Truck Routing for Parcel Delivery

Solving a Multi-depot Pickup and Delivery Problem with Occasional Drivers using ALNS

TIL-5060 Master Thesis

Shuai Wang

Delft University of Technology

TU Delft

Truck Routing for Parcel Delivery

Solving a Multi-depot Pickup and Delivery Problem with Occasional Drivers using ALNS

by

Shuai Wang

Master of Science

in Transport, Infrastructure and Logistics

at Delft University of Technology

Student number:	**5714532**	
Project duration:	**March, 2024 - August, 2024**	
Thesis committee:	**Prof.dr.ir. L.A. (Lóri) Tavasszy,**	**TU Delft, Chair**
	Dr. S. (Stefano) Fazi,	**TU Delft, Supervisor**
	Dr. A. (Alessandro) Bombelli,	**TU Delft, Supervisor**

TU Delft

Preface

In the field of logistics and supply chain management, optimizing delivery routes and scheduling has become increasingly crucial. This research tackles a significant challenge in this domain: optimizing delivery routes by integrating multiple depots, occasional drivers, and multiple depot visits. These features represent the complex realities faced by many companies in their logistics operations today. As the complexity and scale of delivery operations continue to grow, there is a pressing need for more sophisticated models that can accommodate these real-world variables.

The motivation for this study stems from a noticeable gap in existing research, which often focuses on simplified models that do not account for the complexities of modern logistics operations. This research aims to address this gap by developing a comprehensive mathematical model that includes multiple depots and occasional drivers and applies an Adaptive Large Neighborhood Search (ALNS) algorithm to effectively minimize routing costs.

Our approach combines advanced heuristic methods and algorithmic strategies to address routing problems efficiently. The insights gained from this research are of significant importance both to academic research and practical applications in the logistics industry. By providing a robust framework for optimizing delivery routes, this study contributes to the broader field of logistics and supply chain management and offers valuable tools for practitioners aiming to enhance operational efficiency.

This thesis has been completed with the support of my committee. I would first like to thank Dr. S. Fazi for his invaluable insights into the research topic selection and various aspects of the thesis. I am also grateful to Dr. A. Bombelli for his assistance in building the mathematical model and data visualization. The productive and pleasant meetings with both mentors have been a source of new ideas and suggestions. I also wish to thank my chair, Prof.dr.ir. L.A. Tavasszy, for helping me secure an internship in the logistics industry before starting my thesis. This internship provided me with a deeper understanding of the challenges faced by the logistics industry and improved my problem-solving skills from various perspectives. Professor Lori's suggestions on my model, algorithm, report writing, and presentation were immensely helpful throughout the different stages of my thesis.

Additionally, I want to express my gratitude to all my friends in the Netherlands. The two wonderful years spent with you have flown by, and I appreciate all the support you provided in my personal and academic life. Most importantly, I want to thank my family for their comprehensive support, which allowed me the opportunity to study at Delft University of Technology. My family are my strongest support, always behind me silently. I love you all.

This research has been a rewarding experience both intellectually and personally. It has deepened my understanding of complex logistics problems and provided me with valuable skills and insights. I am grateful for the opportunity to contribute to this field and look forward to continuing to explore and develop these ideas in future research.

Shuai Wang

Delft, August 2024

Executive Summary

Introduction In recent years, the exponential growth of e-commerce, accelerated by the Covid-19 pandemic, has significantly increased urban parcel deliveries, creating major logistical challenges for last-mile delivery (LMD) in cities, particularly in densely populated areas. For instance, Amsterdam is expected to handle over 100,000 parcels daily by 2030, putting immense pressure on the logistics system, exacerbating traffic congestion, environmental pollution, and driving up logistics costs, with last-mile delivery accounting for 75% of these costs. Traditional LMD strategies often involve increasing vehicle numbers or capacity, which may reduce efficiency and raise costs. To address these issues, crowdshipping (CS) has emerged as a viable alternative, leveraging occasional drivers (ODs) to complete deliveries on detours at lower costs. This study proposes a new variant of the Pickup and Delivery Problem with Time Windows (PDPTW), integrating multiple depots, regular drivers (RDs), ODs, and intermediate centers (stores) to optimize the total vehicle routing cost. The model, referred to as the Multi-trip Crowd-shipping Split Pickup and Delivery Problem with Time Window and Multi-depots (MC-PDPTW-MD), aims to incorporate these features into a traditional PDPTW model, focusing on minimizing routing costs for both RDs and ODs. The key research question is how to optimize the total vehicle routing cost in the MC-PDPTW-MD problem.

Literature Study The literature review provides insights into the development of models and algorithms related to the Pickup and Delivery Problem with Time Window (PDPTW), Crowd-shipping, and the Open Vehicle Routing Problem (OVRP). Notable algorithmic approaches include Adaptive Large Neighborhood Search (ALNS), Variable Neighborhood Search (VNS), Tabu Search (TS), Stochastic and Genetic algorithms, Simulated Annealing, Greedy Randomized Adaptive Search Procedures (GRASP) for Vehicle Routing, Integer Linear Programming (ILP), and other specialized algorithms. Upon comparison, it has been found that the ALNS algorithm is a good choice for solving this problem. The research methods section delves into the steps of the ALNS algorithm and the criteria used in each step when solving different problems. These integrated studies provide significant reference value for solving the current problem.

Methodology In the methodology chapter, the specific content of the mathematical model and the ALNS algorithm is presented. We first constructed a constraint programming model, which is divided into four main parts based on regular drivers, occasional drivers, and multiple visits to depots. The first part contains the basic model constraints of the pickup and delivery problem, while the second, third, and fourth parts respectively address the constraints for regular drivers, occasional drivers, and the feature of multiple visits to depots by occasional drivers. In the algorithm section, ALNS is applied to solve the proposed problem. The initial solution of the algorithm uses the Basic Greedy Algorithm, which can obtain a reasonable initial solution. In the request removal part, random removal operators and worst removal operators are introduced. The random removal operator has stronger randomness in destroying the initial solution, while the worst removal operator preferentially removes the costliest requests. The combination of these two removal operators can avoid the solution falling into a local optimum to a certain extent. In the inserting requests part, Basic Greedy Insertion and Regret-2 Insertion are introduced. When inserting the removed requests, the Basic Greedy Insertion explores each request waiting to be restored, and the point and position with the lowest insertion cost are given priority. Regret-2 addresses the short-sightedness of Basic Greedy Insertion by calculating the sum of the costs of the top two lowest-cost positions for inserting a request. The request with the highest cost sum is selected first, which helps jump out of the local optimum. Next, we use a roulette wheel method to assign weights to each operator, where better-performing operators have greater weights and higher probabilities of being selected. Acceptance and stopping criteria use simulated annealing acceptance criteria, which also helps fully explore the solution space and avoid falling into local optima.

Computational Experiments The Computational Experiments section uses Cplex and the ALNS algorithm to solve the mathematical model, verifying the effectiveness of both the model and the algorithm. The parameter tuning section finds more universal parameter settings for the algorithm, enhancing its application range. The sensitivity analysis part explores whether the introduction of OD and subsidies for OD can lead to cost savings. In the 20 experimental instances, the results show that introducing one OD, when the OD cost is 90% of the RD cost, the total routing cost can be reduced by 5.18%. When the OD cost is 80% of the RD cost, the total routing cost can be reduced by 15.21%. When using one OD and fixing the OD cost at 80% of the RD cost, the introduction of 60% capacity OD can save 7.86% of the total cost, 70% capacity OD can save 9.97% of the total cost, and 80% capacity OD can save 15.21% of the total cost. We conclude that the introduction of OD can reduce routing costs to varying degrees. The number of ODs, cost, and capacity directly affect the proportion of cost savings. At the same time, cost savings require companies to choose the appropriate number of ODs, the subsidy price for each OD, and the capacity of the ODs based on market research to balance the willingness of ODs, the quality of service, and cost savings.

Case Study Finally, we conducted a case study based on the Dutch e-commerce company Ochama, scaling down the instance to 150 requests and similarly reducing vehicle capacity according to actual vehicle capacity. We mapped the warehouses and intermediate centers to two-dimensional coordinates based on actual physical distances and randomly generated requests within the service area. Considering that vehicles need to run on highways and urban roads, we assumed an average vehicle speed of 60km/h, linking unit path distance to unit path distance cost. Out of 10 instances, 9 were valid. The third instance failed because the ALNS algorithm did not find a solution better than the initial solution within the set 10,000 iterations. The remaining 9 valid instances indicate that the introduction of different OD vehicle types can save total routing costs to varying degrees. Compared to the research by (Hou and Wang, 2021), which introduced OD and reduced the total routing cost by 7.3%, our approach reduced the total routing cost by 11.37% and 6.85% for small and medium-sized vehicles, respectively, when the cost-to-capacity ratio was equal. For small vehicles, with a cost ratio of 50%, the total routing cost could still be reduced by 5.11%.

Contents

List of Figures

List of Tables

1

Introduction

In recent years, due to the exponential growth of e-commerce, the number of urban parcel deliveries has also rapidly increased. This phenomenon has been further accelerated by the Covid-19 pandemic, bringing significant logistical challenges to the last-mile delivery in cities, especially in densely populated urban areas (Bhatti et al., 2020; Gao et al., 2020). Taking Amsterdam as an example, it is anticipated that by the year 2030, the city will see an influx of over 100,000 parcels each day (Guo et al., 2019). To complete the delivery of these parcels, it will not only put immense pressure on the logistics system, thereby exacerbating traffic congestion, environmental pollution, and disturbing the balance of urban life. For logistics companies, the increase in logistics costs will be the main issue faced, as studies have shown that the last-mile delivery costs account for 75% of the total logistics costs (Devari et al., 2017). To solve the last-mile delivery (LMD) problem, traditional parcel delivery has been modeled as the Vehicle Routing Problem (VRP) or the Pickup and Delivery Problem (PDP). For detailed definitions of the VRP and PDP, please refer to (Toth and Vigo, 2014; Parragh et al., 2008). Traditional mathematical models, when faced with the ever-increasing transportation demand, the most direct method is to use more vehicles or vehicles with larger capacities, otherwise, it will lead to reduced delivery efficiency, affecting consumer experience. These two solutions cannot alleviate the problems mentioned above and will cause transportation costs to increase significantly.

To address these challenges, crowdshipping (CS) has been proposed as a viable alternative to traditional LMD strategies (Allahviranloo and Baghestani, 2019; Lee et al., 2016), i.e., by utilizing ordinary people who have already planned routes, namely occasional drivers (OD), to complete pickups and deliveries on a detour and receive a small compensation. The application of crowdshipping in urban parcel delivery is not uncommon, with Amazon launching its first crowdshipping service in 2013, asking in-store customers to detour home after shopping at the grocery store to deliver packages to online shoppers living nearby. Subsequently, the service expanded to include ordinary drivers, who can pick up goods from retail stores, distribution centers, or other designated locations. A new program named Amazon Flex was launched in 2015 and is currently operating in over 50 cities in the United States (Huang and Ardiansyah, 2019). The development of crowdshipping logistics benefits from its own advantages: 1. Compared to regular drivers (RDs), ODs obtain less compensation for making detours to deliver parcels, meaning lower transportation costs (Archetti et al., 2016). 2. Using crowdshipping during peak times to reduce vehicle demand, thereby alleviating traffic congestion and emissions (Arslan et al., 2019; Ren et al., 2019). 3. Utilizing ODs to assist RDs in delivery can enhance delivery efficiency (Dahle et al., 2019). 4. By participating in crowdshipping delivery, ODs can earn money from underutilized assets.

Models combining crowdshipping with traditional VRP and PDP have been continuously proposed, such as the integration of CS with VRP to produce the Open VRP model (Torres et al., 2022), and the combination of CS with PDP to generate the PDP with occasional drivers model (Dahle et al., 2019). Most Open VRP models do not take the pickup process into account. Based on real-life scenarios, consumers may choose to return products in some cases, which is referred to as reverse logistics. Therefore, many logistics companies' fleets, while delivering, also collect parcels as much as possible

according to the vehicle capacity and then deliver them to a designated destination. The PDP with occasional drivers model considers this process. However, the destination in the PDP with occasional drivers model is generally not a depot. In reverse logistics, the delivery destination is usually set as a depot. Combining CS examples has also been studied, namely the Vehicle Routing Problem with Simultaneous Pickup and Delivery and Occasional Drivers (VRPSPDOD) (Vincent et al., 2023), considering each set of pickup and delivery requests can be fulfilled by a fleet of ODs or a pool of capacitated ODs, thus efficiently utilizing ODs to assist RDs in completing all delivery tasks. It is worth noting that situations may arise where the recipient is not at home for door-to-door delivery. In such cases, studies have used nearby grocery stores, train stations, and post offices as alternative addresses, which can be referred to as intermediate centers, offering a significantly lower cost solution compared to returning parcels to the depot (Song et al., 2009). This scheme is also applied in reverse logistics. These intermediate centers can receive a certain number of parcels from nearby areas, possibly exceeding the capacity of RDs or ODs. In this case, the consideration of split pickup and delivery (Nowak, 2005) is necessary, meaning each intermediate center can be served by multiple RDs and ODs. Finally, considering multiple depots is more in line with the reality of logistics distribution than a single depot. According to current research, no studies have combined all the aforementioned features.

Therefore, this research will propose a new variant of the PDP, which considers all the features mentioned above and can be seen as a combination of CS, PDP, and Open VRP. The new variant involves several main features: multi-depots, regular drivers, occasional drivers (crowd shippers), customers, and intermediate centers (stores). In the network, regular drivers (RD) start from the corresponding depot, serving their associated customers and intermediate centers, and appropriately pick up goods from intermediate centers when there is spare capacity. It is important to note that the depot and demand points must strictly correspond. Occasional drivers (OD) start from their own origin and reach the planned destination, which is not a depot, and the origin and destination can be the same. ODs can choose not only to go directly to a depot but also to pick up goods at an intermediate center, and ODs can perform multiple pickups and deliveries and visit depots multiple times. Each customer can only be served by one RD or one OD, while an intermediate center can be served by one or more RDs and ODs. Each demand point is bound to a depot, meaning that parcels are already stored at the specified depot, and parcels from the intermediate center need to be delivered to the specified depot. Finally, vehicle capacity and time window constraints are considered. We refer to this problem as: The Multi-trip Crowd-shipping Split Pickup and Delivery Problem with Time Window and Multi-depots (MC-PDPTW-MD).

The key to this research is to incorporate the new features mentioned above into the traditional PDPTW model, while the introduction of ODs will incur certain operational costs. In this study, we will simplify the objective function, considering only the costs caused by vehicle routing. Therefore, the objective function is to minimize the total routing costs of RDs and ODs. The main research question of this paper is:

How to optimize the total vehicle routing cost of vehicles of MC-PDPTW-MD Problem?

The structure of the remaining parts are as follows:
Chapter 2 is the literature review, which is divided into 2 main parts: related models and algorithms. Each sub-part is divided into three parts based on relevance to this study, exploring the origins and developments of the Crowd-shipping, Open VRP, and PDP problems, respectively. Chapter 3 provides a detailed definition and description of the research problem. Chapter 4 is divided into two parts, solving the proposed research problem using a mathematical model and the ALNS algorithm, respectively. Chapter 5 visualizes a small-scale instance, verifies the consistency of the mathematical model and the algorithm, performs parameter tuning on the algorithm, compares the performance of the mathematical model and the algorithm, and finally conducts a sensitivity analysis. Chapter 6 conducts a case study using the Ochama company as an example. Chapter 7 discusses the results of each part of this study. Chapter 8 is conclusions and recommendations. The conclusion part provides an overall summary of the research. The recommendations part offers some ideas for improving this study and directions for future research. Figure 1.1 provides the research framework of this thesis.

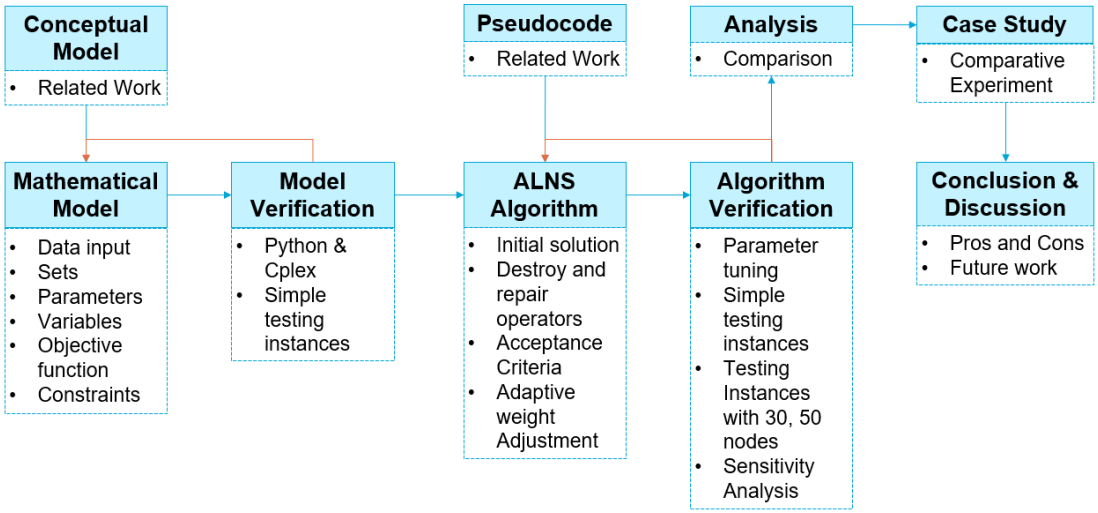

Figure 1.1: Research Framework

2

Literature study

Based on the introduction, the research theme of this research is the combination of multiple aspects of study, including the Crowd-shipping Problem, the Open Vehicle Routing Problem (Open VRP), and the Pickup and Delivery Problem with Time Window (PDPTW). The mathematical models involved in these related studies and the algorithms used to solve these problems will be discussed in detail in this section.

2.1. Related Models

This part will discuss the origin and development of three related problems, discuss in depth the characteristics of the initial model, and different features that are continuously added to the model during the development process to make the model suitable for specific research problems. These features include: Number of depots, RD quantity limit, delivery strategy (single/multiple/split), intermediate center, reverse logistics, etc.

2.1.1. Crowd-shipping Problem

Most research in the crowdsourcing field has focused on virtual tasks that can be completed remotely via the Internet, such as text editing, translation, and debugging (see Doan et al. (2011)). However, with the rapid development of e-commerce, crowdshipping as a delivery method in the sharing economy has garnered significant attention. Subsequently, crowdsourced delivery platforms have been offered as tools for implementing crowdsourced distribution, where occasional drivers can assist in the delivery of parcels or goods while earning compensation. In 2015, over 50 startups were categorized as crowdsourced delivery service providers on global IT platforms (websites and/or mobile applications), engaging in local delivery, freight, and freight brokerage activities (Carbone et al., 2017). In 2018, Walmart tested a new last-mile grocery crowdshipping platform called Spark Delivery, and Amazon introduced the Amazon Flex crowdsourcing service in 79 U.S. cities. These crowdshipping platforms continue to attract attention in academia and industry (Huang and Ardiansyah, 2019). Delivery platforms can generally be divided into two categories: e-retailers and courier companies. The aforementioned Walmart and Amazon belong to e-retailers, while courier companies include DHL MyWays, UberFreight, and PickThisUp, among others. This study will focus on courier companies, as the delivery platforms of Walmart and Amazon are already quite mature, and in the courier parcel field, due to the diversity of delivery forms, such as parcel lockers, drones, crowdshipping, etc., there are still many problems waiting to be researched. The operational logic of crowdsourced delivery platforms is controlled by algorithms written based on mathematical models, which include matching, scheduling, and compensation mechanisms. These crowdsourced delivery models include static and dynamic types; static models consider all vehicles and transportation demands are already determined, while dynamic models evolve over time as temporary drivers may join or leave the crowdsourced delivery platform, and demands are continuously updated over time. Dynamic problems are often solved using methods such as simulation or a rolling horizon framework (Arslan et al., 2019). In this study, since there is no need to make immediate adjustments based on temporary drivers and pickup and delivery

demands, the problem involved in this study can be modeled as a static model.

Regarding the static crowd-shipping problem, in the initial stages of addressing this issue, researchers focused on integrating this concept with traditional delivery models. (Archetti et al., 2016) pioneered this integration by combining RDs with ODs, allowing ODs to perform single pickup and delivery tasks, aiming to minimize total delivery costs including compensations for these ODs. The model introduced a unique variant of the classical capacitated vehicle routing problem. Building on (Archetti et al., 2016)'s research, (Macrina et al., 2017) considered that temporary drivers are not always available for delivery tasks, and the capacity of temporary vehicles allows these drivers to complete multiple delivery tasks. Therefore, (Macrina et al., 2017) introduced time windows into the model and explored scenarios that allow these drivers to make multiple and split deliveries. Results showed that allowing temporary drivers to make multiple deliveries significantly improved solution quality and cost savings. (Kafle et al., 2017) took intermediate centers (grocery stores) into account, but RDs only provided services between depots and intermediate centers, with occasional drivers responsible for delivering parcels from intermediate centers to customers. Due to the growth in the number of parcels, it cannot be guaranteed that enough ODs will deliver parcels from intermediate centers to customers. Therefore, a hybrid delivery model is more reliable. The research question discussed in this research combines features from the studies of (Macrina et al., 2017) and (Kafle et al., 2017), introducing intermediate centers in addition to customers, to address situations where customers are not at home, considering that ODs and RDs serve these customers and intermediate centers, and that temporary drivers can complete multiple delivery tasks. Given that the demand at intermediate centers might exceed the capacity of ODs and RDs, split deliveries are also considered.

2.1.2. Open Vehicle Routing Problem

In 2000, (Sariklis and Powell, 2000) first introduced the variant of the VRP problem known as Open VRP, characterized by vehicles not needing to return to the depot, or if needed, they could revisit customers in the reverse order. A depot has a regular truck fleet with a certain number of vehicles, while companies can subcontract all or part of the product delivery to external courier companies, i.e., rent vehicles to complete delivery tasks. Although the cost per unit driving distance (DT) of rented vehicles is higher, many costs such as capital, maintenance, and depreciation are not incurred (Tarantilis et al., 2004a). Typical OVRP requires that each demand point is served by only one vehicle, and the rented vehicles will be assigned to routes that do not require returning to the depot. The issue of vehicles returning to the station in reverse order, often occurs in the delivery and pickup process, which is the reverse logistics mentioned earlier. Vehicles first visit customers and deliver the goods they ordered. When vehicles reach their last assigned customer and the vehicle is empty, they return to the warehouse along the same route, visiting customers in the reverse order to collect goods that must be returned to the distribution center. This is somewhat similar to the model proposed by (Archetti et al., 2016), but (Sariklis and Powell, 2000) consider: 1. The use of a rented fleet instead of in-store customers as occasional drivers. 2. The limitation on the number of vehicles in the regular fleet, as assuming a sufficiently large number of regular vehicles is unrealistic. 3. Reverse logistics.

(Tarantilis and Kiranoudis, 2002) added new features to the OVRP: multi-depots. The proposed model is based on a case where an industrial company in Greece distributes fresh meat from various warehouses to customers located in nearby areas. (Brandão, 2004; Fleszar et al., 2009) built on (Sariklis and Powell, 2000) by adding vehicle operation time restrictions according to the legal driving hours of drivers. Similarly, in reality, customers cannot receive packages at any time of the day, hence (Repoussis et al., 2007) added time window constraints for customers. (Fu et al., 2005) considered not only the time window restrictions for vehicles but also the maximum operating distance limitations, and split the OVRP problem into three types: Delivery only, Pickup Only, and Both delivery and pickup. While previous research considered that each customer can only be served by one vehicle at a time, (Fu et al., 2005) suggested that perhaps allowing each demand point to be visited by different vehicles multiple times could offer more vehicle route options, higher vehicle capacity utilization, and less vehicle demand, and confirmed these speculations. Given the nature of OVRP, where drivers do not have to return to the depot after completing delivery tasks, not only can delivery tasks be handled by third-party logistics companies, but combining crowdshipping with OVRP might be a better idea. (Torres et al., 2022) introduced OVRP using crowdshippers instead of third-party logistics vehicles, proving that this combination can further reduce costs. Our research will build on (Torres et al., 2022) and combine the

discussed variants of OVRP, setting a certain number of RDs, and adding time window constraints for RDs, ODs, intermediate centers, and customers.

2.1.3. Pickup and Delivery Problem with Time Window

Since the 1990s, the Pickup and Delivery Problem with Time Window (PDPTW) has made significant progress in the field of logistics and transportation planning. VRPTW (Van Landeghem, 1988) is considered a special case of PDPTW, where all destinations are depots. Building on VRPTW and considering simultaneous pickup and delivery, the problem was defined and modeled in 1989 as VRP with simultaneous delivery and pick-up points, which can be regarded as an early type of PDP problems. PDPTW involves constructing optimal routes to satisfy transportation requests, with each route requiring pickup at the origin and delivery at the destination under capacity, time window, and priority constraints (Dumas et al., 1991). Additionally, each route meets pairing constraints, as the corresponding pickup and delivery locations must be serviced by the same vehicle. Considering issues such as increased energy costs, driver shortages, and vehicle capacity utilization, (Nowak et al., 2008) drew on the benefits of split delivery strategies in VRP, i.e., SDVRP, to propose Pickup and Delivery with Split Loads. SDVRP load planning is completed before the vehicle leaves the parking lot with the same fixed capacity. PDPSL is a more complex problem, primarily because the vehicle's available capacity changes with each pickup or delivery, and the vehicle ultimately does not return to the depot. Previous research only considered single depot problems, while (Irnich, 2000) studied a multi-depot pickup and delivery problem with a single hub and heterogeneous vehicles, setting pickups to always involve loading goods at a location and transporting them to the hub, and deliveries defined as loading goods at the hub and transporting them to a location. One characteristic of hub transportation networks is that many different requests need to be transported between a location and the hub. Thus, it is generally assumed that different requests for pickup or delivery occur at a specific location, i.e., requests with different quantities and time windows. Differently, we want to discuss the pickup and delivery problem between depots and intermediate centers, involving demands for pickup and delivery between multiple depots and multiple intermediate centers. Furthermore, (Irnich, 2000) also used heterogeneous vehicles, which leads us to think of crowdshippers.

In 2019, (Dahle et al., 2019) combined Crowdshipping with PDPTW to propose The Pickup and Delivery Problem with Time Windows and Occasional Drivers. Unlike typical heterogeneous vehicles, crowdshipping drivers have individual starting locations, destinations, costs, and time matrices. The study results showed that adding crowdshipping drivers to the traditional PDPTW model could save 10-15% of costs. While considering Crowdshipping, (Voigt and Kuhn, 2022) introduced a new feature: Transhipment Points (TP), similar to the previously mentioned intermediate centers. Drivers can pick up goods from pickup points and transport them to these TPs, then new drivers (including regular drivers and occasional drivers) come to pick up and deliver the parcels to the final recipients. However, we believe that adding TPs makes the delivery process more complex. If these TPs are directly considered as intermediate centers near the recipients, when recipients are inconvenient to receive goods, they can choose to have the goods delivered to nearby intermediate centers in advance, and then go to pick up the goods themselves. This not only facilitates recipients in picking up goods but can also reduce transportation costs. When considering multi-depots, the latest research (Tao et al., 2023) discusses The Pickup and Delivery Problem with Multiple Depots and Dynamic Occasional Drivers in Crowdshipping Delivery. This is a dynamic problem modeling where temporary drivers post their travel planning information and time windows on the crowdshipping platform, and the platform matches based on this information. Since the problem we are studying contains several additional features: multiple or split deliveries, intermediate centers, etc., we believe the platform will first generate temporary driver routes, and temporary drivers will choose the provided routes based on their personal circumstances, therefore, we model the problem we are studying as a static problem.

2.2. Related Algorithms

This section will further discuss the algorithms corresponding to the mathematical models discussed above, including both exact algorithms and metaheuristic algorithms. Common exact algorithms include Mixed-Integer Linear Programming (MILP) and Branch and Bound (B&B), etc. Common metaheuristic algorithms include Tabu Search, Genetic Algorithms, Simulated Annealing, Ant Colony Optimization, and Variable Neighborhood Search, among others.

2.2.1. Crowd-shipping Problem

In terms of algorithms, (Archetti et al., 2016) proposed a multi-start heuristic approach by combining variable neighborhood search (VNS) and tabu search. It used a greedy insertion algorithm to construct a route with only regular drivers, and inserted customers in non-decreasing order of distance from the depot. In the active route, when the capacity limit is exceeded, a new route is restarted, and the customers in the initial route are removed, exchanged, etc. using tabu search and variable neighborhood search mechanisms, and delivered to the temporary driver. Comparing the exact solution of this algorithm with CPLEX for 50 instances of temporary drivers, CPLEX cannot give results in an hour while the heuristic method can solve it in seconds. (Kafle et al., 2017) chose to use the simulated annealing algorithm to solve the proposed model. The reason is that the simulated annealing algorithm is more suitable for large sample instances than the branch and bound algorithm, and is faster than other heuristic algorithms such as genetic algorithms. Provides a pretty good solution to large routing problems. (Gdowska et al., 2018)using meta-heuristics or MIP, all internally employing Tabu Search algorithms. However, research on crowd-shipping problems tends towards the use of more complex and hybrid models, with different studies proposing their unique approaches. To better solve the dynamic crowd-shipping delivery problem proposed by (Dayarian and Savelsbergh, 2020), (Pugliese et al., 2022) developed a Variable Neighborhood Search (VNS) method combined with machine learning techniques, focusing on exploring promising areas in the search space, using reinforcement learning to guide local search actions during the intensification phase. To address the same problem, (Di Puglia Pugliese et al., 2022) designed a Greedy Randomized Adaptive Search Procedure (GRASP) and proposed a hybrid approach where VNS is used as a local search. VNS has been increasingly applied in recent years to solve more complex routing problems.

2.2.2. Open Vehicle Routing Problem

(Sariklis and Powell, 2000) proposed a heuristic method based on a minimum spanning tree with a penalty process to solve Open VRP. The algorithm is divided into three stages. The first stage generates the initial solution, and the second stage uses penalties to modify the minimum generation. Tree Solution,The third stage converts infeasible solutions into feasible,solutions and compares them with the customer exchange heuristic,algorithm. This algorithm was ultimately found to outperform the client exchange heuristic, especially for large and/or loosely constrained problems, but often requires longer computation time. Subsequently, in 2004, (Brandão, 2004) attempted to apply algorithms used for solving VRP to solve Open VRP. Building on (Sariklis and Powell, 2000), they considered the maximum route length and introduced the Tabu Search algorithm. Continuing the research of (Sariklis and Powell, 2000), (Fu et al., 2005) added vehicle capacity constraints. (Tarantilis et al., 2005) identified that the practical problem of goods distribution is Open VRP and proposed a simple yet effective variant of the Threshold Accepting algorithm known as List-Based Threshold Accepting (LBTA), which uses a series of thresholds to intelligently guide the local search and reflects the method's memory. Additionally, they also introduced the Adaptive memory-based tabu search (BR) (Shaw, 1998; Tarantilis et al., 2004a), and Backtracking adaptive threshold accepting (BATA) (Tarantilis et al., 2004b) algorithms for solving Open VRP.

In 2007, (Pisinger and Ropke, 2007) mentioned that the ALNS framework could be used to solve five variants of the vehicle routing problem, including Open VRP, and transformed these variants into PDPTW. The relevance of the PDP problem will be discussed in the following section. The ALNS framework is an extension of the neighborhood search model proposed by (Shaw, 1998), where its algorithm modifies a large number of variables in each iteration. In the ALNS algorithm, the current partial solution of each iteration is destroyed and repaired using heuristic methods. (Fleszar et al., 2009) used the Variable Neighborhood Search (VNS) algorithm to solve Open VRP, differing from ALNS in that it explores the solution space through systematic changes in predefined neighborhood structures. (Fleszar et al., 2009) suggested using a multi-phase oscillated VNS to address OpenVRP, where, unlike ALNS, VNS does not generate new solutions through destruction and repair steps but rather gradually explores the solution space by changing the size of neighborhoods. Following the algorithm of (Fleszar et al., 2009), (Şevkli and Güler, 2017) modeled a real-world newspaper delivery problem as Open VRP and proposed a new multi-phase oscillated VNS algorithm using the Kmeans clustering algorithm to construct an initial solution to solve the problem.

Apart from the ALNS algorithm, researchers have proposed a variety of algorithms to solve Open VRP.

For instance, in 2010, (Cao and Lai, 2010) introduced the concept of OVRP with Fuzzy Demands (OVRPFD), utilizing a model based on fuzzy credibility theory and an enhanced differential evolution algorithm to tackle the ambiguity in demands. In the same year, (Salari et al., 2010) proposed a heuristic improvement procedure based on Integer Linear Programming (ILP) technology for OVRP, achieving new optimal solutions in standard benchmark instances. In 2011, (MirHassani and Abolghasemi, 2011) applied Particle Swarm Optimization (PSO) algorithms to OVRP, further extending its applicability. By 2014, (Cao et al., 2014) focused on OVRP with demand uncertainty and its robust strategies, expanding the concept of OVRP, especially in terms of effective route planning under uncertain demands. In 2016, (Vincent et al., 2016) proposed research combining OVRP with cross-docking technology, aimed at minimizing overall costs in a cross-docking environment while ensuring timely service to all customers. In 2019, (Tavakkoli-Moghaddam et al., 2019) presented various types of VRP, emphasizing the importance of good management in cost and satisfaction. By 2023, (Ahmed and Yousefikhoshbakht, 2023) researched OVRP with time-windowed heterogeneous fixed fleets, showcasing the applicability of the OVRP model in more complex environments. However, ALNS proves to be more effective in dealing with particularly complex optimization problems, especially when the solution space is very large and there is a significant difference in the quality of solutions. This is also the reason why the ALNS algorithm is becoming increasingly popular.

2.2.3. Pickup and Delivery Problem with Time Windows

In 1991, (Dumas et al., 1991) proposed an exact algorithm for PDP with time windows. This algorithm used a column generation method to solve problems with path constraints and was capable of handling multiple sites and different types of vehicles. This study laid the foundation for subsequent solution methods for PDP problems. Entering the 21st century, PDP research began to employ more complex metaheuristic methods. For example, Nanry and Barnes proposed a reactive tabu search method in 2000 (Nanry and Barnes, 2000), which effectively solved the PDP problem using different moving neighborhoods. Similarly, Li and Lim in 2001 (Li and Lim, 2001) proposed a metaheuristic algorithm combining simulated annealing and tabu search, particularly suitable for large-scale multi-vehicle PDP problems.(Bent and Van Hentenryck, 2006) proposed a two-stage hybrid algorithm, which used simulated annealing in the first stage to reduce the number of routes and large neighborhood search in the second stage to lower the total travel cost.

In 2006, (Ropke and Pisinger, 2006a) introduced an adaptive large neighborhood search heuristic method. This method, utilizing the competitive use of sub-heuristics, effectively improved solutions for the Pickup and Delivery Problem (PDP). The standard structure of ALNS provided by (Ropke and Pisinger, 2006a) comprises at least four important parts. These parts are: a the adaptive mechanism for selecting the deployed operators, b the criterion to accept a newly obtained solution, c the stopping criterion of the algorithm, and d the design of destroy and repair operators. (Ropke and Pisinger, 2006a) employed strategies such as Random removal, Worst removal, and Shaw removal during the destroy operator phase, and used Greedy insertion and (k-)Regret insertion during the repair operator phase. For the Acceptance Criteria stage, a simulated annealing algorithm was used, and finally, an appropriate Adaptive Weight Adjustments strategy was proposed. As research on PDP deepened, algorithms began to consider more complex scenarios. For instance, (Ghilas et al., 2016) proposed an adaptive large neighborhood search heuristic algorithm for solving PDP with scheduled lines, utilizing the ALNS algorithm for solution finding, with initial solutions generated using a greedy insertion heuristic algorithm. Various methods were considered for the operator phase, including random removal, route removal, Late-arrival removal, and others. Similarly, the operator insertion stage considered multiple methods such as Greedy insertion, second-best insertion, and Greedy insertion with a noise function. The acceptance stage algorithm adopted the simulated annealing algorithm, similar to (Ropke and Pisinger, 2006a). (Sampaio et al., 2020) applied the ALNS methodology proposed by (Ropke and Pisinger, 2006a) to solve a multi-stop pickup and delivery problem with time windows and transfers.

2.3. Conclusion of Literature Review

The literature review section provides a detailed depiction of the evolution and development of models and algorithms used to solve the Crowd-shipping Problem, Open VRP, and PDPTW. The purpose is to filter the features contained in the model we intend to study, such as time windows, occasional drivers, etc., through the discussion and analysis of these models. By combining these features, a new problem

(Multi-trip Crowd-shipping Split Pickup and Delivery Problem with Time Window and Multi-depots) is proposed. There is no consistent research on the newly proposed problem, which constitutes what is known as a research gap. At the same time, by analyzing the algorithms used in different models and the trend of algorithm development, a suitable algorithm will be selected and discussed. This study will choose ALNS as the preferred algorithm, with the specifics of the ALNS algorithm discussed in chapter 4. Table 2.1 summarizes the features involved in the models reviewed in the literature, with the last row indicating the features included in our research.

Refereces	Type of Problem	Features							Algorithm
		Number of Regular Drivers	Occasional Drivers	Single, Multi, or Split Delivery	Depot	Time Window	Intermediate Centers	Delivery only or Pickup and Delivery	
(Archetti et al., 2016)	VRP & Crowd-shipping	Unlimited	Yes	Single	1	No	No	Delivery only	Multi-start heuristic
(Macrina et al., 2017)	VRP & Crowd-shipping	Limited	Yes	Single, Multi, and Split	1	Yes	No	Delivery only	Branch and Bound
(Kafle et al., 2017)	VRP & Crowd-shipping	Limited	Yes	Single	1	Yes	Yes	Delivery only	Tabu Search
(Sariklis and Powell, 2000)	Open VRP	Limited	Yes (3PL)	Single	1	No	No	Delivery only	Heuristic method
(Tarantilis and Kiranoudis, 2002)	Open VRP	Limited	Yes (3PL)	Single	Multiple	No	No	Delivery only	List-based threshold accepting (LBTA)
(Brandão, 2004) (Fleszar et al., 2009)	Open VRP	Limited	Yes (3PL)	Single	1	Yes (Drivers only)	No	Pickup and Delivery	Tabu Search
(Repoussis et al., 2007)	Open VRP	Limited	Yes (3PL)	Single	1	Yes (Drivers and Customers)	No	Pickup and Delivery	Greedy look-ahead route construction heuristic algorithm
(Fu et al., 2005)	Open VRP Open VRP & Crowd-shipping	Unlimited	No	Single	1	Yes	No	Pickup and Delivery	Tabu search
(Torres et al., 2022)		Limited	Yes	Single	1	Yes	No	Pickup and Delivery	Column generation heuristic
(Dumas et al., 1991)	PDPTW	Limited	No	Single	1	Yes	No	Pickup and Delivery	Forward dynamic programming algorithm.
(Nowak et al., 2008)	PDPSL	Limited	No	Split	1	No	No	Pickup and Delivery	Local Search
(Irnich, 2000)	PDPTW	Limited	No	Single	Multiple	Yes	No	Pickup and Delivery	Covering heuristic
(Dahle et al., 2019)	PDPTW & Crowd-shipping	Limited	Yes	Single	Multiple	Yes	No	Pickup and Delivery	\
(Voigt and Kuhn, 2022)	PDPTW & Crowd-shipping	Limited	Yes	Single	1	Yes	Yes	Pickup and Delivery	ALNS
(Tao et al., 2023)	PDPTW & Crowd-shipping	Limited	Yes	Single	Multiple	Yes	Yes	Pickup and Delivery	Online event-based rolling horizon approach and a simple insertion heuristic
Our Research	**PDPTW & Crowd-shipping**	**Limited**	**Yes**	**Multi and Split**	**Multiple**	**Yes**	**Yes**	**Pickup and Delivery**	**ALNS**

Note: The table lists some of the main features; see the literature review for specific differences.

Table 2.1: Summary of Features and Algorithms for Models

3

Problem Definition

In this section the Multi-trip Crowd-shipping Split Pickup and Delivery Problem with Time Window and Multi-depots (MC-PDPTW-MD) is formally defined. After discussion, in addition to the common time window constraints and vehicle capacity constraints, the model to be studied also contains the following characteristics: Regular drivers, Occasional drivers (crowdshippers), multi-depots, intermediate centers, multi-delivery and split delivery.

The MC-PDPTW-MD is an complex logistics optimization challenge, focusing on the interplay between a regular truck fleet and an occasional driver fleet to minimize total routing costs. Figure 3.1 shows an illustrative example of the multi-trip crowd-shipping split pickup and delivery problem. The operational framework of this problem encompasses a central Distribution Center (DC), various retailers serving as intermediate centers, and customers. At the core of this scenario is the regular truck fleet (RDs), which embarks on its journey from the DC. Each RD is loaded with parcels destined for delivery to intermediate centers and customers. The delivery schedule is intricately planned within specific time windows, ensuring timely and efficient service. While delivering the package, RDs engage in the pickup of return items only from intermediate centers. This aspect of their route is flexible, contingent upon the spare capacity available post-delivery. The RDs' routes culminate back at the DC, where they offload the collected items for further processing. Complementing the regular fleet is the occasional driver (OD) fleet, characterized by its flexibility and adaptability. ODs initiate their routes from varied locations, not bound to the DC as their starting point. OD's primary role is to pick up items from intermediate centers and delivery them back to depot or pickup items from depots and distribute them to customers and intermediate centers. while ODs are delivering, parcels can be picked up at intermediate centers as vehicle capacity allows. ODs head to his/her destination after finishing all delivery tasks. If possible, RD and OD can visit multiple depots multiple times. The essence of the MC-PDPTW-MD lies in the seamless coordination between these two fleets. The system strategically allocates tasks and routes to each fleet, considering real-time dynamics like vehicle capacity, delivery and pickup time windows, and on-the-ground traffic conditions. The objective is a harmonious balance, where the regular fleet's structured routes complement the occasional drivers' flexible pickups, ensuring overall efficiency and cost-effectiveness. The challenge is to manage these split pickups and deliveries within the constraints of time windows, aiming for a solution that minimizes routing costs.

- **Intermediate Center (C):** A intermediate center or a retailer can receive parcels and have some to be sent to depot.
- **Client (S):** Client only receiving parcels at home
- **Distribution Centre (DC):** A distribution center or depot can receive packages from copy centers and store packages for delivery to intermediate centers and customers.
- **Regular Driver (RD):** Regular drivers depart from DC, and end to DC
- **Occasional Driver (OD):** Occasional Drivers start from their origin and have a final destination, they may visit DC multiple times.

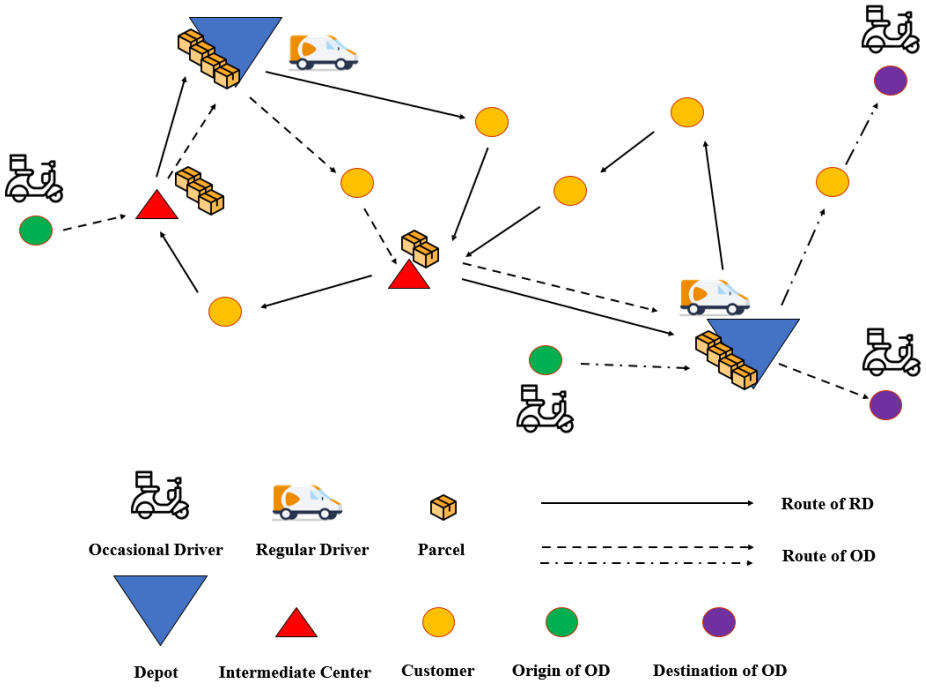

Figure 3.1: An illustrative example of multi-trip crowd-shipping split pickup and delivery problem

4

Methodology

This chapter presents two approaches to solve the problem: an exact method and a metaheuristic algorithm. We begin by formulating a precise mathematical model for the problem in section 4.1. To solve this model, we employ the branch-and-bound technique, a commonly used exact method. Despite the accuracy of exact methods, they often require substantial computational time to find feasible or optimal solutions. This limitation leads us to consider metaheuristic algorithms as a viable alternative.

Metaheuristic algorithms offer a practical solution when the problem scale increases, making exact methods computationally prohibitive. Previous studies have explored various metaheuristic techniques, including Local Search and Tabu Search. In 2005, (Ropke and Pisinger, 2006a) introduced the Adaptive Large Neighborhood Search (ALNS) framework, an enhancement of the Large Neighborhood Search (LNS) algorithm. They successfully applied ALNS to the Pickup and Delivery Problem with Time Windows (PDPTW), demonstrating superior performance compared to standard LNS. Building on this foundation, our research adopts the ALNS algorithm to effectively address the problem in section 4.2. Detailed discussions of the mathematical model and the ALNS implementation are provided in subsequent sections.

4.1. Mathematical Model

Let N denote the set of all nodes, let Δ denote the set of distribution centers and Δ^{dummy} denote the set of dummy nodes make sure OD can visit depot multiple times. Let S denote the set of customers, O^{od} and D^{od} denote the set of origins and destinations of occasional drivers. Let C denote the set of intermediate centers, usually grocery stores.

N consists of DC, C, client and origin and destinations of OD, thus $N = \Delta \cup \Delta^{dummy} \cup C \cup S \cup O^{od} \cup D^{od}$. A regular driver set RD, and a occasional driver set OD, are available to serve the requests. A set of vehicles K consists of RD and OD, thus $K = RD \cup OD$. For RD, each vehicle $k \in RD$ will start from and back to Depot. For OD, each vehicle $k \in OD$ has an origin $i \in O^{od}$ and a destination $j \in D^{od}$. As shown in Figure 3.1, all RDs have origins and destinations at the same depot, while the origins and destinations of the ODs may be spread out.

Each vehicle k has a capacity C_k, and there is a cost factor ρ^k for each vehicle k. When transporting goods from node i to node j, the time distance between node i and node j is given as T_{ij}. For each node $i \in N$ there is a time window $[T_i^0, T_i^1]$ within which it must be serviced. The delivery and pickup quantity at node $i \in N/\Delta$ are D_i and P_i respectively. Regular vehicle k has a time window for its origin depot $[T_0^0, T_0^1]$. For the ODs it is assumed that the time window are wide enough to allow for a direct travel.

The variable x_{ij}^k is a binary variable denoting if vehicle $k \in K$ uses arc (i, j) when $i, j \in N$. While variable y_{ij}^{kse} is pickup and delivery flow variable denotes load on vehicle $k \in K$ for pickup or delivery request from node s to node e on arc (i, j), $ij \in N$. Variable z_i^{kse} denotes pickup or delivery quantity at node i for vehicle $k \in K$ and request from start point s to end point e, $se \in N$. Variable dd_{ij}^k and pd_{ij}^k

are decision variables for delivery and pickup requests respectively denoting if vehicle $k \in K$ serve request from start node i and end node j. For variable dd_{ij}^k, $n \in \Delta, i \in \Delta_n, j \in DV_n$. For variable pd_{ij}^k, $n \in \Delta, j \in \Delta_n, i \in PV_n$. t_i^k means the time of vehicle $k \in K$ arrive at node $i \in N$.

Table 4.1 provide an overview about all the mentioned sets, parameters and variables.

Sets	
N	Set of all nodes
Δ	Set of depots
Δ^{dummy}	Set of dummy depots
Δ_n	Set of depot and its dummy depot for depot n
S	Set of customers
C	Set of Intermediate centers
DV	Set of delivery requests
PV	Set of pickup requests
DV_n	Set of delivery requests of depot n
PV_n	Set of pickup requests of depot n
K	Set of vehicles
K_n	Regular vehicle set belongs to depot n
RD	Set of regular drivers
OD	Set of occasional drivers
O^{OD}	Set of origins occasional drivers
D^{OD}	Set of destinations occasional drivers
Parameters	
T_{ij}	Time distance between nodes i and j
DD_{ij}	Pickup and delivery quantity between nodes i and j
C_k	Capacity vehicle k
V_i	Depot that provides service for request i
T_i^0, T_i^1	Time windows of node i
ρ^k	Cost factor of vehicle k
M	Big value
Variables	
x_{ij}^k	Routing variable
y_{ij}^{kse}	Pickup and delivery flow variable
z_i^{kse}	Pickup and delivery quantity variable
dd_{ij}^k	Decision variable for delivery requests
pd_{ij}^k	Decision variable for pickup requests
t_i^k	Time of vehicle k at node i

Table 4.1: Set up of Sets, Parameters and Variables

The formulate of the problem is given as follows:

Objective Function:

$$\min \sum_{k \in K} \sum_{(ij) \in N} x_{ij}^k T_{ij} \rho^k \tag{4.1}$$

Constraints for all vehicles:

$$\sum_{j \in N} x_{ij}^k - \sum_{j \in N} x_{ji}^k = 0 \qquad \forall i \in N / \left(O^{OD} \cup D^{OD}\right), k \in K \tag{4.2}$$

$$\sum_{k\in K}\sum_{j\in N} x_{ij}^{k} = 1 \qquad \forall i \in S \cup C \quad (4.3)$$

$$\sum_{j\in N} x_{ij}^{k} \leq 1 \qquad \forall k \in K, i \in \Delta \quad (4.4)$$

$$T_i^0 \leq t_i^k \leq T_i^1 \qquad \forall i \in N, k \in K \quad (4.5)$$

$$x_{ii}^{k} = 0 \qquad \forall i \in N, k \in K \quad (4.6)$$

$$t_n^k + T_{ni} \leq t_i^k \qquad \forall n \in \Delta, i \in DV_n, k \in K, k \notin K_n \quad (4.7)$$

$$\sum_{j\in N} x_{nj}^{k} \geq \sum_{j\in N} x_{ji}^{k} \qquad \forall n \in \Delta, i \in DV_n, k \in K \quad (4.8)$$

$$t_n^k \geq t_i^k + T_{in} \qquad \forall n \in \Delta, i \in PV_n, k \in K, k \notin K_n \quad (4.9)$$

$$\sum_{j\in N} x_{jn}^{k} \geq \sum_{j\in N} x_{ji}^{k} \qquad \forall n \in \Delta, i \in PV_n, k \in K \quad (4.10)$$

$$\sum_{j\in N} y_{ji}^{k,s,e} - \sum_{j\in N} y_{ij}^{k,s,e} = z_i^{k,s,e} \qquad \forall e \in DV, s \in \Delta_{V_e}, i \in N, k \in K \quad (4.11)$$

$$\sum_{k\in K} z_i^{k,s,e} = DD_{V_e,i} * \sum_{k\in K} dd_{s,i,k} \qquad \forall e \in DV, i \in N, s \in \Delta_{V_e} \quad (4.12)$$

$$\sum_{j\in N} y_{ji}^{k,s,e} - \sum_{j\in N} y_{ij}^{k,s,e} = z_i^{k,s,e} \qquad \forall s \in PV, e \in \Delta_{V_s}, i \in N, k \in K \quad (4.13)$$

$$\sum_{k\in K} z_i^{k,s,e} = DD_{i,V_s} * \sum_{k\in K} pd_{i,e,k} \qquad \forall s \in PV, i \in N, e \in \Delta_{V_s} \quad (4.14)$$

$$\sum_{s\in N}\sum_{e\in N} y_{ij}^{k,s,e} \leq C_k * x_{ij}^{k} \qquad \forall i, j \in N, k \in K \quad (4.15)$$

Constraints for regular drivers:

$$t_j^k \geq t_i^k + T_{ij} - M * (1 - x_{ij}^k) \qquad \forall n \in \Delta, k \in K_n, i, j \in N, j \neq n \quad (4.16)$$

$$t_j^k \geq (T_i^0 + T_{ij}) * x_{ij}^k \qquad \forall n \in \Delta, k \in K_n, i, j \in N, j \neq n \quad (4.17)$$

$$t_n^k \leq t_i^k \qquad \forall n \in \Delta, k \in K_n, i \in N, i \neq n \quad (4.18)$$

$$\sum_{j \in N} x_{ij}^k \leq 0 \qquad \forall i \in \Delta_{dummy}, k \in RD \quad (4.19)$$

$$\sum_{j \in N} x_{ij}^k \leq 0 \qquad \forall k \in RD, i \in \left(O^{OD} \cup D^{OD}\right) \quad (4.20)$$

Constraints for occasional drivers:

$$\sum_{j \in N/(O^{OD} \cup D^{OD})} x_{O_k^{OD} j}^k - \sum_{j \in N/(O^{OD} \cup D^{OD})} x_{j D_k^{OD}}^k = 0 \qquad \forall k \in OD \quad (4.21)$$

$$\sum_{i \in \Delta} \sum_{j \in N} x_{ij}^k - \sum_{j \in N/(O^{OD} \cup D^{OD})} x_{O_k^{OD} j}^k \geq 0 \qquad \forall k \in OD \quad (4.22)$$

$$\sum_{j \in N} x_{O_k^{OD} j}^k \leq 1 \qquad \forall k \in OD \quad (4.23)$$

$$\sum_{j \in N} x_{j O_k^{OD}}^k \leq 0 \qquad \forall k \in OD \quad (4.24)$$

$$\sum_{j \in S} x_{O_k^{OD} j}^k \leq 0 \qquad \forall k \in OD \quad (4.25)$$

$$\sum_{j \in N} x_{D_k^{OD} j}^k \leq 0 \qquad \forall k \in OD \quad (4.26)$$

$$x_{O_k^{OD} D_k^{OD}}^k \leq 0 \qquad \forall k \in OD \quad (4.27)$$

$$x^k_{i,i+n_{depots}} \leq 0 \qquad \forall i \in \Delta, k \in OD \quad (4.28)$$

$$x^k_{i+n_{depots},i} \leq 0 \qquad \forall i \in \Delta, k \in OD \quad (4.29)$$

$$\sum_{j \in N} x^k_{ij} \leq 0 \qquad \forall k \in OD, i \in O^{OD}/O^{OD}_k \quad (4.30)$$

$$\sum_{j \in N} x^k_{ji} \leq 0 \qquad \forall k \in OD, i \in D^{OD}/D^{OD}_k \quad (4.31)$$

$$t^k_j \geq t^k_i + T_{ij} - \left(1 - x^k_{ij}\right) M \qquad \forall i, j \in N, k \in OD \quad (4.32)$$

$$t^k_j \geq \left(T^0_i + T_{ij}\right) x^k_{ij} \qquad \forall i, j \in N, k \in OD \quad (4.33)$$

Constraints generated by allowing vehicles to visit the depot multiple times:

$$z^{k,n,e}_i = 0 \qquad \forall e \in DV, n \in \Delta_{V_e}, k \in K, i \in (\Delta \cup \Delta_{dummy}), i \neq n \quad (4.34)$$

$$z^{k,s,n}_i = 0 \qquad \forall s \in PV, n \in \Delta_{V_s}, k \in K, i \in (\Delta \cup \Delta_{dummy}), i \neq n \quad (4.35)$$

$$\sum_{k \in K} \sum_{i \in \Delta_n} dd^k_{i,j} = 1 \qquad \forall n \in \Delta, j \in DV_n \quad (4.36)$$

$$\sum_{k \in K} \sum_{j \in \Delta_n} pd^k_{i,j} = 1 \qquad \forall n \in \Delta, i \in PV_n \quad (4.37)$$

$$\sum_{m \in \Delta_n} dd^k_{m,i} \geq \sum_{j \in N} x^k_{ij} \qquad \forall n \in \Delta, i \in DV_n, k \in K \quad (4.38)$$

$$\sum_{m \in \Delta_n} pd^k_{i,m} \geq \sum_{j \in N} x^k_{ij} \qquad \forall n \in \Delta, i \in PV_n, k \in K \quad (4.39)$$

$$t^k_i \leq t^k_j + \left(1 - dd^k_{i,j}\right) * M \qquad \forall n \in \Delta, i \in \Delta_n, j \in DV_n, k \in K \quad (4.40)$$

$$t_i^k \leq t_j^k + \left(1 - pd_{i,j}^k\right) * M \qquad \forall n \in \Delta, j \in \Delta_n, i \in PV_n, k \in K \quad (4.41)$$

$$\sum_{m \in \Delta_n} \sum_{j \in N} x_{mj}^k \geq \sum_{j \in N} x_{ji}^k \qquad \forall n \in \Delta, i \in DV_n, k \in K \quad (4.42)$$

$$\sum_{m \in \Delta_n} \sum_{j \in N} x_{jm}^k \geq \sum_{j \in N} x_{ji}^k \qquad \forall n \in \Delta, i \in PV_n, k \in K \quad (4.43)$$

The objective function 4.1 aims to minimize the total routing cost. Constraints 4.2 - 4.15 are linked to all vehicles. Constraint 4.2 serves as the vehicle flow conservation constraint. Constraint 4.3 ensures every request must be served once. Constraint 4.4 restricts each vehicle to depart from the depot on at most one route. Constraint 4.5 is the time window restriction for vehicles arriving at each point. Constraint 4.6 prevents vehicles from passing through the same point consecutively. Constraints 4.7-4.8 ensure that vehicles must pass through the corresponding depot before meeting the delivery requirements. Similarly, Constraints 4.9-4.10 ensure that vehicles must pass through the corresponding depot after meeting the pickup requirements. Constraints 4.11 and 4.13 record the flow for each pickup and delivery requirement. Constraints 4.12 and 4.14 satisfy the pickup and delivery requirements at all points. Constraint 4.15 is the vehicle capacity limit.

Constraints 4.16-4.20 are linked to regular drivers. Constraints 4.16-4.17 are the time constraints between two points for RDs. Constraint 4.18 limits the departure time from the depot for RDs to not exceed the time to any point. Constraint 4.19 RDs are not allowed to visit dummy depots. Constraint 4.20 RDs are not allowed to visit O^{OD} and D^{OD}.

Constraints 4.21 - 4.35 are linked to occasional drivers. Constraint 4.21 flow conservation for origins and destinations of ODs. Constraint 4.22 if OD departs, then it must visit the depot. Constraint 4.23 at most one arc after the origin of the ODs. Constraint 4.24 no arc entering the origin of ODs. Constraint 4.25 first arc after origin cannot go to a customer. Constraint 4.26 no arc exiting from destination of ODs. Constraint 4.27 no link between origins and destinations of ODs. Constraints Constraints 4.28-4.29 no link between depots and dummy depots. Constraints 4.30-4.31 ODs cannot visit the origin and destination of other ODs. Constraints 4.32-4.33 are the time constraints between two points for ODs.

Constraints 4.34-4.35 no cross-docking during the pickup and delivery process. Constraints 4.36-4.37 for a delivery or pickup demand, it must be served by its depot or dummy depot. Constraints 4.38-4.39 the relationship between pickup and delivery variables and path variable. Constraints 4.40-4.41 if the delivery or pickup request is serviced by the corresponding depot/dummy, the time limit is met. 4.42-4.43 if the delivery or demand is served by vehicle k, then vehicle k must have passed through its depot/dummy.

4.2. ALNS Algorithm

According to literature research, the ALNS algorithm has been widely applied to vehicle routing problems, and ALNS is suitable for our problem. The strategy of each step in the ALNS algorithm, as well as the involved parameters, directly affect the computation time and the generation of optimal solutions. Therefore, the selection of strategies and the tuning of parameters is a labor-intensive and time-consuming process. As discussed in the section 2.1.3, (Mara et al., 2022) summarized the model proposed by (Ropke and Pisinger, 2006a) into four steps: (a) Adaptive mechanism, (b) Acceptance criteria, (c) Stopping criteria, and (d) Design of destroy and repair operators. Regarding the Adaptive mechanism, based on the 251 papers surveyed in (Mara et al., 2022), 250 papers utilized the Roulette Wheel ((Laporte et al., 2010), (Hemmelmayr et al., 2012), (Braaten et al., 2017)), meaning the probability of each operator being selected is the same. For Acceptance criteria, most articles adopt the Metropolis Criterion, allowing the algorithm to accept solutions slightly worse than the current one with a certain probability to avoid local optima, thereby increasing the algorithm's capability to explore a

broader area of the solution space. Some articles also utilize the Greedy Mechanism and Record-to-Record. Regarding the Termination Criterion, the majority of articles use the most common Number of Iterations, with research also employing the Number of Non-Improving Iterations, Running Time Limit, and Annealing Temperature.

The following table 4.2 summarizes the strategies chosen in some of the research studies that utilized the ALNS algorithm. In the table, each criterion encompasses additional methods. For example, among the removal operators, it includes Time-based removal ((Demir et al., 2012)), Demand-based removal ((Demir et al., 2012)), and Cluster removal ((Ropke and Pisinger, 2006a), (Pisinger and Ropke, 2007), and (Pisinger and Ropke, 2019)). Similarly, within the repair operators, it comprises Sequential insertion ((Kovacs et al., 2012)), Swap insertion ((Coelho et al., 2012b)), and Cluster insertion ((Maknoon and Laporte, 2017) and others). This study will compare some of these criteria and aim to find the criterion with the shortest computation time.

Algorithm 1 illustrates the framework of the used ALNS algorithm. Each step of this framework will be discussed in detail in the subsequent sections. The algorithm requires two input parameters: the initial solution s, which is generated using a greedy algorithm as described in section 4.2.1, and a number q, representing the count of demand nodes to be removed. The core of the algorithm is from lines 5 to 8. In line 5, an appropriate removal operator is selected based on the current weights of the removal operators. This paper uses two removal operators: random removal and worst removal, detailed in section 4.2.2. In line 6, the selected requests are removed from s. In line 7, a suitable repair operator is chosen based on the current weights of the repair operators. The repair operators used are greedy repair heuristic and regret-2 repair heuristic, detailed in section 4.2.3. In line 7, the requests are reinserted into the destroyed solution. The choice of removal and repair operators directly affects the performance of the algorithm. Following the strategy of (Ropke and Pisinger, 2006a). we use a roulette wheel method for operator selection, detailed in section 4.2.5. Considering some unnecessary depot visits in the initial solution, the depot, serving as both a pickup and delivery point, may be permanently removed during the destruction process. While this can help find a better solution, it may also lead to infeasible solutions. Therefore, in line 9, we need to check if the repaired solution is feasible. If feasible, we then determine whether the new solution should be accepted. This paper uses a simulated annealing acceptance criteria to decide whether to accept the new feasible solution, detailed in section 4.2.6.

Algorithm 1 ALNS Heuristic

```
1:  function ALNS(s ∈ {solutions}, q ∈ ℕ)
2:      solution s_best = s;
3:      repeat
4:          s' = s
5:          choose a destroy operator
6:          remove q requests from s'
7:          choose a repair operator
8:          reinsert removed requests into s'
9:          if s' is feasible then
10:             if f(s') < f(s_best) then
11:                 s_best = s'
12:             end if
13:             if accept(s', s) then
14:                 s = s'
15:             end if
16:         else
17:             continue
18:         end if
19:         operator adaptive weight adjustment
20:     until stop-criterion met
21:     return s_best
22: end function
```

Table 4.2: Summary of ALNS Algorithm Criteria

Criterion Name	Description	Reference(s)
Adaptive Mechanism		
Roulette Wheel	The probability of each operator (both destroy and repair) being selected is equal.	(Ropke and Pisinger, 2006a), (Laporte et al., 2010), (Hemmelmayr et al., 2012), (Braaten et al., 2017)
Acceptance Criterion		
Metropolis Criterion	To avoid local optima, the algorithm is allowed to accept slightly worse solutions than the current one with a certain probability.	(Kirkpatrick et al., 1983), (Ropke and Pisinger, 2006a)
Greedy Mechanism	Only accept solutions that are better than the current best solution.	(Muklason et al., 2018)
Record-to-Record	The algorithm is allowed to accept worse solution within a pre-determined threshold (R).	(Dueck, 1993)
Threshold acceptance	The algorithm is allowed to accept worse solution within a pre-determined threshold (R) and the value of R decreases at every iteration	(Sarasola and Doerner, 2020)
Termination Criterion		
Number of Iterations	Algorithm returns the best found solution after a fixed number of iterations.	(Ropke and Pisinger, 2006a), (Ropke and Pisinger, 2006b), (Demir et al., 2012)
Annealing Temperature	Algorithm returns the best solution found when the set minimum temperature is reached.	(Li et al., 2016), (Santos and de Carvalho, 2018) (Li et al., 2020)
Destroy Operators		
Random Removal	Randomly select an operator to remove.	(Ropke and Pisinger, 2006a), (Ropke and Pisinger, 2006b) (Ghilas et al., 2016)
Shaw Removal	Remove the node with the highest similarity index. The similarity index of a node is calculated by comparing it to a selected seed node based on a set of predefined criteria.	(Ropke and Pisinger, 2006a), (Ropke and Pisinger, 2006b), (Shaw, 1998), (Ghilas et al., 2016)
Worst Removal	Remove nodes that significantly contribute to the total cost of the solution.	(Ropke and Pisinger, 2006a), (Ropke and Pisinger, 2006b)
Route removal	This operator is commonly found in the vehicle routing domain randomly and removes a number of routes with all the associated nodes from a solution.	(Demir et al., 2012), (Ghilas et al., 2016)
Repair Operators		
Greedy insertion	Select and insert the node with the smallest insertion cost from all remaining nodes in the deleted nodes list into the solution.	(Ropke and Pisinger, 2006a), (Ropke and Pisinger, 2006b), (Pisinger and Ropke, 2007), (Pisinger and Ropke, 2019)
(k-)Regret insertion	Select and insert the node with the highest regret value from the remaining nodes in the deleted nodes list, where the regret value is the difference in cost between the best and the k-th best insertion positions.	(Ropke and Pisinger, 2006a), (Ropke and Pisinger, 2006b), (Pisinger and Ropke, 2007), (Pisinger and Ropke, 2019)
Random insertion	Randomly select a node from the list of deleted nodes and insert it into the position with the lowest incremental cost.	(Coelho et al., 2012b), (Coelho et al., 2012a), (Qu and Bard, 2012), (Qu and Bard, 2013)
Shaw insertion	Select the next node to be inserted into the solution from the list of removed nodes using the concept of similarity index as described in Shaw removal.	(Coelho et al., 2012b), (Coelho et al., 2012a)

4.2.1. Initial Solution

The initial solution is a necessary input for the ALNS algorithm. We use a simple Greedy algorithm to generate the initial solution. The algorithm begins at the vehicle's starting point and iteratively finds the nearest unvisited point, checking capacity and time window constraints to decide whether to visit that point. For delivery points, the algorithm prioritizes finding the nearest depot that can service the point, ensuring the route remains valid. For pickup points, it ensures that a depot can service the point. After visiting each point, the vehicle's load and time are updated until all points are serviced, and the vehicle returns to its destination, forming a complete route.

The initial solution for Algorithm 2 can be simply divided into two parts. The first part uses occasional drivers (OD) to handle demand nodes, and the second part uses regular drivers (RD) to handle the remaining demand nodes. The reason for this separation is that occasional drivers and regular drivers have different characteristics. Compared to regular drivers who start from the depot, occasional drivers have specific origins and destinations. Additionally, the cost for OD is lower, so OD is given a higher priority. In each part, for every nearest point found, we first determine whether the point is a pickup point, delivery point, or depot, and then execute different handling methods based on the type of point. Additionally, we must ensure that there are enough vehicles to meet all demands, otherwise, the generated initial solution will result in some demands not being serviced. Any surplus vehicles in the initial solution will be removed during the ALNS algorithm.

4.2.2. Request Removal

This section introduces two removal operators: random removal and worst removal. In addition, (Ropke and Pisinger, 2006a) also used Shaw removal proposed by (Shaw, 1997). However, random removal can be seen as a special form of Shaw removal and has been proven to be more efficient, thus we adopt random removal. Unlike random removal, worst removal selects those points that seem to be misplaced, i.e., service points that incur higher costs.

Random Removal

The basic idea of the random removal algorithm is to randomly select q requests and remove them from the solution. The number of requests q to be removed is obtained by multiplying a destruction factor κ between 0 and 1 by the number of points in the current solution. These randomly selected requests include pickup points, delivery points, and depots. As mentioned earlier, the solution may contain some unnecessary visits to depots. Therefore, we need to reselect the nodes chosen for removal. If the removed points are pickup or delivery points, it means that these points need to be served, so we retain these points. If the removed points are depots, we randomly select from the removed depots again. The selected depots are kept for the repair operation, and the unselected depots are permanently removed from the solution. This ensures unnecessary paths are eliminated, leading to a lower cost. Algorithm 3 shows the pseudocode for random removal.

Worst Removal

The basic idea of worst removal is that for a pickup or delivery request i served by some vehicle in a solution s, we define the cost of the request as cost$(i, s) = f(s) - f_{-i}(s)$, where $f_{-i}(s)$ is the cost of the solution without request i. All requests are sorted by cost in descending order, and the top q requests with the highest costs are removed and reinserted in the repair operator to obtain a better solution. On this basis, we have made some modifications. Since the solution path includes some depots, before using worst removal, we first select the depots in the current solution and use random removal to remove some depots. Suppose the number removed is n_{removed}, which ranges from 0 to $\min\{\text{number of depots in current solution}, q\}$. The remaining number of requests to be removed using worst removal is $q - n_{\text{removed}}$. This allows the algorithm to permanently delete some depots to achieve a better solution. Algorithm 4 shows the pseudocode for worst removal. To increase the randomness of the worst removal, a new parameter $p \geq 1$ is added, as can be seen in line 11 of the code. A lower value of p corresponds to greater randomness, so with a certain probability, we will choose the $y^p|L|$-th worst request.

4.2.3. Inserting Requests

According to the research by (Potvin and Rousseau, 1993), insertion heuristics for the vehicle routing problem can be broadly divided into two categories: sequential and parallel insertion heuristics. The

Algorithm 2 Initial Solution - Basic Greedy

```
Initialization
for each vehicle od do
    if no more unvisited points then
        Break
    end if
    while unvisited points exist do
        Find nearest point
        if nearest point is delivery point then
            Handle delivery point
        else if nearest point is depot point then
            Handle depot point
        elsenearest point is pickup point
            Handle pickup point
        end if
        Update unvisited points
    end while
end for
if all points are visited then
    Check solution completeness
    if complete then
        Add end point to solution
    else
        Find nearest unvisited points and update solution
    end if
else
    for each vehicle rd do
        while unvisited points exist do
            Find nearest point from start
            if nearest point is None then
                Break
            end if
            if nearest point is delivery point then
                Handle delivery point
            else if nearest point is depot point then
                Handle depot point
            elsenearest point is pickup point
                Handle pickup point
            end if
        end while
    end for
end if
```

Algorithm 3 Random Removal

function RandomRemoval($s \in \{solutions\}, q \in \mathbb{N}$)
    Copy the current solution s to a new state $destroyed$
    Randomly select q customers from the routes and move them to the unassigned list
    Filter out the depots from the unassigned list and store them separately
    if only one depot is unassigned **then**
        Select the depot
    else if multiple depots are unassigned **then**
        Randomly select a number of depots to keep unassigned
    else
        No depots are selected
    end if
    Add the selected depots back to the unassigned list
    return new state $destroyed$
end function

Algorithm 4 Worst Removal

function RandomRemoval($s \in \{solutions\}, q \in \mathbb{N}, p \in \mathbb{R}_+$)
    Copy the current solution s to a new state $destroyed$
    Select the depots in the current solution
    Randomly select a number of depots to remove and move them to the unassigned list
    Update the number of customers to remove, q
    while $q > 0$ **do**
        $q \leftarrow (q - 1)$
        Calculate the cost difference for removing each customer
        Sort customers (L) by cost difference (largest to smallest)
        Choose a random number y in the interval [0, 1)
        Select a customer $r = L[y^p|L|]$
        Remove the selected customer and move them to the unassigned list
    end while
    return the new state $destroyed$
end function

difference between them is that sequential heuristics construct one route at a time, whereas parallel heuristics can construct multiple routes simultaneously. The insertion heuristics involved in this research are all parallel heuristics. Basic Greedy heuristic and Regret-2 heuristic are applied, as they are the most common insertion heuristics.

Basic Greedy Heuristic

The principle of the Basic Greedy Heuristic is the same as the greedy strategy used when generating the initial solution. The program iterates over the requests removed in the previous section. For each request, we try to insert it into a feasible position in the route, recording the cost change $\Delta f_{i,k,idx}$ each time a feasible insertion position is found, $\Delta f_{i,k,idx}$ represents the cost difference when inserting the request i at position idx in route k compared to not inserting it. The request i is inserted at the position with the smallest cost until all requests are inserted into the routes. If no suitable insertion position is found in the existing routes, the request is inserted into an empty route of the vehicle corresponding to the depot of that request. Algorithm 5 shows the pseudocode for the basic greedy heuristic.

Algorithm 5 Basic Greedy Heuristic

```
function RreedyRepair(state : destroyed)
    Randomly shuffle the unassigned customers
    while destroyed.unassigned is not None do
        Pop a customer from the unassigned list
        route, index = best_insert(customer, destroyed)
        if route is not None then
            route.insert(index, customer)
        else
            depot = get_depot_for_customer(customer)
            if a corresponding depot is found then
                Find a route associated with the depot
                if a suitable route is found then
                    Insert the customer into the route
                else
                    Continue
                end if
            else
                Continue
            end if
        end if
    end while
    return the repaired state
end function
```

Regret Heuristic

The class of regret heuristics was proposed to overcome the main weakness of greedy heuristics, i.e., the myopic behavior. In greedy insertion, we always focus on one request at a time, which may result in the position idx where request i is inserted in the route being the optimal position for another request j to be inserted later. If request j is inserted at position idx in the route, it may result in a better solution. For example, the cost of inserting request i at idx 1 in route 1 is 20, and at idx 2 in route 2 is 15. Similarly, the cost of inserting request j at idx 1 in route 1 is 40, and at idx 2 in route 2 is 18. According to the basic greedy insertion, request i is inserted at idx 2 in route 2 because it has the lowest insertion cost, and then request j is considered. Since idx 2 in route 2 is already occupied by request i, request j can only be inserted at idx 1 in route 1, resulting in a total insertion cost of 55. This research considers the Regret-2 insertion heuristic, which improves the basic greedy heuristic by incorporating a look-ahead information when selecting requests to insert. First, we iterate over each request to be inserted and find all possible insertion points and insertion costs. For each request, we set a regret value $c_i^* = \Delta f_{i,k_1,idx_1} - \Delta f_{i,k_2,idx_2}$, where the regret value is the difference in the cost of inserting the request in its best route-index and its second best route-index. In each iteration, we select the request with the maximum regret value and insert it into the optimal route. The heuristic can be

extended to regret-k heuristic, but as k increases, the improvement efficiency of the algorithm slows down. Therefore, this research only uses the regret-2 heuristic. Algorithm 6 shows the pseudocode for the regret-2 heuristic.

Algorithm 6 Regret-2 Heuristic

function Regret-2Repair($state : destroyed$)
    Randomly shuffle the unassigned customers
    while $destroyed.unassigned$ is not None **do**
        Pop a $customer$ from the unassigned list
        $flag \leftarrow 0$ ▷ No feasible insertion found if $flag = 0$, otherwise $flag = 1$
        for $route$ in $destroyed.routes$ **do**
            if $route$ is None **then**
                Continue
            else
                if $can\_insert(destroyed, customer, route, index)$ **then**
                    Record $route$, $index$ and $cost$
                    $flag \leftarrow 1$
                end if
            end if
            if $flag == 0$ **then**
                Find depot for customer: $get\_depot\_for\_customer(customer)$
                Calculate cost inserting customer into a empty route
                Record $route$, $index$ and $cost$
            end if
        end for
        Sort insertion $costs$ for every customer in descending way
        Calculate $regret\_value$ for every customer
        Choose the $customer$ with largest $regret\_value$
        Insert the $customer$ into its best position
    end while
    return the repaired state
end function

4.2.4. Choosing a Removal and an Insertion Heuristic

Two removal operators and two repair operators are defined in section 4.2.2 and section 4.2.3. In one iteration, only one removal and one repair operator need to be used, but ALNS involves multiple removal and repair operators because different operators may be suitable for different sizes of problems. Even for problems of the same size, different operators may perform differently. For a specific problem, we do not know which operator is more suitable, so we let the algorithm choose. To select the appropriate removal and repair operators, we assign weights to each repair and removal operator and use the roulette wheel selection principle. Suppose we have k heuristics with weights $w_i, i \in \{1, 2, ..., k\}$, we select heuristic j with probability

$$\frac{w_j}{\sum_{i=1}^{k} w_i} \tag{4.44}$$

Note that the selection of insertion heuristic is independent of the removal heuristic (and vice versa). We can manually set these weights, but if many removal and insertion heuristics are used, this can be a very complex process. Instead, Section 4.2.5 proposes an adaptive weight adjustment algorithm.

4.2.5. Adaptive Weight Adjustment

This section details how to introduce weights and adaptively adjust them. The Adaptive Weight Adjustment part draws on the method of Adaptive Weight Adjustment in the ALNS framework of (Ropke and

Pisinger, 2006a). The basic idea is to track the score of each heuristic, which measures the recent performance of the heuristic. High scores correspond to successful heuristics. However, we made some modifications. First, we set the initial score of each removal and repair operator to 0, so the probability of each operator being selected is equal. We will call the ALNS algorithm multiple times, each time called a segment. Each segment has an inner loop for iterating new solutions. In each segment, each time a new solution is accepted, the used removal and repair operators are scored according to the following scoring rules in Table 4.3.

Table 4.3: Score Adjustment Parameters

Parameter	Description
σ_1	The last remove-insert operation resulted in a new global best solution.
σ_2	The last remove-insert operation resulted in a solution that has not been accepted before. The cost of the new solution is better than the cost of the current solution.

In each iteration, we apply two heuristics: a removal heuristic and an insertion heuristic. Both heuristics' scores are updated by the same amount because we cannot determine whether the removal or insertion is the successful reason. At the same time, we record the number of successful runs of the heuristic and calculate the weight of each heuristic before executing the next segment based on the scores and successful runs of each operator. The weight calculation formula is as follows,

$$w_{i,j+1} = w_{ij}(1-r) + r\frac{\pi_i}{\theta_i} \tag{4.45}$$

where $w_{i,j+1}$ represents the weight of operator i in segment $j+1$, $w_{i,j}$ represents the weight of operator i in segment j, π_i is the score obtained by operator i in the last segment, and θ_i is the number of successes of the operator. The reaction factor r controls the speed of the weight adjustment algorithm in response to changes in operator effectiveness. If r is zero, then we do not use the score at all and stick to the initial weight. If r is set to 1, then the weight is determined by the score obtained in the last segment.

4.2.6. Acceptance and Stopping Criteria

The simplest Acceptance and Stopping Criteria is to accept only solutions better than the current solution, see (Shaw, 1997). This is likely to lead to the solution getting stuck in a local optimum. Therefore, it is sometimes wiser to accept solutions worse than the current one (Ropke and Pisinger, 2006a). Therefore, we choose to use simulated annealing acceptance criteria, which means that we accept a solution s' worse than the current solution s with a certain probability, given by $e^{(f(s)-f(s'))/T}$, where $T > 0$ represents the temperature. The temperature starts from T_{start} and is updated at the end of each iteration by $T = T \cdot c$, where $0 < c < 1$ is the cooling rate. Similarly, if the new solution s' is better than the current solution, we accept this solution with 100% probability. The stopping criterion of the algorithm is to meet the set maximum number of iterations. Of course, for different sizes of instances, the initial temperature and number of iterations will affect the performance of the results, so parameter tuning is necessary. See Section 5.3 for details.

5

Computational Experiments

5.1. Instance Generation

The instances used in this chapter are virtual instances generated by code for the purpose of parameter tuning and testing algorithm performance. Each instance has fixed characteristics and is created and stored in a text file. Each instance file starts with 11 lines whose meanings are shown in Figure 5.1. Following this, there is a matrix containing all the points, where each row of the matrix represents a point. According to Figure 5.1, there are two depots, corresponding to the first two rows of the matrix. The dummy depots corresponding to these two depots are represented by the third and fourth rows of the matrix. Next, there are 6 customers, each with delivery demands. Following these are 4 pickup and delivery demands for the same intermediate center. Finally, there are the start and end points for two ODs (Origin-Destination pairs). Rows 14-15 represent the start and end points for the OD 1, while rows 16-17 represent the start and end points for the OD 2. For the columns, the first two columns represent the two-dimensional coordinates of each node, the third and fourth columns represent the time window for each node, the fifth column represents the pickup or delivery quantity (positive for delivery, negative for pickup), and the last column represents the depot corresponding to each pickup or delivery demand. The naming rule of Instance is composed of three numbers, such as **Instance 2_1_10**, which means there are 2 depots, 1 intermediate center and 10 requests.

5.2. Example Instance

Before parameter tuning, we want to verify whether the model and algorithm outputs are consistent through a simple example, meaning whether they can achieve the same optimal solution. In fact, we used several requests ranging from 10 to 20 to debug the model and code. Upon completing the debugging, we have verified the consistency between the mathematical model and the ALNS algorithm. Therefore, this demonstration of the optimal solution generation, along with its description and visualization, helps the readers better understand the model. The test example mentioned here is Instance 2_1_10. All experiments were performed on a PC running Microsoft Windows 10 with the following specifications:

- Processor: Intel Core i7-8750H 2.20GHz
- RAM: 8 GB
- GPU: NVIDIA GeForce GTX 1050 Ti
- Software: VS Code 1.78.0, Python 3.8.18

Using the CPLEX package in Python to run the mathematical model, we solve it with the branch-and-bound method, setting a time limit of 3600 seconds. The optimal cost of 303.5 was obtained in 40 minutes, which is the same result as that obtained using the ALNS algorithm, but the ALNS algorithm only took 8.9 seconds. In practice, the ALNS algorithm should take a shorter time. Typically, when the number of requests is 10, we set the iteration to 200, and the runtime is approximately 300ms. To prevent the algorithm from getting stuck in a local optimum during a single loop, we add an outer loop,

Table 5.1: Test Instance 2_1_10

Parameter	Value
Number of Nodes	18
Number of Depots	2
Number of RD in depot 0	1
Capacity of RD in depot 0	10
Number of RD in depot 1	1
Capacity of RD in depot 1	10
Number of OD	2
Capacity of OD	8
Number of requests in intermediate centers	4
Max number visits depot	2
Coeff Cost OD	0.5

Coor_1	Coor_2	TW_1	TW_2	PD	V
50.0	40.0	0	1236	0	0
125.0	110.0	0	1236	0	0
50.0	40.0	0	1236	0	0
125.0	110.0	0	1236	0	0
45.0	20.0	20	870	2	0
18.0	50.0	30	880	8	0
35.0	100.0	30	880	2	0
125.0	20.0	30	880	2	1
150.0	100.0	30	880	2	1
110.0	140.0	30	880	2	1
85.0	65.0	0	1236	2	0
85.0	65.0	0	1236	-2	0
85.0	65.0	0	1236	-2	1
85.0	65.0	0	1236	2	0
0.0	0.0	100	300	0	0
0.0	0.0	0	1236	0	0
0.0	0.0	100	300	0	0
0.0	0.0	0	1236	0	0

running the ALNS algorithm 30 times and taking the best performance among the 30 runs, as shown in Figure 5.1. In the figure, the best iteration found 50 feasible solutions within 200 iterations. The algorithm found the optimal solution within 10 iterations and then attempted to accept worse solutions to explore the solution space further. Since the optimal solution had already been found, no lower cost were discovered afterward. The red line in the figure represents the best solution found in the current iteration, and the blue line represents the current feasible solution. If a current feasible solution is better than the current best solution, the current best solution is updated. The optimal solution is visualized in Figure 5.2, and the optimal routes are:

- **RD 1**: [0]
- **RD 2**: [1]
- **OD 1**: [14, 15]
- **OD 2**: [16, 0, 6, 10, 11, 12, 13, 3, 9, 1, 8, 7, 4, 2, 5, 17]

Since the OD vehicles have sufficient capacity, the RDs were not used, RD 1 and RD 2 stayed at their respective depots. Here, we assume that the OD vehicles are provided by a third-party logistics company, so two OD vehicles with the same start and end points were set up to ensure that all demands can be fully serviced. OD 1 departs from the start point directly to the end point, indicating that OD 1 was not used, while OD 2 handled all the demands.

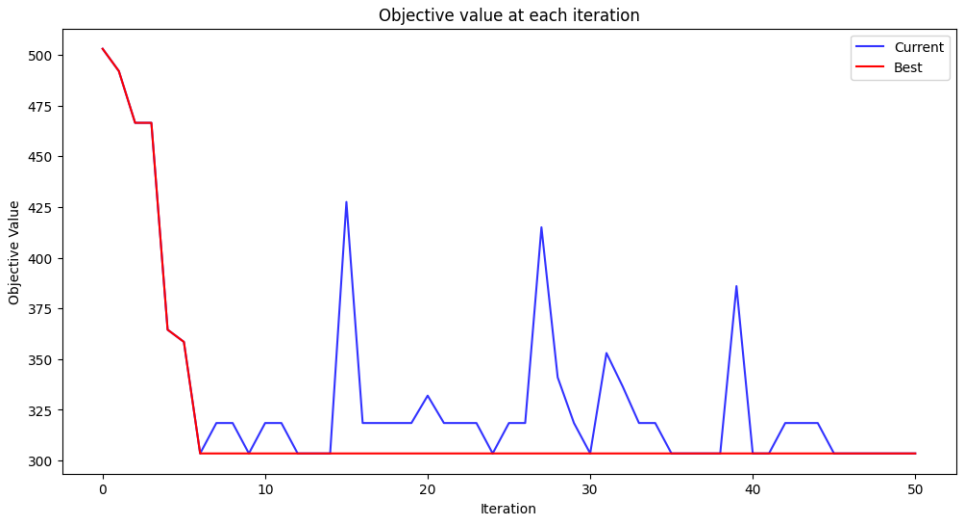

Figure 5.1: ALNS Result of test instance 2_1_10

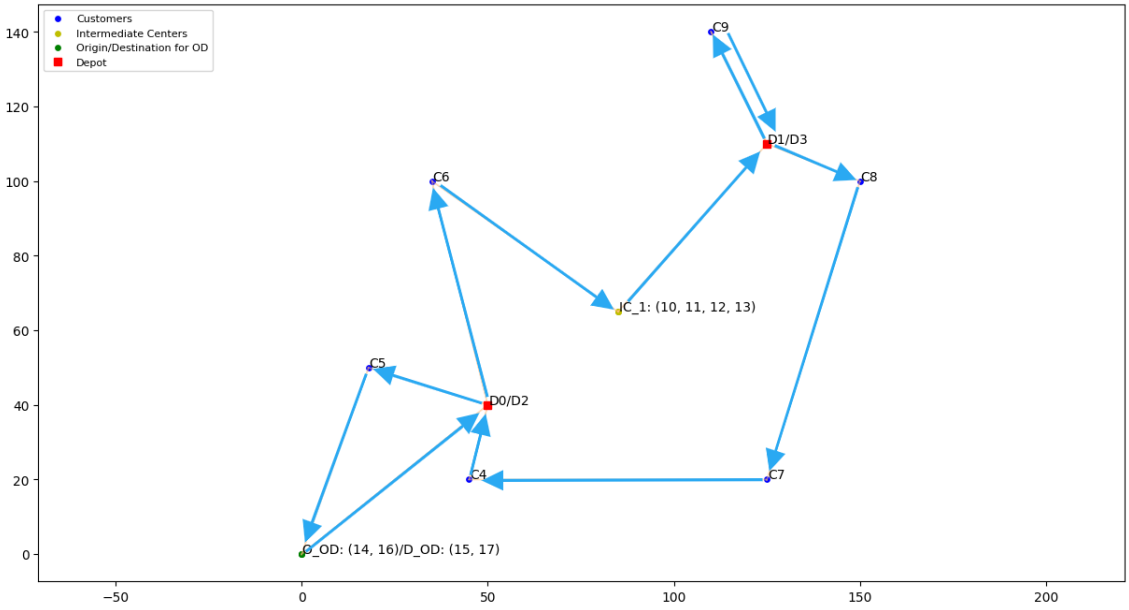

Figure 5.2: Result map of test instance 2_1_10

5.3. Parameter Tuning

Before parameter tuning, a representative set of tuning instances is required. As our research problem is a new pickup and delivery problem, no existing instances found that could be directly referenced and used. Therefore, we will refer to the method of generating instances in (Solomon, 1987) with certain modifications to suit the specifics of our study. Initially, the scale of the tuning instances needs to be sufficiently diverse. Our tuning set consists of 6 instances, where the number of requests starts from 10, with 2 instances added for every increase of 20 requests. The largest instances contain 50 requests. Request coordinates are uniformly distributed around the depots. Alongside increasing requests, we will concurrently adjust the vehicle capacity, number of intermediate center, number of OD, and number of depot. Specific parameter settings for generating tuning instances are shown in Table 5.2.

The primary objective of parameter tuning is to subsequently conduct large-scale testing and comparison of the performance between ALNS algorithm and exact algorithm, and to reveal which heuristic part or which parameter contributes the most to solution quality. Firstly, it is necessary to determine

Table 5.2: Tuning Instances Information

Requests	Nodes	Depots	ICs	Requests per Depot	Requests per IC	RDs each Depot	Capacity of RD	ODs	Capacity of OD
10	18	2	1	3	4	1	20	2	16
30	38	2	2	10	5	2	30	2	24
50	60	3	2	10	10	1	50	2	40

which parameters need adjustment. Before executing removal operators, we need to determine how many requests to remove, i.e., the destruction factor κ. Random removal involves no parameters, while worst removal involves a random parameter p. No parameters are involved in repair operators. For acceptance criteria, two parameters are involved: initial temperature T_{start} and cooling rate c. For the Adaptive weight adjustment criteria, three parameters are involved: reaction factor r, score adjustment parameters σ_1, and σ_2. For these seven parameters, adjustments are made sequentially in a certain order, with each parameter set to 4 reasonable values as listed in Table 5.3. Each tuning instance is run 5 times for each parameter value. Thus, for 6 tuning instances, we obtain 30 costs per parameter, and the parameter corresponding to the lowest average cost is selected as the optimal parameter. Table 5.4 shows the tuning results for the destruction factor κ, recording the runtime and cost for each instance, and calculating the average runtime and average cost accordingly. According to Table 5.4, when $\kappa = 0.1$, the average cost is minimized at 1073.33, whereas the average costs for the other three parameter values are 1108.33, 1176.23, and 1192.2, respectively. The tuning results for the other six parameters are presented in Appendix A. Based on the tuning results, it is evident that only the destruction factor κ shows sensitivity to different parameter values, while the remaining six parameters exhibit lower sensitivity. Therefore, these six parameters are deemed suitable for solving instances within a certain scale range. The optimal parameter value for the destruction factor κ is found to be 0.1. Given that 0.1 represents a relatively small destruction ratio (for instance, in cases with 10 requests, only one request is destroyed per iteration), we choose not to conduct further in-depth analysis on the Destruction factor. The best values for each parameter are listed in Table 5.3.

Table 5.3: Parameter Settings

Parameter name	Symbol	Parameter range	Selected values	Best Value
Destruction factor	κ	(0, 1)	[0.1, 0.15, 0.2, 0.5]	0.1
Random Parameter	p	≥ 1	[1, 5, 10, 20]	1
Initial Temperature	T_{start}	$\mathbb{R}^+$	[100, 200, 300, 400]	300
Cooling rate	c	(0, 1)	[0.8, 0.85, 0.9, 0.95]	0.95
Reaction factor	r	[0, 1)	[0.2, 0.4, 0.6, 0.8]	0.2
Score adjustment factor	σ_1	$\mathbb{R}^+$	[10, 20, 30, 40]	30
Score adjustment factor	σ_2	$\mathbb{R}^+$	[5, 10, 15, 20]	5

5.4. Computational Experiments

In this section, we conduct a comparative experiment between the ALNS algorithm with tuned parameters and an exact method to evaluate the performance of the ALNS algorithm. Using the same logic as parameter tuning, we generate 30 instances with request quantities of 10, 30, and 50 (10 instances each). The results of the computational experiments are presented in Table 5.5. The bold fonts in the table indicate the known optimal solutions. For instances with 10 requests, the exact method using CPLEX's built-in branch and bound algorithm found 4 optimal solutions, with other solutions showing relatively good performance but longer average runtimes, mostly exceeding 20 minutes. Although the ALNS algorithm confirmed only one instance as optimal, it demonstrated overall good performance across the 10 instances. It consistently found solutions within 10 seconds that were better than those found by the exact algorithm after 1 hour of computation. As the instance sizes increased to 30 and 50 requests, the advantages of the ALNS algorithm became apparent. We configured it to run 10000 and 15000 iterations respectively. The ALNS algorithm significantly improved upon initial solutions in a short time frame, whereas the exact algorithm achieved only a best bound after running for the set 1

Table 5.4: Instances Parameter Tuning Results - Destruction Factor κ

Fixed Parameter setting: p **= 1,** T_{start} **= 200,** c **= 0.9,** r **= 0.2,** σ_1 **= 10,** σ_2 **= 5**														
Tuning Instances	**Run Time**					**Avg. Time**	**Total Avg. Time**	**Cost**					**Avg. Cost**	**Total Avg. Cost**
	1	**2**	**3**	**4**	**5**			**1**	**2**	**3**	**4**	**5**		
Parameter setting: κ **= 0.1**														
2-1-10-1	0.8	0.9	0.7	0.9	0.8	0.82	**22.02**	357	357	294	357	328.5	338.7	**1073.33**
2-1-10-2	0.7	0.7	0.7	0.8	0.7	0.72		389	389	389	389	389	389	
2-2-30-1	13.6	13.6	10.8	10	10.5	11.7		1250	1210	1337	1492	1427.5	1343.3	
2-2-30-2	10.1	10.7	9.7	8.7	8.9	9.62		1042.5	1017.5	1095.5	1067.5	1052.5	1055.1	
3-2-50-1	50.8	46.5	46.2	46	46	47.1		1979	1979	1892	1979	1979	1961.6	
3-2-50-2	67.9	66.8	60	57.5	58.5	62.14		1433.5	1302	1298	1430	1298	1352.3	
Parameter setting: κ **= 0.15**														
2-1-10-1	1	0.9	0.8	0.9	0.8	0.88	31.34	294	294	357	357	357	331.8	1108.33
2-1-10-2	0.7	0.8	0.8	0.8	0.9	0.8		430	316	389	389	389	382.6	
2-2-30-1	15	16.9	15.9	15.9	16.3	16		1875	1539	1449.5	1511.5	1509	1576.8	
2-2-30-2	13.5	12.5	11.3	11.9	12.3	12.3		1084.5	1030.5	1069.5	1065.5	1039	1057.8	
3-2-50-1	75.4	73.3	77.3	77.3	75	75.66		1979	1979	1979	1979	1979	1979	
3-2-50-2	72.5	83.3	87.8	84.1	84.2	82.38		1362	1307	1304	1330	1307	1322	
Parameter setting: κ **= 0.2**														
2-1-10-1	1	1	1.3	1.1	1.2	1.12	41.04	294	294	294	294	391.5	313.5	1176.23
2-1-10-2	1	0.9	0.9	0.8	0.9	0.9		389	389	389	346	389	380.4	
2-2-30-1	17.1	16.8	14.9	15.5	12	15.26		1875	1875	1875	1875	1660	1832	
2-2-30-2	14	14.5	14.4	14.9	14.3	14.42		1126.5	1126.5	1126.5	1126.5	1126.5	1126.5	
3-2-50-1	106.2	103.8	105.6	104.5	106.3	105.28		1979	1979	1979	1979	1979	1979	
3-2-50-2	101.2	99.4	107.6	107.5	130.6	109.26		1439.5	1493	1477	1375	1345.5	1426	
Parameter setting: κ **= 0.25**														
2-1-10-1	1.3	1.3	1.4	1.2	1.2	1.28	57.74	294	294	294	328.5	294	300.9	1192.2
2-1-10-2	1.2	1.1	1	1.1	1	1.08		389	389	389	389	389	389	
2-2-30-1	24.1	22.7	23.1	29.4	24.2	24.7		1875	1875	1875	1735.5	1875	1847.1	
2-2-30-2	21.3	23.9	21.8	21.7	23.4	22.42		1126.5	1126.5	1126.5	1126.5	1126.5	1126.5	
3-2-50-1	143.3	146.5	142.3	138.8	139	141.98		1979	1979	1979	1979	1979	1979	
3-2-50-2	152.3	150.8	148.4	144.3	179.2	155		1519.5	1519.5	1519.5	1519.5	1476	1510.8	

hour and 1.5 hours for the respective instance sizes.

Due to hardware limitations, the computational experiments and subsequent sensitivity analysis experiments were conducted on a workstation equipped with an Intel Core i9-14900K 24-core CPU, 64 GB RAM, operating at a frequency of 4800 MHz, and running on Windows 10. The ALNS algorithm was implemented using VS Code in Python version 3.8.18.

5.5. Sensitivity Analysis

In the business domain, cost is one of the primary concerns for managers. In the process of parcel delivery, reducing costs and increasing efficiency is a common goal for almost all companies. (Hou and Wang, 2021)'s research shows that the average delivery cost can be reduced by 7.30% compared to delivery by dedicated vehicles when incorporating a compensation scheme based on crowdshippers' acceptance behavior. In this study, there are four factors that influence parcel delivery costs: the number of occasional drivers, the capacity of occasional vehicles, the transportation cost of occasional vehicles, and the location of occasional vehicles. Since the location of occasional vehicles is difficult to evaluate, it will not be discussed in this section. Generally, the capacity of occasional vehicles is positively correlated with their transportation cost per unit distance or per unit time. This chapter will also analyze the correlation between vehicle capacity and vehicle transportation cost. Finally, the sensitivity analysis sequence is to first select different numbers of occasional drivers, determine the number of occasional drivers, then set vehicle capacity and cost factor within a reasonable range, and analyze the cost optimization brought by different values.

5.5.1. Instances for Sensitivity Analysis

Sensitivity analysis requires a new set of instances. We decided to modify the instances used for parameter tuning. First, we only consider instances with 30 and 50 requests, as instances with 10 requests are considered too small to be of analytical value. We retain all coordinates, time windows, and demands of the 20 instances with 30 and 50 requests from the parameter tuning section. We modify the values of regular drivers, occasional drivers, occasional vehicle capacity, and cost factors for these instances. The parameter values for the occasional driver sensitivity analysis are shown in Table 5.6. For the sensitivity analysis of occasional vehicle capacity and cost factor, except for the different values of vehicle capacity and cost factor, all other parameters remain the same. Each instance is run once, with each run having 10000 iterations.

Table 5.5: Instance Performance Comparison - ALNS vs Branch and Bound

Instance	ALNS				CPLEX			
	Initial Cost	Best Cost	Time (s)	Iteration	Best Cost	Best Bound	MIP Gap	Solve Time (s)
2.1.10.1	663	**429**	5	5000	436	199.64	54.21%	3600
2.1.10.2	384	**252**	17	5000	252	252	0%	476
2.1.10.3	583.5	377.5	7.6	5000	305.5	305.5	0%	2114
2.1.10.4	676	315.5	7.8	5000	300	231.79	22.74%	3644
2.1.10.5	427.5	**294**	5.7	5000	294	254.53	13.42%	3036
2.1.10.6	546	231.5	5.4	5000	**204.5**	204.5	0%	2085
2.1.10.7	575.5	348.5	5.6	5000	**249**	206.1	17.23%	3617
2.1.10.8	615.5	273	5.7	5000	**236**	236	0%	1215
2.1.10.9	701	336.5	6.9	5000	**320.5**	244	23.87%	3725
2.1.10.10	395	278.5	5.8	5000	**267.5**	224.82	15.95%	3607
2.2.30.1	1778	**801**	63.6	10000		275.92		3600
2.2.30.2	1035	**853.5**	48.3	10000		261.08		3600
2.2.30.3	1339.5	**1017.5**	49.5	10000		293.19		3600
2.2.30.4	1210.5	**837**	61	10000		278.6		3600
2.2.30.5	1076.5	**881**	66.5	10000		332.31		3600
2.2.30.6	989	**879.5**	47.5	10000		270.42		3600
2.2.30.7	1137.5	**950**	51.6	10000		327.38		3600
2.2.30.8	1334.5	**962**	47.9	10000		311.44		3600
2.2.30.9	1047.5	**912**	42.1	10000		278.13		3600
2.2.30.10	1120	**950**	65.8	10000		234.45		3600
3.2.50.1	1585.5	**1048.5**	304.4	15000		374.98		5400
3.2.50.2	1519.5	**1290.5**	273.8	15000		379.96		5400
3.2.50.3	1984	**1357.5**	242.1	15000		409.74		5400
3.2.50.4	2001.5	**1440**	244.1	15000		394.61		5400
3.2.50.5	2019.5	**1410**	315.1	15000		372.4		5400
3.2.50.6	1967.5	**1045**	318	15000		346.03		5400
3.2.50.7	1652	**1104**	264.3	15000		374.89		5400
3.2.50.8	1633	**1170**	242	15000		363.4		5400
3.2.50.9	1531.5	**1204.5**	319	15000		327.26		5400
3.2.50.10	1379.5	**1088.5**	239.4	15000		320.11		5400

Table 5.6: Sensitivity Analysis Instance Information

Requests	Nodes	Depot	Depot 1 RD	Depot 2 RD	Depot 3 RD	Capacity RD	OD	Capacity OD	Cost Factor
OD = 0									
30	34	2	3	3		30	0	0	
50	60	2	2	2	2	50	0	0	
OD = 1									
30	36	2	2	3		30	1	24	0.8
50	58	3	2	2	2	50	1	40	0.8
OD = 2									
30	38	2	2	2		30	2	24	0.8
50	60	3	2	2	2	50	2	40	0.8
OD = 3									
30	40	2	1	2		30	3	24	0.8
50	62	3	2	2	2	50	3	40	0.8

5.5.2. Sensitivity Analysis - Number of OD

The results of the sensitivity analysis for occasional drivers are shown in Table 5.7. We conducted four sets of comparative analyses based on the number of occasional drivers, which is $OD = [0, 1, 2, 3]$, corresponding to the four cost columns in the table. We used the no OD group as the baseline and calculated the optimization ratio of the other three groups containing ODs compared to the baseline cost,

shown in the three columns in the table. A negative ratio indicates a worse result, as the ALNS algorithm may get stuck in local optima when solving instances. The optimal solutions and their corresponding optimization ratios are highlighted in bold. Among the 20 instances with optimal costs, the no OD group accounts for 4 instances, the group with 1 OD accounts for 7 instances, the group with 2 ODs accounts for 6 instances, and the group with 3 ODs only accounts for 3 instances. When OD = 1, even though some instance results are not optimal, the cost differences are relatively small, and the results are more stable. Therefore, we consider 1 OD to be a good choice for instances with 30 and 50 requests. In subsequent sensitivity analyses of OD capacity and cost, the parameter settings for 1 OD will be used.

In actual operations, company decision-makers can choose either the number of ODs with a cost advantage or the lowest-cost path each time. In this experiment, when a fixed number of ODs is selected, having 1 OD increases the total cost of all instances by 19.09% compared to having no OD. With 2 ODs, the total cost increases by 17.19%, and with 3 ODs, the total cost increases by 18.48%. It is worth noting that when the cost is worse with a fixed number of OD vehicles, we will continue to choose the cost without ODs. If each instance selects its corresponding optimal cost, the total cost of the 20 instances increases by 22.07%.

Table 5.7: Sensitivity Analysis - Number of OD

Instances	OD = 0	OD = 1	Gap - OD=1	OD = 2	Gap - OD=2	OD = 3	Gap - OD=3
2.2.30.1	1091	1029	5.7%	**955.4**	**12.4%**	1041.6	4.5%
2.2.30.2	1303.5	1229	5.7%	**1018**	**21.9%**	1073.6	17.6%
2.2.30.3	1433	1281	10.6%	**1154.2**	**19.5%**	1170	18.4%
2.2.30.4	**839**	939	-11.9%	1405.8	-67.6%	1085	-29.3%
2.2.30.5	1105	1266.2	-14.6%	1098.2	0.6%	**1067**	**3.4%**
2.2.30.6	967	**884.4**	**8.5%**	1134	-17.3%	1057	-9.3%
2.2.30.7	1377	**1141**	**17.1%**	1181.2	14.2%	**1122.2**	**18.5%**
2.2.30.8	1264	**1214**	**4.0%**	1264.2	0.0%	1177.8	6.8%
2.2.30.9	**778**	854.6	-9.8%	1182	-51.9%	1123	-44.3%
2.2.30.10	**979**	994	-1.5%	1217	-24.3%	1128.4	-15.3%
3.2.50.1	2021.5	1942.4	3.9%	**1836**	**9.2%**	1967	2.7%
3.2.50.2	2581	**1718.8**	**33.4%**	1811.6	29.8%	1814	29.7%
3.2.50.3	3516	2568.4	27.0%	**2211**	**37.1%**	**2053**	**41.6%**
3.2.50.4	3260	2151.8	34.0%	**2044.4**	**37.3%**	2158.6	33.8%
3.2.50.5	1828	**1716**	**6.1%**	2206.6	-20.7%	2041	-11.7%
3.2.50.6	2577	**1580**	**38.7%**	1929.2	25.1%	1973.2	23.4%
3.2.50.7	2541	**1875**	**26.2%**	2180.8	14.2%	2111	16.9%
3.2.50.8	2365.5	**1518.4**	**35.8%**	2240.8	5.3%	1556.6	34.2%
3.2.50.9	2280.5	1755.8	23.0%	**1668**	**26.9%**	1846.8	19.0%
3.2.50.10	**1516**	1567.4	-3.4%	1699.6	-12.1%	2058.8	-35.8%

5.5.3. Sensitivity Analysis - OD Capacity and Cost

For the capacity of ODs, we set up seven comparative experiments, reducing the capacity of ODs relative to the capacity of RDs by a certain percentage. The seven comparative experimental percentage parameters are set to $Capacity = [40\%, 50\%, 60\%, 70\%, 80\%, 90\%, 100\%]$. For the unit distance cost of RDs, we set up five comparative experiments, reducing the unit distance cost relative to that of RDs by a certain percentage. Different vehicle capacity parameter settings correspond to different unit distance cost ratios for RDs, as detailed below:

- **OD Capacity Parameter = 40%**: [20%, 30%, 40%, 50%, 60%]
- **OD Capacity Parameter = 50%**: [30%, 40%, 50%, 60%, 70%]
- **OD Capacity Parameter = 60%**: [40%, 50%, 60%, 70%, 80%]
- **OD Capacity Parameter = 70%**: [50%, 60%, 70%, 80%, 90%]
- **OD Capacity Parameter = 80%**: [50%, 60%, 70%, 80%, 90%]

- **OD Capacity Parameter = 90%**: [50%, 60%, 70%, 80%, 90%]
- **OD Capacity Parameter = 100%**: [50%, 60%, 70%, 80%, 90%]

A total of 100 cost values were obtained for each OD capacity parameter setting, resulting in 700 cost values across 7 parameter configurations. All results are stored in 7 tables, where Table 5.8 presents the results for OD Capacity = 40%. The results for the remaining 6 OD Capacity values can be found in Appendix B. In addition to the costs obtained for different parameter values, the lowest cost parameter results serves as the base cost for calculating the percentage increase in OD costs per 10% increase in capacity. Positive percentages indicate increased costs, while negative percentages indicate decreased costs. The percentage increase in OD costs for all 7 OD capacity values is shown in Table 5.9. For each row, if capacity correlates positively with vehicle costs (i.e., when OD vehicle capacity is 40% of RD vehicle capacity, OD per unit distance costs should also be 40% of RD costs), we use this as a benchmark to explore scenarios where OD costs are below or above 40%. The same applies to the other 6 experimental groups. Taking OD capacities of 60% and 70% as examples, the calculated cost increase percentages are [50%, 60%, 70%, 80%, 90%, 100%] = [11.76%, 4.89%, 9.08%, 8.92%, 8.47%, 7.81%], with an average increase of 8.47%.

Table 5.8: Sensitivity Analysis - Capacity and Cost - 40% Capacity

40% Capacity									
Instances	**20% Cost**	**30% Cost**	**Gap Cost=30%**	**40% Cost**	**Gap Cost=40%**	**50% Cost**	**Gap Cost=50%**	**60% Cost**	**Gap Cost=60%**
2.2.30.1	915.6	966.3	5.54%	1191.6	30.14%	1119	22.21%	1288.6	40.74%
2.2.30.2	612	832.6	36.05%	823.4	34.54%	920	50.33%	949.2	55.10%
2.2.30.3	1001.2	1058.9	5.76%	1123.4	12.21%	1138	13.66%	889.8	-11.13%
2.2.30.4	786.8	801	1.80%	927	17.82%	980.5	24.62%	1047	33.07%
2.2.30.5	1113.4	1201.8	7.94%	1183.4	6.29%	1205.5	8.27%	1179	5.89%
2.2.30.6	811.4	898.3	10.71%	1023.6	26.15%	1120	38.03%	1011.8	24.70%
2.2.30.7	923.8	1019.8	10.39%	1084.4	17.38%	1141	23.51%	1223	32.39%
2.2.30.8	851.8	880	3.31%	950.2	11.55%	1150.5	35.07%	1283.6	50.69%
2.2.30.9	699.2	770	10.13%	747.8	6.95%	794.5	13.63%	840.2	20.17%
2.2.30.10	823.8	961.7	16.74%	964.6	17.09%	1099	33.41%	1115.6	35.42%
3.2.50.1	1525.6	1476.7	-3.21%	1236.4	-18.96%	1787.5	17.17%	1658.2	8.69%
3.2.50.2	1411.2	1379.8	-2.23%	1527.2	8.22%	1865	32.16%	1759.2	24.66%
3.2.50.3	1741.8	1875.2	7.66%	2008.6	15.32%	1862	6.90%	1864.2	7.03%
3.2.50.4	1860	1761.3	-5.31%	1089.8	-41.41%	2229	19.84%	2352	26.45%
3.2.50.5	1831	1975	7.86%	2119	15.73%	2263	23.59%	2407	31.46%
3.2.50.6	1530	1155.2	-24.50%	1538.6	0.56%	1634.5	6.83%	1492	-2.48%
3.2.50.7	1434.8	1563.2	8.95%	1691.6	17.90%	1820	26.85%	1948.4	35.80%
3.2.50.8	1660.4	1787.1	7.63%	1913.8	15.26%	2040.5	22.89%	1662.2	0.11%
3.2.50.9	1409.6	1428.3	1.33%	1571.8	11.51%	1434	1.73%	1423.8	1.01%
3.2.50.10	1221.8	1357.7	11.12%	1493.6	22.25%	1629.5	33.37%	1765.4	44.49%

Table 5.9: Cost Sensitivity Analysis

Capacity	**20% Cost**	**30% Cost**	**40% Cost**	**50% Cost**	**60% Cost**	**70% Cost**	**80% Cost**	**90% Cost**
40%	Base Cost	5.88%	11.33%	22.70%	23.21%			
50%		Base Cost	9.52%	18.48%	21.13%	32.89%		
60%			Base Cost	6.22%	15.76%	20.66%	26.63%	
70%				Base Cost	11.95%	21.03%	28.29%	33.71%
80%				Base Cost	6.38%	15.30%	16.81%	30.63%
90%				Base Cost	2.68%	11.01%	20.74%	28.50%
100%				Base Cost	9.96%	17.78%	23.30%	30.22%

Next, we explore the performance of OD capacity or OD cost as single variables compared to path costs without OD inclusion. As mentioned earlier, both the number of ODs and the sensitivity analysis in this section use the same instances, adjusting only relevant parameters to facilitate direct comparison of these results. In Section 5.7, we used OD capacity and OD cost ratios of 80%, as shown in Table 5.6 and Table B.4 in Appendix B. Referring to the path costs when OD = 0 in Table 5.7, we calculated the cost decrease brought by different OD cost ratios when OD capacity = 80% across 20 instances as [90%, 80%, 70%, 60%, 50%] = [5.18%, 15.21%, 16.47%, 22.54%, 27.01%]. These results indicate that path total costs save 5.18% when OD costs are 90% of RD costs, and save 15.21% when OD costs are 80% of RD costs. We then explore the effect of varying OD capacity on path total costs with fixed OD costs at 80% of RD costs. Tables B.2, B.3, B.4, B.5, and B.6 in Appendix B correspond to OD capacities of [60%, 70%, 80%, 90%, 100%], with path total cost decreases relative to OD = 0 as

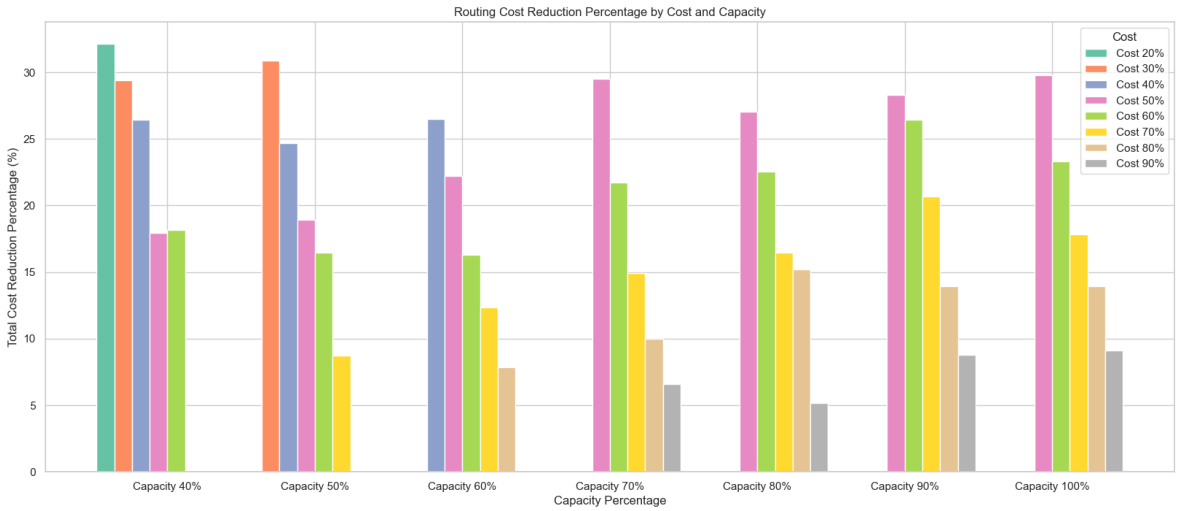

Figure 5.3: Sensitivity Analysis Results

[7.86%, 9.97%, 15.21%, 13.96%, 13.94%]. According to the results, using OD with 60% RD capacity saves 7.86% costs, 70% RD capacity saves 9.97% costs, and 80% RD capacity saves 15.21% costs, with further capacity increases not resulting in further cost reductions. Figure 5.3 illustrates the total cost savings ratio across all combinations of vehicle capacities and costs for 20 test instances.

6

Case Study

6.1. Background

Ochama is an omnichannel retailer, part of the Chinese e-commerce giant JD.com. Headquartered in the Netherlands, Ochama operates primarily in the Netherlands, Belgium, France, and Germany. The platform offers over 10,000 high-quality products across various categories. Utilizing an advanced automated warehousing system, Ochama provides dual fulfillment services of pickup and delivery. Consumers can place orders online through the Ochama platform, with the option to pick up their orders from pickup points or have them delivered directly to their homes. Ochama has successfully implemented advanced warehouse network and logistics system with over 700 ochama pickup points in major cities and towns in the Netherlands, Belgium, and Germany.

Ochama currently operates two depots: a self-operated warehouse and a crossdocking warehouse located in Rotterdam and Venray, Netherlands, respectively (see Map 6.1). The self-operated warehouse in Berkel en Rodenrijs, Rotterdam, handles all storage and shipping of goods, as well as distribution for orders in western and northern Netherlands. Orders from eastern Netherlands, western Germany, and northern Belgium are consolidated and transported to Ochama's crossdocking facility in Venray for distribution after crossdocking. Ochama has over 700 pickup points, primarily within the Netherlands, with the remainder in western Germany, Belgium, and parts of France. These pickup points do not operate every day but rather on select days of the week, servicing an average of 150 to 200 pickup points daily. In China, June 18th annually marks a major e-commerce promotion day, similar to Black Friday. Ochama continues this tradition, and statistics show that during the recent 2024 June 18th promotion, Ochama shipped 13,000 parcels. Parcels are delivered using vans sized 5.45m × 2.263m × 2.29m, with an average parcel size of 40cm × 25cm × 30cm. Each van can accommodate an average of 200 parcels.

6.2. Instances for Case Study

In this study, we scale down the problem based on real-world scenarios. Each instance retains the positions of two depots and randomly sets 5 intermediate centers (ICs). Each IC contains 20 requests, and each depot is randomly assigned 25 demand points, totaling 150 demands per instance. The positions of ICs and demand points are within a range of 80 km from the two depots.

According to research, vehicles are categorized into three types: large, medium, and small. The RDs use large vehicles, whereas ODs, typically freelance drivers, have a higher proportion of medium and small vehicles compared to large vehicles, see Figure 6.2. Therefore, ODs consider medium and small vehicles. The capacities of these three types of vehicles are set in certain proportions, with medium and small vehicles having cargo capacities of 60% and 45% of a large vehicle's capacity, respectively. This design is based on the prevalence of large vehicles in Dutch logistics companies, such as Ford Transit and Renault, which typically have cargo capacities of 8-9 cubic meters. Medium vehicles are modeled after Peugeot Expert and Renault models, which usually have cargo capacities of 5-6 cubic meters. Small vehicles are modeled after Caddy Cargo and Renault Cargo models, which typically have cargo

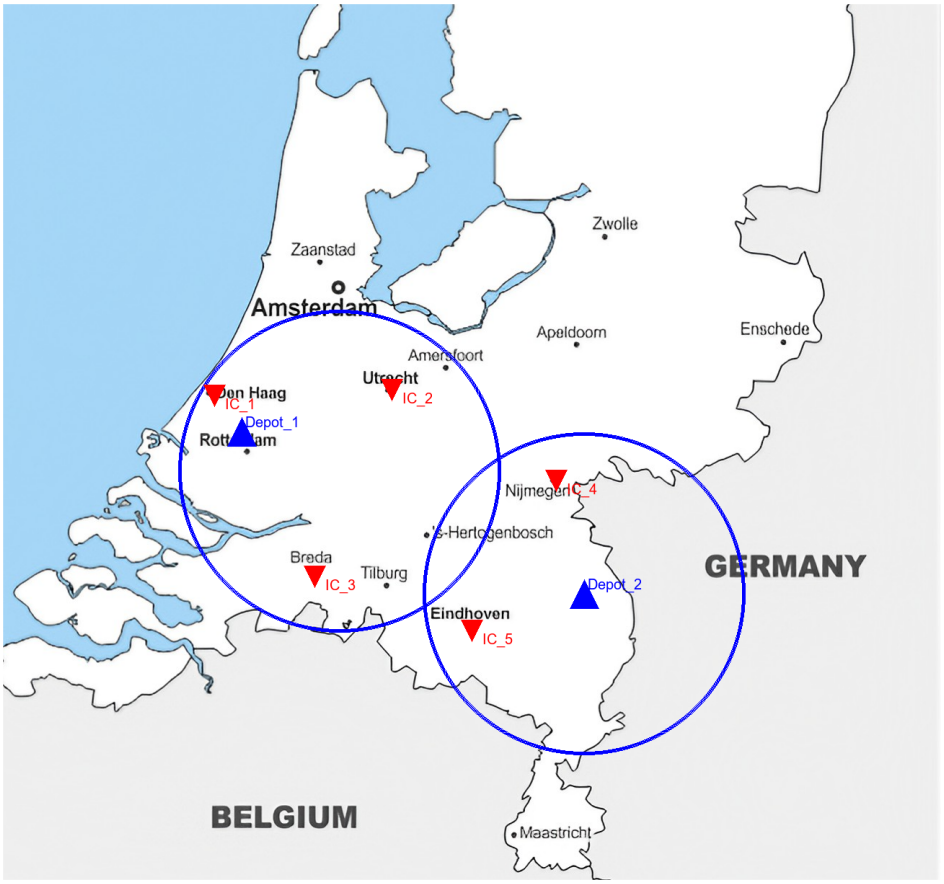

Figure 6.1: Case Study Map - NL, BE and DE

capacities of 4-5 cubic meters. The capacity factors for medium and small vehicles are estimated based on their cargo capacities. Each warehouse is equipped with two RD vehicles, each with a capacity of 200, and the capacity of each parcel is randomly chosen from [2, 3, 4]. In the comparative experiment, we chose the strategy of replacing RD with OD instead of adding OD as an additional strategy. This is because the ALNS algorithm in this study prioritizes the use of OD vehicles. If the results show that the cost with sufficient use of OD vehicles is higher than that with only RD vehicles, it indicates that adding OD vehicles does not bring cost benefits to a certain extent.

Below summarizes the common characteristics of the Case Study instances:

- **Total number of requests**: 150
- **Number of requests for customer**: 50
- **Number of requests for IC**: 100
- **Number of RD in each depot**: 2
- **Number of OD (if needed)**: 2
- **Capacity RD**: 200
- **Capacity OD**: 45% and 60% Capacity RD
- **Maximum number of times OD visits the depot**: 2

6.3. Comparative Experiment

We aim to compare employing ODs with different vehicle types and whether providing different levels of subsidies to ODs can reduce overall routing costs. For the former, we set three scenarios: no OD involvement, 2 medium-sized OD, and 2 small-sized OD. For the latter, when choosing small-sized

Large: Ford Transit

Medium: Peugeot Expert

Small: Caddy Cargo

Figure 6.2: Comparative View of Different Cargo Vans

vehicles as ODs, we set 5 cost scenarios with OD unit routing costs at 45%, 50%, 60%, 70%, and 80% of RD routing costs. When choosing medium-sized vehicles as ODs, we set 4 cost scenarios with OD unit routing costs at 50%, 60%, 70%, and 80% of RD routing costs.

We use the no OD experiment group as a baseline. We compare the total routing costs of instances using different types of vehicles and different subsidy costs with the total costs of the experimental group. The results of no OD experiment group are shown in Table 6.1. Due to the infeasible solution of instance ”2.5.150.3”, there are 9 valid instances. The last row of the table gives the total routing costs of the 9 valid instances. Similarly, the results of introducing small-sized car ODs and medium-sized car ODs groups also remove instance ”2.5.150.3”. Their results are shown in Table 6.2 and Table 6.3, respectively.

Table 6.1: Case Study - RD only

RD only	
Instances	**Routing Cost**
2.5.150.1	1437
2.5.150.2	1270
2.5.150.4	1388
2.5.150.5	1084
2.5.150.6	1253
2.5.150.7	1302
2.5.150.8	1360
2.5.150.9	1274
2.5.150.10	1407.4
Sum	11775.4

In the group where 2 RDs were replaced by 2 small-sized cars (45% RD capacity), according to the data in the last row of the table, when the cost is also 45% of the RD unit distance cost, the total routing cost decreased by 11.37%, and when the cost is 50% of the RD unit distance cost, the total routing cost decreased by 5.11%. The costs obtained for other OD costs were higher than the experimental group costs without OD. In the group where 2 RDs were replaced by 2 medium-sized cars (60% RD capacity), according to the data in the last row of the table, when the cost is also 50% of the RD unit distance cost, the total routing cost decreased by 16.57%, and when the cost is 60% of the RD unit

distance cost, the total routing cost decreased by 6.85%. The costs obtained for other OD costs were higher than the experimental group costs without OD. From the results of the two experimental groups, it can be concluded that introducing OD vehicles can optimize the total routing cost if the ratio of OD capacity to RD capacity and the unit distance cost of OD to RD are consistent. Ochama can better optimize delivery costs based on market research to balance the acceptance of OD compensation and the required number of ODs.

Table 6.2: Case Study - 45% RD Capacity

45% Capacity					
Instances	**Cost 45%**	**Cost 50%**	**Cost 60%**	**Cost 70%**	**Cost 80%**
2.5.150.1	1302.2	1401.5	1498.8	1630.6	1718.6
2.5.150.2	1077.9	1137	1233.6	1268.2	1594.2
2.5.150.4	1142.2	1228	1265.8	1327	1611.6
2.5.150.5	1148.5	1201.5	1382.2	1450.6	1592.4
2.5.150.6	1197.2	1249.5	1406.6	1501.8	1574.6
2.5.150.7	1099	1268	1316.4	1427.4	1531
2.5.150.8	1163.3	1229.5	1335	1449.5	1565.6
2.5.150.9	1193.1	1242.5	1340.2	1325.6	1566.4
2.5.150.10	1112.7	1216	1291.4	1462.9	1496.4
Sum	10436.1	11173.5	12070	12843.6	14250.8
Gap	11.37%	5.11%	-2.50%	-9.07%	-21.02%

Table 6.3: Case Study - 60% RD Capacity

60% Capacity				
Instances	**Cost 50%**	**Cost 60%**	**Cost 70%**	**Cost 80%**
2.5.150.1	1111	1264.4	1380	1431.6
2.5.150.2	1203	1305.8	1472	1532.4
2.5.150.4	1263	1389.2	1582.9	1645.4
2.5.150.5	956	1106.6	1182.2	1266
2.5.150.6	1261	1324.4	1467.7	1595
2.5.150.7	901	1030.4	1099.8	1165.6
2.5.150.8	1275	1446.6	1618.2	1789.8
2.5.150.9	974	1090.4	1229.7	1366.4
2.5.150.10	880	1011	1157.7	1253.2
Sum	9824	10968.8	12190.2	13045.4
Gap	16.57%	6.85%	-3.52%	-10.79%

Figure 6.3 and 6.4 present the visualized results of all instances' costs relative to the baseline cost without OD, with the addition of small-sized OD and middle-sized OD, respectively. In each figure, the grey dashed line represents the baseline without OD consideration. When OD is introduced, data points above the baseline indicate cost savings in route expenses, whereas points below the baseline indicate higher route costs. The x-axis represents each instance, and each differently colored line corresponds to different OD cost settings. The figures provide a clear visualization of whether cost savings occur for each instance across different OD cost values.

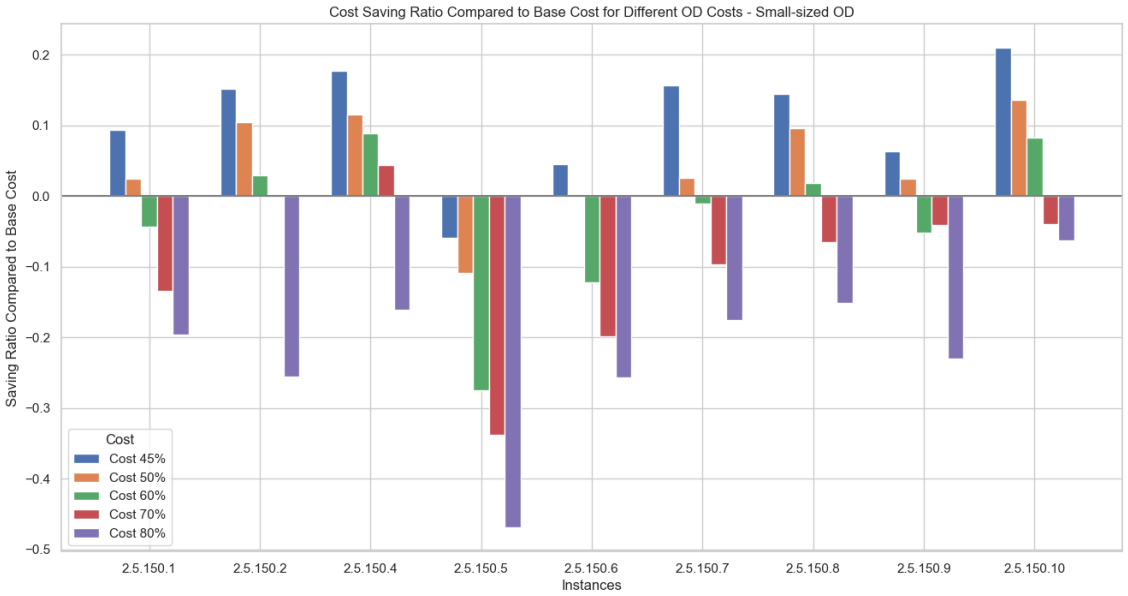

Figure 6.3: Results for adding Small-sized OD

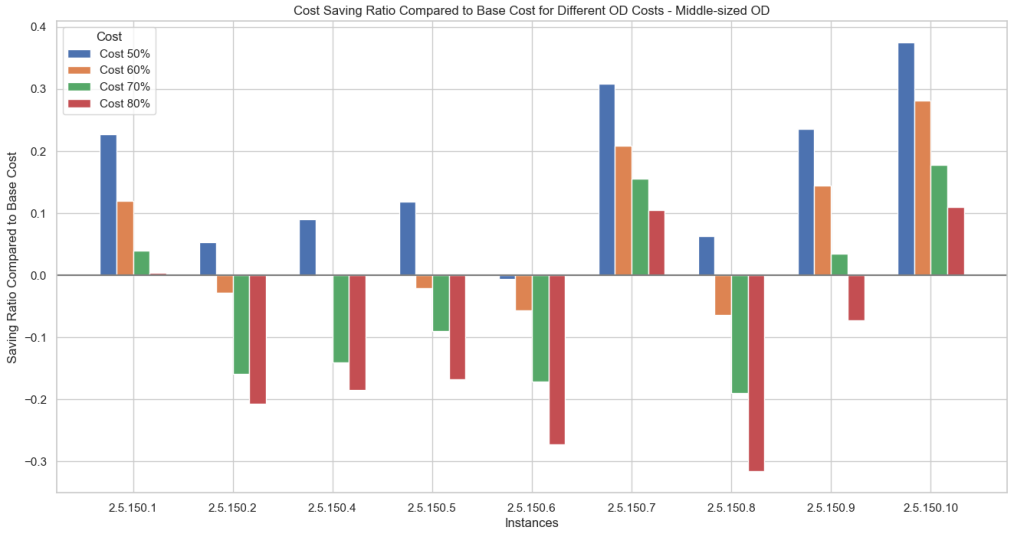

Figure 6.4: Results for adding Middle-sized OD

7

Discussion

We first consider constructing a mathematical model to solve the research problem. Considering the difficulty of directly developing the algorithm, the mathematical model can help better understand the constraints needed to solve the problem, and during the algorithm part, these constraints can be quickly converted into algorithm functions, making the logic of the algorithm generation phase clearer. Since this study incorporates multiple features based on actual applications, we have to introduce more variables to handle the situations of multi-depots and multiple visits to depots by OD. The addition of variables requires us to impose more constraints to restrict these variables. Therefore, we use a relatively simple arc-based modeling approach and divide the constraints into four parts based on the features. This segmentation greatly facilitates the debugging of the mathematical model, allowing us to quickly locate and solve problems. Next, we generated instances of three request sizes and used Cplex to evaluate the mathematical model. The evaluation results show that when the number of requests is 10, the optimal solution can be obtained within the set one-hour runtime. However, as the problem size increases, the mathematical model cannot obtain a feasible solution within a short time and can only get the best bound.

Based on the literature review, ALNS is widely used to solve pickup and delivery problems and related variants, making it the first choice for our research problem. This method was proposed by (Ropke and Pisinger, 2006a). We modified the ALNS algorithm framework to suit our study. The first step of the ALNS algorithm is to generate an initial feasible solution. We used the common Basic Greedy Algorithm, which, after testing, can generally generate a feasible solution within 1 second. This algorithm requires us to ensure that there are enough vehicles since the basic greedy has a certain short-sighted behavior, leading to an initial solution that is always more costly than the optimal solution. The initial solution will be re-destroyed and repaired in the subsequent part of the algorithm, so the number of vehicles will be minimized during this process.

In the requests removal part, (Ropke and Pisinger, 2006a) studied the problem of single-visit pickup and delivery by vehicles, and their ALNS algorithm framework cannot be directly applied to our research. Our study considers the problem of multiple visits to depots by OD. Thus, in the generation of the initial solution, the Basic Greedy Algorithm frequently adds depots as pickup or delivery points because we cannot determine the number of visits of each OD to each depot in the optimal solution in advance. In fact, the total number of visits to depots by all vehicles in the initial solution must be greater than or equal to the total number of visits to depots by all vehicles in the optimal solution. If we follow the original ALNS algorithm framework, these redundant visits to depots will be retained, meaning that vehicles may make some meaningless trips between depots. We devised a strategy to remove depots and requests separately. First, a certain number of depots are randomly selected, and then a second random selection is made from these depots. The depots not selected are permanently deleted, ensuring that the improved ALNS algorithm can fully explore the solution space to obtain a better solution. However, this setting also has certain limitations. If too many depots are deleted, the repaired solution will be unreasonable. In this case, we check whether each repaired solution is feasible. If feasible, we perform the acceptance criteria; if not, we skip this iteration and move to the next one. In

this study, we considered two removal operators: Random Removal and Worst Removal. During the experiments, the probabilities of selecting these two removal operators were similar, indicating that the probabilities of the new solutions being accepted by these two removal operations were similar.

In the requests insertion part, we considered two repair operators: Basic Greedy Heuristic and Regret-2 Heuristic. During the experiments, the probability of selecting Basic Greedy Heuristic was significantly higher than that of Regret-2 Heuristic, indicating that the probability of the solution repaired by Basic Greedy Heuristic being accepted was higher. This is because Regret-2 Heuristic calculates the cost of all feasible insertion positions for each request, sorts the insertion costs for different positions from small to large, sums the smallest two values for each request, and gives priority to the one with the largest sum. Since we considered two types of vehicles, RD and OD, the cost of inserting into the RD route is sometimes higher than that of inserting into the OD route. Therefore, if a request had fewer feasible insertion positions, its probability of being selected is greater, which disrupt our selection to a certain extent, making the generated solution more difficult to accept. However, we also found that once Regret-2 Heuristic generated a feasible solution, it would jump out of the short-sighted behavior of Basic Greedy Heuristic. Therefore, the introduction of Regret-2 Heuristic is necessary.

In the Computational Experiments section, we firstly verified the consistency of the mathematical model and the algorithm. We then tuned the 7 parameters included in the ALNS algorithm. The tuning results showed that certain parameter values were at the boundaries of the values we set. We also tried values beyond the boundaries but within a reasonable range, finding that further exploration of these parameters reduced their universality. Therefore, the parameter values chosen did not exceed the set range. The parameter combinations performed well in the subsequent experiments and case study sections. Then, we generated 20 instances for sensitivity analysis, showing good results, indicating that the introduction of OD can optimize route costs to some extent. This finding was also verified in the case study.

In the Case Study section, we defined the total cost performance of using different types of vehicles and different subsidy ratios for OD. The results showed that introducing OD optimized the total cost under certain parameter combinations. The pricing strategy for OD by logistics companies and the willingness of temporary drivers can be further studied. (Kung and Zhong, 2017) discussed a richer study of OD pricing strategies, including employee pricing, transaction pricing, and cross-subsidy, which can provide references for further enriching this study.

8

Conclusions and Recommendations

8.1. Conclusion

The motivation for this research arises from the lack of models in related studies that integrate the features of multiple depots, occasional drivers, and multiple visits to depots. These features are added to better align with real-world situations. Our research objective is defined as: developing a mathematical model to solve the proposed operations research problem, and selecting an appropriate algorithm to convert and solve the mathematical model, finding a solution to the routing problem at the lowest possible cost within a short time.

The literature review provides insights into the development of models and algorithms related to the Pickup and Delivery Problem with Time Window (PDPTW), Crowd-shipping, and the Open Vehicle Routing Problem (OVRP). Notable algorithmic approaches include Adaptive Large Neighborhood Search (ALNS), Variable Neighborhood Search (VNS), Tabu Search (TS), Stochastic and Genetic algorithms, Simulated Annealing, Greedy Randomized Adaptive Search Procedures (GRASP) for Vehicle Routing, Integer Linear Programming (ILP), and other specialized algorithms. Upon comparison, it has been found that the ALNS algorithm is a good choice for solving this problem. The research methods section delves into the steps of the ALNS algorithm and the criteria used in each step when solving different problems. These integrated studies provide significant reference value for solving the current problem.

In the methodology chapter, the specific content of the mathematical model and the ALNS algorithm is presented. We first constructed a constraint programming model, which is divided into four main parts based on regular drivers, occasional drivers, and multiple visits to depots. The first part contains the basic model constraints of the pickup and delivery problem, while the second, third, and fourth parts respectively address the constraints for regular drivers, occasional drivers, and the feature of multiple visits to depots by occasional drivers. In the algorithm section, ALNS is applied to solve the proposed problem. The initial solution of the algorithm uses the Basic Greedy Algorithm, which can obtain a reasonable initial solution. In the request removal part, random removal operators and worst removal operators are introduced. The random removal operator has stronger randomness in destroying the initial solution, while the worst removal operator preferentially removes the costliest requests. The combination of these two removal operators can avoid the solution falling into a local optimum to a certain extent. In the inserting requests part, Basic Greedy Insertion and Regret-2 Insertion are introduced. When inserting the removed requests, the Basic Greedy Insertion explores each request waiting to be restored, and the point and position with the lowest insertion cost are given priority. Regret-2 addresses the short-sightedness of Basic Greedy Insertion by calculating the sum of the costs of the top two lowest-cost positions for inserting a request. The request with the highest cost sum is selected first, which helps jump out of the local optimum. Next, we use a roulette wheel method to assign weights to each operator, where better-performing operators have greater weights and higher probabilities of being selected. Acceptance and stopping criteria use simulated annealing acceptance criteria, which also helps fully explore the solution space and avoid falling into local optima.

The Computational Experiments section uses Cplex and the ALNS algorithm to solve the mathematical

model, verifying the effectiveness of both the model and the algorithm. The parameter tuning section finds more universal parameter settings for the algorithm, enhancing its application range. The sensitivity analysis part explores whether the introduction of OD and subsidies for OD can lead to cost savings. In the 20 experimental instances, the results show that introducing one OD, when the OD cost is 90% of the RD cost, the total routing cost can be reduced by 5.18%. When the OD cost is 80% of the RD cost, the total routing cost can be reduced by 15.21%. When using one OD and fixing the OD cost at 80% of the RD cost, the introduction of 60% capacity OD can save 7.86% of the total cost, 70% capacity OD can save 9.97% of the total cost, and 80% capacity OD can save 15.21% of the total cost. We conclude that the introduction of OD can reduce routing costs to varying degrees. The number of ODs, cost, and capacity directly affect the proportion of cost savings. At the same time, cost savings require companies to choose the appropriate number of ODs, the subsidy price for each OD, and the capacity of the ODs based on market research to balance the willingness of ODs, the quality of service, and cost savings.

Finally, we conducted a case study based on the Dutch e-commerce company Ochama, scaling down the instance to 150 requests and similarly reducing vehicle capacity according to actual vehicle capacity. We mapped the warehouses and intermediate centers to two-dimensional coordinates based on actual physical distances and randomly generated requests within the service area. Considering that vehicles need to run on highways and urban roads, we assumed an average vehicle speed of 60km/h, linking unit path distance to unit path distance cost. Out of 10 instances, 9 were valid. The third instance failed because the ALNS algorithm did not find a solution better than the initial solution within the set 10,000 iterations. The remaining 9 valid instances indicate that the introduction of different OD vehicle types can save total routing costs to varying degrees. Compared to the research by (Hou and Wang, 2021), which introduced OD and reduced the total routing cost by 7.3%, our approach reduced the total routing cost by 11.37% and 6.85% for small and medium-sized vehicles, respectively, when the cost-to-capacity ratio was equal. For small vehicles, with a cost ratio of 50%, the total routing cost could still be reduced by 5.11%.

8.2. Recommendations

This study addresses the proposed a more realistic pickup and delivery problem through modeling and algorithm development. However, this is the beginning. This section highlights some areas for improvement in this study and suggests directions for future research.

- **Improving the Mathematical Model**
  By changing the modeling strategy, this research uses an Arc-based Formulation, which has the advantage of detailing each step along the path, making it easier to understand. However, it requires more constraint variables. Alternative modeling approaches, such as Node-based Formulation, Path-based Formulation, or Column Generation, could be explored to reframe the problem, reducing the number of constraint variables and potentially shortening the solution time.

- **Further Optimization of Total Route Cost for Instances**
  For the ALNS algorithm, we obtained a relatively good solution. For large-scale problems, we might obtain a local optimal solution. To explore the optimal solution, we could consider using the variable values of the current best solution of the ALNS algorithm as input for exact algorithms, such as branch and bound, to further explore the solution space and obtain better solutions.

- **Adding More Destruction and Repair Operators**
  For the ALNS algorithm, only two repair and destruction operators are used. More repair and destruction operators could be added later to enhance the algorithm's selectivity, thereby further improving its performance.

- **Adjusting the Strategy for Selecting Destruction Coefficients**
  In the parameter tuning section, there is a parameter for the destruction coefficient κ. This study uses a destruction coefficient of 0.1, which is suitable for scenarios with a small number of requests. For example, with 50 requests, 5 requests will be selected for removal each time a

destruction operator is executed. However, when the number of requests is large, such as 500 requests, 50 requests will be removed, leading to a lengthy repair process for finding new feasible solutions or better solutions. Therefore, it is advisable to link the value of the destruction operator to the number of requests in the problem. In large-scale cases, controlling the removal of a smaller number of requests can help quickly explore the solution space in a short time.

- **Solving Real-world Package Delivery Problems**
  The case study in this research randomly generates demand points based on a selected company. The actual situation involves a larger number of demands. Therefore, future work could apply this research to real-world problems. Additionally, companies can conduct market research to determine the proportion of drivers with each type of vehicle and the subsidies that crowdsourced vehicle drivers can accept per unit distance or unit time. Based on the research results, companies can decide whether to introduce OD. If OD is introduced, this study can help analyze which type of OD to choose and the OD pricing strategy.

Bibliography

Ahmed, Z. H. and Yousefikhoshbakht, M. (2023). An improved tabu search algorithm for solving heterogeneous fixed fleet open vehicle routing problem with time windows. *Alexandria Engineering Journal*, 64:349–363.

Allahviranloo, M. and Baghestani, A. (2019). A dynamic crowdshipping model and daily travel behavior. *Transportation Research Part E: Logistics and Transportation Review*, 128:175–190.

Archetti, C., Savelsbergh, M., and Speranza, M. G. (2016). The vehicle routing problem with occasional drivers. *European Journal of Operational Research*, 254(2):472–480.

Arslan, A. M., Agatz, N., Kroon, L., and Zuidwijk, R. (2019). Crowdsourced delivery—a dynamic pickup and delivery problem with ad hoc drivers. *Transportation Science*, 53(1):222–235.

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

Bhatti, A., Akram, H., Basit, H. M., Khan, A. U., Raza, S. M., Naqvi, M. B., et al. (2020). E-commerce trends during covid-19 pandemic. *International Journal of Future Generation Communication and Networking*, 13(2):1449–1452.

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

Brandão, J. (2004). A tabu search algorithm for the open vehicle routing problem. *European Journal of Operational Research*, 157(3):552–564.

Cao, E. and Lai, M. (2010). The open vehicle routing problem with fuzzy demands. *Expert Systems with Applications*, 37(3):2405–2411.

Cao, E., Lai, M., and Yang, H. (2014). Open vehicle routing problem with demand uncertainty and its robust strategies. *Expert Systems with Applications*, 41(7):3569–3575.

Carbone, V., Rouquet, A., and Roussat, C. (2017). The rise of crowd logistics: a new way to co-create logistics value. *Journal of Business Logistics*, 38(4):238–252.

Coelho, L. C., Cordeau, J.-F., and Laporte, G. (2012a). Consistency in multi-vehicle inventory-routing. *Transportation Research Part C: Emerging Technologies*, 24:270–287.

Coelho, L. C., Cordeau, J.-F., and Laporte, G. (2012b). The inventory-routing problem with transshipment. *Computers & Operations Research*, 39(11):2537–2548.

Dahle, L., Andersson, H., Christiansen, M., and Speranza, M. G. (2019). The pickup and delivery problem with time windows and occasional drivers. *Computers & Operations Research*, 109:122–133.

Dayarian, I. and Savelsbergh, M. (2020). Crowdshipping and same-day delivery: Employing in-store customers to deliver online orders. *Production and Operations Management*, 29(9):2153–2174.

Demir, E., Bektaş, T., and Laporte, G. (2012). An adaptive large neighborhood search heuristic for the pollution-routing problem. *European journal of operational research*, 223(2):346–359.

Devari, A., Nikolaev, A. G., and He, Q. (2017). Crowdsourcing the last mile delivery of online orders by exploiting the social networks of retail store customers. *Transportation Research Part E: Logistics and Transportation Review*, 105:105–122.

Di Puglia Pugliese, L., Ferone, D., Festa, P., Guerriero, F., and Macrina, G. (2022). Solution approaches for the vehicle routing problem with occasional drivers and time windows. *Optimization Methods and Software*, 37(4):1384–1414.

Doan, A., Ramakrishnan, R., and Halevy, A. Y. (2011). Crowdsourcing systems on the world-wide web. *Communications of the ACM*, 54(4):86–96.

Dueck, G. (1993). New optimization heuristics: The great deluge algorithm and the record-to-record travel. *Journal of Computational physics*, 104(1):86–92.

Dumas, Y., Desrosiers, J., and Soumis, F. (1991). The pickup and delivery problem with time windows. *European journal of operational research*, 54(1):7–22.

Fleszar, K., Osman, I. H., and Hindi, K. S. (2009). A variable neighbourhood search algorithm for the open vehicle routing problem. *European Journal of Operational Research*, 195(3):803–809.

Fu, Z., Eglese, R., and Li, L. Y. (2005). A new tabu search heuristic for the open vehicle routing problem. *Journal of the operational Research Society*, 56(3):267–274.

Gao, X., Shi, X., Guo, H., and Liu, Y. (2020). To buy or not buy food online: The impact of the covid-19 epidemic on the adoption of e-commerce in china. *PloS one*, 15(8):e0237900.

Gdowska, K., Viana, A., and Pedroso, J. P. (2018). Stochastic last-mile delivery with crowdshipping. *Transportation research procedia*, 30:90–100.

Ghilas, V., Demir, E., and Van Woensel, T. (2016). An adaptive large neighborhood search heuristic for the pickup and delivery problem with time windows and scheduled lines. *Computers & Operations Research*, 72:12–30.

Guo, X., Jaramillo, Y. J. L., Bloemhof-Ruwaard, J., and Claassen, G. (2019). On integrating crowd-sourced delivery in last-mile logistics: A simulation study to quantify its feasibility. *Journal of Cleaner Production*, 241:118365.

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

Hou, S. and Wang, C. (2021). Matching models for crowd-shipping considering shipper's acceptance uncertainty. In *2021 IEEE International Conference on Autonomous Systems (ICAS)*, pages 1–6. IEEE.

Huang, K. and Ardiansyah, M. N. (2019). A decision model for last-mile delivery planning with crowd-sourcing integration. *Computers & Industrial Engineering*, 135:898–912.

Irnich, S. (2000). A multi-depot pickup and delivery problem with a single hub and heterogeneous vehicles. *European Journal of Operational Research*, 122(2):310–328.

Kafle, N., Zou, B., and Lin, J. (2017). Design and modeling of a crowdsource-enabled system for urban parcel relay and delivery. *Transportation research part B: methodological*, 99:62–82.

Kirkpatrick, S., Gelatt Jr, C. D., and Vecchi, M. P. (1983). Optimization by simulated annealing. *science*, 220(4598):671–680.

Kovacs, A. A., Parragh, S. N., Doerner, K. F., and Hartl, R. F. (2012). Adaptive large neighborhood search for service technician routing and scheduling problems. *Journal of scheduling*, 15:579–600.

Kung, L.-C. and Zhong, G.-Y. (2017). The optimal pricing strategy for two-sided platform delivery in the sharing economy. *Transportation Research Part E: Logistics and Transportation Review*, 101:1–12.

Laporte, G., Musmanno, R., and Vocaturo, F. (2010). An adaptive large neighbourhood search heuristic for the capacitated arc-routing problem with stochastic demands. *Transportation Science*, 44(1):125–135.

Lee, S., Kang, Y., and Prabhu, V. V. (2016). Smart logistics: distributed control of green crowdsourced parcel services. *International Journal of Production Research*, 54(23):6956–6968.

Li, B., Krushinsky, D., Van Woensel, T., and Reijers, H. A. (2016). The share-a-ride problem with stochastic travel times and stochastic delivery locations. *Transportation Research Part C: Emerging Technologies*, 67:95–108.

Li, H. and Lim, A. (2001). A metaheuristic for the pickup and delivery problem with time windows. In *Proceedings 13th IEEE International Conference on Tools with Artificial Intelligence. ICTAI 2001*, pages 160–167. IEEE.

Li, W., Wu, Y., Kumar, P. R., and Li, K. (2020). Multi-trip vehicle routing problem with order release time. *Engineering Optimization*, 52(8):1279–1294.

Macrina, G., Di Puglia Pugliese, L., Guerriero, F., and Laganà, D. (2017). The vehicle routing problem with occasional drivers and time windows. In *Optimization and Decision Science: Methodologies and Applications: ODS, Sorrento, Italy, September 4-7, 2017 47*, pages 577–587. Springer.

Maknoon, Y. and Laporte, G. (2017). Vehicle routing with cross-dock selection. *Computers & Operations Research*, 77:254–266.

Mara, S. T. W., Norcahyo, R., Jodiawan, P., Lusiantoro, L., and Rifai, A. P. (2022). A survey of adaptive large neighborhood search algorithms and applications. *Computers & Operations Research*, 146:105903.

MirHassani, S. and Abolghasemi, N. (2011). A particle swarm optimization algorithm for open vehicle routing problem. *Expert Systems with Applications*, 38(9):11547–11551.

Muklason, A., Bwananesia, P. C., YT, S. H., Angresti, N. D., and Supoyo, V. A. (2018). Automated examination timetabling optimization using greedy-late acceptance-hyperheuristic algorithm. In *2018 International Conference on Electrical Engineering and Computer Science (ICECOS)*, pages 201–206. IEEE.

Nanry, W. P. and Barnes, J. W. (2000). Solving the pickup and delivery problem with time windows using reactive tabu search. *Transportation Research Part B: Methodological*, 34(2):107–121.

Nowak, M., Ergun, Ö., and White III, C. C. (2008). Pickup and delivery with split loads. *Transportation science*, 42(1):32–43.

Nowak, M. A. (2005). *The pickup and delivery problem with split loads*. Georgia Institute of Technology.

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

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

Pisinger, D. and Ropke, S. (2019). Large neighborhood search. *Handbook of metaheuristics*, pages 99–127.

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

Pugliese, L. D. P., Ferone, D., Festa, P., Guerriero, F., and Macrina, G. (2022). Combining variable neighborhood search and machine learning to solve the vehicle routing problem with crowd-shipping. *Optimization Letters*, pages 1–23.

Qu, Y. and Bard, J. F. (2012). A grasp with adaptive large neighborhood search for pickup and delivery problems with transshipment. *Computers & Operations Research*, 39(10):2439–2456.

Qu, Y. and Bard, J. F. (2013). The heterogeneous pickup and delivery problem with configurable vehicle capacity. *Transportation Research Part C: Emerging Technologies*, 32:1–20.

Ren, S., Luo, F., Lin, L., Hsu, S.-C., and Li, X. I. (2019). A novel dynamic pricing scheme for a large-scale electric vehicle sharing network considering vehicle relocation and vehicle-grid-integration. *International Journal of Production Economics*, 218:339–351.

Repoussis, P. P., Tarantilis, C. D., and Ioannou, G. (2007). The open vehicle routing problem with time windows. *Journal of the Operational Research Society*, 58(3):355–367.

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

Ropke, S. and Pisinger, D. (2006b). A unified heuristic for a large class of vehicle routing problems with backhauls. *European Journal of Operational Research*, 171(3):750–775.

Salari, M., Toth, P., and Tramontani, A. (2010). An ilp improvement procedure for the open vehicle routing problem. *Computers & Operations Research*, 37(12):2106–2120.

Sampaio, A., Savelsbergh, M., Veelenturf, L. P., and Van Woensel, T. (2020). Delivery systems with crowd-sourced drivers: A pickup and delivery problem with transfers. *Networks*, 76(2):232–255.

Santos, V. G. M. and de Carvalho, M. A. M. (2018). Adaptive large neighborhood search applied to the design of electronic circuits. *Applied Soft Computing*, 73:14–23.

Sarasola, B. and Doerner, K. F. (2020). Adaptive large neighborhood search for the vehicle routing problem with synchronization constraints at the delivery location. *Networks*, 75(1):64–85.

Sariklis, D. and Powell, S. (2000). A heuristic method for the open vehicle routing problem. *Journal of the Operational Research Society*, 51:564–573.

Şevkli, A. Z. and Güler, B. (2017). A multi-phase oscillated variable neighbourhood search algorithm for a real-world open vehicle routing problem. *Applied Soft Computing*, 58:128–144.

Shaw, P. (1997). A new local search algorithm providing high quality solutions to vehicle routing problems. *APES Group, Dept of Computer Science, University of Strathclyde, Glasgow, Scotland, UK*, 46.

Shaw, P. (1998). Using constraint programming and local search methods to solve vehicle routing problems. In *International conference on principles and practice of constraint programming*, pages 417–431. Springer.

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

Song, L., Cherrett, T., McLeod, F., and Guan, W. (2009). Addressing the last mile problem: transport impacts of collection and delivery points. *Transportation research record*, 2097(1):9–18.

Tao, Y., Zhuo, H., and Lai, X. (2023). The pickup and delivery problem with multiple depots and dynamic occasional drivers in crowdshipping delivery. *Computers & Industrial Engineering*, 182:109440.

Tarantilis, C. and Kiranoudis, C. (2002). Distribution of fresh meat. *Journal of Food Engineering*, 51(1):85–91.

Tarantilis, C. D., Diakoulaki, D., and Kiranoudis, C. T. (2004a). Combination of geographical information system and efficient routing algorithms for real life distribution operations. *European Journal of Operational Research*, 152(2):437–453.

Tarantilis, C. D., Ioannou, G., Kiranoudis, C. T., and Prastacos, G. P. (2004b). A threshold accepting approach to the open vehicle routing problem. *RAIRO-Operations Research*, 38(4):345–360.

Tarantilis, C. D., Ioannou, G., Kiranoudis, C. T., and Prastacos, G. P. (2005). Solving the open vehicle routeing problem via a single parameter metaheuristic algorithm. *Journal of the Operational research Society*, 56:588–596.

Tavakkoli-Moghaddam, R., Meskini, M., Nasseri, H., and Tavakkoli-Moghaddam, H. (2019). A multi-depot close and open vehicle routing problem with heterogeneous vehicles. In *2019 international conference on industrial engineering and systems management (IESM)*, pages 1–6. IEEE.

Torres, F., Gendreau, M., and Rei, W. (2022). Crowdshipping: An open vrp variant with stochastic destinations. *Transportation Research Part C: Emerging Technologies*, 140:103677.

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

Van Landeghem, H. (1988). A bi-criteria heuristic for the vehicle routing problem with time windows. *European Journal of Operational Research*, 36(2):217–226.

Vincent, F. Y., Aloina, G., Jodiawan, P., Gunawan, A., and Huang, T.-C. (2023). The vehicle routing problem with simultaneous pickup and delivery and occasional drivers. *Expert Systems with Applications*, 214:119118.

Vincent, F. Y., Jewpanya, P., and Redi, A. P. (2016). Open vehicle routing problem with cross-docking. *Computers & Industrial Engineering*, 94:6–17.

Voigt, S. and Kuhn, H. (2022). Crowdsourced logistics: The pickup and delivery problem with transshipments and occasional drivers. *Networks*, 79(3):403–426.

A

Parameters Tuning Result

Table A.1: Instances Parameter Tuning Results - Random Parameter p

Fixed parameter setting: κ **= 0.1,** T_{start} **= 200,** c **= 0.9,** r **= 0.2,** σ_1 **= 10,** σ_2 **= 5**														
Tuning Instances	**Run Time**					**Avg. Time**	**Total Avg. Time**	**Cost**					**Avg. Cost**	**Total Avg. Cost**
	1	**2**	**3**	**4**	**5**			**1**	**2**	**3**	**4**	**5**		
Parameter setting: p **= 1**														
2-1-10-1	0.7	0.7	0.7	0.6	0.6	0.66	22.00	294	357	314	357	357	335.8	**1057.87**
2-1-10-2	0.7	0.7	0.8	0.7	0.7	0.72		400.5	400.5	389	389	389	393.6	
2-2-30-1	11.9	12.9	11.2	11.8	11.3	11.82		1343.5	1278	1310	1176	1332	1287.9	
2-2-30-2	9.6	8.0	7.7	7.6	7.8	8.14		1049	1030	1009	1067.5	1092.5	1049.6	
3-2-50-1	49.0	49.3	47.5	46.8	46.6	47.84		1979	1979	1979	1979	1979	1979	
3-2-50-2	66.7	62.9	54.2	61.9	68.4	62.82		1294.5	1306	1304	1304	1298	1301.3	
Parameter setting: p **= 5**														
2-1-10-1	0.8	0.7	0.7	0.8	0.6	0.72	**20.82**	357	391.5	357	357	294	351.3	1060.95
2-1-10-2	0.6	0.7	0.7	0.7	0.7	0.68		389	389	400.5	400.5	389	393.6	
2-2-30-1	11.1	12.5	11.8	12.7	11.5	11.92		1620.5	1331.5	1221	1006	1557.5	1347.3	
2-2-30-2	6.4	6.1	5.7	5.6	6.3	6.02		1008	1033.5	1091.5	1031.5	1046	1042.1	
3-2-50-1	45.6	59.2	47.4	48.3	45.7	49.24		1979	1640	1979	1979	1979	1911.2	
3-2-50-2	69.5	48.8	46.4	55.1	62.0	56.36		1364.5	1302	1298	1304	1332.5	1320.2	
Parameter setting: p **= 10**														
2-1-10-1	0.7	0.6	0.6	0.6	0.6	0.62	20.29	357	294	357	357	357	344.4	1063.32
2-1-10-2	0.6	0.7	0.7	0.8	0.9	0.74		389	389	400.5	389	389	391.3	
2-2-30-1	9.9	11.1	10.7	10.5	11.0	10.64		1549.5	970	1132	1145.5	1572	1273.8	
2-2-30-2	7.8	8.5	8.4	9.1	9.3	8.62		1051	1098.5	1038	1022	1066	1055.1	
3-2-50-1	46.4	44.0	43.5	44.7	44.4	44.60		1979	1979	1979	1979	1979	1979	
3-2-50-2	63.0	62.4	66.2	41.7	49.3	56.52		1457.5	1320	1304	1302	1298	1336.3	
Parameter setting: p **= 20**														
2-1-10-1	0.6	0.6	0.7	0.7	0.6	0.64	23.25	357	357	357	357	328.5	351.3	1064.87
2-1-10-2	0.8	0.8	0.7	0.7	0.6	0.72		389	400.5	389	389	389	391.3	
2-2-30-1	11.6	10.7	10.9	11.0	11.0	11.04		1328.5	1428.5	1582.5	1277	1028	1328.9	
2-2-30-2	7.2	6.9	6.6	6.8	6.7	6.84		1023	1049.5	995	1029.5	1104.5	1040.3	
3-2-50-1	45.0	47.7	45.7	44.8	45.3	45.70		1979	1979	1979	1979	1979	1979	
3-2-50-2	66.6	64.7	65.5	62.4	68.7	65.58		1294.5	1306	1306	1306	1298	1302.9	

Table A.2: Instances Parameter Tuning Results - Reaction Factor r

Fixed parameter setting: κ **= 0.1,** p **= 1,** T_{start} **= 200,** c **= 0.9,** σ_1 **= 10,** σ_2 **= 5**														
Tuning Instances	**Run Time**					**Avg. Time**	**Total Avg. Time**	**Cost**					**Avg. Cost**	**Total Avg. Cost**
	1	**2**	**3**	**4**	**5**			**1**	**2**	**3**	**4**	**5**		
Parameter setting: r **= 0.2**														
2-1-10-1	1.0	1.0	1.0	1.1	0.8	0.98	28.97	357	357	294	357	357	344.4	**1064.63**
2-1-10-2	1.2	1.0	0.9	1.3	1.5	1.18		389	389	464	389	464	419.0	
2-2-30-1	16.1	15.5	14.9	14.4	14.6	15.10		1556	1483	1136	1176	1285	1327.2	
2-2-30-2	13.7	9.6	9.7	10.7	9.8	10.70		1065.5	1115.5	1025	1011	1016.5	1046.7	
3-2-50-1	73.8	70.3	57.4	56.4	60.0	63.58		1833	1979	1979	1979	1979	1949.8	
3-2-50-2	73.3	80.4	85.9	87.7	84.2	82.30		1304	1304.5	1298	1293	1304	1300.7	
Parameter setting: r **= 0.4**														
2-1-10-1	1.7	1.5	1.5	1.0	0.8	1.30	**26.30**	357	357	357	357	357	357.0	1079.62
2-1-10-2	1.8	1.7	1.2	0.8	0.9	1.28		389	400.5	389	389	389	391.3	
2-2-30-1	14.5	12.8	12.8	13.4	11.2	12.94		1559	1464.5	1176.5	1404	1423	1405.4	
2-2-30-2	10.3	9.9	10.5	11.2	10.0	10.38		1080.5	1054.5	1013.5	1027.5	1055	1046.2	
3-2-50-1	58.1	59.7	57.3	60.8	63.2	59.82		1979	1979	1979	1979	1979	1979.0	
3-2-50-2	78.5	73.5	63.1	74.6	70.8	72.10		1323.5	1298	1304	1293	1275.5	1298.8	
Parameter setting: r **= 0.6**														
2-1-10-1	0.8	0.8	0.8	0.8	0.9	0.82	29.14	357	357	357	391.5	357	363.9	1070.48
2-1-10-2	0.8	0.9	0.8	0.9	1.0	0.88		464	389	389	389		407.8	
2-2-30-1	12.9	14.5	14.1	12.9	13.9	13.66		1530	1405	1495	1483	1514	1485.4	
2-2-30-2	9.9	10.7	10.8	10.6	10.9	10.58		1052	1038	1026	1009.5	1026.5	1030.4	
3-2-50-1	63.7	61.1	77.7	69.2	59.0	66.14		1979	1979	1706.5	1566.5	1979	1842.0	
3-2-50-2	81.8	63.2	67.3	97.3	104.1	82.74		1295.5	1287.5	1290.5	1295.5	1298	1293.4	
Parameter setting: r **= 0.8**														
2-1-10-1	0.9	0.8	0.8	0.9	0.7	0.82	27.30	357	357	357	357	357	357.0	1093.55
2-1-10-2	1.0	1.0	0.9	1.3	0.9	1.02		389	389	389	389	389	389.0	
2-2-30-1	11.5	16.6	18.6	13.8	16.0	15.30		1590	1425	1637.5	1393	1379	1484.9	
2-2-30-2	10.6	9.9	10.1	9.9	11.5	10.40		1093.5	1082	1050.5	1005.5	1032.5	1052.8	
3-2-50-1	60.3	55.5	56.3	55.3	56.0	56.68		1979	1979	1979	1979	1979	1979.0	
3-2-50-2	87.5	76.0	77.6	80.2	76.7	79.60		1298	1304	1298	1295	1298	1298.6	

Table A.3: Instances Parameter Tuning Results - Score Adjustment Factor σ_1

Fixed parameter setting: κ **= 0.1,** p **= 1,** T_{start} **= 200,** c **= 0.9,** r **= 0.2,** σ_2 **= 5**														
Tuning Instances	**Run Time**					**Avg. Time**	**Total Avg. Time**	**Cost**					**Avg. Cost**	**Total Avg. Cost**
	1	**2**	**3**	**4**	**5**			**1**	**2**	**3**	**4**	**5**		
Parameter setting: σ_1 **= 10**														
2-1-10-1	1.2	1.1	1.1	0.9	1.0	1.06	**26.25**	357	357	357	357	357	357.0	1086.8
2-1-10-2	1.0	1.0	1.2	1.3	1.7	1.24		389	389	389	389	389	389.0	
2-2-30-1	16.6	15.3	12.9	14.2	13.0	14.40		1524	1470.5	1521.5	1516	1433.5	1493.1	
2-2-30-2	11.0	8.7	9.1	8.9	9.4	9.42		1126.5	1126.5	1091	1052.5	1126.5	1104.6	
3-2-50-1	56.8	55.6	56.9	58.5	82.7	62.10		1979	1979	1979	1832.5	1590	1871.9	
3-2-50-2	86.9	83.9	57.2	62.1	56.3	69.28		1324.5	1304	1294.5	1310	1293	1305.2	
Parameter setting: σ_1 **= 20**														
2-1-10-1	1.0	1.2	1.2	1.2	1.0	1.12	26.35	357	357	391.5	357	357	363.9	1077.2
2-1-10-2	1.4	1.3	1.1	0.8	0.7	1.06		389	389	389	389	389	389.0	
2-2-30-1	12.5	14.4	14.6	13.5	12.7	13.54		1164.5	1734.5	1253.5	1298.5	1475.5	1385.3	
2-2-30-2	8.8	10.3	10.3	8.8	9.2	9.48		1068	1026	1016	1051.5	1068	1045.9	
3-2-50-1	54.0	55.1	54.0	54.9	55.7	54.74		1979	1979	1979	1979	1979	1979.0	
3-2-50-2	78.7	80.8	82.1	70.7	78.4	78.14		1302	1287.5	1286.5	1326.5	1298	1300.1	
Parameter setting: σ_1 **= 30**														
2-1-10-1	0.8	0.8	0.7	0.7	0.8	0.76	26.29	357	357	357	357	357	357.0	**1060.1**
2-1-10-2	0.8	0.9	0.8	0.9	0.8	0.84		389	389	389	389	389	389.0	
2-2-30-1	15.3	12.7	15.3	13.6	14.9	14.36		1475.5	1530	1493	1424	1557	1495.9	
2-2-30-2	8.4	8.5	10.2	9.4	10.1	9.32		1068	1059	1003.5	1032.5	1001.5	1032.9	
3-2-50-1	56.1	57.8	66.7	49.0	67.5	59.42		1979	1979	1483.5	1979	1469.5	1778.0	
3-2-50-2	83.8	84.1	81.8	58.1	57.4	73.04		1307	1298	1304	1326	1304	1307.8	
Parameter setting: σ_1 **= 40**														
2-1-10-1	0.8	0.8	0.8	0.8	0.9	0.82	27.33	357	357	357	357	357	357.0	1088.73
2-1-10-2	0.8	0.8	0.8	0.7	0.7	0.76		389	389	400.5	389	389	391.3	
2-2-30-1	15.6	14.5	15.4	14.6	10.7	14.16		1442	1560	1461.5	1319.5	1493	1455.2	
2-2-30-2	10.3	9.7	10.1	9.2	9.4	9.74		1089.5	1070	1016	1031.5	1032.5	1047.9	
3-2-50-1	56.2	56.8	54.3	54.9	60.5	56.54		1979	1979	1979	1979	1979	1979.0	
3-2-50-2	81.0	93.0	83.2	76.8	75.9	81.98		1298	1298	1304	1312	1298	1302.0	

Table A.4: Instances Parameter Tuning Results - Score Adjustment Factor σ_2

Fixed parameter setting: κ **= 0.1,** p **= 1,** T_{start} **= 200,** c **= 0.9,** r **= 0.2,** σ_1 **= 30**														
Tuning Instances	**Run Time**					**Avg. Time**	**Total Avg. Time**	**Cost**					**Avg. Cost**	**Total Avg. Cost**
	1	**2**	**3**	**4**	**5**			**1**	**2**	**3**	**4**	**5**		
Parameter setting: σ_2 **= 5**														
2-1-10-1	1.0	1.1	1.2	1.0	1.0	1.06	27.83	294	357	357	357	357	344.4	**1048.48**
2-1-10-2	1.1	1.2	1.2	1.5	1.7	1.34		389	389	389	389	389	389.0	
2-2-30-1	17.9	12.7	13.8	14.7	15.0	14.82		1281.5	1483.0	1404.0	1430.0	1521.5	1424.0	
2-2-30-2	10.8	10.2	12.3	11.7	11.8	11.36		1084.0	1093.0	988.5	1108.5	996.0	1054.0	
3-2-50-1	62.5	61.9	81.2	49.8	61.8	63.44		1979.0	1979.0	1476.0	1868.5	1569.5	1774.4	
3-2-50-2	76.0	73.9	79.1	74.3	71.4	74.94		1326.0	1326.0	1290.0	1293.0	1290.5	1305.1	
Parameter setting: σ_2 **= 10**														
2-1-10-1	1.6	1.4	1.4	1.2	0.8	1.28	**25.67**	357	294	391.5	357	357	351.3	1067.48
2-1-10-2	1.5	1.3	1.1	0.8	0.9	1.12		389	389	389	389	389	389.0	
2-2-30-1	12.8	12.9	12.6	12.4	12.1	12.56		1469	1492	1420	1410	1529	1464.0	
2-2-30-2	9.9	9.4	9.0	10.6	10.8	9.94		1009.0	1059.5	1017.5	1114.0	1017.5	1043.5	
3-2-50-1	56.6	56.6	55.4	83.0	55.2	61.36		1979.0	1979.0	1979.0	1372.0	1979.0	1857.6	
3-2-50-2	68.4	52.6	63.7	73.4	80.6	67.74		1302.0	1292.5	1298.0	1307.0	1298.0	1299.5	
Parameter setting: σ_2 **= 15**														
2-1-10-1	0.8	0.8	0.8	0.8	0.7	0.78	26.19	357	357	392.5	357	391.5	371.0	1072.67
2-1-10-2	0.9	0.9	0.9	0.9	0.9	0.90		400.5	389.0	389.0	389.0	389.0	391.3	
2-2-30-1	13.2	16.8	18.1	20.0	20.2	17.66		1261.0	1496.5	1308.5	1437.0	1313.0	1363.2	
2-2-30-2	8.4	9.0	8.5	9.4	8.4	8.74		1067.5	995.0	1052.5	1009.0	1037.5	1032.3	
3-2-50-1	56.8	55.9	54.1	55.6	55.9	55.66		1979.0	1979.0	1979.0	1979.0	1979.0	1979.0	
3-2-50-2	86.9	79.4	69.3	71.2	60.3	73.42		1290.5	1298.0	1302.0	1298.0	1307.5	1299.2	
Parameter setting: σ_2 **= 20**														
2-1-10-1	0.8	1.0	0.9	0.8	1.0	0.90	28.7	357	357	357	357	357	357.0	1110.27
2-1-10-2	1.0	0.9	0.8	0.9	1.0	0.92		389	389	389	389	389	389.0	
2-2-30-1	12.5	12.0	18.9	15.1	12.1	14.12		1875.0	1528.5	1483.0	1483.0	1518.0	1577.5	
2-2-30-2	8.7	9.2	8.9	9.3	9.5	9.12		1126.5	1022.0	1023.0	1126.5	1025.0	1064.6	
3-2-50-1	57.7	54.9	55.2	54.3	56.4	55.70		1979.0	1979.0	1979.0	1979.0	1979.0	1979.0	
3-2-50-2	79.9	81.7	85.2	105.9	104.4	91.42		1294.0	1299.0	1298.0	1291.0	1290.5	1294.5	

Table A.5: Instances Parameter Tuning Results - Initial Temperature T_{start}

Fixed parameter setting: κ **= 0.1,** p **= 1,** c **= 0.9,** r **= 0.2,** σ_1 **= 30,** σ_2 **= 5**														
Tuning Instances	**Run Time**					**Avg. Time**	**Total Avg. Time**	**Cost**					**Avg. Cost**	**Total Avg. Cost**
	1	**2**	**3**	**4**	**5**			**1**	**2**	**3**	**4**	**5**		
Parameter setting: T_{start} **= 100**														
2-1-10-1	2.0	1.8	1.4	1.1	1.0	1.46	**27.93**	357	357	357	357	357	357.0	1073.97
2-1-10-2	1.1	1.0	1.1	1.2	1.4	1.16		389	389	389	389	400.5	391.3	
2-2-30-1	20.9	14.1	17.4	16.4	14.9	16.74		1134.5	1530.5	1545.5	1264	1368.5	1368.6	
2-2-30-2	14.6	9.3	9.3	7.3	7.6	9.62		1061	1003.5	1001.5	1066.5	1032.5	1033.0	
3-2-50-1	64.6	58.2	62.1	61.7	61.9	61.70		1979	1979	1979	1979	1979	1979.0	
3-2-50-2	100.0	86.3	61.6	68.2	68.4	76.90		1364.5	1304.0	1304.0	1298	1304	1314.9	
Parameter setting: T_{start} **= 200**														
2-1-10-1	1.1	1.1	1.2	1.3	1.3	1.20	28.20	357	357	357	357	357	357.0	1081.87
2-1-10-2	1.7	1.3	1.2	1.1	0.7	1.20		400.5	389	389	389	400.5	393.6	
2-2-30-1	12.2	12.2	11.7	12.3	13.4	12.36		1340.5	1502.5	1522.5	1152	1483	1400.1	
2-2-30-2	10.8	10.5	10.0	10.6	10.4	10.46		1082	1030.5	1078	1108.5	1013.5	1062.5	
3-2-50-1	61.9	61.4	62.4	61.8	63.1	62.12		1979	1979	1979	1979	1979	1979.0	
3-2-50-2	85.8	82.3	79.8	80.3	81.0	81.84		1293.5	1304	1295.5	1298	1304	1299.0	
Parameter setting: T_{start} **= 300**														
2-1-10-1	0.8	0.8	0.8	1.0	1.0	0.88	30.18	357	357	391.5	357	357	363.9	**1042.35**
2-1-10-2	0.7	0.8	0.8	0.9	1.0	0.84		400.5	389	389	389	389	391.3	
2-2-30-1	16.4	21.3	20.0	19.0	16.9	18.72		1332.5	1146	1170.5	1279.5	1144.5	1214.6	
2-2-30-2	10.3	9.8	10.8	9.8	10.5	10.24		1107.5	1091	1054.5	1052.5	1068	1074.7	
3-2-50-1	63.6	94.4	61.8	59.4	58.8	67.60		1979	1635	1979	1979	1979	1910.2	
3-2-50-2	77.0	80.7	88.9	81.6	85.9	82.82		1298	1299	1298	1304	1298	1299.4	
Parameter setting: T_{start} **= 400**														
2-1-10-1	0.9	0.8	0.8	0.9	0.8	0.84	28.89	357	357	357	357	357	357.0	1081.47
2-1-10-2	0.9	0.9	1.0	1.2	0.9	0.98		389	389	389	389	389	389.0	
2-2-30-1	14.7	14.3	14.6	11.9	15.3	14.16		1875	1328.5	1471	1492	1459	1525.1	
2-2-30-2	9.4	10.8	12.7	13.0	12.6	11.70		1055.5	1017.5	990.5	1029	1067	1031.9	
3-2-50-1	60.0	70.9	64.5	57.6	57.3	62.06		1979	1741.5	1719.5	1979	1979	1879.6	
3-2-50-2	89.3	100.7	73.5	80.7	73.8	83.60		1334	1294	1298	1298	1307	1306.2	

Table A.6: Instances Parameter Tuning Results - Cooling Rate c

Fixed parameter setting: κ **= 0.1,** p **= 1,** T_{start} **= 300,** r **= 0.2,** σ_1 **= 30,** σ_2 **= 5**														
Tuning Instances	**Run Time**					**Avg. Time**	**Total Avg. Time**	**Cost**					**Avg. Cost**	**Total Avg. Cost**
	1	**2**	**3**	**4**	**5**			**1**	**2**	**3**	**4**	**5**		
Parameter setting: c **= 0.8**														
2-1-10-1	1.4	1.2	1.2	1.5	1.4	1.34	27.21	357	357	357	357	357	357.0	1074.72
2-1-10-2	1.4	1.3	1.0	1.0	1.0	1.14		389	389	389	389	389	389.0	
2-2-30-1	15.7	9.6	12.1	12.6	8.3	11.66		1734	1205	1692.5	1483	1475.5	1518.0	
2-2-30-2	14.6	11.3	10.1	10.7	12.5	11.84		1083.5	1001.5	1050.5	1066.5	1023.0	1045.0	
3-2-50-1	70.8	70.0	60.8	56.9	65.8	64.86		1786.5	1848.5	1979	1979	1620.0	1842.6	
3-2-50-2	68.1	66.5	68.5	83.3	75.6	72.4		1287.5	1304	1296	1298	1298	1296.7	
Parameter setting: c **= 0.85**														
2-1-10-1	1.4	1.6	1.2	0.9	1.1	1.24	26.98	294	357	357	357	357	344.4	1083.68
2-1-10-2	0.9	1.0	1.1	1.1	1.0	1.02		389	389	389	389	389	389.0	
2-2-30-1	12.0	12.3	11.7	12.1	13.3	12.28		1424.5	1529	1439	1334.5	1481	1441.6	
2-2-30-2	10.5	10.9	9.7	10.2	9.5	10.16		1076.5	1068	1022	1017.5	1033.5	1043.5	
3-2-50-1	56.8	58.3	55.8	57.2	58.5	57.32		1979	1979	1979	1979	1979	1979.0	
3-2-50-2	79.6	83.6	81.2	79.4	75.4	79.84		1326	1297	1304	1298	1298	1304.6	
Parameter setting: c **= 0.9**														
2-1-10-1	1.0	1.0	1.0	1.0	1.0	1.00	**26.43**	357	357	357	357	357	357.0	1073.1
2-1-10-2	0.8	0.9	0.9	0.8	0.9	0.86		389	389	389	389	389	389.0	
2-2-30-1	14.6	13.2	13.3	13.7	13.2	13.60		1252.5	1185.5	1641.5	1393.5	1304.5	1355.5	
2-2-30-2	11.0	10.2	9.7	8.8	8.8	9.70		1070	1001.5	1126.5	1052.5	1041.5	1058.4	
3-2-50-1	57.7	56.5	56.9	56.6	56.7	56.88		1979	1979	1979	1979	1979	1979.0	
3-2-50-2	82.7	88.8	74.0	71.2	66.0	76.54		1298	1295.5	1305.5	1295.5	1304	1299.7	
Parameter setting: c **= 0.95**														
2-1-10-1	1.0	0.9	0.9	0.9	1.0	0.94	30.42	391.5	357	357	357	357	363.9	**1056.3**
2-1-10-2	0.8	0.8	0.7	0.7	0.7	0.74		389	389	389	464	389	404.0	
2-2-30-1	16.2	15.3	15.4	13.8	12.9	14.72		1362.5	989.5	1244.5	1461	1483	1308.1	
2-2-30-2	14.0	15.3	14.1	16.8	15.0	15.04		1001.5	1036	1013.5	1001.5	1024.5	1015.4	
3-2-50-1	57.4	58.2	53.9	56.4	60.5	57.28		1979	1979	1979	1979	1779	1939.0	
3-2-50-2	93.8	87.8	94.5	101.3	91.5	93.78		1402	1318.5	1261.5	1287	1268	1307.4	

B

Sensitivity Analysis Results - Capacity and Cost

Table B.1: Sensitivity Analysis - Capacity and Cost - 50% Capacity

50% Capacity									
Instances	**30% Cost**	**40% Cost**	**Gap - Cost=40%**	**50% Cost**	**Gap - Cost=50%**	**60% Cost**	**Gap - Cost=60%**	**70% Cost**	**Gap - Cost=70%**
2.2.30.1	1026	1060.6	3.37%	1094	6.63%	1035.6	0.94%	1470.6	43.33%
2.2.30.2	695.7	836.6	20.25%	950	36.55%	922.6	32.61%	1080.5	55.31%
2.2.30.3	814.1	1214.2	49.15%	948.5	16.51%	1088.2	33.67%	1051.4	29.15%
2.2.30.4	819.5	879	7.26%	946	15.44%	1029	25.56%	1139.5	39.05%
2.2.30.5	872.3	974.4	11.70%	1076.5	23.41%	1146.2	31.40%	1136.8	30.32%
2.2.30.6	904.4	817	-9.66%	961.5	6.31%	1067.4	18.02%	968.9	7.13%
2.2.30.7	884.9	1036.4	17.12%	1078	21.82%	1157	30.75%	1276.7	44.28%
2.2.30.8	1108.1	938.8	-15.28%	1227.5	10.78%	1102.2	-0.53%	1057.3	-4.58%
2.2.30.9	666.9	708.2	6.19%	893.5	33.98%	877.4	31.56%	832.1	24.77%
2.2.30.10	791.1	810.6	2.46%	1033.5	30.64%	766.2	-3.15%	1109.4	40.24%
3.2.50.1	1550.2	1727.8	11.46%	1688	8.89%	1764	13.79%	2002.4	29.17%
3.2.50.2	1535.6	1713.2	11.57%	1741.5	13.41%	2114.4	37.69%	1881.6	22.53%
3.2.50.3	1555.1	1759.8	13.16%	2008.5	29.16%	1895.2	21.87%	2076.4	33.52%
3.2.50.4	1828.1	1861.4	1.82%	2255	23.35%	2378.8	30.12%	2022.2	10.62%
3.2.50.5	1975	2119	7.29%	2263	14.58%	2407	21.87%	2551	29.16%
3.2.50.6	1257.6	1482.2	17.86%	1671	32.87%	1909	51.80%	2326.5	85.00%
3.2.50.7	1834.7	1977.6	7.79%	2120.5	15.58%	2263.4	23.37%	2406.3	31.15%
3.2.50.8	1622.2	1756.6	8.29%	1501	-7.47%	1578.8	-2.68%	2159.8	33.14%
3.2.50.9	1550.5	1679	8.29%	1807.5	16.58%	1936	24.86%	2064.5	33.15%
3.2.50.10	1340.5	1479	10.33%	1617.5	20.66%	1328	-0.93%	1894.5	41.33%

Table B.2: Sensitivity Analysis - Capacity and Cost - 60% Capacity

60% Capacity									
Instances	**40% Cost**	**50% Cost**	**Gap - Cost=50%**	**60% Cost**	**Gap - Cost=60%**	**70% Cost**	**Gap - Cost=70%**	**80% Cost**	**Gap - Cost=80%**
2.2.30.1	1086.8	1356	24.77%	1219.6	12.22%	1438.1	32.32%	1229	13.08%
2.2.30.2	866	760.5	-12.18%	913.4	5.47%	1121.4	29.49%	1153	33.14%
2.2.30.3	1155.6	1137.5	-1.57%	1150.2	-0.47%	1145	-0.92%	1135.2	-1.77%
2.2.30.4	892.6	936	4.86%	994.6	11.43%	1032.8	15.71%	1073.4	20.26%
2.2.30.5	905	915	1.10%	1100.8	21.64%	1228.3	35.72%	1312.4	45.02%
2.2.30.6	859	775	-9.78%	1025.4	19.37%	983.5	14.49%	1144.8	33.27%
2.2.30.7	796.4	1042	30.84%	1256	57.71%	1187.8	49.15%	1300.6	63.31%
2.2.30.8	883.2	1038	17.53%	1255.8	42.19%	1071.6	21.33%	996.8	12.86%
2.2.30.9	859.2	949.5	10.51%	1062.8	23.70%	1144	33.15%	1195	39.08%
2.2.30.10	945.8	1031	9.01%	1099.4	16.24%	1140	20.53%	1152.8	21.89%
3.2.50.1	1700.6	1904.5	11.99%	1751.4	2.99%	1956	15.02%	2034.4	19.63%
3.2.50.2	1472	1576.5	7.10%	1794	21.88%	1734.3	17.82%	1967.2	33.64%
3.2.50.3	1691	1801.5	6.53%	1826	7.98%	2105.2	24.49%	2015	19.16%
3.2.50.4	1793.6	1786.5	-0.40%	1791.8	-0.10%	1980.8	10.44%	2149.8	19.86%
3.2.50.5	2119	2263	6.80%	2407	13.59%	2551	20.39%	2695	27.18%
3.2.50.6	1370.2	1465	6.92%	1310.6	-4.35%	1416	3.34%	1672.6	22.07%
3.2.50.7	1977.6	2120.5	7.23%	2263.4	14.45%	2406.3	21.68%	2136.8	8.05%
3.2.50.8	1674.6	1433.5	-14.40%	1949.4	16.41%	1614.6	-3.58%	2224.2	32.82%
3.2.50.9	1645.8	1777.5	8.00%	1874.6	13.90%	2040.9	24.01%	2172.6	32.01%
3.2.50.10	1494.2	1636.5	9.52%	1778.8	19.05%	1921.1	28.57%	2063.4	38.09%

Table B.3: Sensitivity Analysis - Capacity and Cost - 70% Capacity

70% Capacity									
Instances	**50% Cost**	**60% Cost**	**Gap - Cost=60%**	**70% Cost**	**Gap - Cost=70%**	**80% Cost**	**Gap - Cost=80%**	**90% Cost**	**Gap - Cost=90%**
2.2.30.1	884	1274.2	44.14%	1328.5	50.28%	1141	29.07%	1404.8	58.91%
2.2.30.2	856	913.4	6.71%	1025	19.74%	1155.4	34.98%	1313	53.39%
2.2.30.3	1126.5	1177.2	4.50%	1182.4	4.96%	1188.8	5.53%	1178.2	4.59%
2.2.30.4	928.5	995.6	7.23%	1048.8	12.96%	1280.6	37.92%	1165.9	25.57%
2.2.30.5	826	1073.2	29.93%	1068.7	29.38%	1148.6	39.06%	1336.8	61.84%
2.2.30.6	959	1001.2	4.40%	1051.6	9.66%	1172.2	22.23%	1265.6	31.97%
2.2.30.7	1031	1094.2	6.13%	1189.7	15.39%	1264.6	22.66%	1340.2	29.99%
2.2.30.8	1033.5	1166.2	12.84%	1211.7	17.24%	1257	21.63%	1189.7	15.11%
2.2.30.9	965	1034.8	7.23%	1037.6	7.52%	1190.8	23.40%	1228.8	27.34%
2.2.30.10	1040.5	1049.4	0.86%	1113.4	7.01%	1252.2	20.35%	1279.8	23.00%
3.2.50.1	1357	1670	23.07%	1822.4	34.30%	2008.8	48.03%	1945.7	43.38%
3.2.50.2	1528	1719.8	12.55%	1889.1	23.63%	1774.6	16.14%	1859.3	21.68%
3.2.50.3	1838.5	1779.2	-3.23%	2121.3	15.38%	2098	14.11%	2072.7	12.74%
3.2.50.4	1488	1866.8	25.46%	2054.6	38.08%	2035.6	36.80%	2335.7	56.97%
3.2.50.5	1791	1877.2	4.81%	1715.7	-4.20%	2073.8	15.79%	2284.5	27.55%
3.2.50.6	1255	1438	14.58%	1449.7	15.51%	1500.2	19.54%	1772.9	41.27%
3.2.50.7	2135	2285.4	7.04%	2435.8	14.09%	2586.2	21.13%	2736.6	28.18%
3.2.50.8	1411.5	1438	1.88%	2111.4	49.59%	2252.6	59.59%	1830.7	29.70%
3.2.50.9	1329.5	1447.4	8.87%	1610.3	21.12%	1705	28.24%	1604.2	20.66%
3.2.50.10	1328	1593	19.95%	1844.1	38.86%	1986.4	49.58%	2128.7	60.29%

Table B.4: Sensitivity Analysis - Capacity and Cost - 80% Capacity

80% Capacity									
Instances	**50% Cost**	**60% Cost**	**Gap - Cost=60%**	**70% Cost**	**Gap - Cost=70%**	**80% Cost**	**Gap - Cost=80%**	**90% Cost**	**Gap - Cost=90%**
2.2.30.1	1142.5	1143.6	0.10%	1248.6	9.29%	1368.8	19.81%	1572.1	37.60%
2.2.30.2	856	920	7.48%	1011.6	18.18%	1129.8	31.99%	1205.3	40.81%
2.2.30.3	1171	1163.4	-0.65%	1240.2	5.91%	1227.2	4.80%	1234.1	5.39%
2.2.30.4	809.5	983.6	21.51%	1034.8	27.83%	934.8	15.48%	1082.9	33.77%
2.2.30.5	971	1037.4	6.84%	1132.1	16.59%	1092.8	12.54%	1320.3	35.97%
2.2.30.6	922.5	951.2	3.11%	1084.1	17.52%	1049.8	13.80%	1151.5	24.82%
2.2.30.7	968	935	-3.41%	988.5	2.12%	1173.2	21.20%	1279.6	32.19%
2.2.30.8	1116.5	1227.6	9.95%	1293.3	15.84%	1215.6	8.88%	1514.4	35.64%
2.2.30.9	1047.5	1060.4	1.23%	1200.5	14.61%	1236.8	18.07%	1246.1	18.96%
2.2.30.10	1011	1026.6	1.54%	1208.4	19.53%	1236.2	22.27%	1246.8	23.32%
3.2.50.1	1349.5	1815.4	34.52%	1810.5	34.16%	1593.2	18.06%	1997.9	48.05%
3.2.50.2	1550	1628.6	5.07%	1743.4	12.48%	1818	17.29%	1979.4	27.70%
3.2.50.3	1889	1916	1.43%	2121.3	12.30%	2300.4	21.78%	2054.3	8.75%
3.2.50.4	1793.5	1871.4	4.34%	2011.8	12.17%	2034.4	13.43%	2227.5	24.20%
3.2.50.5	1973	1948.8	-1.23%	1948.4	-1.25%	2084	5.63%	2280.5	15.59%
3.2.50.6	1286	1324.6	3.00%	1753.4	36.35%	1712.8	33.19%	1935.5	50.51%
3.2.50.7	1647.5	2063.6	25.26%	2145.3	30.22%	2005.2	21.71%	2371	43.92%
3.2.50.8	1723.5	1524	-11.58%	1593.3	-7.55%	1652	-4.15%	2389.4	38.64%
3.2.50.9	1475.5	1715.6	16.27%	1675.2	13.53%	1793.6	21.56%	1879	27.35%
3.2.50.10	1299.5	1335	2.73%	1510	16.20%	1546	18.97%	1811.2	39.38%

Table B.5: Sensitivity Analysis - Capacity and Cost - 90% Capacity

90% Capacity									
Instances	**50% Cost**	**60% Cost**	**Gap - Cost=60%**	**70% Cost**	**Gap - Cost=70%**	**80% Cost**	**Gap - Cost=80%**	**90% Cost**	**Gap - Cost=90%**
2.2.30.1	1109	813.2	-26.67%	1031.3	-7.01%	1528.6	37.84%	1634.6	47.39%
2.2.30.2	857	959	11.90%	1079.9	26.01%	1081.6	26.21%	1218.1	42.14%
2.2.30.3	1208.5	1148.6	-4.96%	1140.9	-5.59%	1277.8	5.73%	1293.8	7.06%
2.2.30.4	907	962.2	6.09%	1041.2	14.80%	934.8	3.07%	1154.9	27.33%
2.2.30.5	963.5	1020.8	5.95%	1102.4	14.42%	1215.4	26.14%	1321.5	37.16%
2.2.30.6	894.5	964.8	7.86%	1046.7	17.02%	1229	37.40%	1249.4	39.68%
2.2.30.7	1035.5	1066.8	3.02%	1233.4	19.11%	1275	23.13%	1306.8	26.20%
2.2.30.8	1079	1070.2	-0.82%	1291.4	19.68%	1251.8	16.01%	1424.4	32.01%
2.2.30.9	954.5	994	4.14%	1071.2	12.23%	1243.4	30.27%	1289.8	35.13%
2.2.30.10	1024.5	1064.8	3.93%	1163.7	13.59%	1279.6	24.90%	1253.2	22.32%
3.2.50.1	1612	1655.8	2.72%	1659	2.92%	1879.6	16.60%	1954.3	21.23%
3.2.50.2	1507	1593.6	5.75%	1724.3	14.42%	1798	19.31%	1955.9	29.79%
3.2.50.3	1836	1929.4	5.09%	2024.2	10.25%	2162	17.76%	2388.3	30.08%
3.2.50.4	1723.5	1761.2	2.19%	2011.2	16.69%	2139.6	24.14%	2042.4	18.50%
3.2.50.5	1847.5	1867.6	1.09%	2270.6	22.90%	1731	-6.31%	2140.7	15.87%
3.2.50.6	1354	1637.2	20.92%	1286.8	-4.96%	1324.6	-2.17%	1542.8	13.94%
3.2.50.7	1395.5	1500.4	7.52%	1514.3	8.51%	1744	24.97%	1854.7	32.91%
3.2.50.8	1523	1492	-2.04%	1543.4	1.34%	1827.2	19.97%	1885.1	23.78%
3.2.50.9	1532.5	1504.4	-1.83%	1665.8	8.70%	2375.4	55.00%	1854.4	21.00%
3.2.50.10	1179	1199.4	1.73%	1359	15.27%	1352.8	14.74%	1727.8	46.55%

Table B.6: Sensitivity Analysis - Capacity and Cost - 100% Capacity

100% Capacity									
Instances	**50% Cost**	**60% Cost**	**Gap - Cost=60%**	**70% Cost**	**Gap - Cost=70%**	**80% Cost**	**Gap - Cost=80%**	**90% Cost**	**Gap - Cost=90%**
2.2.30.1	1131	1272.8	12.54%	1277.9	12.99%	1436.6	27.02%	1335.8	18.11%
2.2.30.2	866	919.4	6.17%	995.2	14.92%	1051	21.36%	1271.4	46.81%
2.2.30.3	1049	1310.4	24.92%	1188.7	13.32%	1180.8	12.56%	1279.8	22.00%
2.2.30.4	732.5	962.2	31.36%	895	22.18%	1098.2	49.92%	949.8	29.67%
2.2.30.5	947.5	937.2	-1.09%	1193.8	25.99%	1162.8	22.72%	1245.8	31.48%
2.2.30.6	879.5	956.6	8.77%	1118.2	27.14%	1116	26.89%	1165.9	32.56%
2.2.30.7	968	1042.8	7.73%	1108.6	14.52%	1181.8	22.09%	1226.6	26.71%
2.2.30.8	1231	1127.2	-8.43%	1290.7	4.85%	1353.2	9.93%	1530.9	24.36%
2.2.30.9	945.5	1029.8	8.92%	1032.6	9.21%	1088.6	15.13%	1193.2	26.20%
2.2.30.10	969	1092.4	12.73%	1137.8	17.42%	1242.6	28.24%	1245	28.48%
3.2.50.1	1523.5	1740	14.21%	1786	17.23%	2006.8	31.72%	1910.1	25.38%
3.2.50.2	1526	1582.4	3.70%	1856.7	21.67%	1990.8	30.46%	2029.2	32.98%
3.2.50.3	2054.5	1917.2	-6.68%	1875.1	-8.73%	2048.8	-0.28%	2365.8	15.15%
3.2.50.4	1601.5	1878.6	17.30%	2057.8	28.49%	2080.4	29.90%	2267.5	41.59%
3.2.50.5	1880.5	1925.8	2.41%	2014.5	7.13%	2529.2	34.50%	2108.7	12.14%
3.2.50.6	1374.5	1323.8	-3.69%	1557.4	13.31%	1510	9.86%	2400.9	74.67%
3.2.50.7	1398	1512.4	8.18%	1715.2	22.69%	1733.6	24.01%	1858.5	32.94%
3.2.50.8	1345.5	1643.6	22.16%	1645.7	22.31%	1603.2	19.15%	1646.8	22.39%
3.2.50.9	1498.5	2024	35.07%	2178.5	45.38%	1877.8	25.31%	1852.3	23.61%
3.2.50.10	1088.5	1120.8	2.97%	1344.1	23.48%	1365.4	25.44%	1492.7	37.13%

C

Scientific Paper

The scientific paper starts on the next page.

Truck Routing for Parcel Delivery:

Solving a Multi-depot Pickup and Delivery Problem with Occasional Drivers using ALNS

S. Wang, Dr. S. Fazi, Dr. A. Bombelli, Prof.dr.ir. L.A. Tavasszy

***Abstract*— This research presents a mathematical model for routing that incorporates multiple depots, occasional drivers, and multiple depot visits, a problem faced by many companies in reality. An Adaptive Large Neighborhood Search (ALNS) algorithm was employed, using Random and Worst removal operators, Basic Greedy and Regret-2 insertion strategies, a roulette wheel for operator selection, and simulated annealing for acceptance criteria. Computational experiments validated the effectiveness of the ALNS algorithm and model. Sensitivity analysis revealed that adding depots (ODs) can reduce routing costs by up to 15.21%, with various OD capacities offering additional savings (7.86% for 60% capacity and 9.97% for 70% capacity). A case study with the Dutch e-commerce company Ochama, scaled to 150 requests, confirmed practical applicability, achieving cost reductions of 11.37% for small vehicles and 6.85% for medium-sized vehicles. The results highlight that integrating ODs into routing strategies can significantly lower costs, with optimal outcomes dependent on market research to balance savings, service quality, and OD capacity.**

I. INTRODUCTION

In recent years, the exponential growth of e-commerce, accelerated by the Covid-19 pandemic, has significantly increased urban parcel deliveries, creating major logistical challenges for last-mile delivery (LMD) in cities, particularly in densely populated areas. For instance, Amsterdam is expected to handle over 100,000 parcels daily by 2030, putting immense pressure on the logistics system, exacerbating traffic congestion, environmental pollution, and driving up logistics costs, with last-mile delivery accounting for 75% of these costs. Traditional LMD strategies often involve increasing vehicle numbers or capacity, which may reduce efficiency and raise costs. To address these issues, crowdshipping (CS) has emerged as a viable alternative, leveraging occasional drivers (ODs) to complete deliveries on detours at lower costs. This study proposes a new variant of the Pickup and Delivery Problem with Time Windows (PDPTW), integrating multiple depots, regular drivers (RDs), ODs, and intermediate centers (stores) to optimize the total vehicle routing cost. The model, referred to as the Multi-trip Crowd-shipping Split Pickup and Delivery Problem with Time Window and Multi-depots (MC-PDPTW-MD), aims to incorporate these features into a traditional PDPTW model, focusing on minimizing routing costs for both RDs and ODs. The key research question is how to optimize the total vehicle routing cost in the MC-PDPTW-MD problem.

II. LITERATURE STUDY

The research theme of this research is the combination of multiple aspects of study, including the Crowd-shipping Problem, the Open Vehicle Routing Problem (Open VRP), and the Pickup and Delivery Problem with Time Window (PDPTW). The mathematical models involved in these related studies and the algorithms used to solve these problems will be discussed in detail in this section.

A. Related Models

This part will discuss the origin and development of three related problems, discuss in depth the characteristics of the initial model, and different features that are continuously added to the model during the development process to make the model suitable for specific research problems. These features include: Number of depots, RD quantity limit, delivery strategy (single/multiple/split), intermediate center, reverse logistics, etc.

1) Crowd-shipping Problem: Most research in crowdsourcing has focused on virtual tasks like text editing, translation, and debugging, which can be completed remotely via the Internet [11]. However, with the rise of e-commerce, crowdshipping has gained significant attention as a delivery method in the sharing economy. Platforms such as Walmart's Spark Delivery and Amazon Flex have popularized crowdsourced delivery, where occasional drivers assist in parcel delivery while earning compensation [7], [18]. This study focuses on courier companies, which face diverse delivery challenges, including parcel lockers, drones, and crowdshipping, areas still rich for research. The operational logic of crowdsourced delivery platforms, governed by algorithms, can be divided into static and dynamic models. While dynamic models adapt to real-time changes, static models assume fixed vehicles and demands [2]. This study addresses a static crowdshipping problem, integrating elements from traditional delivery models. Building on Archetti et al. [1] and Macrina et al. [23], it considers intermediate centers and scenarios where occasional drivers (ODs) and regular drivers (RDs) work together, allowing for multiple and split deliveries to enhance reliability when customer demand exceeds capacity [20].

2) Open Vehicle Routing Problem: In 2000, Sariklis & Powell introduced the Open Vehicle Routing Problem (OVRP), where vehicles do not need to return to the depot or may revisit customers in reverse order [35]. In OVRP, a depot with a limited regular truck fleet may subcontract delivery tasks to external couriers, whose higher per-unit driving costs are offset by savings in capital, maintenance, and depreciation [41]. OVRP typically assigns rented vehicles to routes without return to the depot and often involves reverse logistics, where vehicles deliver goods and then return to the depot in reverse order to collect returns. This model

is similar to Archetti et al.'s research [1] but focuses on rented fleets and vehicle limitations rather than occasional drivers. Tarantilis & Kiranoudis [40] expanded OVRP by incorporating multi-depots, applied in real-world cases like fresh meat distribution in Greece, while Brandão [4] and Fleszar et al. [13] added operational time restrictions, and Repoussis et al. [31] introduced customer time windows. Fu et al. [14] further diversified OVRP by splitting it into delivery-only, pickup-only, and combined delivery and pickup models, allowing multiple vehicles to serve the same demand point, thus improving route options, vehicle capacity utilization, and reducing vehicle needs. Combining OVRP with crowdshipping, as explored by Torres et al. [43], demonstrated cost reductions by using crowdshippers instead of third-party logistics vehicles. Our research builds on this by integrating various OVRP variants, incorporating time window constraints for regular drivers (RDs), occasional drivers (ODs), intermediate centers, and customers.

3) Pickup and Delivery Problem with Time Window: Since the 1990s, the Pickup and Delivery Problem with Time Window (PDPTW) has made significant strides in logistics and transportation planning. A special case of PDPTW is the Vehicle Routing Problem with Time Window (VRPTW), where all destinations are depots [44]. The problem was extended in 1989 to include simultaneous pickup and delivery, marking an early form of the PDP problems [12]. PDPTW involves creating optimal routes that satisfy transportation requests under capacity, time window, and priority constraints, with the additional requirement that the same vehicle handles both pickup and delivery. To address challenges such as rising energy costs, driver shortages, and vehicle capacity utilization, Nowak et al. [27] introduced Pickup and Delivery with Split Loads (PDPSL), drawing on the benefits of split delivery strategies from SDVRP. PDPSL is more complex because the vehicle's capacity changes with each pickup or delivery, and vehicles do not return to the depot. While early research focused on single-depot problems, Irnich [19] expanded the scope to a multi-depot setup with a single hub and heterogeneous vehicles, emphasizing the transport of diverse requests between a location and the hub.

In 2019, Dahle et al. [8] combined Crowdshipping with PDPTW to introduce The Pickup and Delivery Problem with Time Windows and Occasional Drivers, where crowdshipping drivers have individual starting locations, destinations, costs, and time matrices. This model showed a 10-15% cost savings by incorporating crowdshipping drivers. Expanding on this, Voigt & Kuhn [45] introduced Transhipment Points (TPs), similar to intermediate centers, where goods are transferred between drivers. However, we propose that treating TPs as intermediate centers near recipients simplifies the process, allowing recipients to pick up goods themselves if direct delivery is inconvenient, thus reducing transportation costs. Considering multi-depot scenarios, Tao et al. [39] discussed The Pickup and Delivery Problem with Multiple Depots and Dynamic Occasional Drivers in Crowdshipping Delivery, a dynamic problem where temporary drivers post their travel plans on a crowdshipping platform for route matching. Given our focus on multiple and split deliveries and intermediate centers, we model our problem as a static one, where temporary drivers select pre-generated routes based on their availability.

B. Related Algorithms

This section will further discuss the algorithms corresponding to the mathematical models discussed above, including both exact algorithms and metaheuristic algorithms. Common exact algorithms include Mixed-Integer Linear Programming (MILP) and Branch and Bound (B&B), etc. Common metaheuristic algorithms include Tabu Search, Genetic Algorithms, Simulated Annealing, Ant Colony Optimization, and Variable Neighborhood Search, among others.

1) Crowd-shipping Problem: Archetti et al. [1] introduced a multi-start heuristic combining variable neighborhood search (VNS) and tabu search, which constructs routes with regular drivers and adjusts them using tabu and VNS mechanisms. This heuristic method outperformed CPLEX in solving instances involving temporary drivers, providing solutions in seconds compared to CPLEX's inability to solve within an hour. Kafle et al. [20] opted for simulated annealing due to its suitability for large instances and faster performance compared to other heuristics like genetic algorithms. More recent approaches, such as Dayarian & Savelsbergh's research [9], have utilized advanced methods like VNS combined with machine learning [30] and hybrid models integrating GRASP with VNS for local search [10], highlighting the growing trend towards using VNS in complex routing problems.

2) Open Vehicle Routing Problem: Sariklis & Powell [35] introduced a heuristic method based on a minimum spanning tree with penalties to solve Open VRP, showing strong performance but with longer computation times for large problems. Building on this, Brandão [4] applied Tabu Search, adding maximum route length constraints, while Fu et al. [14] incorporated vehicle capacity limits. Later, Tarantilis et al. [42] proposed the List-Based Threshold Accepting (LBTA) algorithm, Adaptive Memory-based Tabu Search (BR), and Backtracking Adaptive Threshold Accepting (BATA) for solving Open VRP. In 2007, Pisinger & Ropke [28] used the Adaptive Large Neighborhood Search (ALNS) framework, which systematically modifies solutions in each iteration, to solve Open VRP variants. Fleszar et al. [13] introduced Variable Neighborhood Search (VNS) for Open VRP, exploring the solution space by varying neighborhood sizes, which was further enhanced by Sevkli & Güler [36] using a multi-phase VNS approach. Other notable approaches include algorithms like Particle Swarm Optimization [25], Integer Linear Programming-based heuristics [33], and models addressing fuzzy demands [5] and demand uncertainty [6]. Despite these advancements, ALNS has emerged as particularly effective for complex optimization problems, making it increasingly popular for solving Open VRP.

3) Pickup and Delivery Problem with Time Windows: In 1991, Dumas et al. [12] introduced an exact algorithm

for the Pickup and Delivery Problem (PDP) with time windows, using a column generation method to handle multiple sites and vehicle types, laying the foundation for future PDP solutions. Entering the 21st century, more complex metaheuristics emerged, such as the reactive tabu search by Nanry & Barnes [26] and the combination of simulated annealing and tabu search by Li & Lim [22], which were particularly effective for large-scale PDPs. In 2006, Ropke & Pisinger [32] significantly advanced the field by introducing the Adaptive Large Neighborhood Search (ALNS) heuristic, which employs sub-heuristics for solution improvement, using methods like random and worst removal during destruction phases and greedy insertion during repair phases, with a simulated annealing-based acceptance criterion. This ALNS framework was later expanded to tackle more complex scenarios, as seen in [15] for PDP with scheduled lines and [34] for multi-stop PDP with time windows and transfers.

The literature review section provides a detailed depiction of the evolution and development of models and algorithms used to solve the Crowd-shipping Problem, Open VRP, and PDPTW. The purpose is to filter the features contained in the model we intend to study, such as time windows, occasional drivers, etc., through the discussion and analysis of these models. By combining these features, a new problem (Multi-trip Crowd-shipping Split Pickup and Delivery Problem with Time Window and Multi-depots) is proposed. There is no consistent research on the newly proposed problem, which constitutes what is known as a research gap. At the same time, by analyzing the algorithms used in different models and the trend of algorithm development, a suitable algorithm will be selected and discussed. This study will choose ALNS as the preferred algorithm, with the specifics of the ALNS algorithm discussed in methodology chapter.

III. Problem Definition

In this section the Multi-trip Crowd-shipping Split Pickup and Delivery Problem with Time Window and Multi-depots (MC-PDPTW-MD) is formally defined. After discussion, in addition to the common time window constraints and vehicle capacity constraints, the model to be studied also contains the following characteristics: Regular drivers, Occasional drivers (crowdshippers), multi-depots, intermediate centers, multi-delivery and split delivery.

The MC-PDPTW-MD is an complex logistics optimization challenge, focusing on the interplay between a regular truck fleet and an occasional driver fleet to minimize total routing costs. Figure 1 shows an illustrative example of the multi-trip crowd-shipping split pickup and delivery problem. The operational framework of this problem encompasses a central Distribution Center (DC), various retailers serving as intermediate centers, and customers. At the core of this scenario is the regular truck fleet (RDs), which embarks on its journey from the DC. Each RD is loaded with parcels destined for delivery to intermediate centers and customers. The delivery schedule is intricately planned within specific time windows, ensuring timely and efficient service. While delivering the package, RDs engage in the pickup of return items only from intermediate centers. This aspect of their route is flexible, contingent upon the spare capacity available post-delivery. The RDs' routes culminate back at the DC, where they offload the collected items for further processing. Complementing the regular fleet is the occasional driver (OD) fleet, characterized by its flexibility and adaptability. ODs initiate their routes from varied locations, not bound to the DC as their starting point. OD's primary role is to pick up items from intermediate centers and delivery them back to depot or pickup items from depots and distribute them to customers and intermediate centers. while ODs are delivering, parcels can be picked up at intermediate centers as vehicle capacity allows. ODs head to his/her destination after finishing all delivery tasks. If possible, RD and OD can visit multiple depots multiple times. The essence of the MC-PDPTW-MD lies in the seamless coordination between these two fleets. The system strategically allocates tasks and routes to each fleet, considering real-time dynamics like vehicle capacity, delivery and pickup time windows, and on-the-ground traffic conditions. The objective is a harmonious balance, where the regular fleet's structured routes complement the occasional drivers' flexible pickups, ensuring overall efficiency and cost-effectiveness. The challenge is to manage these split pickups and deliveries within the constraints of time windows, aiming for a solution that minimizes routing costs.

- **Intermediate Center (C):** A intermediate center or a retailer can receive parcels and have some to be sent to depot.
- **Client (S):** Client only receiving parcels at home
- **Distribution Centre (DC):** A distribution center or depot can receive packages from copy centers and store packages for delivery to intermediate centers and customers.
- **Regular Driver (RD):** Regular drivers depart from DC, and end to DC
- **Occasional Driver (OD):** Occasional Drivers start from their origin and have a final destination, they may visit DC multiple times.

IV. Methodology

This chapter presents two approaches to solve the problem: an exact method and a metaheuristic algorithm. We begin by formulating a precise mathematical model for the problem. To solve this model, we employ the branch-and-bound technique, a commonly used exact method. Despite the accuracy of exact methods, they often require substantial computational time to find feasible or optimal solutions. This limitation leads us to consider metaheuristic algorithms as a viable alternative.

Metaheuristic algorithms offer a practical solution when the problem scale increases, making exact methods computationally prohibitive. Previous studies have explored various metaheuristic techniques, including Local Search and Tabu Search. In 2005, Ropke & Pisinger [32] introduced the Adaptive Large Neighborhood Search (ALNS) framework,

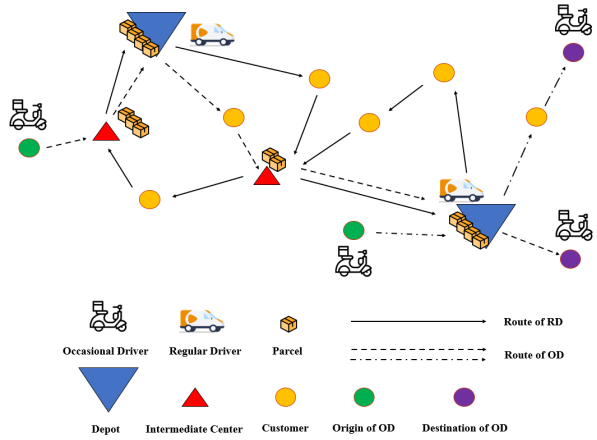

Fig. 1. An illustrative example of multi-trip crowd-shipping split pickup and delivery problem

an enhancement of the Large Neighborhood Search (LNS) algorithm. They successfully applied ALNS to the Pickup and Delivery Problem with Time Windows (PDPTW), demonstrating superior performance compared to standard LNS. Building on this foundation, our research adopts the ALNS algorithm to effectively address the problem. Detailed discussions of the mathematical model and the ALNS implementation are provided in subsequent sections.

A. Mathematical Model

Let N denote the set of all nodes, let Δ denote the set of distribution centers and Δ^{dummy} denote the set of dummy nodes make sure OD can visit depot multiple times. Let S denote the set of customers, O^{od} and D^{od} denote the set of origins and destinations of occasional drivers. Let C denote the set of intermediate centers, usually grocery stores.

N consists of DC, C, client and origin and destinations of OD, thus $N = \Delta \cup \Delta^{dummy} \cup C \cup S \cup O^{od} \cup D^{od}$. A regular driver set RD, and a occasional driver set OD, are available to serve the requests. A set of vehicles K consists of RD and OD, thus $K = RD \cup OD$. For RD, each vehicle $k \in RD$ will start from and back to Depot. For OD, each vehicle $k \in OD$ has an origin $i \in O^{od}$ and a destination $j \in D^{od}$. As shown in Figure 1, all RDs have origins and destinations at the same depot, while the origins and destinations of the ODs may be spread out.

Each vehicle k has a capacity C_k, and there is a cost factor ρ^k for each vehicle k. When transporting goods from node i to node j, the time distance between node i and node j is given as T_{ij}. For each node $i \in N$ there is a time window $[T_i^0, T_i^1]$ within which it must be serviced. The delivery and pickup quantity at node $i \in N/\Delta$ are D_i and P_i respectively. Regular vehicle k has a time window for its origin depot $[T_0^0, T_0^1]$. For the ODs it is assumed that the time window are wide enough to allow for a direct travel.

The variable x_{ij}^k is a binary variable denoting if vehicle $k \in K$ uses arc (i, j) when $i, j \in N$. While variable y_{ij}^{kse} is pickup and delivery flow variable denotes load on vehicle $k \in K$ for pickup or delivery request from node s to node e on arc (i, j), $ij \in N$. Variable z_i^{kse} denotes pickup or delivery quantity at node i for vehicle $k \in K$ and request from start point s to end point e, $se \in N$. Variable dd_{ij}^k and pd_{ij}^k are decision variables for delivery and pickup requests respectively denoting if vehicle $k \in K$ serve request from start node i and end node j. For variable dd_{ij}^k, $n \in \Delta, i \in \Delta_n, j \in DV_n$. For variable pd_{ij}^k, $n \in \Delta, j \in \Delta_n, i \in PV_n$. t_i^k means the time of vehicle $k \in K$ arrive at node $i \in N$.

TABLE I
SET UP OF SETS, PARAMETERS AND VARIABLES

Sets	
N	Set of all nodes
Δ	Set of depots
Δ^{dummy}	Set of dummy depots
Δ_n	Set of depot and its dummy depot for depot n
S	Set of customers
C	Set of Intermediate centers
DV	Set of delivery requests
PV	Set of pickup requests
DV_n	Set of delivery requests of depot n
PV_n	Set of pickup requests of depot n
K	Set of vehicles
K_n	Regular vehicle set belongs to depot n
RD	Set of regular drivers
OD	Set of occasional drivers
O^{OD}	Set of origins occasional drivers
D^{OD}	Set of destinations occasional drivers
Parameters	
T_{ij}	Time distance between nodes i and j
DD_{ij}	Pickup and delivery quantity between nodes i and j
C_k	Capacity vehicle k
V_i	Depot that provides service for request i
T_i^0, T_i^1	Time windows of node i
ρ^k	Cost factor of vehicle k
M	Big value
Variables	
x_{ij}^k	Routing variable
y_{ij}^{kse}	Pickup and delivery flow variable
z_i^{kse}	Pickup and delivery quantity variable
dd_{ij}^k	Decision variable for delivery requests
pd_{ij}^k	Decision variable for pickup requests
t_i^k	Time of vehicle k at node i

Table I provide an overview about all the mentioned sets, parameters and variables.

The formulate of the problem is given as follows:

Objective Function:

$$\min \sum_{k \in K} \sum_{(ij) \in N} x_{ij}^k T_{ij} \rho^k \tag{1}$$

Constraints for all vehicles:

$$\sum_{j \in N} x_{ij}^k - \sum_{j \in N} x_{ji}^k = 0 \qquad \forall i \in N/\left(O^{OD} \cup D^{OD}\right), k \in K \tag{2}$$

$$\sum_{k \in K} \sum_{j \in N} x_{ij}^k = 1 \qquad \forall i \in S \cup C \tag{3}$$

$$\sum_{j \in N} x_{ij}^k \leq 1 \qquad \forall k \in K, i \in \Delta \quad (4)$$

$$T_i^0 \leq t_i^k \leq T_i^1 \qquad \forall i \in N, k \in K \quad (5)$$

$$x_{ii}^k = 0 \qquad \forall i \in N, k \in K \quad (6)$$

$$t_n^k + T_{ni} \leq t_i^k \qquad \forall n \in \Delta, i \in DV_n, k \in K, k \notin K_n \quad (7)$$

$$\sum_{j \in N} x_{nj}^k \geq \sum_{j \in N} x_{ji}^k \qquad \forall n \in \Delta, i \in DV_n, k \in K \quad (8)$$

$$t_n^k \geq t_i^k + T_{in} \qquad \forall n \in \Delta, i \in PV_n, k \in K, k \notin K_n \quad (9)$$

$$\sum_{j \in N} x_{jn}^k \geq \sum_{j \in N} x_{ji}^k \qquad \forall n \in \Delta, i \in PV_n, k \in K \quad (10)$$

$$\sum_{j \in N} y_{ji}^{k,s,e} - \sum_{j \in N} y_{ij}^{k,s,e} = z_i^{k,s,e}$$
$$\forall e \in DV,\ s \in \Delta_{V_e},\ i \in N,\ k \in K \quad (11)$$

$$\sum_{k \in K} z_i^{k,s,e} = DD_{V_e,i} * \sum_{k \in K} dd_{s,i,k}$$
$$\forall e \in DV, i \in N, s \in \Delta_{V_e} \quad (12)$$

$$\sum_{j \in N} y_{ji}^{k,s,e} - \sum_{j \in N} y_{ij}^{k,s,e} = z_i^{k,s,e}$$
$$\forall s \in PV, e \in \Delta_{V_s}, i \in N, k \in K \quad (13)$$

$$\sum_{k \in K} z_i^{k,s,e} = DD_{i,V_s} * \sum_{k \in K} pd_{i,e,k}$$
$$\forall s \in PV, i \in N, e \in \Delta_{V_s} \quad (14)$$

$$\sum_{s \in N} \sum_{e \in N} y_{ij}^{k,s,e} \leq C_k * x_{ij}^k \qquad \forall i, j \in N, k \in K \quad (15)$$

Constraints for regular drivers:

$$t_j^k \geq t_i^k + T_{ij} - M * (1 - x_{ij}^k)$$
$$\forall n \in \Delta, k \in K_n, i, j \in N, j \neq n \quad (16)$$

$$t_j^k \geq (T_i^0 + T_{ij}) * x_{ij}^k \qquad \forall n \in \Delta, k \in K_n, i, j \in N, j \neq n \quad (17)$$

$$t_n^k \leq t_i^k \qquad \forall n \in \Delta, k \in K_n, i \in N, i \neq n \quad (18)$$

$$\sum_{j \in N} x_{ij}^k \leq 0 \qquad \forall i \in \Delta_{dummy}, k \in RD \quad (19)$$

$$\sum_{j \in N} x_{ij}^k \leq 0 \qquad \forall k \in RD, i \in \left(O^{OD} \cup D^{OD}\right) \quad (20)$$

Constraints for occasional drivers:

$$\sum_{j \in N/(O^{OD} \cup D^{OD})} x_{O_k^{OD} j}^k - \sum_{j \in N/(O^{OD} \cup D^{OD})} x_{j D_k^{OD}}^k = 0$$
$$\forall k \in OD \quad (21)$$

$$\sum_{i \in \Delta} \sum_{j \in N} x_{ij}^k - \sum_{j \in N/(O^{OD} \cup D^{OD})} x_{O_k^{OD} j}^k \geq 0$$
$$\forall k \in OD \quad (22)$$

$$\sum_{j \in N} x_{O_k^{OD} j}^k \leq 1 \qquad \forall k \in OD \quad (23)$$

$$\sum_{j \in N} x_{j O_k^{OD}}^k \leq 0 \qquad \forall k \in OD \quad (24)$$

$$\sum_{j \in S} x_{O_k^{OD} j}^k \leq 0 \qquad \forall k \in OD \quad (25)$$

$$\sum_{j \in N} x_{D_k^{OD} j}^k \leq 0 \qquad \forall k \in OD \quad (26)$$

$$x_{O_k^{OD} D_k^{OD}}^k \leq 0 \qquad \forall k \in OD \quad (27)$$

$$x_{i,i+n_{depots}}^k \leq 0 \qquad \forall i \in \Delta, k \in OD \quad (28)$$

$$x_{i+n_{depots},i}^k \leq 0 \qquad \forall i \in \Delta, k \in OD \quad (29)$$

$$\sum_{j \in N} x_{ij}^k \leq 0 \qquad \forall k \in OD, i \in O^{OD}/O_k^{OD} \quad (30)$$

$$\sum_{j \in N} x_{ji}^k \leq 0 \qquad \forall k \in OD, i \in D^{OD}/D_k^{OD} \quad (31)$$

$$t_j^k \geq t_i^k + T_{ij} - \left(1 - x_{ij}^k\right) M \qquad \forall i, j \in N, k \in OD \quad (32)$$

$$t_j^k \geq \left(T_i^0 + T_{ij}\right) x_{ij}^k \qquad \forall i, j \in N, k \in OD \quad (33)$$

Constraints generated by allowing vehicles to visit the depot multiple times:

$$z_i^{k,n,e} = 0$$
$$\forall e \in DV, n \in \Delta_{V_e}, k \in K, i \in (\Delta \cup \Delta_{dummy}), i \neq n \tag{34}$$

$$z_i^{k,s,n} = 0$$
$$\forall s \in PV, n \in \Delta_{V_s}, k \in K, i \in (\Delta \cup \Delta_{dummy}), i \neq n \tag{35}$$

$$\sum_{k \in K} \sum_{i \in \Delta_n} dd_{i,j}^k = 1 \qquad \forall n \in \Delta, j \in DV_n \tag{36}$$

$$\sum_{k \in K} \sum_{j \in \Delta_n} pd_{i,j}^k = 1 \qquad \forall n \in \Delta, i \in PV_n \tag{37}$$

$$\sum_{m \in \Delta_n} dd_{m,i}^k \geq \sum_{j \in N} x_{ij}^k \qquad \forall n \in \Delta, i \in DV_n, k \in K \tag{38}$$

$$\sum_{m \in \Delta_n} pd_{i,m}^k \geq \sum_{j \in N} x_{ij}^k \qquad \forall n \in \Delta, i \in PV_n, k \in K \tag{39}$$

$$t_i^k \leq t_j^k + \left(1 - dd_{i,j}^k\right) * M$$
$$\forall n \in \Delta, i \in \Delta_n, j \in DV_n, k \in K \tag{40}$$

$$t_i^k \leq t_j^k + \left(1 - pd_{i,j}^k\right) * M$$
$$\forall n \in \Delta, j \in \Delta_n, i \in PV_n, k \in K \tag{41}$$

$$\sum_{m \in \Delta_n} \sum_{j \in N} x_{mj}^k \geq \sum_{j \in N} x_{ji}^k \qquad \forall n \in \Delta, i \in DV_n, k \in K \tag{42}$$

$$\sum_{m \in \Delta_n} \sum_{j \in N} x_{jm}^k \geq \sum_{j \in N} x_{ji}^k \qquad \forall n \in \Delta, i \in PV_n, k \in K \tag{43}$$

The objective function 1 aims to minimize the total routing cost. Constraints 2 - 15 are linked to all vehicles. Constraint 2 serves as the vehicle flow conservation constraint. Constraint 3 ensures every request must be served once. Constraint 4 restricts each vehicle to depart from the depot on at most one route. Constraint 5 is the time window restriction for vehicles arriving at each point. Constraint 6 prevents vehicles from passing through the same point consecutively. Constraints 7-8 ensure that vehicles must pass through the corresponding depot before meeting the delivery requirements. Similarly, Constraints 9-10 ensure that vehicles must pass through the corresponding depot after meeting the pickup requirements. Constraints 11 and 13 record the flow for each pickup and delivery requirement. Constraints 12 and 14 satisfy the pickup and delivery requirements at all points. Constraint 15 is the vehicle capacity limit.

Constraints 16-20 are linked to regular drivers. Constraints 16-17 are the time constraints between two points for RDs. Constraint 18 limits the departure time from the depot for RDs to not exceed the time to any point. Constraint 19 RDs are not allowed to visit dummy depots. Constraint 20 RDs are not allowed to visit O^{OD} and D^{OD}.

Constraints 21 - 35 are linked to occasional drivers. Constraint 21 flow conservation for origins and destinations of ODs. Constraint 22 if OD departs, then it must visit the depot. Constraint 23 at most one arc after the origin of the ODs. Constraint 24 no arc entering the origin of ODs. Constraint 25 first arc after origin cannot go to a customer. Constraint 26 no arc exiting from destination of ODs. Constraint 27 no link between origins and destinations of ODs. Constraints Constraints 28-29 no link between depots and dummy depots. Constraints 30-31 ODs cannot visit the origin and destination of other ODs. Constraints 32-33 are the time constraints between two points for ODs.

Constraints 34-35 no cross-docking during the pickup and delivery process. Constraints 36-37 for a delivery or pickup demand, it must be served by its depot or dummy depot. Constraints 38-39 the relationship between pickup and delivery variables and path variable. Constraints 40-41 if the delivery or pickup request is serviced by the corresponding depot/dummy, the time limit is met. 42-43 if the delivery or demand is served by vehicle k, then vehicle k must have passed through its depot/dummy.

B. ALNS Algorithm

According to literature research, the ALNS algorithm has been widely applied to vehicle routing problems, and ALNS is suitable for our problem. The strategy of each step in the ALNS algorithm, as well as the involved parameters, directly affect the computation time and the generation of optimal solutions. Therefore, the selection of strategies and the tuning of parameters is a labor-intensive and time-consuming process. Windras Mara et al. [24] summarized the model proposed by Ropke & Pisinger [32] into four steps: (a) Adaptive mechanism, (b) Acceptance criteria, (c) Stopping criteria, and (d) Design of destroy and repair operators. Regarding the Adaptive mechanism, based on the 251 papers surveyed [24], 250 papers utilized the Roulette Wheel ([21], [16], [3]), meaning the probability of each operator being selected is the same. For Acceptance criteria, most articles adopt the Metropolis Criterion, allowing the algorithm to accept solutions slightly worse than the current one with a certain probability to avoid local optima, thereby increasing the algorithm's capability to explore a broader area of the solution space. Some articles also utilize the Greedy Mechanism and Record-to-Record. Regarding the Termination Criterion, the majority of articles use the most common Number of Iterations, with research also employing the Number of Non-Improving Iterations, Running Time Limit, and Annealing Temperature.

Algorithm 1 illustrates the framework of the used ALNS algorithm. Each step of this framework will be discussed in detail in the subsequent sections. The algorithm requires two

input parameters: the initial solution s, which is generated using a greedy algorithm, and a number q, representing the count of demand nodes to be removed. The core of the algorithm is from lines 5 to 8. In line 5, an appropriate removal operator is selected based on the current weights of the removal operators. This paper uses two removal operators: random removal and worst removal. In line 6, the selected requests are removed from s'. In line 7, a suitable repair operator is chosen based on the current weights of the repair operators. The repair operators used are greedy repair heuristic and regret-2 repair heuristic. In line 7, the requests are reinserted into the destroyed solution. The choice of removal and repair operators directly affects the performance of the algorithm. Following the strategy [32], we use a roulette wheel method for operator selection. Considering some unnecessary depot visits in the initial solution, the depot, serving as both a pickup and delivery point, may be permanently removed during the destruction process. While this can help find a better solution, it may also lead to infeasible solutions. Therefore, in line 9, we need to check if the repaired solution is feasible. If feasible, we then determine whether the new solution should be accepted. This paper uses a simulated annealing acceptance criteria to decide whether to accept the new feasible solution.

Algorithm 1 ALNS Heuristic

```
1:  function ALNS(s ∈ {solutions}, q ∈ ℕ)
2:      solution s_best = s;
3:      repeat
4:          s' = s
5:          choose a destroy operator
6:          remove q requests from s'
7:          choose a repair operator
8:          reinsert removed requests into s'
9:          if s' is feasible then
10:             if f(s') < f(s_best) then
11:                 s_best = s'
12:             end if
13:             if accept(s', s) then
14:                 s = s'
15:             end if
16:         else
17:             continue
18:         end if
19:         operator adaptive weight adjustment
20:     until stop-criterion met
21:     return s_best
22: end function
```

1) Initial Solution: The initial solution is a necessary input for the ALNS algorithm. We use a simple Greedy algorithm to generate the initial solution. The algorithm begins at the vehicle's starting point and iteratively finds the nearest unvisited point, checking capacity and time window constraints to decide whether to visit that point. For delivery points, the algorithm prioritizes finding the nearest depot that can service the point, ensuring the route remains valid. For pickup points, it ensures that a depot can service the point. After visiting each point, the vehicle's load and time are updated until all points are serviced, and the vehicle returns to its destination, forming a complete route.

The initial solution for Algorithm can be simply divided into two parts. The first part uses occasional drivers (OD) to handle demand nodes, and the second part uses regular drivers (RD) to handle the remaining demand nodes. The reason for this separation is that occasional drivers and regular drivers have different characteristics. Compared to regular drivers who start from the depot, occasional drivers have specific origins and destinations. Additionally, the cost for OD is lower, so OD is given a higher priority. In each part, for every nearest point found, we first determine whether the point is a pickup point, delivery point, or depot, and then execute different handling methods based on the type of point. Additionally, we must ensure that there are enough vehicles to meet all demands, otherwise, the generated initial solution will result in some demands not being serviced. Any surplus vehicles in the initial solution will be removed during the ALNS algorithm.

2) Request Removal: This section introduces two removal operators: random removal and worst removal. In addition, Ropke & Pisinger [32] also used Shaw removal proposed by Shaw [37]. However, random removal can be seen as a special form of Shaw removal and has been proven to be more efficient, thus we adopt random removal. Unlike random removal, worst removal selects those points that seem to be misplaced, i.e., service points that incur higher costs.

Random Removal The basic idea of the random removal algorithm is to randomly select q requests and remove them from the solution. The number of requests q to be removed is obtained by multiplying a destruction factor κ between 0 and 1 by the number of points in the current solution. These randomly selected requests include pickup points, delivery points, and depots. As mentioned earlier, the solution may contain some unnecessary visits to depots. Therefore, we need to reselect the nodes chosen for removal. If the removed points are pickup or delivery points, it means that these points need to be served, so we retain these points. If the removed points are depots, we randomly select from the removed depots again. The selected depots are kept for the repair operation, and the unselected depots are permanently removed from the solution. This ensures unnecessary paths are eliminated, leading to a lower cost.

Worst Removal The basic idea of worst removal is that for a pickup or delivery request i served by some vehicle in a solution s, we define the cost of the request as $\text{cost}(i, s) = f(s) - f_{-i}(s)$, where $f_{-i}(s)$ is the cost of the solution without request i. All requests are sorted by cost in descending order, and the top q requests with the highest costs are removed and reinserted in the repair operator to obtain a better solution. On this basis, we have made some modifications. Since the solution path includes some depots, before using worst removal,

we first select the depots in the current solution and use random removal to remove some depots. Suppose the number removed is n_{removed}, which ranges from 0 to $\min\{\text{number of depots in current solution}, q\}$. The remaining number of requests to be removed using worst removal is $q - n_{\text{removed}}$. This allows the algorithm to permanently delete some depots to achieve a better solution. To increase the randomness of the worst removal, a new parameter $p \geq 1$ is added. A lower value of p corresponds to greater randomness, so with a certain probability, we will choose the $y^p|L|$-th worst request.

3) Inserting Requests: According to the research by Potvin & Rousseau [29], insertion heuristics for the vehicle routing problem can be broadly divided into two categories: sequential and parallel insertion heuristics. The difference between them is that sequential heuristics construct one route at a time, whereas parallel heuristics can construct multiple routes simultaneously. The insertion heuristics involved in this research are all parallel heuristics. Basic Greedy heuristic and Regret-2 heuristic are applied, as they are the most common insertion heuristics.

Basic Greedy Heuristic The principle of the Basic Greedy Heuristic is the same as the greedy strategy used when generating the initial solution. The program iterates over the requests removed in the previous section. For each request, we try to insert it into a feasible position in the route, recording the cost change $\Delta f_{i,k,idx}$ each time a feasible insertion position is found, $\Delta f_{i,k,idx}$ represents the cost difference when inserting the request i at position idx in route k compared to not inserting it. The request i is inserted at the position with the smallest cost until all requests are inserted into the routes. If no suitable insertion position is found in the existing routes, the request is inserted into an empty route of the vehicle corresponding to the depot of that request.

Regret Heuristic The class of regret heuristics was proposed to overcome the main weakness of greedy heuristics, i.e., the myopic behavior. In greedy insertion, we always focus on one request at a time, which may result in the position idx where request i is inserted in the route being the optimal position for another request j to be inserted later. If request j is inserted at position idx in the route, it may result in a better solution. For example, the cost of inserting request i at idx 1 in route 1 is 20, and at idx 2 in route 2 is 15. Similarly, the cost of inserting request j at idx 1 in route 1 is 40, and at idx 2 in route 2 is 18. According to the basic greedy insertion, request i is inserted at idx 2 in route 2 because it has the lowest insertion cost, and then request j is considered. Since idx 2 in route 2 is already occupied by request i, request j can only be inserted at idx 1 in route 1, resulting in a total insertion cost of 55. This research considers the Regret-2 insertion heuristic, which improves the basic greedy heuristic by incorporating a look-ahead information when selecting requests to insert. First, we iterate over each request to be inserted and find all possible insertion points and insertion costs. For each request, we set a regret value $c_i^* = \Delta f_{i,k_1,idx_1} - \Delta f_{i,k_2,idx_2}$, where the regret value is the difference in the cost of inserting the request in its best route-index and its second best route-index. In each iteration, we select the request with the maximum regret value and insert it into the optimal route. The heuristic can be extended to regret-k heuristic, but as k increases, the improvement efficiency of the algorithm slows down. Therefore, this research only uses the regret-2 heuristic.

4) Choosing a Removal and an Insertion Heuristic: Two removal operators and two repair operators are defined in section IV-B.2 and section IV-B.3. In one iteration, only one removal and one repair operator need to be used, but ALNS involves multiple removal and repair operators because different operators may be suitable for different sizes of problems. Even for problems of the same size, different operators may perform differently. For a specific problem, we do not know which operator is more suitable, so we let the algorithm choose. To select the appropriate removal and repair operators, we assign weights to each repair and removal operator and use the roulette wheel selection principle. Suppose we have k heuristics with weights $w_i, i \in \{1, 2, ..., k\}$, we select heuristic j with probability

$$\frac{w_j}{\sum_{i=1}^{k} w_i} \tag{44}$$

Note that the selection of insertion heuristic is independent of the removal heuristic (and vice versa). We can manually set these weights, but if many removal and insertion heuristics are used, this can be a very complex process. Instead, Section IV-B.5 proposes an adaptive weight adjustment algorithm.

5) Adaptive Weight Adjustment: This section details how to introduce weights and adaptively adjust them. The Adaptive Weight Adjustment part draws on the method of Adaptive Weight Adjustment in the ALNS framework [32]. The basic idea is to track the score of each heuristic, which measures the recent performance of the heuristic. High scores correspond to successful heuristics. However, we made some modifications. First, we set the initial score of each removal and repair operator to 0, so the probability of each operator being selected is equal. We will call the ALNS algorithm multiple times, each time called a segment. Each segment has an inner loop for iterating new solutions. In each segment, each time a new solution is accepted, the used removal and repair operators are scored according to the following scoring rules in Table II.

In each iteration, we apply two heuristics: a removal heuristic and an insertion heuristic. Both heuristics' scores are updated by the same amount because we cannot determine whether the removal or insertion is the successful reason. At the same time, we record the number of successful runs of the heuristic and calculate the weight of each heuristic before executing the next segment based on the scores and

TABLE II
SCORE ADJUSTMENT PARAMETERS

Parameter	Description
σ_1	The last remove-insert operation resulted in a new global best solution.
σ_2	The last remove-insert operation resulted in a solution that has not been accepted before. The cost of the new solution is better than the cost of the current solution.

successful runs of each operator. The weight calculation formula is as follows,

$$w_{i,j+1} = w_{ij}(1-r) + r\frac{\pi_i}{\theta_i} \tag{45}$$

where $w_{i,j+1}$ represents the weight of operator i in segment $j+1$, $w_{i,j}$ represents the weight of operator i in segment j, π_i is the score obtained by operator i in the last segment, and θ_i is the number of successes of the operator. The reaction factor r controls the speed of the weight adjustment algorithm in response to changes in operator effectiveness. If r is zero, then we do not use the score at all and stick to the initial weight. If r is set to 1, then the weight is determined by the score obtained in the last segment.

6) Acceptance and Stopping Criteria: The simplest Acceptance and Stopping Criteria is to accept only solutions better than the current solution, see [37]. This is likely to lead to the solution getting stuck in a local optimum. Therefore, it is sometimes wiser to accept solutions worse than the current one [32]. Therefore, we choose to use simulated annealing acceptance criteria, which means that we accept a solution s' worse than the current solution s with a certain probability, given by $e^{(f(s)-f(s'))/T}$, where $T > 0$ represents the temperature. The temperature starts from T_{start} and is updated at the end of each iteration by $T = T \cdot c$, where $0 < c < 1$ is the cooling rate. Similarly, if the new solution s' is better than the current solution, we accept this solution with 100% probability. The stopping criterion of the algorithm is to meet the set maximum number of iterations. Of course, for different sizes of instances, the initial temperature and number of iterations will affect the performance of the results, so parameter tuning is necessary.

V. COMPUTATIONAL EXPERIMENTS

A. Instance Generation

The instances used in this chapter are virtual instances generated by code for the purpose of parameter tuning and testing algorithm performance. Each instance has fixed characteristics and is created and stored in a text file. Each instance file starts with 11 lines whose meanings are shown in Figure III. Following this, there is a matrix containing all the points, where each row of the matrix represents a point. According to Figure III, there are two depots, corresponding to the first two rows of the matrix. The dummy depots corresponding to these two depots are represented by the third and fourth rows of the matrix. Next, there are 6 customers, each with delivery demands. Following these are 4 pickup and delivery demands for the same intermediate center. Finally, there are the start and end points for two ODs (Origin-Destination pairs). Rows 14-15 represent the start and end points for the OD 1, while rows 16-17 represent the start and end points for the OD 2. For the columns, the first two columns represent the two-dimensional coordinates of each node, the third and fourth columns represent the time window for each node, the fifth column represents the pickup or delivery quantity (positive for delivery, negative for pickup), and the last column represents the depot corresponding to each pickup or delivery demand. The naming rule of Instance is composed of three numbers, such as **Instance 2_1_10**, which means there are 2 depots, 1 intermediate center and 10 requests.

TABLE III
TEST INSTANCE 2_1_10

Parameter	Value
Number of Nodes	18
Number of Depots	2
Number of RD in depot 0	1
Capacity of RD in depot 0	10
Number of RD in depot 1	1
Capacity of RD in depot 1	10
Number of OD	2
Capacity of OD	8
Number of requests in intermediate centers	4
Max number visits depot	2
Coeff Cost OD	0.5

Coor_1	Coor_2	TW_1	TW_2	PD	V
50.0	40.0	0	1236	0	0
125.0	110.0	0	1236	0	0
50.0	40.0	0	1236	0	0
125.0	110.0	0	1236	0	0
45.0	20.0	20	870	2	0
18.0	50.0	30	880	8	0
35.0	100.0	30	880	2	0
125.0	20.0	30	880	2	1
150.0	100.0	30	880	2	1
110.0	140.0	30	880	2	1
85.0	65.0	0	1236	2	0
85.0	65.0	0	1236	-2	0
85.0	65.0	0	1236	-2	1
85.0	65.0	0	1236	2	0
0.0	0.0	100	300	0	0
0.0	0.0	0	1236	0	0
0.0	0.0	100	300	0	0
0.0	0.0	0	1236	0	0

B. Example Instance

Before parameter tuning, we want to verify whether the model and algorithm outputs are consistent through a simple example, meaning whether they can achieve the same optimal solution. In fact, we used several requests ranging from 10 to 20 to debug the model and code. Upon completing the debugging, we have verified the consistency between the mathematical model and the ALNS algorithm. Therefore, this demonstration of the optimal solution generation, along with its description and visualization, helps the readers better understand the model. The test example mentioned here is Instance 2_1_10. All experiments were performed on

a PC running Microsoft Windows 10 with the following specifications:

- Processor: Intel Core i7-8750H 2.20GHz
- RAM: 8 GB
- GPU: NVIDIA GeForce GTX 1050 Ti
- Software: VS Code 1.78.0, Python 3.8.18

Using the CPLEX package in Python to run the mathematical model, we solve it with the branch-and-bound method, setting a time limit of 3600 seconds. The optimal cost of 303.5 was obtained in 40 minutes, which is the same result as that obtained using the ALNS algorithm, but the ALNS algorithm only took 8.9 seconds. In practice, the ALNS algorithm should take a shorter time. Typically, when the number of requests is 10, we set the iteration to 200, and the runtime is approximately 300ms. To prevent the algorithm from getting stuck in a local optimum during a single loop, we add an outer loop, running the ALNS algorithm 30 times and taking the best performance among the 30 runs, as shown in Figure 2. In the figure, the best iteration found 50 feasible solutions within 200 iterations. The algorithm found the optimal solution within 10 iterations and then attempted to accept worse solutions to explore the solution space further. Since the optimal solution had already been found, no lower cost were discovered afterward. The red line in the figure represents the best solution found in the current iteration, and the blue line represents the current feasible solution. If a current feasible solution is better than the current best solution, the current best solution is updated. The optimal solution is visualized in Figure 3, and the optimal routes are:

- **RD 1**: [0]
- **RD 2**: [1]
- **OD 1**: [14, 15]
- **OD 2**: [16, 0, 6, 10, 11, 12, 13, 3, 9, 1, 8, 7, 4, 2, 5, 17]

Since the OD vehicles have sufficient capacity, the RDs were not used, RD 1 and RD 2 stayed at their respective depots. Here, we assume that the OD vehicles are provided by a third-party logistics company, so two OD vehicles with the same start and end points were set up to ensure that all demands can be fully serviced. OD 1 departs from the start point directly to the end point, indicating that OD 1 was not used, while OD 2 handled all the demands.

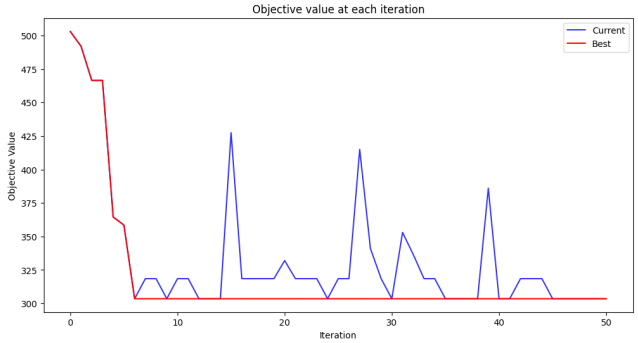

Fig. 2. ALNS Result of test instance 2_1_10

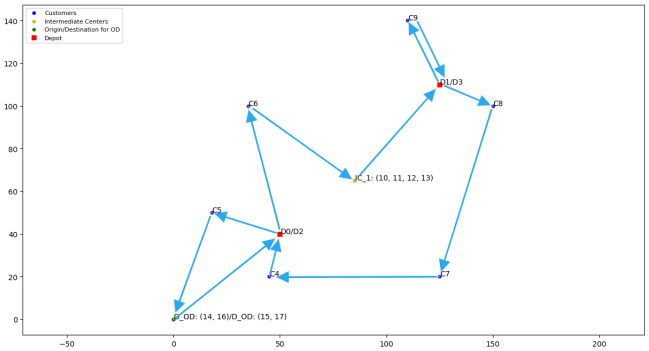

Fig. 3. Result map of test instance 2_1_10

C. Parameter Tuning

Before parameter tuning, a representative set of tuning instances is required. As our research problem is a new pickup and delivery problem, no existing instances found that could be directly referenced and used. Therefore, we will refer to the method of generating instances in [38] with certain modifications to suit the specifics of our study. Initially, the scale of the tuning instances needs to be sufficiently diverse. Our tuning set consists of 6 instances, where the number of requests starts from 10, with 2 instances added for every increase of 20 requests. The largest instances contain 50 requests. Request coordinates are uniformly distributed around the depots. Alongside increasing requests, we will concurrently adjust the vehicle capacity, number of intermediate center, number of OD, and number of depot. Specific parameter settings for generating tuning instances are shown in Table IV.

The primary objective of parameter tuning is to subsequently conduct large-scale testing and comparison of the performance between ALNS algorithm and exact algorithm, and to reveal which heuristic part or which parameter contributes the most to solution quality. Firstly, it is necessary to determine which parameters need adjustment. Before executing removal operators, we need to determine how many requests to remove, i.e., the destruction factor κ. Random removal involves no parameters, while worst removal involves a random parameter p. No parameters are involved in repair operators. For acceptance criteria, two parameters are involved: initial temperature T_{start} and cooling rate c. For the Adaptive weight adjustment criteria, three parameters are involved: reaction factor r, score adjustment parameters σ_1, and σ_2. For these seven parameters, adjustments are made sequentially in a certain order, with each parameter set to 4 reasonable values as listed in Table V. Each tuning instance is run 5 times for each parameter value. Thus, for 6 tuning instances, we obtain 30 costs per parameter, and the parameter corresponding to the lowest average cost is selected as the optimal parameter. Table **??** shows the tuning results for the destruction factor κ, recording the runtime and cost for each instance, and calculating the average runtime and average cost accordingly. Based on the tuning results, it is evident that only the destruction factor κ shows

TABLE IV
TUNING INSTANCES INFORMATION

Requests	Nodes	Depots	ICs	Requests per Depot	Requests per IC	RDs each Depot	Capacity of RD	ODs	Capacity of OD
10	18	2	1	3	4	1	20	2	16
30	38	2	2	10	5	2	30	2	24
50	60	3	2	10	10	1	50	2	40

sensitivity to different parameter values, while the remaining six parameters exhibit lower sensitivity. Therefore, these six parameters are deemed suitable for solving instances within a certain scale range. The optimal parameter value for the destruction factor κ is found to be 0.1. Given that 0.1 represents a relatively small destruction ratio (for instance, in cases with 10 requests, only one request is destroyed per iteration), we choose not to conduct further in-depth analysis on the Destruction factor. The best values for each parameter are listed in Table V.

TABLE V
PARAMETER SETTINGS

Parameter name	Symbol	Parameter range	Selected values	Best value
Destruction factor	κ	(0, 1)	[0.1, 0.15, 0.2, 0.5]	0.1
Random Parameter	p	≥ 1	[1, 5, 10, 20]	1
Initial Temperature	T_{start}	$\mathbb{R}^+$	[100, 200, 300, 400]	300
Cooling rate	c	(0, 1)	[0.8, 0.85, 0.9, 0.95]	0.95
Reaction factor	r	[0, 1)	[0.2, 0.4, 0.6, 0.8]	0.2
Score adjustment factor	σ_1	$\mathbb{R}^+$	[10, 20, 30, 40]	30
Score adjustment factor	σ_2	$\mathbb{R}^+$	[5, 10, 15, 20]	5

D. Computational Experiments

In this section, we conduct a comparative experiment between the ALNS algorithm with tuned parameters and an exact method to evaluate the performance of the ALNS algorithm. Using the same logic as parameter tuning, we generate 30 instances with request quantities of 10, 30, and 50 (10 instances each). The results of the computational experiments are presented in Table VI. The bold fonts in the table indicate the known optimal solutions. For instances with 10 requests, the exact method using CPLEX's built-in branch and bound algorithm found 4 optimal solutions, with other solutions showing relatively good performance but longer average runtimes, mostly exceeding 20 minutes. Although the ALNS algorithm confirmed only one instance as optimal, it demonstrated overall good performance across the 10 instances. It consistently found solutions within 10 seconds that were better than those found by the exact algorithm after 1 hour of computation. As the instance sizes increased to 30 and 50 requests, the advantages of the ALNS algorithm became apparent. We configured it to run 10000 and 15000 iterations respectively. The ALNS algorithm significantly improved upon initial solutions in a short time frame, whereas the exact algorithm achieved only a best bound after running for the set 1 hour and 1.5 hours for the respective instance sizes.

E. Sensitivity Analysis

In the business domain, cost is one of the primary concerns for managers. In the process of parcel delivery, reducing costs and increasing efficiency is a common goal for almost all companies. Hou & Wang's [17] research shows that the average delivery cost can be reduced by 7.30% compared to delivery by dedicated vehicles when incorporating a compensation scheme based on crowdshippers' acceptance behavior. In this study, there are four factors that influence parcel delivery costs: the number of occasional drivers, the capacity of occasional vehicles, the transportation cost of occasional vehicles, and the location of occasional vehicles. Since the location of occasional vehicles is difficult to evaluate, it will not be discussed in this section. Generally, the capacity of occasional vehicles is positively correlated with their transportation cost per unit distance or per unit time. This chapter will also analyze the correlation between vehicle capacity and vehicle transportation cost. Finally, the sensitivity analysis sequence is to first select different numbers of occasional drivers, determine the number of occasional drivers, then set vehicle capacity and cost factor within a reasonable range, and analyze the cost optimization brought by different values.

1) Instances for Sensitivity Analysis: Sensitivity analysis requires a new set of instances. We decided to modify the instances used for parameter tuning. First, we only consider instances with 30 and 50 requests, as instances with 10 requests are considered too small to be of analytical value. We retain all coordinates, time windows, and demands of the 20 instances with 30 and 50 requests from the parameter tuning section. We modify the values of regular drivers, occasional drivers, occasional vehicle capacity, and cost factors for these instances. The parameter values for the occasional driver sensitivity analysis are shown in Table VII. For the sensitivity analysis of occasional vehicle capacity and cost factor, except for the different values of vehicle capacity and cost factor, all other parameters remain the same. Each instance is run once, with each run having 10000 iterations.

2) Sensitivity Analysis - Number of OD: The results of the sensitivity analysis for occasional drivers are shown in Table VIII. We conducted four sets of comparative analyses based on the number of occasional drivers, which is $OD = [0, 1, 2, 3]$, corresponding to the four cost columns in the table. We used the no OD group as the baseline and calculated the optimization ratio of the other three groups containing ODs compared to the baseline cost, shown in the three columns in the table. A negative ratio indicates

TABLE VI

INSTANCE PERFORMANCE COMPARISON - ALNS VS BRANCH AND BOUND

Instance	ALNS Initial Cost	Best Cost	Time (s)	Iteration	CPLEX Best Cost	Best Bound	MIP Gap	Solve Time (s)
2.1.10.1	663	**429**	5	5000	436	199.64	54.21%	3600
2.1.10.2	384	**252**	17	5000	252	252	0%	476
2.1.10.3	583.5	377.5	7.6	5000	305.5	305.5	0%	2114
2.1.10.4	676	315.5	7.8	5000	300	231.79	22.74%	3644
2.1.10.5	427.5	**294**	5.7	5000	294	254.53	13.42%	3036
2.1.10.6	546	231.5	5.4	5000	**204.5**	204.5	0%	2085
2.1.10.7	575.5	348.5	5.6	5000	**249**	206.1	17.23%	3617
2.1.10.8	615.5	273	5.7	5000	**236**	236	0%	1215
2.1.10.9	701	336.5	6.9	5000	**320.5**	244	23.87%	3725
2.1.10.10	395	278.5	5.8	5000	**267.5**	224.82	15.95%	3607
2.2.30.1	1778	**801**	63.6	10000		275.92		3600
2.2.30.2	1035	**853.5**	48.3	10000		261.08		3600
2.2.30.3	1339.5	**1017.5**	49.5	10000		293.19		3600
2.2.30.4	1210.5	**837**	61	10000		278.6		3600
2.2.30.5	1076.5	**881**	66.5	10000		332.31		3600
2.2.30.6	989	**879.5**	47.5	10000		270.42		3600
2.2.30.7	1137.5	**950**	51.6	10000		327.38		3600
2.2.30.8	1334.5	**962**	47.9	10000		311.44		3600
2.2.30.9	1047.5	**912**	42.1	10000		278.13		3600
2.2.30.10	1120	**950**	65.8	10000		234.45		3600
3.2.50.1	1585.5	**1048.5**	304.4	15000		374.98		5400
3.2.50.2	1519.5	**1290.5**	273.8	15000		379.96		5400
3.2.50.3	1984	**1357.5**	242.1	15000		409.74		5400
3.2.50.4	2001.5	**1440**	244.1	15000		394.61		5400
3.2.50.5	2019.5	**1410**	315.1	15000		372.4		5400
3.2.50.6	1967.5	**1045**	318	15000		346.03		5400
3.2.50.7	1652	**1104**	264.3	15000		374.89		5400
3.2.50.8	1633	**1170**	242	15000		363.4		5400
3.2.50.9	1531.5	**1204.5**	319	15000		327.26		5400
3.2.50.10	1379.5	**1088.5**	239.4	15000		320.11		5400

TABLE VII

SENSITIVITY ANALYSIS INSTANCE INFORMATION

Requests	Nodes	Depot	Depot 1 RD	Depot 2 RD	Depot 3 RD	Capacity RD	OD	Capacity OD	Cost Factor
OD = 0									
30	34	2	3	3		30	0	0	
50	60	2	2	2	2	50	0	0	
OD = 1									
30	36	2	2	3		30	1	24	0.8
50	58	3	2	2	2	50	1	40	0.8
OD = 2									
30	38	2	2	2		30	2	24	0.8
50	60	3	2	2	2	50	2	40	0.8
OD = 3									
30	40	2	1	2		30	3	24	0.8
50	62	3	2	2	2	50	3	40	0.8

a worse result, as the ALNS algorithm may get stuck in local optima when solving instances. The optimal solutions and their corresponding optimization ratios are highlighted in bold. Among the 20 instances with optimal costs, the no OD group accounts for 4 instances, the group with 1 OD accounts for 7 instances, the group with 2 ODs accounts for 6 instances, and the group with 3 ODs only accounts for 3 instances. When OD = 1, even though some instance results are not optimal, the cost differences are relatively small, and the results are more stable. Therefore, we consider 1 OD to be a good choice for instances with 30 and 50 requests. In subsequent sensitivity analyses of OD capacity and cost, the parameter settings for 1 OD will be used.

In actual operations, company decision-makers can choose either the number of ODs with a cost advantage or the lowest-cost path each time. In this experiment, when a fixed number of ODs is selected, having 1 OD increases the total cost of all instances by 19.09% compared to having no OD.

With 2 ODs, the total cost increases by 17.19%, and with 3 ODs, the total cost increases by 18.48%. It is worth noting that when the cost is worse with a fixed number of OD vehicles, we will continue to choose the cost without ODs. If each instance selects its corresponding optimal cost, the total cost of the 20 instances increases by 22.07%.

3) Sensitivity Analysis - OD Capacity and Cost: For the capacity of ODs, we set up seven comparative experiments, reducing the capacity of ODs relative to the capacity of RDs by a certain percentage. The seven comparative experimental percentage parameters are set to $Capacity = [40\%, 50\%, 60\%, 70\%, 80\%, 90\%, 100\%]$. For the unit distance cost of RDs, we set up five comparative experiments, reducing the unit distance cost relative to that of RDs by a certain percentage. Different vehicle capacity parameter settings correspond to different unit distance cost ratios for RDs, as detailed below:

- **OD Capacity Parameter = 40%**: [20%, 30%, 40%, 50%, 60%]
- **OD Capacity Parameter = 50%**: [30%, 40%, 50%, 60%, 70%]
- **OD Capacity Parameter = 60%**: [40%, 50%, 60%, 70%, 80%]
- **OD Capacity Parameter = 70%**: [50%, 60%, 70%, 80%, 90%]
- **OD Capacity Parameter = 80%**: [50%, 60%, 70%, 80%, 90%]
- **OD Capacity Parameter = 90%**: [50%, 60%, 70%, 80%, 90%]
- **OD Capacity Parameter = 100%**: [50%, 60%, 70%, 80%, 90%]

A total of 100 cost values were obtained for each OD capacity parameter setting, resulting in 700 cost values across 7 parameter configurations. In addition to the costs obtained for different parameter values, the lowest cost parameter results serves as the base cost for calculating the percentage increase in OD costs per 10% increase in capacity. Positive percentages indicate increased costs, while negative percentages indicate decreased costs. The percentage increase in OD costs for all 7 OD capacity values is shown in Table IX. For each row, if capacity correlates positively with vehicle costs (i.e., when OD vehicle capacity is 40% of RD vehicle capacity, OD per unit distance costs should also be 40% of RD costs), we use this as a benchmark to explore scenarios where OD costs are below or above 40%. The same applies to the other 6 experimental groups. Taking OD capacities of 60% and 70% as examples, the calculated cost increase percentages are [50%, 60%, 70%, 80%, 90%, 100%] = [11.76%, 4.89%, 9.08%, 8.92%, 8.47%, 7.81%], with an average increase of 8.47%.

Next, we explore the performance of OD capacity or OD cost as single variables compared to path costs without OD inclusion. As mentioned earlier, both the number of ODs and the sensitivity analysis in this section use the same instances, adjusting only relevant parameters to facilitate direct comparison of these results. In Section VIII, we used OD capacity and OD cost ratios of 80%. Referring to the path costs when OD = 0, we calculated the cost decrease brought by different OD cost ratios when OD capacity = 80% across 20 instances as [90%, 80%, 70%, 60%, 50%] = [5.18%, 15.21%, 16.47%, 22.54%, 27.01%]. These results indicate that path total costs save 5.18% when OD costs are 90% of RD costs, and save 15.21% when OD costs are 80% of RD costs. We then explore the effect of varying OD capacity on path total costs with fixed OD costs at 80% of RD costs.

When the OD cost is fixed at 80%, and the OD capacity takes values of [60%, 70%, 80%, 90%, 100%], the total routing cost reduction relative to no OD situation is [7.86%, 9.97%, 15.21%, 13.96%, 13.94%]. According to the results, using OD with 60% RD capacity saves 7.86% costs, 70% RD capacity saves 9.97% costs, and 80% RD capacity saves 15.21% costs, with further capacity increases not resulting in further cost reductions. Figure 4 illustrates the total cost savings ratio across all combinations of vehicle capacities and costs for 20 test instances.

VI. Case Study

A. Background

Ochama is an omnichannel retailer, part of the Chinese e-commerce giant JD.com. Headquartered in the Netherlands, Ochama operates primarily in the Netherlands, Belgium, France, and Germany. The platform offers over 10,000 high-quality products across various categories. Utilizing an advanced automated warehousing system, Ochama provides dual fulfillment services of pickup and delivery. Consumers can place orders online through the Ochama platform, with the option to pick up their orders from pickup points or have them delivered directly to their homes. Ochama has successfully implemented advanced warehouse network and logistics system with over 700 ochama pickup points in major cities and towns in the Netherlands, Belgium, and Germany.

Ochama currently operates two depots: a self-operated warehouse and a crossdocking warehouse located in Rotterdam and Venray, Netherlands, respectively (see Map 5). The self-operated warehouse in Berkel en Rodenrijs, Rotterdam, handles all storage and shipping of goods, as well as distribution for orders in western and northern Netherlands. Orders from eastern Netherlands, western Germany, and northern Belgium are consolidated and transported to Ochama's crossdocking facility in Venray for distribution after crossdocking. Ochama has over 700 pickup points, primarily within the Netherlands, with the remainder in western Germany, Belgium, and parts of France. These pickup points do not operate every day but rather on select days of the week, servicing an average of 150 to 200 pickup points daily. In China, June 18th annually marks a major e-commerce promotion day, similar to Black Friday. Ochama continues this tradition, and statistics show that during the recent 2024 June 18th promotion, Ochama shipped 13,000 parcels. Parcels are delivered using vans sized 5.45m × 2.263m × 2.29m, with an average parcel size of 40cm × 25cm × 30cm. Each van can accommodate an average of 200 parcels.

B. Instances for Case Study

In this study, we scale down the problem based on real-world scenarios. Each instance retains the positions of two depots and randomly sets 5 intermediate centers (ICs). Each IC contains 20 requests, and each depot is randomly assigned 25 demand points, totaling 150 demands per instance. The

TABLE VIII

SENSITIVITY ANALYSIS - NUMBER OF OD

Instances	OD = 0	OD = 1	Gap - OD=1	OD = 2	Gap - OD=2	OD = 3	Gap - OD=3
2.2.30.1	1091	1029	5.7%	**955.4**	**12.4%**	1041.6	4.5%
2.2.30.2	1303.5	1229	5.7%	**1018**	**21.9%**	1073.6	17.6%
2.2.30.3	1433	1281	10.6%	**1154.2**	**19.5%**	1170	18.4%
2.2.30.4	**839**	939	-11.9%	1405.8	-67.6%	1085	-29.3%
2.2.30.5	1105	1266.2	-14.6%	1098.2	0.6%	**1067**	**3.4%**
2.2.30.6	967	**884.4**	**8.5%**	1134	-17.3%	1057	-9.3%
2.2.30.7	1377	**1141**	**17.1%**	1181.2	14.2%	**1122.2**	**18.5%**
2.2.30.8	1264	**1214**	**4.0%**	1264.2	0.0%	1177.8	6.8%
2.2.30.9	**778**	854.6	-9.8%	1182	-51.9%	1123	-44.3%
2.2.30.10	**979**	994	-1.5%	1217	-24.3%	1128.4	-15.3%
3.2.50.1	2021.5	1942.4	3.9%	**1836**	**9.2%**	1967	2.7%
3.2.50.2	2581	**1718.8**	**33.4%**	1811.6	29.8%	1814	29.7%
3.2.50.3	3516	2568.4	27.0%	**2211**	**37.1%**	**2053**	**41.6%**
3.2.50.4	3260	2151.8	34.0%	**2044.4**	**37.3%**	2158.6	33.8%
3.2.50.5	1828	**1716**	**6.1%**	2206.6	-20.7%	2041	-11.7%
3.2.50.6	2577	**1580**	**38.7%**	1929.2	25.1%	1973.2	23.4%
3.2.50.7	2541	**1875**	**26.2%**	2180.8	14.2%	2111	16.9%
3.2.50.8	2365.5	**1518.4**	**35.8%**	2240.8	5.3%	1556.6	34.2%
3.2.50.9	2280.5	1755.8	23.0%	**1668**	**26.9%**	1846.8	19.0%
3.2.50.10	**1516**	1567.4	-3.4%	1699.6	-12.1%	2058.8	-35.8%

TABLE IX

COST SENSITIVITY ANALYSIS

Capacity	20% Cost	30% Cost	40% Cost	50% Cost	60% Cost	70% Cost	80% Cost	90% Cost
40%	Base Cost	5.88%	11.33%	22.70%	23.21%			
50%		Base Cost	9.52%	18.48%	21.13%	32.89%		
60%			Base Cost	6.22%	15.76%	20.66%	26.63%	
70%				Base Cost	11.95%	21.03%	28.29%	33.71%
80%				Base Cost	6.38%	15.30%	16.81%	30.63%
90%				Base Cost	2.68%	11.01%	20.74%	28.50%
100%				Base Cost	9.96%	17.78%	23.30%	30.22%

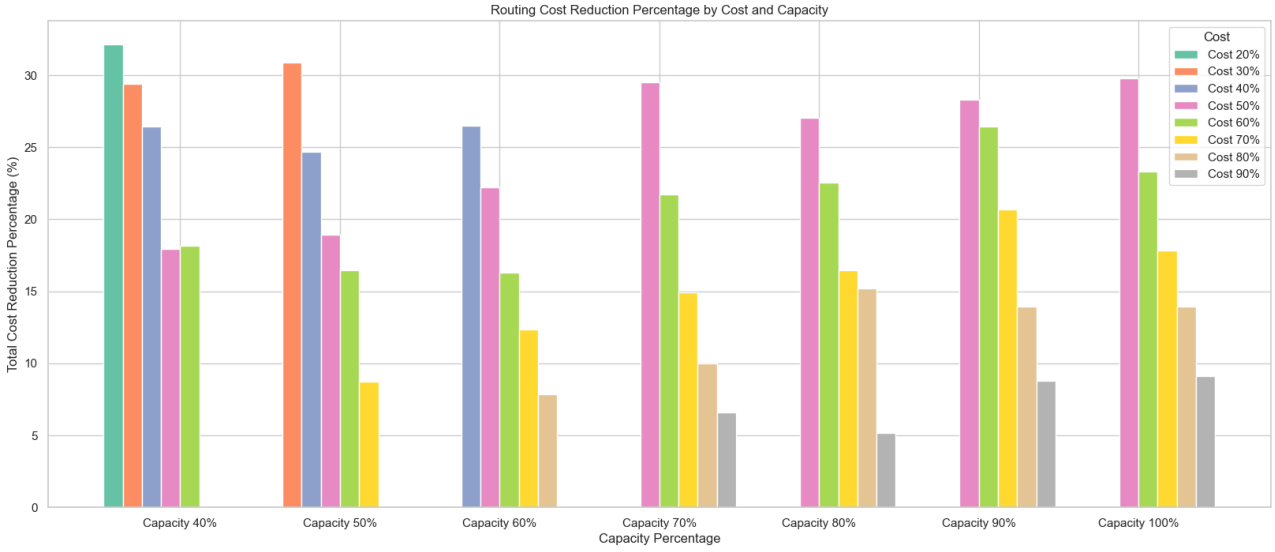

Fig. 4. Sensitivity Analysis Results

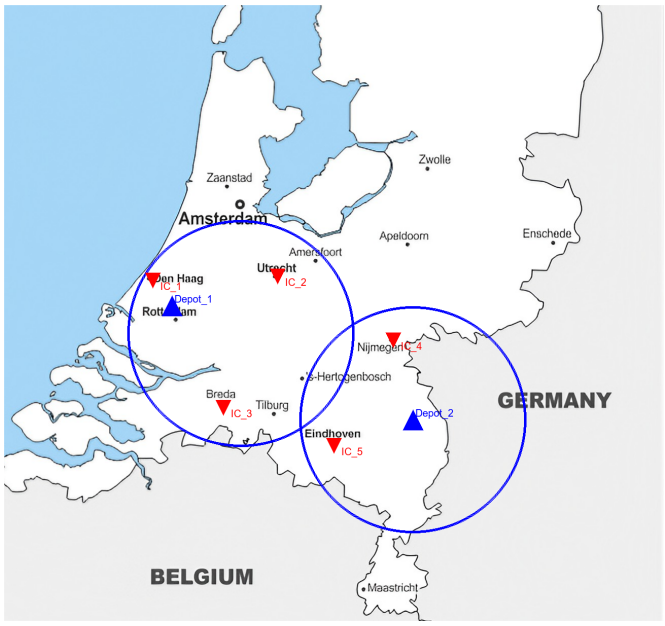

Fig. 5. Case Study Map - NL, BE and DE

positions of ICs and demand points are within a range of 80 km from the two depots.

According to research, vehicles are categorized into three types: large, medium, and small. The RDs use large vehicles, whereas ODs, typically freelance drivers, have a higher proportion of medium and small vehicles compared to large vehicles, see Figure 9. Therefore, ODs consider medium and small vehicles. The capacities of these three types of vehicles are set in certain proportions, with medium and small vehicles having cargo capacities of 60% and 45% of a large vehicle's capacity, respectively. This design is based on the prevalence of large vehicles in Dutch logistics companies, such as Ford Transit and Renault, which typically have cargo capacities of 8-9 cubic meters. Medium vehicles are modeled after Peugeot Expert and Renault models, which usually have cargo capacities of 5-6 cubic meters. Small vehicles are modeled after Caddy Cargo and Renault Cargo models, which typically have cargo capacities of 4-5 cubic meters. The capacity factors for medium and small vehicles are estimated based on their cargo capacities. Each warehouse is equipped with two RD vehicles, each with a capacity of 200, and the capacity of each parcel is randomly chosen from [2, 3, 4]. In the comparative experiment, we chose the strategy of replacing RD with OD instead of adding OD as an additional strategy. This is because the ALNS algorithm in this study prioritizes the use of OD vehicles. If the results show that the cost with sufficient use of OD vehicles is higher than that with only RD vehicles, it indicates that adding OD vehicles does not bring cost benefits to a certain extent.

Below summarizes the common characteristics of the Case Study instances:

- **Total number of requests**: 150
- **Number of requests for customer**: 50
- **Number of requests for IC**: 100
- **Number of RD in each depot**: 2
- **Number of OD (if needed)**: 2
- **Capacity RD**: 200
- **Capacity OD**: 45% and 60% Capacity RD
- **Maximum number of times OD visits the depot**: 2

Fig. 6. Large: Ford Transit

Fig. 7. Medium: Peugeot Expert

Fig. 8. Small: Caddy Cargo

Fig. 9. Comparative View of Different Cargo Vans

C. Comparative Experiment

We aim to compare employing ODs with different vehicle types and whether providing different levels of subsidies to ODs can reduce overall routing costs. For the former, we set three scenarios: no OD involvement, 2 medium-sized OD, and 2 small-sized OD. For the latter, when choosing small-sized vehicles as ODs, we set 5 cost scenarios with OD unit routing costs at 45%, 50%, 60%, 70%, and 80% of RD routing costs. When choosing medium-sized vehicles as ODs, we set 4 cost scenarios with OD unit routing costs at 50%, 60%, 70%, and 80% of RD routing costs.

We use the no OD experiment group as a baseline. We compare the total routing costs of instances using different types of vehicles and different subsidy costs with the total costs of the experimental group. The results of no OD experiment group are shown in Table X. Due to the infeasible solution of instance "2.5.150.3", there are 9 valid instances. The last row of the table gives the total routing costs of the 9 valid instances. Similarly, the results of introducing small-sized car ODs and medium-sized car ODs groups also remove instance "2.5.150.3". Their results are shown in Table XI and Table XII, respectively.

In the group where 2 RDs were replaced by 2 small-sized cars (45% RD capacity), according to the data in the last row of the table, when the cost is also 45% of the RD unit distance cost, the total routing cost decreased by 11.37%, and when the cost is 50% of the RD unit distance cost, the total routing cost decreased by 5.11%. The costs obtained for other OD costs were higher than the experimental group costs without OD. In the group where 2 RDs were replaced by 2 medium-sized cars (60% RD capacity), according to the data in the last row of the table, when the cost is also 50% of the RD unit distance cost, the total routing cost decreased by 16.57%, and when the cost is 60% of the RD unit distance cost, the total routing cost decreased by

TABLE X
CASE STUDY - RD ONLY

RD only	
Instances	**Routing Cost**
2.5.150.1	1437
2.5.150.2	1270
2.5.150.4	1388
2.5.150.5	1084
2.5.150.6	1253
2.5.150.7	1302
2.5.150.8	1360
2.5.150.9	1274
2.5.150.10	1407.4
Sum	11775.4

6.85%. The costs obtained for other OD costs were higher than the experimental group costs without OD. From the results of the two experimental groups, it can be concluded that introducing OD vehicles can optimize the total routing cost if the ratio of OD capacity to RD capacity and the unit distance cost of OD to RD are consistent. Ochama can better optimize delivery costs based on market research to balance the acceptance of OD compensation and the required number of ODs.

TABLE XI
CASE STUDY - 45% RD CAPACITY

45% Capacity					
Instances	**Cost 45%**	**Cost 50%**	**Cost 60%**	**Cost 70%**	**Cost 80%**
2.5.150.1	1302.2	1401.5	1498.8	1630.6	1718.6
2.5.150.2	1077.9	1137	1233.6	1268.2	1594.2
2.5.150.4	1142.2	1228	1265.8	1327	1611.6
2.5.150.5	1148.5	1201.5	1382.2	1450.6	1592.4
2.5.150.6	1197.2	1249.5	1406.6	1501.8	1574.6
2.5.150.7	1099	1268	1316.4	1427.4	1531
2.5.150.8	1163.3	1229.5	1335	1449.5	1565.6
2.5.150.9	1193.1	1242.5	1340.2	1325.6	1566.4
2.5.150.10	1112.7	1216	1291.4	1462.9	1496.4
Sum	10436.1	11173.5	12070	12843.6	14250.8
Gap	11.37%	5.11%	-2.50%	-9.07%	-21.02%

TABLE XII
CASE STUDY - 60% RD CAPACITY

60% Capacity				
Instances	**Cost 50%**	**Cost 60%**	**Cost 70%**	**Cost 80%**
2.5.150.1	1111	1264.4	1380	1431.6
2.5.150.2	1203	1305.8	1472	1532.4
2.5.150.4	1263	1389.2	1582.9	1645.4
2.5.150.5	956	1106.6	1182.2	1266
2.5.150.6	1261	1324.4	1467.7	1595
2.5.150.7	901	1030.4	1099.8	1165.6
2.5.150.8	1275	1446.6	1618.2	1789.8
2.5.150.9	974	1090.4	1229.7	1366.4
2.5.150.10	880	1011	1157.7	1253.2
Sum	9824	10968.8	12190.2	13045.4
Gap	16.57%	6.85%	-3.52%	-10.79%

Figure 10 and 11 present the visualized results of all instances' costs relative to the baseline cost without OD, with the addition of small-sized OD and middle-sized OD, respectively. In each figure, the grey dashed line represents the baseline without OD consideration. When OD is introduced, data points above the baseline indicate cost savings in route expenses, whereas points below the baseline indicate higher route costs. The x-axis represents each instance, and each differently colored line corresponds to different OD cost settings. The figures provide a clear visualization of whether cost savings occur for each instance across different OD cost values.

VII. CONCLUSIONS AND FUTURE RESEARCH

A. CONCLUSIONS

This research addresses the need for models that integrate multiple depots, occasional drivers, and multiple depot visits, features often overlooked in existing studies. We developed a mathematical model and applied an Adaptive Large Neighborhood Search (ALNS) algorithm to address the routing problem, aiming to minimize costs efficiently. Our literature review highlighted various algorithms for the Pickup and Delivery Problem with Time Windows (PDPTW), Crowdshipping, and Open Vehicle Routing Problem (OVRP), including ALNS, Variable Neighborhood Search (VNS), and Integer Linear Programming (ILP). We found ALNS to be particularly effective for our problem, offering a robust solution through a combination of random and worst removal operators, and Basic Greedy and Regret-2 insertion strategies. The use of a roulette wheel method for operator selection and simulated annealing for acceptance criteria further improved solution quality. Computational experiments validated the model and ALNS algorithm, demonstrating their effectiveness. Parameter tuning and sensitivity analysis revealed significant cost savings with the introduction of additional depots (ODs). Specifically, introducing an OD with an 80% cost relative to regular depots (RDs) can reduce total routing costs by up to 15.21%. Varying OD capacities also showed potential savings, with 60% capacity ODs saving 7.86%, and 70% capacity ODs saving 9.97%. A case study with the Dutch e-commerce company Ochama, scaled to 150 requests, confirmed the practical applicability of our approach. Out of ten instances, nine were valid. Our method achieved total routing cost reductions of 11.37% for small vehicles and 6.85% for medium-sized vehicles, outperforming previous research which reported a 7.3% reduction. Overall, our findings indicate that integrating ODs into routing strategies can substantially lower costs. The choice of the number, cost, and capacity of ODs should be guided by market research to optimize cost savings while balancing service quality and OD willingness.

B. Future Research

This study addresses the proposed a more realistic pickup and delivery problem through modeling and algorithm development. However, this is the beginning. This section highlights some areas for improvement in this study and suggests directions for future research.

- **Improving the Mathematical Model**
  This study employs an Arc-based Formulation, which

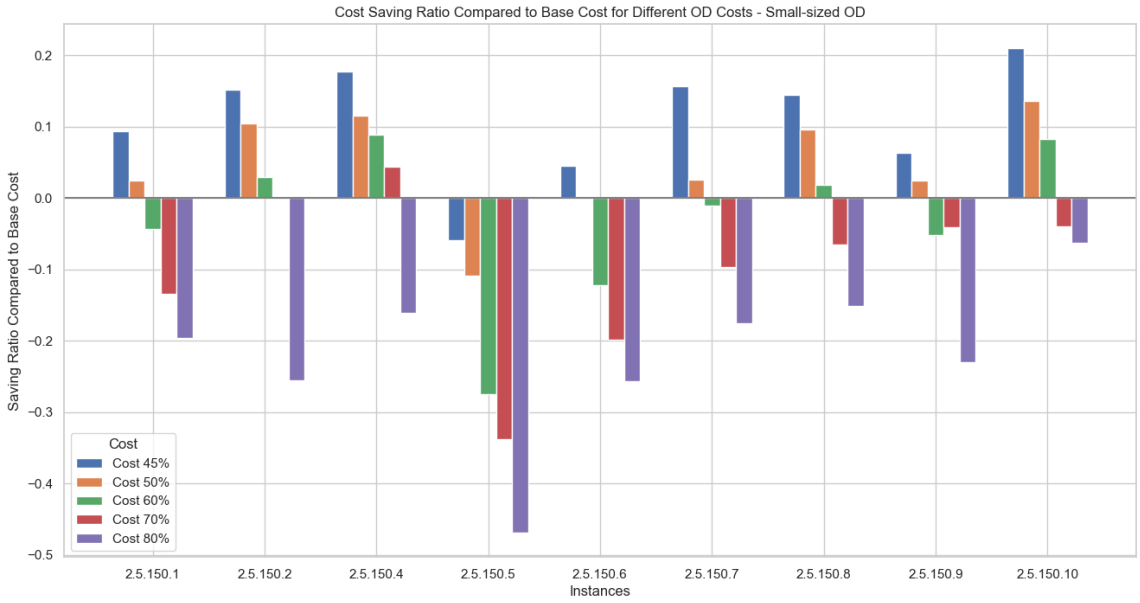

Fig. 10. Results for adding Small-sized OD

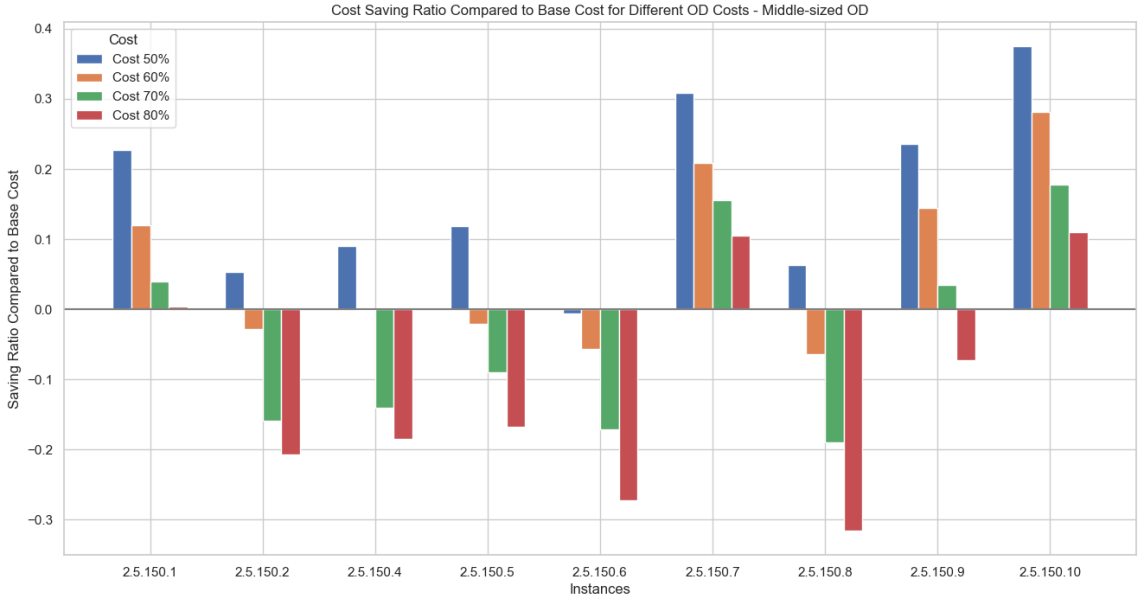

Fig. 11. Results for adding Middle-sized OD

provides detailed path analysis but increases the number of constraints. Exploring alternative modeling approaches, such as Node-based or Path-based Formulations, or Column Generation, might reduce constraints and solution time.

- **Further Optimization of Route Cost**
  While the ALNS algorithm provides good solutions, it may only yield local optima for large problems. Using the best solution from ALNS as input for exact algorithms like branch and bound could help in exploring the solution space further and achieving better results.
- **Adding More Operators**
  Currently, the ALNS algorithm uses only two repair and destruction operators. Incorporating additional operators could enhance selectivity and improve algorithm performance.
- **Adjusting Destruction Coefficient Strategy**
  The destruction coefficient κ is set to 0.1, which works well for fewer requests. For larger instances, adjusting κ in relation to the number of requests can help avoid excessive removal, speeding up the repair process and improving solution exploration.
- **Application to Real-world Problems**
  This study used simulated demand points. Future work

should apply these methods to real-world scenarios, considering actual demand volumes and market conditions. Companies can use this research to determine optimal OD types and pricing strategies based on market research and driver preferences.

REFERENCES

[1] Claudia Archetti, Martin Savelsbergh, and M Grazia Speranza. The vehicle routing problem with occasional drivers. *European Journal of Operational Research*, 254(2):472–480, 2016.

[2] Alp M Arslan, Niels Agatz, Leo Kroon, and Rob Zuidwijk. Crowdsourced delivery—a dynamic pickup and delivery problem with ad hoc drivers. *Transportation Science*, 53(1):222–235, 2019.

[3] Simen Braaten, Ola Gjønnes, Lars Magnus Hvattum, and Gregorio Tirado. Heuristics for the robust vehicle routing problem with time windows. *Expert Systems with Applications*, 77:136–147, 2017.

[4] José Brandão. A tabu search algorithm for the open vehicle routing problem. *European Journal of Operational Research*, 157(3):552–564, 2004.

[5] Erbao Cao and Mingyong Lai. The open vehicle routing problem with fuzzy demands. *Expert Systems with Applications*, 37(3):2405–2411, 2010.

[6] Erbao Cao, Mingyong Lai, and Hongming Yang. Open vehicle routing problem with demand uncertainty and its robust strategies. *Expert Systems with Applications*, 41(7):3569–3575, 2014.

[7] Valentina Carbone, Aurélien Rouquet, and Christine Roussat. The rise of crowd logistics: a new way to co-create logistics value. *Journal of Business Logistics*, 38(4):238–252, 2017.

[8] Lars Dahle, Henrik Andersson, Marielle Christiansen, and M Grazia Speranza. The pickup and delivery problem with time windows and occasional drivers. *Computers & Operations Research*, 109:122–133, 2019.

[9] Iman Dayarian and Martin Savelsbergh. Crowdshipping and same-day delivery: Employing in-store customers to deliver online orders. *Production and Operations Management*, 29(9):2153–2174, 2020.

[10] Luigi Di Puglia Pugliese, Daniele Ferone, Paola Festa, Francesca Guerriero, and Giusy Macrina. Solution approaches for the vehicle routing problem with occasional drivers and time windows. *Optimization Methods and Software*, 37(4):1384–1414, 2022.

[11] Anhai Doan, Raghu Ramakrishnan, and Alon Y Halevy. Crowdsourcing systems on the world-wide web. *Communications of the ACM*, 54(4):86–96, 2011.

[12] Yvan Dumas, Jacques Desrosiers, and Francois Soumis. The pickup and delivery problem with time windows. *European journal of operational research*, 54(1):7–22, 1991.

[13] Krzysztof Fleszar, Ibrahim H Osman, and Khalil S Hindi. A variable neighbourhood search algorithm for the open vehicle routing problem. *European Journal of Operational Research*, 195(3):803–809, 2009.

[14] Zhuo Fu, Richard Eglese, and Leon YO Li. A new tabu search heuristic for the open vehicle routing problem. *Journal of the operational Research Society*, 56(3):267–274, 2005.

[15] Veaceslav Ghilas, Emrah Demir, and Tom Van Woensel. An adaptive large neighborhood search heuristic for the pickup and delivery problem with time windows and scheduled lines. *Computers & Operations Research*, 72:12–30, 2016.

[16] Vera C Hemmelmayr, Jean-François Cordeau, and Teodor Gabriel Crainic. An adaptive large neighborhood search heuristic for two-echelon vehicle routing problems arising in city logistics. *Computers & operations research*, 39(12):3215–3228, 2012.

[17] Shixuan Hou and Chun Wang. Matching models for crowd-shipping considering shipper's acceptance uncertainty. In *2021 IEEE International Conference on Autonomous Systems (ICAS)*, pages 1–6. IEEE, 2021.

[18] Kuancheng Huang and Muhammad Nashir Ardiansyah. A decision model for last-mile delivery planning with crowdsourcing integration. *Computers & Industrial Engineering*, 135:898–912, 2019.

[19] Stefan Irnich. A multi-depot pickup and delivery problem with a single hub and heterogeneous vehicles. *European Journal of Operational Research*, 122(2):310–328, 2000.

[20] Nabin Kafle, Bo Zou, and Jane Lin. Design and modeling of a crowdsource-enabled system for urban parcel relay and delivery. *Transportation research part B: methodological*, 99:62–82, 2017.

[21] Gilbert Laporte, Roberto Musmanno, and Francesca Vocaturo. An adaptive large neighbourhood search heuristic for the capacitated arc-routing problem with stochastic demands. *Transportation Science*, 44(1):125–135, 2010.

[22] Haibing Li and Andrew Lim. A metaheuristic for the pickup and delivery problem with time windows. In *Proceedings 13th IEEE International Conference on Tools with Artificial Intelligence. ICTAI 2001*, pages 160–167. IEEE, 2001.

[23] Giusy Macrina, Luigi Di Puglia Pugliese, Francesca Guerriero, and Demetrio Laganà. The vehicle routing problem with occasional drivers and time windows. In *Optimization and Decision Science: Methodologies and Applications: ODS, Sorrento, Italy, September 4-7, 2017 47*, pages 577–587. Springer, 2017.

[24] Setyo Tri Windras Mara, Rachmadi Norcahyo, Panca Jodiawan, Luluk Lusiantoro, and Achmad Pratama Rifai. A survey of adaptive large neighborhood search algorithms and applications. *Computers & Operations Research*, 146:105903, 2022.

[25] SA MirHassani and N Abolghasemi. A particle swarm optimization algorithm for open vehicle routing problem. *Expert Systems with Applications*, 38(9):11547–11551, 2011.

[26] William P Nanry and J Wesley Barnes. Solving the pickup and delivery problem with time windows using reactive tabu search. *Transportation Research Part B: Methodological*, 34(2):107–121, 2000.

[27] Maciek Nowak, Özlem Ergun, and Chelsea C White III. Pickup and delivery with split loads. *Transportation science*, 42(1):32–43, 2008.

[28] David Pisinger and Stefan Ropke. A general heuristic for vehicle routing problems. *Computers & operations research*, 34(8):2403–2435, 2007.

[29] Jean-Yves Potvin and Jean-Marc Rousseau. A parallel route building algorithm for the vehicle routing and scheduling problem with time windows. *European Journal of Operational Research*, 66(3):331–340, 1993.

[30] Luigi Di Puglia Pugliese, Daniele Ferone, Paola Festa, Francesca Guerriero, and Giusy Macrina. Combining variable neighborhood search and machine learning to solve the vehicle routing problem with crowd-shipping. *Optimization Letters*, pages 1–23, 2022.

[31] Panagiotis P Repoussis, Christos D Tarantilis, and George Ioannou. The open vehicle routing problem with time windows. *Journal of the Operational Research Society*, 58(3):355–367, 2007.

[32] Stefan Ropke and David Pisinger. An adaptive large neighborhood search heuristic for the pickup and delivery problem with time windows. *Transportation science*, 40(4):455–472, 2006.

[33] Majid Salari, Paolo Toth, and Andrea Tramontani. An ilp improvement procedure for the open vehicle routing problem. *Computers & Operations Research*, 37(12):2106–2120, 2010.

[34] Afonso Sampaio, Martin Savelsbergh, Lucas P Veelenturf, and Tom Van Woensel. Delivery systems with crowd-sourced drivers: A pickup and delivery problem with transfers. *Networks*, 76(2):232–255, 2020.

[35] Dimitrios Sariklis and Susan Powell. A heuristic method for the open vehicle routing problem. *Journal of the Operational Research Society*, 51:564–573, 2000.

[36] Aişe Zülal Şevkli and Bekir Güler. A multi-phase oscillated variable neighbourhood search algorithm for a real-world open vehicle routing problem. *Applied Soft Computing*, 58:128–144, 2017.

[37] Paul Shaw. A new local search algorithm providing high quality solutions to vehicle routing problems. *APES Group, Dept of Computer Science, University of Strathclyde, Glasgow, Scotland, UK*, 46, 1997.

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

[39] Yi Tao, Haibing Zhuo, and Xiaofan Lai. The pickup and delivery problem with multiple depots and dynamic occasional drivers in crowdshipping delivery. *Computers & Industrial Engineering*, 182:109440, 2023.

[40] CD Tarantilis and CT Kiranoudis. Distribution of fresh meat. *Journal of Food Engineering*, 51(1):85–91, 2002.

[41] Christos D Tarantilis, Danae Diakoulaki, and Chris T Kiranoudis. Combination of geographical information system and efficient routing algorithms for real life distribution operations. *European Journal of Operational Research*, 152(2):437–453, 2004.

[42] Christos D Tarantilis, George Ioannou, Chris T Kiranoudis, and Gregory P Prastacos. Solving the open vehicle routeing problem via a single parameter metaheuristic algorithm. *Journal of the Operational research Society*, 56:588–596, 2005.

[43] Fabian Torres, Michel Gendreau, and Walter Rei. Crowdshipping: An open vrp variant with stochastic destinations. *Transportation Research Part C: Emerging Technologies*, 140:103677, 2022.

[44] HRG Van Landeghem. A bi-criteria heuristic for the vehicle routing problem with time windows. *European Journal of Operational Research*, 36(2):217–226, 1988.

[45] Stefan Voigt and Heinrich Kuhn. Crowdsourced logistics: The pickup and delivery problem with transshipments and occasional drivers. *Networks*, 79(3):403–426, 2022.