-5273.

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

[85] P. Grangier, M. Gendreau, F. Lehuédé, and L.-M. Rousseau. A matheuristic based on large neighborhood search for the vehicle routing problem with cross-docking. *Computers & Operations Research*, 84:116–126, 2017.

[86] L. Grieco, M. Utley, and S. Crowe. Operational research applied to decisions in home health care: A systematic literature review. *Journal of the Operational Research Society*, pages 1–32, 2020.

[87] D. Gupta and B. Denton. Appointment scheduling in health care: Challenges and opportunities. *IIE Transactions*, 40(9):800–819, 2008.

[88] A. Gutierrez, L. Dieulle, N. Labadie, and N. Velasco. A multi-population algorithm to solve the vrp with stochastic service and travel times. *Computers & Industrial Engineering*, 125:144–156, 2018. ISSN 0360-8352.

[89] E. V. Gutiérrez and C. J. Vidal. Home health care logistics management problems: A critical review of models and methods. *Revista Facultad de Ingeniería Universidad de Antioquia*, (68): 160–175, 2013.

[90] M. H. Hà, T. D. Nguyen, T. N. Duy, H. G. Pham, T. Do, and L.-M. Rousseau. A new constraint programming model and a linear programming-based adaptive large neighborhood search for the vehicle routing problem with synchronization constraints. *Computers & Operations Research*, 124:105085, 2020.

[91] W. L. Hamilton, Z. Ying, and J. Leskovec. Inductive Representation Learning on Large Graphs. In *NeurIPS*, 2017.

[92] S. Han, L. Zhao, K. Chen, Z.-w. Luo, and D. Mishra. Appointment scheduling and routing optimization of attended home delivery system with random customer behavior. *European Journal of Operational Research*, 262(3):966–980, 2017. doi: 10.1016/j.ejor.2017.03.060.

[93] S. D. Handoko, D. T. Nguyen, Z. Yuan, and H. C. Lau. Reinforcement learning for adaptive operator selection in memetic search applied to quadratic assignment problem. In *GECCO Companion*, 2014.

[94] P. Hansen, N. Mladenović, and J. A. Moreno Perez. Variable neighbourhood search: methods and applications. *Annals of Operations Research*, 175:367–407, 2010.

[95] M. Hartman, A. B. Martin, J. Benson, A. Catlin, and National Health Expenditure Accounts Team. National health care spending in 2018: Growth driven by accelerations in medicare and private insurance spending: Us health care spending increased 4.6 percent to reach $3.6 trillion in 2018, a faster growth rate than that of 4.2 percent in 2017 but the same rate as in 2016. *Health Affairs*, 39(1):8–17, 2020.

[96] A. Hasani and A. Khosrojerdi. Robust global supply chain network design under disruption and uncertainty considering resilience strategies: A parallel memetic algorithm for a real-life case study. *Transportation Research. Part E: Logistics and Transportation Review*, 87:20–52, 2016.

[97] H. Hashemi Doulabi, G. Pesant, and L.-M. Rousseau. Vehicle routing problems with synchronized visits and stochastic travel and service times: Applications in healthcare. *Transportation Science*, 54(4):1053–1072, 2020.

[98] L. He, M. de Weerdt, and N. Yorke-Smith. Time/sequence-dependent scheduling: the design and evaluation of a general purpose tabu-based adaptive large neighbourhood search algorithm. *Journal of Intelligent Manufacturing*, 31(4):1051–1078, 2020.

[99] V. Hemmelmayr, K. Smilowitz, and L. de la Torre. A periodic location routing problem for collaborative recycling. *IISE Transactions*, 49(4):414–428, 2017. ISSN 2472-5854.

[100] V. C. Hemmelmayr. Sequential and parallel large neighborhood search algorithms for the periodic location routing problem. *European Journal of Operational Research*, 243(1):52–60, 2015. ISSN 0377-2217.

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

[102] G. Hendel. Adaptive large neighborhood search for mixed integer programming. *Mathematical Programming Computation*, 14(2):185–221, 2022.

[103] S. W. Hess, J. B. Weaver, H. J. Siegfeldt, J. N. Whelan, and P. A. Zitlau. Nonpartisan political redistricting by computer. *Operations Research*, 13(6):998–1006, 1965. ISSN 0030-364X.

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

[105] H. Hojabri, M. Gendreau, J.-Y. Potvin, and L.-M. Rousseau. Large neighborhood search with constraint programming for a vehicle routing problem with synchronization constraints. *Computers & Operations Research*, 92:87–97, 2018.

[106] Y. Hoogendoorn and R. Spliet. An improved integer l-shaped method for the vehicle routing problem with stochastic demands. *INFORMS Journal on Computing*, 35(2):423–439, 2023.

[107] J. N. Hooker and G. Ottosson. Logic-based benders decomposition. *Mathematical Programming*, 96(1):33–60, 2003.

[108] K. Hornik, M. Stinchcombe, and H. White. Multilayer feedforward networks are universal approximators. *Neural networks*, 2(5):359–366, 1989.

[109] A. Hottung and K. Tierney. Neural large neighborhood search for the capacitated vehicle routing problem. In *ECAI*, 2020.

[110] Y. Huang, M. Savelsbergh, and L. Zhao. Designing logistics systems for home delivery in densely populated urban areas. *Transportation Research. Part B: Methodological*, 115:95–125, 2018. ISSN 0191-2615.

[111] P. J. Hulshof, N. Kortbeek, R. J. Boucherie, E. W. Hans, and P. J. Bakker. Taxonomic classification of planning decisions in health care: a structured review of the state of the art in or/ms. *Health systems*, 1(2):129–175, 2012.

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

[113] S.-N. Johnn, Y. Zhu, A. Miniguano-Trujillo, and A. Gupte. Solving the home service assignment, routing, and appointment scheduling (h-sara) problem with uncertainties. In M. Müller-Hannemann and F. Perea, editors, *21st Symposium on Algorithmic Approaches for Transportation Modelling, Optimization, and Systems (ATMOS 2021)*, pages 4:1–4:21. Schloss Dagstuhl-Leibniz-Zentrum für Informatik, 2021.

[114] S.-N. Johnn, V.-A. Darvariu, J. Handl, and J. Kalcsics. Graph reinforcement learning for operator selection in the alns metaheuristic. In *International Conference on Optimization and Learning*, pages 200–212. Springer, 2023.

[115] P. Kalatzantonakis, A. Sifaleras, and N. Samaras. A reinforcement learning-variable neighborhood search method for the capacitated vehicle routing problem. *Expert Systems with Applications*, 213:118812, 2023.

[116] J. Kalcsics and R. Z. Rios-Mercado. Districting problems. In *Location Science*, pages 705–743. Springer, 2019.

[117] J. Kallestad, R. Hasibi, A. Hemmati, and K. Sörensen. A general deep reinforcement learning hyperheuristic framework for solving combinatorial optimization problems. *European Journal of Operational Research*, 309(1):446–468, 2023.

[118] M. Karimi-Mamaghan. *Hybridizing metaheuristics with machine learning for combinatorial optimization: a taxonomy and learning to select operators*. PhD thesis, Ecole nationale supérieure Mines-Télécom Atlantique Bretagne Pays de la Loire, 2022.

[119] M. Karimi-Mamaghan, B. Pasdeloup, M. Mohammadi, and P. Meyer. A learning-based iterated local search algorithm for solving the traveling salesman problem. In *International Conference on Optimization and Learning (OLA)*, 2021.

[120] M. Karimi-Mamaghan, M. Mohammadi, P. Meyer, A. M. Karimi-Mamaghan, and E.-G. Talbi. Machine learning at the service of meta-heuristics for solving combinatorial optimization problems: A state-of-the-art. *European Journal of Operational Research*, 296(2):393–422, 2022.

[121] M. Karimi-Mamaghan, M. Mohammadi, B. Pasdeloup, and P. Meyer. Learning to select operators in meta-heuristics: An integration of q-learning into the iterated greedy algorithm for the permutation flowshop scheduling problem. *European Journal of Operational Research*, 2022.

[122] S. Katoch, S. S. Chauhan, and V. Kumar. A review on genetic algorithm: past, present, and future. *Multimedia tools and applications*, 80:8091–8126, 2021.

[123] J. Kennedy and R. Eberhart. Particle swarm optimization. In *Proceedings of ICNN'95-international conference on neural networks*, volume 4, pages 1942–1948. IEEE, 1995.

[124] M. Keskin and B. Çatay. Partial recharge strategies for the electric vehicle routing problem with time windows. *Transportation Research. Part C: Emerging Technologies*, 65:111–127, 2016.

[125] E. Khalil, H. Dai, Y. Zhang, B. Dilkina, and L. Song. Learning combinatorial optimization algorithms over graphs. In *NeurIPS*, 2017.

[126] M. Kim, J. Park, et al. Learning collaborative policies to solve np-hard routing problems. *NeurIPS*, 2021.

[127] S. Kır, H. R. Yazgan, and E. Tüncel. A novel heuristic algorithm for capacitated vehicle routing problem. *Journal of Industrial Engineering International*, 13:323–330, 2017.

[128] P. Kitjacharoenchai, B.-C. Min, and S. Lee. Two echelon vehicle routing problem with drones in last mile delivery. *International Journal of Production Economics*, 225:107598, 2020.

[129] V. Konda and J. Tsitsiklis. Actor-critic algorithms. *Advances in neural information processing systems*, 12, 1999.

[130] W. Kool, H. Van Hoof, and M. Welling. Attention, learn to solve routing problems! In *ICLR*, 2018.

[131] A. A. Kovacs, B. L. Golden, R. F. Hartl, and S. N. Parragh. Vehicle routing problems in which consistency considerations are important: A survey. *Networks*, 64(3):192–213, 2014.

[132] Koç. A unified-adaptive large neighborhood search metaheuristic for periodic location-routing problems. *Transportation Research. Part C: Emerging Technologies*, 68:265–284, 2016. ISSN 0968-090X.

[133] F. Lagos and J. Pereira. Multi-armed bandit-based hyper-heuristics for combinatorial optimization problems. *European Journal of Operational Research*, 2023.

[134] E. Lanzarone, A. Matta, and G. Scaccabarozzi. A patient stochastic model to support human resource planning in home care. *Production Planning and Control*, 21(1):3–25, 2010.

[135] G. Laporte. Fifty years of vehicle routing. *Transportation Science*, 43(4):408–416, 2009.

[136] G. Laporte and F. V. Louveaux. The integer l-shaped method for stochastic integer programs with complete recourse. *Operations Research Letters*, 13(3):133–142, 1993.

[137] G. Laporte and Y. Nobert. An exact algorithm for minimizing routing and operating costs in depot location. *European Journal of Operational Research*, 6(2):224–226, 1981.

[138] G. Laporte and Y. Nobert. A branch and bound algorithm for the capacitated vehicle routing problem. *Operations Research Spektrum*, 5(2):77–85, 1983.

[139] G. Laporte, Y. Nobert, and D. Arpin. An exact algorithm for solving a capacitated location-routing problem. *Annals of Operations Research*, 6(9):291–310, 1986.

[140] G. Laporte, R. Musmanno, and F. Vocaturo. An adaptive large neighbourhood search heuristic for the capacitated arc-routing problem with stochastic demands. *Transportation Science*, 44(1): 125–135, 2010.

[141] G. Laporte, S. Nickel, and F. Saldanha da Gama, editors. *Location Science*. Springer International Publishing, Cham, 2015 edition, 2015.

[142] F. Lehuédé, R. Masson, S. N. Parragh, O. Péton, and F. Tricoire. A multi-criteria large neighbourhood search for the transportation of disabled people. *Journal of the Operational Research Society*, 65(7):983–1000, 2014.

[143] B. Li, D. Krushinsky, T. Van Woensel, and H. A. Reijers. An adaptive large neighborhood search heuristic for the share-a-ride problem. *Computers & Operations Research*, 66:170–180, 2016.

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

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

[146] C. Lin, K. Choy, G. Ho, S. Chung, and H. Lam. Survey of green vehicle routing problem: Past and future trends. *Expert Systems with Applications*, 41(4):1118–1138, 2014.

[147] C.-C. Lin, L.-P. Hung, W.-Y. Liu, and M.-C. Tsai. Jointly rostering, routing, and rerostering for home health care services: A harmony search approach with genetic, saturation, inheritance, and immigrant schemes. *Computers & Industrial Engineering*, 115:151–166, 2018.

[148] J.-R. Lin and H.-C. Lei. Distribution systems design with two-level routing considerations. *Annals of Operations Research*, 172(1):329–347, 2009. ISSN 0254-5330.

[149] R. Liu, B. Yuan, and Z. Jiang. A branch-and-price algorithm for the home-caregiver scheduling and routing problem with stochastic travel and service times. *Flexible Services and Manufacturing Journal*, 31(4):989–1011, 2019. doi: 10.1007/s10696-018-9328-8.

[150] S. C. Liu and S. B. Lee. A two-phase heuristic method for the multi-depot location routing problem taking inventory control decisions into consideration. *International Journal of Advanced Manufacturing Technology*, 22(11-12):941–950, 2003.

[151] T. Liu, Z. Luo, H. Qin, and A. Lim. A branch-and-cut algorithm for the two-echelon capacitated vehicle routing problem with grouping constraints. *European Journal of Operational Research*, 266(2):487–497, 2018. ISSN 0377-2217.

[152] R. Lopes, C. Ferreira, B. Santos, and S. Barreto. A taxonomical analysis, current methods and objectives on location-routing problems. *International Transactions in Operational Research*, 20 (6), 2013.

[153] M. López-Ibáñez, J. Dubois-Lacoste, L. P. Cáceres, M. Birattari, and T. Stützle. The irace package: Iterated racing for automatic algorithm configuration. *Operations Research Perspectives*, 3:43–58, 2016.

[154] F. V. Louveaux and J.-J. Salazar-González. Exact approach for the vehicle routing problem with stochastic demands and preventive returns. *Transportation Science*, 52(6):1463–1478, 2018.

[155] H. Lu, X. Zhang, and S. Yang. A learning-based iterative method for solving vehicle routing problems. In *ICLR*, 2019.

[156] Z. Luo, H. Qin, D. Zhang, and A. Lim. Adaptive large neighborhood search heuristics for the vehicle routing problem with stochastic demands and weight-related cost. *Transportation Research. Part E: Logistics and Transportation Review*, 85:69–89, 2016.

[157] S. Mancini. A real-life multi depot multi period vehicle routing problem with a heterogeneous fleet: Formulation and adaptive large neighborhood search based matheuristic. *Transportation Research. Part C: Emerging Technologies*, 70:100–112, 2016.

[158] S. T. W. Mara, R. Kuo, and A. M. S. Asih. Location-routing problem: a classification of recent research. *International Transactions in Operational Research*, 2021.

[159] S. T. W. Mara, R. Norcahyo, P. Jodiawan, L. Lusiantoro, and A. P. Rifai. A survey of adaptive large neighborhood search algorithms and applications. *Computers & Operations Research*, page 105903, 2022.

[160] R. Masson, F. Lehuédé, and O. Péton. An adaptive large neighborhood search for the pickup and delivery problem with transfers. *Transportation Science*, 47(3):344–355, 2013.

[161] N. Mazyavkina, S. Sviridov, S. Ivanov, and E. Burnaev. Reinforcement learning for combinatorial optimization: A survey. *Computers & Operations Research*, 134:105400, 2021.

[162] G. P. McCormick. Computability of global solutions to factorable nonconvex programs: Part i—convex underestimating problems. *Mathematical Programming*, 10(1):147–175, 1976.

[163] Y. Z. Mehrjerdi and A. Nadizadeh. Using greedy clustering method to solve capacitated location-routing problem with fuzzy demands. *International journal of industrial engineering production research*, 27(1):1–19, 2016. ISSN 2008-4889.

[164] J. Melechovský, C. Prins, and R. W. Calvo. A metaheuristic to solve a location-routing problem with non-linear costs. *Journal of Heuristics*, 11(5-6 SPEC. ISS.):375–391, 2005.

[165] H. Min. Consolidation terminal location-allocation and consolidated routing problems. *Journal of Business Logistics*, 17(2):235–263, 1996.

[166] V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, et al. Human-level control through deep reinforcement learning. *Nature*, 518(7540):529–533, 2015.

[167] H. Mosadegh, S. F. Ghomi, and G. A. Süer. Stochastic mixed-model assembly line sequencing problem: Mathematical modeling and q-learning based simulated annealing hyper-heuristics. *European Journal of Operational Research*, 282(2):530–544, 2020.

[168] F. Mühlbauer and P. Fontaine. A parallelised large neighbourhood search heuristic for the asymmetric two-echelon vehicle routing problem with swap containers for cargo-bicycles. *European Journal of Operational Research*, 289(2):742–757, 2021.

[169] L. F. Muller, S. Spoorendonk, and D. Pisinger. A hybrid adaptive large neighborhood search heuristic for lot-sizing with setup times. *European Journal of Operational Research*, 218(3): 614–623, 2012.

[170] B. Naderi, M. A. Begen, G. S. Zaric, and V. Roshanaei. A novel and efficient exact technique for integrated staffing, assignment, routing, and scheduling of home care services under uncertainty. *Omega*, 116:102805, 2023.

[171] G. Nagy and S. Salhi. Location-routing: Issues, models and methods. *European Journal of Operational Research*, 177(2):649–672, 2007.

[172] V. Nair, S. Bartunov, F. Gimeno, I. von Glehn, P. Lichocki, I. Lobov, B. O'Donoghue, N. Sonnerat, C. Tjandraatmadja, P. Wang, et al. Solving mixed integer programs using neural networks. *arXiv:2012.13349*, 2020.

[173] J. M. Nambiar, L. F. Gelders, and L. N. Van Wassenhove. A large scale location-allocation problem in the natural rubber industry. *European Journal of Operational Research*, 6(2):183–189, 1981.

[174] J. A. Nasir and C. Dang. Solving a more flexible home health care scheduling and routing problem with joint patient and nursing staff selection. *Sustainability*, 10(1):148, 2018.

[175] M. Nazari, A. Oroojlooy, L. Snyder, and M. Takác. Reinforcement learning for solving the vehicle routing problem. In *NeurIPS*, 2018.

[176] V.-P. Nguyen, C. Prins, and C. Prodhon. Solving the two-echelon location routing problem by a grasp reinforced by a learning process and path relinking. *European Journal of Operational Research*, 216(1):113–126, 2012.

[177] S. Nickel, M. Schröder, and J. Steeg. Mid-term and short-term planning support for home health care services. *European Journal of Operational Research*, 219(3):574–587, 2012.

[178] E. Nikzad, M. Bashiri, and B. Abbasi. A matheuristic algorithm for stochastic home health care planning. *European Journal of Operational Research*, 288(3):753–774, 2021.

[179] F. F. Oberweger, G. R. Raidl, E. Rönnberg, and M. Huber. A learning large neighborhood search for the staff rerostering problem. In *CPAIOR*, 2022.

[180] P. J. Palomo-Martínez, M. A. Salazar-Aguilar, and G. Laporte. Planning a selective delivery schedule through adaptive large neighborhood search. *Computers & Industrial Engineering*, 112: 368–378, 2017.

[181] B. Pan, Z. Zhang, and A. Lim. Multi-trip time-dependent vehicle routing problem with time windows. *European Journal of Operational Research*, 291(1):218–231, 2021.

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

[183] J. Perl and M. S. Daskin. A warehouse location-routing problem. *Transportation Research. Part B: Methodological*, 19(5):381–396, 1985.

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

[185] C. Prins, C. Prodhon, and R. W. Calvo. Solving the capacitated location-routing problem by a grasp complemented by a learning process and a path relinking. *4OR*, 4(3):221–238, 2006. ISSN 1614-2411.

[186] C. Prodhon. A hybrid evolutionary algorithm for the periodic location-routing problem. *European Journal of Operational Research*, 210(2):204–212, 2011. ISSN 0377-2217.

[187] C. Prodhon and C. Prins. A memetic algorithm with population management (ma—pm) for the periodic location-routing problem. In *Hybrid Metaheuristics*, volume 5296 of *Lecture Notes in Computer Science*, pages 43–57, Berlin, Heidelberg, 2008. Springer Berlin Heidelberg. ISBN 3540884386.

[188] C. Prodhon and C. Prins. A survey of recent research on location-routing problems. *European Journal of Operational Research*, 238(1):1–17, 2014.

[189] H. Qiu, S. Wang, Y. Yin, D. Wang, and Y. Wang. A deep reinforcement learning-based approach for the home delivery and installation routing problem. *International Journal of Production Economics*, 244:108362, 2022.

[190] Y. Qu and J. F. Bard. A grasp with adaptive large neighborhood search for pickup and delivery problems with transshipment. *Computers & Operations Research*, 39(10):2439–2456, 2012.

[191] J. Quiñonero-Candela, M. Sugiyama, A. Schwaighofer, and N. D. Lawrence. *Dataset Shift in Machine Learning*. MIT Press, 2008.

[192] M. Rabbani, H. Farrokhi-Asl, and B. Asgarian. Solving a bi-objective location routing problem by a nsga-ii combined with clustering approach: application in waste collection problem. *Journal of industrial engineering international*, 13(1):13–27, 2016. ISSN 2251-712X.

[193] M. Rabbani, R. Heidari, and R. Yazdanparast. A stochastic multi-period industrial hazardous waste location-routing problem: Integrating nsga-ii and monte carlo simulation. *European Journal of Operational Research*, 272(3):945–961, 2019.

[194] R. Rahmaniani, T. G. Crainic, M. Gendreau, and W. Rei. The benders decomposition algorithm: A literature review. *European Journal of Operational Research*, 259(3):801–817, 2017.

[195] S. M. Raza, M. Sajid, and J. Singh. Vehicle routing problem using reinforcement learning: Recent advancements. In *Advanced Machine Intelligence and Signal Processing*, pages 269–280. Springer, 2022.

[196] C. Rego, D. Gamboa, F. Glover, and C. Osterman. Traveling salesman problem heuristics: Leading methods, implementations and latest advances. *European Journal of Operational Research*, 211(3):427–441, 2011.

[197] R. Reijnen, Y. Zhang, H. C. Lau, and Z. Bukhsh. Operator selection in adaptive large neighborhood search using deep reinforcement learning. *arXiv:2211.00759*, 2022.

[198] K.-D. Rest and P. Hirsch. Daily scheduling of home health care services using time-dependent public transport. *Flexible Services and Manufacturing Journal*, 28(3):495–525, 2016.

[199] G. M. Ribeiro and G. Laporte. An adaptive large neighborhood search heuristic for the cumulative capacitated vehicle routing problem. *Computers & Operations Research*, 39(3):728–735, 2012.

[200] C. Rodriguez, T. Garaix, X. Xie, and V. Augusto. Staff dimensioning in homecare services with uncertain demands. *International Journal of Production Research*, 53(24):7396–7410, 2015. doi: 10.1080/00207543.2015.1081427.

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

[202] R. A. Rutenbar. Simulated annealing algorithms: An overview. *IEEE Circuits and Devices magazine*, 5(1):19–26, 1989.

[203] G. K. Saharidis and M. G. Ierapetritou. Improving benders decomposition using maximum feasible subsystem (mfs) cut generation strategy. *Computers & Chemical Engineering*, 34(8): 1237–1245, 2010.

[204] M. Salavati-Khoshghalb, M. Gendreau, O. Jabali, and W. Rei. An exact algorithm to solve the vehicle routing problem with stochastic demands under an optimal restocking policy. *European Journal of Operational Research*, 273(1):175–189, 2019.

[205] E. Sanci and M. S. Daskin. An integer l-shaped algorithm for the integrated location and network restoration problem in disaster relief. *Transportation Research Part B: Methodological*, 145:152–184, 2021.

[206] A. Santini, S. Ropke, and L. M. Hvattum. A comparison of acceptance criteria for the adaptive large neighbourhood search metaheuristic. *Journal of Heuristics*, 24(5):783–815, 2018.

[207] F. Santos, A. Cunha, and G. Mateus. Branch-and-price algorithms for the two-echelon capacitated vehicle routing problem. *Optimization Letters*, 7(7):1537–1547, 2013.

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

[209] F. Scarselli, M. Gori, A. C. Tsoi, M. Hagenbuchner, and G. Monfardini. The graph neural network model. *IEEE Transactions on Neural networks*, 20(1):61–80, 2009.

[210] M. Schiffer and G. Walther. An adaptive large neighborhood search for the location-routing problem with intra-route facilities. *Transportation Science*, 52(2):331–352, 2018.

[211] S. Shahnejat-Bushehri, R. Tavakkoli-Moghaddam, M. Boronoos, and A. Ghasemkhani. A robust home health care routing-scheduling problem with temporal dependencies under uncertainty. *Expert Systems with Applications*, 182:115209, 2021.

[212] P. Shaw. Using constraint programming and local search methods to solve vehicle routing problems. In *Principles and Practice of Constraint Programming — CP98*, 1999.

[213] K. S. Shehadeh and M. Chiriki. 13th AIMMS-MOPTA Optimization Modeling Competition. In *Modeling and Optimization: Theory and Applications (MOPTA)*, 2021. https://coral.ise.lehigh.edu/~mopta2021/competition, Date accessed: 7 Aug 2021.

[214] Y. Shi, T. Boudouh, and O. Grunder. A hybrid genetic algorithm for a home health care routing problem with time window and fuzzy demand. *Expert Systems with Applications*, 72:160–176, 2017.

[215] Y. Shi, T. Boudouh, O. Grunder, and D. Wang. Modeling and solving simultaneous delivery and pick-up problem with stochastic travel and service times in home health care. *Expert Systems with Applications*, 102:218–233, 2018. ISSN 0957-4174.

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

[217] E. A. Silver. An overview of heuristic solution methods. *Journal of the Operational Research Society*, 55(9):936–956, 2004.

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

[219] J. Song, Y. Yue, B. Dilkina, et al. A general large neighborhood search framework for solving integer linear programs. In *NeurIPS*, 2020.

[220] W. Song, X. Chen, Q. Li, and Z. Cao. Flexible job-shop scheduling via graph neural network and deep reinforcement learning. *IEEE Transactions on Industrial Informatics*, 19(2):1600–1610, 2022.

[221] K. Sörensen and F. Glover. Metaheuristics. *Encyclopedia of Operations Research and Management Science*, 62:960–970, 2013.

[222] A. Sperduti and A. Starita. Supervised neural networks for the classification of structures. *IEEE Transactions on Neural Networks*, 8(3):714–735, 1997.

[223] R. S. Sutton and A. G. Barto. *Reinforcement learning: An introduction*. MIT press, 2018.

[224] A. A. Syed, K. Akhnoukh, B. Kaltenhaeuser, and K. Bogenberger. Neural network based large neighborhood search algorithm for ride hailing services. In *EPIA Conference on Artificial Intelligence*, 2019.

[225] K. Sörensen and M. Sevaux. A practical approach for robust and flexible vehicle routing using metaheuristics and monte carlo sampling. *Journal of Mathematical Modelling and Algorithms*, 8 (4):387, 2009. ISSN 1570-1166.

[226] E.-G. Talbi. Machine learning into metaheuristics: A survey and taxonomy. *ACM Computing Surveys (CSUR)*, 54(6):1–32, 2021.

[227] O. Tellez, S. Vercraene, F. Lehuédé, O. Péton, and T. Monteiro. The fleet size and mix dial-a-ride problem with reconfigurable vehicle capacity. *Transportation Research. Part C: Emerging Technologies*, 91:99–123, 2018.

[228] A. Teymourifar, A. M. Rodrigues, and J. S. Ferreira. A comparison between simultaneous and hierarchical approaches to solve a multi-objective location-routing problem. In *Graphs and Combinatorial Optimization: from Theory to Applications*, pages 251–263. Springer, 2021.

[229] C. Thomas and P. Schaus. Revisiting the self-adaptive large neighborhood search. In *CPAIOR*, 2018.

[230] A. Trautsamwieser, M. Gronalt, and P. Hirsch. Securing home health care in times of natural disasters. *OR spectrum*, 33(3):787–813, 2011.

[231] R. Tunalıoğlu, Koç, and T. Bektaş. A multiperiod location-routing problem arising in the collection of olive oil mill wastewater. *The Journal of the Operational Research Society*, 67(7): 1012–1024, 2016. ISSN 0160-5682.

[232] R. Turkeš, K. Sörensen, and L. M. Hvattum. Meta-analysis of metaheuristics: Quantifying the effect of adaptiveness in adaptive large neighborhood search. *European Journal of Operational Research*, 292(2):423–442, 2021.

[233] D. Tuzun and L. I. Burke. A two-phase tabu search approach to the location routing problem. *European Journal of Operational Research*, 116(1):87–99, 1999. ISSN 0377-2217.

[234] B. Vahdani, B. Vahdani, D. Veysmoradi, D. Veysmoradi, N. Shekari, N. Shekari, S. M. Mousavi, and S. M. Mousavi. Multi-objective, multi-period location-routing model to distribute relief after earthquake by considering emergency roadway repair. *Neural computing applications*, 30 (3):835–854, 2018. ISSN 0941-0643.

[235] P. Veličković, G. Cucurull, A. Casanova, A. Romero, P. Lio, and Y. Bengio. Graph attention networks. In *ICLR*, 2018.

[236] J. Verstichel, J. Kinable, P. De Causmaecker, and G. V. Berghe. A combinatorial benders decomposition for the lock scheduling problem. *Computers & Operations Research*, 54:117–128, 2015.

[237] N. Vesselinova, R. Steinert, D. F. Perez-Ramirez, and M. Boman. Learning combinatorial optimization on graphs: A survey with applications to networking. *IEEE Access*, 8:120388–120416, 2020.

[238] M. Vidović, B. Ratković, N. Bjelić, and D. Popović. A two-echelon location-routing model for designing recycling logistics networks with profit: Milp and heuristic approach. *Expert Systems with Applications*, 51:34–48, 2016.

[239] O. Vinyals, M. Fortunato, and N. Jaitly. Pointer networks. *NeurIPS*, 2015.

[240] J. Wang, D. Lei, and J. Cai. An adaptive artificial bee colony with reinforcement learning for distributed three-stage assembly scheduling with maintenance. *Applied Soft Computing*, 117: 108371, 2022.

[241] Y. Wang, K. Assogba, Y. Liu, X. Ma, M. Xu, and Y. Wang. Two-echelon location-routing optimization with time windows based on customer clustering. *Expert Systems With Applications*, 104:244–260, 2018.

[242] C. Watkins and P. Dayan. Q-learning. *Machine Learning*, 8(3-4):279–292, 1992.

[243] G. Watkins, G. Montana, and J. Branke. Generating a graph colouring heuristic with deep q-learning and graph neural networks. *arXiv preprint arXiv:2304.04051*, 2023.

[244] M. Wen, E. Linde, S. Ropke, P. Mirchandani, and A. Larsen. An adaptive large neighborhood search heuristic for the electric vehicle scheduling problem. *Computers & Operations Research*, 76:73–83, 2016.

[245] M. Winkenbach, P. R. Kleindorfer, and S. Spinler. Enabling urban logistics services at la poste through multi-echelon location-routing. *Transportation Science*, 50(2):520–540, 2016.

[246] T.-H. Wu, C. Low, and J.-W. Bai. Heuristic solutions to multi-depot location-routing problems. *Computers & Operations Research*, 29(10):1393–1415, 2002.

[247] X. Wu, L. Nie, and M. Xu. Designing an integrated distribution system for catering services for high-speed railways: A three-echelon location routing model with tight time windows and time deadlines. *Transportation Research. Part C: Emerging Technologies*, 74:212–244, 2017.

[248] Y. Wu, W. Song, Z. Cao, and J. Zhang. Learning large neighborhood search policy for integer programming. In *NeurIPS*, 2021.

[249] Y. Wu, W. Song, Z. Cao, J. Zhang, and A. Lim. Learning improvement heuristics for solving routing problems.. *IEEE Transactions on Neural Networks and Learning Systems*, 2021.

[250] E. Yakıcı. Solving location and routing problem for uavs. *Computers & Industrial Engineering*, 102:294–301, 2016.

[251] M. Yang, Y. Ni, and L. Yang. A multi-objective consistent home healthcare routing and scheduling problem in an uncertain environment. *Computers & Industrial Engineering*, 160:107560, 2021.

[252] J. You, B. Liu, R. Ying, V. Pande, and J. Leskovec. Graph Convolutional Policy Network for Goal-Directed Molecular Graph Generation. In *NeurIPS*, 2018.

[253] B. Yuan, R. Liu, and Z. Jiang. A branch-and-price algorithm for the home health care scheduling and routing problem with stochastic service times and skill requirements. *International Journal of Production Research*, 53(24):7450–7464, 2015.

[254] B. Yuan, R. Liu, and Z. Jiang. Daily scheduling of caregivers with stochastic times. *International Journal of Production Research*, 56(9):3245–3261, 2018.

[255] Y. Zhan and G. Wan. Vehicle routing and appointment scheduling with team assignment for home services. *Computers & Operations Research*, 100:1–11, 2018. ISSN 0305-0548. doi: 10.1016/j.cor.2018.07.006.

[256] Y. Zhan, Z. Wang, and G. Wan. Home service routing and appointment scheduling with stochastic service times. *European Journal of Operational Research*, 288(1):98–110, 2021. ISSN 0377-2217. doi: https://doi.org/10.1016/j.ejor.2020.05.037.

[257] K. Zhang, F. He, Z. Zhang, X. Lin, and M. Li. Multi-vehicle routing problems with soft time windows: A multi-agent reinforcement learning approach. *Transportation Research. Part C: Emerging Technologies*, 121:102861, 2020.

[258] R. Zhang, C. Zhang, Z. Cao, W. Song, P. S. Tan, J. Zhang, B. Wen, and J. Dauwels. Learning to solve multiple-tsp with time window and rejections via deep reinforcement learning. *IEEE Transactions on Intelligent Transportation Systems*, 2022.

[259] F. Zhao, L. Zhang, J. Cao, and J. Tang. A cooperative water wave optimization algorithm with reinforcement learning for the distributed assembly no-idle flowshop scheduling problem. *Computers & Industrial Engineering*, 153:107082, 2021.

[260] J. Zhou, G. Cui, S. Hu, Z. Zhang, C. Yang, Z. Liu, L. Wang, C. Li, and M. Sun. Graph neural networks: A review of methods and applications. *AI open*, 1:57–81, 2020.

[261] Y. Zhou, J.-K. Hao, and B. Duval. Reinforcement learning based local search for grouping problems: A case study on graph coloring. *Expert Systems with Applications*, 64:412–422, 2016.