VIETNAM NATIONAL UNIVERSITY – HO CHI MINH CITY
INTERNATIONAL UNIVERSITY

SCHOOL OF INDUSTRIAL ENGINEERING & MANAGEMENT

CAPACITATED MULT-ALLOCATION HUB LOCATION ROUTING PROBLEM: A CASE STUDY OF VIETTEL POST

Submitted in partial fulfillment of the requirements for the Degree of Bachelor of Engineering in Logistics and Supply Chain Management

Student: LY THUC VAN

ID: IELSIU19306

Thesis Advisor: Assoc. Prof. NGUYEN VAN HOP

Ho Chi Minh City, Vietnam

August 2023

CAPACITATED MULTI-ALLOCATION HUB LOCATION ROUTING PROBLEM: A CASE STUDY OF VIETTEL POST

Submitted in partial fulfillment of the requirements for the Degree of Bachelor of Engineering in Logistics and Supply Chain Management

Student: LY THUC VAN

ID: IELSIU19306

Thesis Advisor: Assoc. Prof. Nguyen Van Hop

Ho Chi Minh City, Vietnam

August 2023

CAPACITATED MULTI-ALLOCATION HUB LOCATION ROUTING PROBLEM: A CASE STUDY OF VIETTEL POST

By

LY THUC VAN

Submitted in partial fulfillment of the requirements for the Degree of Bachelor of Engineering in Logistics and Supply Chain Management

International University, Ho Chi Minh City
August 2023

Signature of Student: _____

Ly Thuc Van

Certified by _____

Thesis Advisor: Assoc. Prof. Nguyen Van Hop

Approved by _____

Dean of IEM school: Assoc. Prof. Nguyen Van Hop

Abstract

The current research mentions the ineffective delivery rate of Viettel Post recently due to a prominent factor of long shipment duration or high distribution's travel time which is possibly defined by the large distance between hub locations and assigned post offices. Thus, a hub location problem is established to further investigate the optimal locations in the hub potential set along with the non-hub nodes' multi-allocation and their corresponding routing between those assigned non-hubs and hubs. The multi-allocation mechanism allows the non-hub nodes to be allocated to at least one hub which is much practical in real scenarios. The expected outcome considers the number of hubs to be opened, the locations and equivalent allocations, and vehicle routing to the problem with the goal of operation network costs minimization.

A Mixed Integer Linear Programming (MILP) is proposed to obtain the exact solutions and a metaheuristic – Adaptive Large Neighborhood Decomposition Search (ALNDS) is then applied for validation whether which method gives more high-quality performance. Furthermore, the six datasets with small, medium, and large-sized instances are generated to better evaluate the logic and accuracy of the mathematical model as well as compare the corresponding sizes with the algorithm. The results present the ALNDS yields more optimal solutions under shorter computational time for the medium to large-sized instances and slightly equal to or bigger than the exact solutions in small-sized instances. In addition, the sensitivity analysis on both the unit transportation costs and vehicle capacity along with the fleet size delivers higher costs when increasing those parameters and vice versa.

Keywords:

Hub location-routing problem, multiple allocations, exact method, ALNDS.

Acknowledgements

Comprehending the graduate thesis is the final challenge on the journey of graduation, the implementation duration is thus strongly not a comfortable, pleasant, and straightforward path. I want to present appreciation and gratitude to life and to all beloved people and companions for believing in me and mentally support during this tough journey.

First, without the wholeheartedly continuous and enthusiastic support of my thesis advisor, Assoc. Prof. Nguyen Van Hop, throughout the course, I myself cannot fully overcome the challenging thesis. I deeply respect and am thankful for his expertise and mental support even in the seemingly stuck and impassable situation, especially in formulating the model and adjusting coding aspects. Thanks to his genuine and helpful advice on my coping problem, I can finally modify and obtain the outcome and thus learn from mistakes and study way-findings for those stuck circumstances.

Second, I want to thank all my sincere and beloved friends from high school and new ones at university for creating beautiful memories and mentally supporting each other. They have consistently advised and given me helpful tips and mindsets to help me deliver a much more positive outlook in this darkest time with multiple pressures in life and studying. I am also grateful to the other friends who are distant but are always there to listen and share about the hardships we have encountered.

Third, I am grateful and treasure my family, especially my parents for laying their belief in me and encouraging me in completing the final thesis no matter what the result. I truly love and cherish my family for always being there for me until the end.

Finally, I am proud and grateful to myself for not giving up, strictly fighting till the end with all the accumulative problems throughout my university journey.

Table of Contents

List of Tables

List of Figures

List of Abbreviations

ACO	Ant Colony Optimization
AMOSA	Archieved multi-objective simulated annealing
BD	Bender Decomposition
EEA	Endosymbiotic Evolutionary Algorithm
ILS	Iterative Large Search
HLP	Hub location problem
HLRP	Hub location – routing problem
MAHLRP	Multi-allocation Hub location – routing problem
MSSA	Multi-start Simulated Annealing
NSGA - II	Non-dominated sorting genetic algorithm
SAHLRP	Single-allocation Hub location – routing problem
SA	Simulated Annealing
TS	Tabu Search
VNDS	Variable Neighborhood Decomposition Search

Chapter 1: INTRODUCTION

1.1 Background

In the parcel and postal services, all activities must be covered in three stages including first, middle, and last-mile deliveries. First-mile delivery is the process of receiving goods from senders to the goods collection point. Followingly, middle-mile delivery entails transportation to the gathering point (receiving hub/ sub), sortation, and transfer to the delivery point (hub/ sub delivery). The last mile will be in charge of picking up the goods from the delivery point and delivering them to the consignee.

Figure 1.1 General, first, middle and last-mile delivery process

Last-mile delivery has been gradually playing a decisive role in the customer service quality management strategy as it directly affects customer experience satisfaction from the first time they receive the product, particularly in the burgeoning industry of E-commerce. Given the Vietnamese people's desire for amusement, E-commerce emerges as a solution for huge organizations to generate sales and interact with customers, and last-mile delivery is one of the keys to success. The last-mile delivery is the final part of the delivery process involving a series of activities and processes required for transporting goods from the point of final transshipment to the last drop-off point in the delivery chain. Last-mile delivery expenses, on the other hand, account for 53% of total e-commerce expenditures, and 39% of customers are willing to switch suppliers if they do not obtain acceptable last-mile delivery services [1]. There are a number of studies that have proposed criteria for assessing the quality of delivery services last mile such as on-time delivery rate, cargo damage rate, service flexibility,

service price, service delivery speed, etc. As expedited shipping becomes the new practice, users tend to expect their orders to be delivered the same day or the next day. Nonetheless, achieving this fast-paced transportation requires the companies to plan the last mile in detail for optimal delivery efficiency to keep customers engaged.

The logistics delivery network contributes a substantial role in the effective last-mile delivery operation entailing Hubs and branch offices, or specifically post offices in postal industry. In reality, the usage of hubs can result in cheaper network costs, but determining where hubs should be situated and how users should be assigned to them can be difficult. Also, the cost of delivery/ pickup to or from the consumers for these systems is connected to vehicle routing (Tuzun & Burke, 1999) [6]. The location-routing problem (LRP) also incorporates depot placement and vehicle routing considerations. Thus, optimizing hub location problems (HLPs) for more instant last-mile deliveries and related processes is the requirement for a successful and thriving postal services future.

Viettel Post (VTPost) has concentrated on investing in last-mile delivery with a long-term vision and an emphasis on sustainable growth. VTPost has become one of the main delivery brands in Vietnam with a delivery network covering 100% of countrywide provinces and cities [2]. VTPost is recognized as a sustainable development company and is the sole enterprise in Vietnam in the postal delivery to overseas markets and international integration. Nonetheless, the low delivery success rate of VTPost has raised concerns for the enterprise in recent years and yet effectively solved the problem leading to great financial loss. The faster the delivery process, the more likely a customer will stay loyal to the brand.

Some qualitative and quantitative factors have been assessed for their impact on the decreased successful delivery rate and the high return rate. For qualitative elements, the unsuccessful delivery rate can be obtained from the sender's mistakes (with size, color, etc), government or ineffective operational policy, inconsistent delivery demand, consignee's obscure address, poor customer service, shippers' motivation, etc. The measurement of quantitative factors is also critically taken into consideration

with the failed committed service quality (i.e. delivery and receiving's speed and quality have not followed the required KPI, postal connection time increases, KPI segments are unimplemented resulting in error identification difficulty in each stage, etc), unavailable interactive real-time system between users and shippers, fluctuated fuel prices, unoptimized hub location and transportation routing, ineffective Strategic Business Unit (SBU) model, and so on. Thus, to sufficiently analyze and select the most optimal and cost-effective location for smooth and quick goods distribution with the aim of improving the successful delivery rate, the capacitated multiple hub alocation-routing problem for the simultaneous pickup and delivery problem is investigated.

1.2 Problem Statement – The Need of Study

1.2.1 The gap needed to fill in the case

Since the E-commerce strong growth in Vietnam, the number of goods surged in 2016 – 2018; nevertheless, VTPost quality afterward has been slow down despite the improved quality in three stages with the first, middle, and last mile. While other competitors are executing well, VTPost has been facing a low delivery success rate and the high return rate.

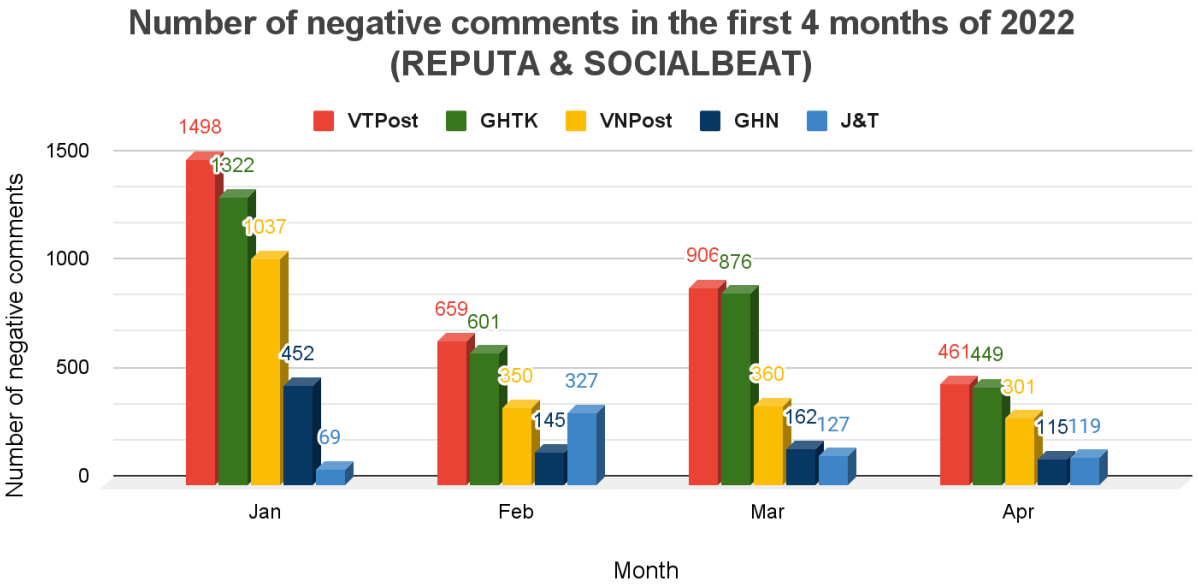
Table 1.1 Distinct criteria rates of VTPost, GHTK, and GHN up to 2022

Criteria	VTPost	GHTK	GHN
Successful E-commerce delivery rate	86,15%	90,24%	91,20%
E-commerce return rate	10%	6,60%	8,60%
Average parcel process delivery time	110 hr	50 hr	55 hr
Average delivery speed at Ha Noi & Ho Chi Minh (same day delivery service)	20 - 21 hr	8 hr	12 hr

Note: Process delivery time is measured from the customer's order creation. Particularly

for VTPost, it is measured when the order is successfully received (converted to 12h and 24h milestones).

Source: Viettel Post



Source: Viettel Post

Figure 1.2 Number of negative comments for VTPost, GHTK, VNPost, GHN, and J&T in the first 4 months of 2022

According to VTPost, the successful E-commerce delivery rate is comparatively lesser by 4 - 5% than that of other competitors such as GHTK and GHN, meanwhile, the return rate and average parcel delivery time have been higher compared to GHTK and GHN, specifically the delivery speed. These key quality indicators in the logistics field of VTPost underperform the other postal enterprises. Additionally, negative feedback on social networks in the first 4 months of 2022 of VTPost is also at a higher level compared to other enterprises in the postal industry. Since the delivery rate is one of the uttermost concerns of VTPost, acquiring a successful one can be challenging depending on those mentioned above quantitative and qualitative factors.

Tailored to this paper, the approach for the problem aims at optimized technical solutions; thus, the quantitative factors are studied, mainly the hub location, regional

demand, and vehicle routes. The travel time from senders or post offices to Megahub or Hub requires long-distance leading to frequently about 6 to 12 hours, which is significantly not an optimal location routing problem. As claimed by Viettel Post, the average process delivery time of VTPost was twofold than that of other competitors, indicating insufficient management in the last-mile delivery. One dominant reason for this originates from the non-optimal locations of hubs. The remote Hub location to Sub causes rapid delivery time management struggles since the parcel has to transfer from the receiving point known as Sub to Hubs for consolidation and sortation moving to the destination Subs which lately distributes to specific districts or wards. After solving the problem of hub covering location which resulted in five potential hub locations in Lam Dong, Binh Phuoc, Kien Giang, Ben Tre, and Bac Lieu. Therefore, the next considered issue is solution for the allocation and corresponding routing flow for simultaneous collection and delivery.

1.2.2 What needs to be solved or achieved?

The hub network optimization in the postal industry has been a difficult problem where cost and travel time minimization is placed as the top priority. Thus, the objective of this paper is to propose a Mixed-Integer Linear Programming Model to reduce the total costs including inter-hub transportation costs, local-tour, fixed arc and employed vehicle costs between Origin-Destination (O-D) pairs while ensuring particular restraints with the demand in the studied region, hub and vehicle capacity as well as the travel flows specified among the nodes in the network topology. The allocation with exact coordinates is illustrated on the map satisfying the needed covered demand (non-hub nodes). Therefore, the delivery time issue can be mitigated since the strategic hub points are established for distance reduction from post offices while satisfying the needs which helps improve the consumers' attitude towards VTPost services.

1.3 The Design Study Objectives and Requirements

1.3.1 The benefitted stakeholders with the results of this study

- VTPost itself: The enterprise benefits from the study in controlling the smooth and quick movement of the parcel distribution network from the senders to recipients. This could serve as a primary tool for the enterprise in allocation-routing optimization and cost reduction in last-mile delivery and help expand the market share.

- Customer: The study's purpose is to establish the allocation-routing solution improving the process delivery time from senders to receivers. Thus, consumers can obtain the goods on time, enhancing their attitudes and loyalty.

- Logistics companies: In real-life scenarios, the allocation-routing in the delivery network model decision are thoroughly and frequently considered and evaluated by logistics companies to better increase service level for competitive remaining in this industry. Hence, they can reference the approach in modeling and utilize proper techniques to solve based on their situation.

- HLRPs researchers: Through this study concerning the capacitated multiple hub allocation-routing problem of VTPost in finalizing the optimal allocations and routing flows, the researchers are able to validate the result's effectiveness and suggest more enhanced algorithms to solve the model better. Grasping the overall conceptual model, they can develop and apply it to solve medium to large size problems in business relations.

1.3.2 Expected output and application

The paper's desired output aims to redesign the distribution supply chain network consisting of Hubs, Post offices by assigning demand nodes to the defined Hubs, and the flow quantity from the supply to the received point, constructing the optimized network topology, routing the demand flow to meet the needs between the O-D pairs that were chosen to be supplied from the proposed obtained new Hub locations in

Southern region in order to minimize the total costs and travel time in the Southern zone by considering the limited capacity of each Hub node, vehicle, regional demand, cost, distance, etc. Accordingly, the managers can better manage the goods flow from the first-mile to last-mile delivery for high demand satisfaction and thus enhance the service level requirement in tackling order movement.

Deteriorating transportation infrastructure, security issues, and the inescapable occurrence of natural disasters and traffic events have put the importance of better mathematical modeling for delivery systems. The significance of the logistics delivery network establishment requires the company to continuously research and update the region's characteristics towards infrastructure, set up costs, the density of inhabitants, businesses, licensing permits, and covered network of other competitors, etc to adjust and propose potential alternatives timely.

1.3.3 Design requirements

The study system design entails significant comprehension of the delivery network model of business, functions of each component, clear input and output definition, optimization techniques, and required constraints for solving allocation-routing problem with a view to have cost-effectiveness and shorten travel time among the hub and non-hub nodes.

1.4 Scope and Limitation

1.4.1 Scope

This problem likewise takes into account the multiple allocation technique imposes capacity limitations, and assumes that establishing hubs and hub arcs has fixed costs.

Considering the case of VTPost about the logistics network delivery design, the scopes, and limitations are defined for a specific solution:

- Five potential hub locations are considered which are in Lam Dong, Binh Phuoc, Kien Giang, Ben Tre, Long An.

- The investigated region: the southern region from Lam Dong to Ca Mau province including 22 provinces: Dak Nong, Lam Dong, Binh Thuan, Binh Phuoc, Dong Nai, Ba Ria – Vung Tau, Tay Ninh, Binh Duong, Ho Chi Minh, Long An, Tien Giang, Ben Tre, Dong Thap, Vinh Long, Tra Vinh, An Giang, Can Tho, Kien Giang, Hau Giang, Soc Trang, Bac Lieu, Ca Mau to redesign the downstream supply chain network for time and cost-saving distribution.

- The level number: the investigated paper considers one level from Hub to non-hub nodes, which are the post offices.

    + Hub: the Distribution Center or Hub is in charge of the goods sorting, consolidation, and delivery.

    + Post office: the Post offices are relatively defined as the Regional Warehouse but comprise receiving, distribution, and coordination processes.

- The interconnection network: the multiple-allocation is examined in this paper where the post offices are assigned to multiple near-located hubs with full interconnection among Hub nodes.

- Goods: Those are small, unbreakable and easy to transport, however, no required for specific type of goods. The commodity is grouped into 5kg/ homogeneous pack.

- Vehicle: the homogeneous vehicle is considered with specific capacity and assumed to be available instead of outsourcing.

- Capacity: Total collection and the total delivery load of nodes assigned to a hub do not exceed the hub's capacity.

- Historical demand: from April to June in 2022.

- Shipper: full-time driver of Viettel Post.

1.4.2 Limitation

- No direct connections between non-hub nodes are allowed due to the characteristics of the warmed-up transportations system studied for this type of problem.
- One echelon is taken into consideration from Hub to Post Offices.
- Data size is not large enough to tackle for practical cases.
- External conditions are not studied such as time windows, traffic congestion, surrounding environment to establish new hub, vehicle availability, etc.

Chapter 2: RELATED WORKS

2.1 Viettel Post Overview

Viettel Post is a leading enterprise that provides domestic and international express delivery services of parcels in Vietnam. Regarding infrastructure investment, VTPost aims to build a logistics network to invest in warehousing activities (fulfillment, distribution), connection, and division. The advantage of Viettel Logistics is the network and infrastructure to form the BACKBONE national infrastructure with the Mega Hub - Sub - Post Office system spanning 63 provinces across the country. Concerning means of transportation serving logistics activities, in addition to new self-invested vehicles, VTPost aims to outsource to optimize costs, and simultaneously promote the development of the MyGo Multimodal Transport platform. It optimizes the performance of Viettel Post's existing trucks and serves as a transportation service business platform for truck partners and customers.

Up to now, Viettel Post Corporation has owned 63 branches, 09 functional departments, 07 sorting centers, 05 successful companies members, 03 central warehouses, 96 provincial warehouses providing 209,000m^2 warehouse capacity, 1,120 post offices, 819 telecom stores, 771 self-owned trucks, and 550 contracted trucks. The enterprise has currently delivered 4 main service sectors comprising domestic, international services, Logistics, and E-commerce services, which are illustrated in the following figure:

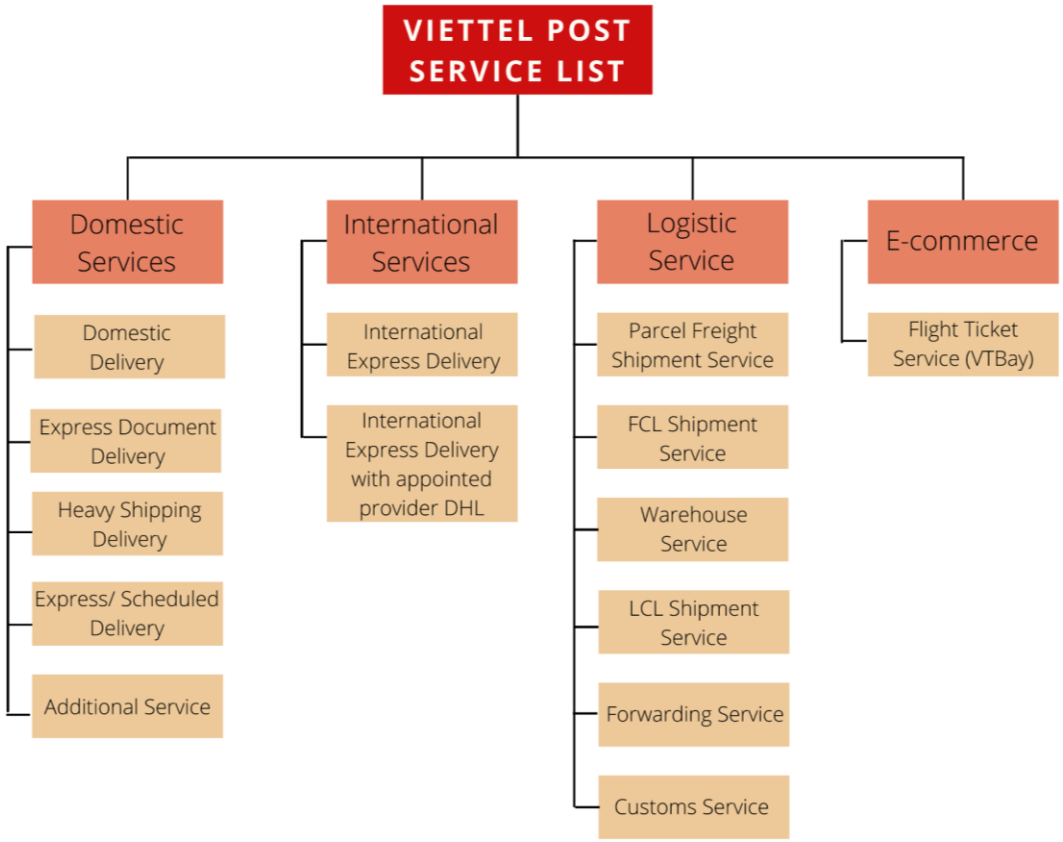
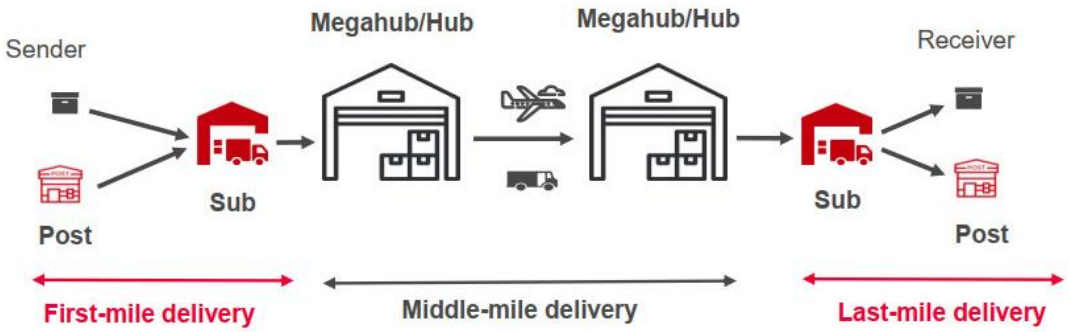


Figure 2.1 Viettel Post's service list



Source: Viettel Post

Figure 2.2 Viettel Post Logistics Delivery Network

VTPost's current procedures for handling goods from senders to recipients are presented as follows:

First, the customer (sender) creates an order on the Viettel Post app or website. Then, the sender can either choose to get a parcel collected at home or go to the post office to

send it. Additionally, senders can also send goods directly to the Sub instead of moving through the intermediary side. In this step, there will be two cases with order handling:

- If the order is over the time limit and has not been processed, the post office is in charge of directing the shippers to handle the order urgently.

- The order is under the time limit.

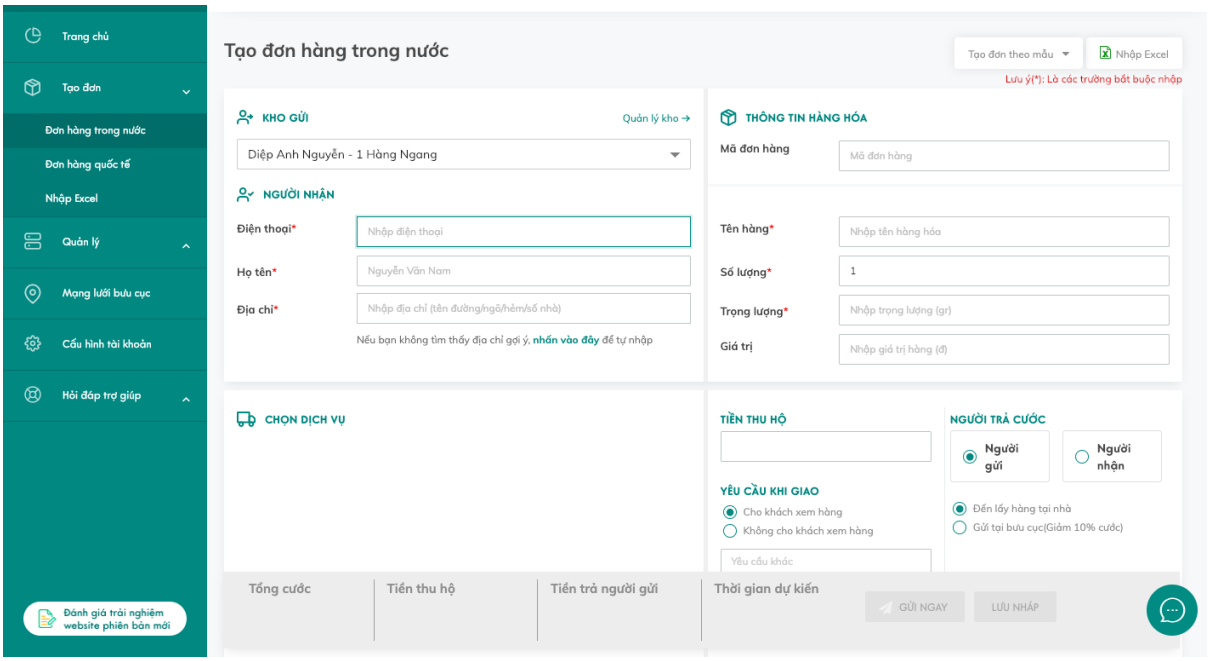


Figure 2.3 Order formation with Viettel Post on the website

Now, the order has been delivered for goods receipt. The salesperson prints the bill of consignment or supports printing with the request of the shippers to pick up the customer's goods. In case the salespersons are unavailable to print, checking the address, phone number, and order data is mandatory. Then, they have to sign the client’s and their receipt book.

Figure 2.4 Viettel Post's bill of consignment

There are three cases for order processing:

- The order has been received successfully: The shippers will update the status of receipt of the goods on the system.
- The order has been rejected: The order is refused due to one of the following reasons:
    + Order good is a prohibited item.
    + Customer packaging is not guaranteed.
    + Customers are not at home.
    + Customer is unable to contact.
    + Customers make an appointment to pick it up at another time.
    + The content and weight of goods are incompatible between the actual and the bill of consignment.

    + The sales staff updated other reasons on the system for not receiving the goods.

- Partner requests to take back the goods: Those are the rejected orders for the above-mentioned reasons. When the customer has finished preparing and is available, they can ask the shippers to receive the order again.

- The order requires cancellation from the partner: When the customers have no need to send the goods anymore, they cancel orders on the system or contact the post office directly.

Second, the order has been processed from sending Sub to Hub for storage, fulfillment, and delivery to the destination Hub with multi-modal transportation modes comprising truck, rail, or airway.

Third, the destination Hub will distribute goods to the regional Sub, and then deliver them to the post office in specific districts and wards.

Finally, the receipt and successful input order will be transported to the delivery location, which is received at the delivered post office or logistics centers. The bill of consignment is signed when the order is successfully delivered to the recipient.

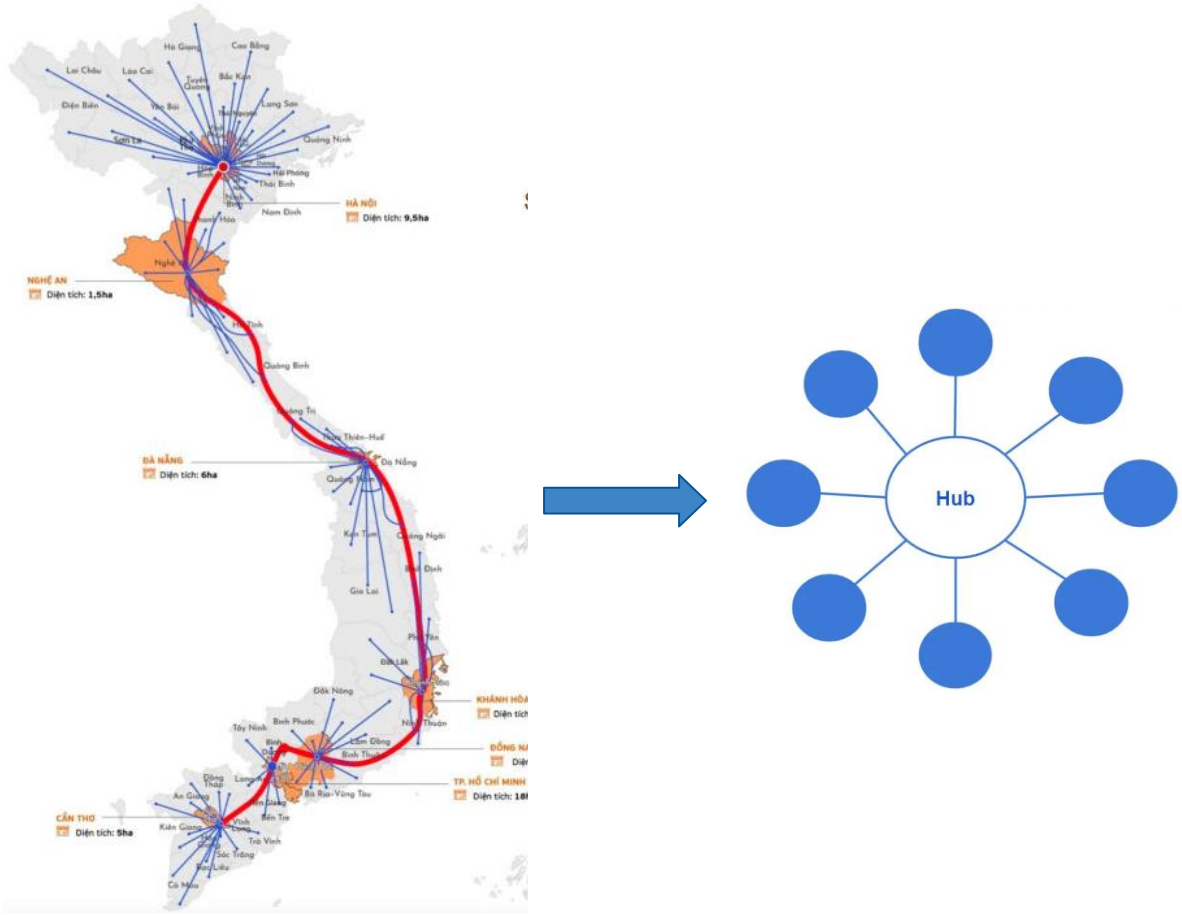


Figure 2.5 Hub-and-Spoke model

The covered distribution network of VTPost follows the Hub-and-Spoke model, in which the hub is located in the center and allows each spoke to proceed in one direction of delivery before meeting at a central location. The central hub is designed to accept, route, load, and launch several shipments to a single destination. The most obvious advantage of this network model is the reduction in transportation costs. Each day, a hub can accept significant numbers of Less Than Truckload (LTL) shipments and store them until a proper route, vehicle, and destination are available. It also advantages LTL companies to establish a network of distribution hubs where vehicles may be loaded with numerous various shipments headed to the same location, saving fuel, wages, and time. And the savings can be passed on to customers. Customer

service also benefits when the hub and spoke arrangement is automated. Automated communication is quick, straightforward, less expensive, and easy to track and trace.

2.2 Literature Review

2.2.1 Current research on HLRP

The HLRP is very difficult to be solved since it is composed of two NP-hard problems, the HLP and the MH-VRP. In comparison to HLP, HLRP has gotten far less attention (Alumur, S. et al. 2020) [3]. HLRP Additionally, the majority of investigations have concentrated on the SAHLRP. In their seminal work, Nagy and Salhi (1998) proposed the introduction of the HLRP, which they referred to as the MMLRP, incorporating a constraint on route length [4]. This constraint serves to effectively regulate the driving hours of drivers. The researchers formulated the problem as a Mixed Integer Programming (MIP) model with a size that is not polynomial. They then proposed an iterative heuristic algorithm to address the problem. Sun (2013) presented the p-hub location routing problem (pHLRP) for a fixed number of open hubs [5]. Simultaneous pickup and delivery were permitted, and hub and vehicle capacity limits were taken into account. Bostel et al. (2015) investigated a SAHLRP variation in which vehicle capacity was limited by restricting the number of nodes visited by each vehicle [6]. Kartal et al. (2017) provided three MIP formulations of a single allocation p-hub median hub placement problem (SApHLRP) without taking into account vehicle and hub capacity [7]. Karimi and Setak (2018) propose a flow shipment scheduling and incomplete hub location-routing issue with two competing objectives: maximize total flow supplied in a set period and minimize total expenditures [8]. Furthermore, in the SAHLRP, Karimi (2018) implemented a maximum travel time constraint between OD pairings [18]. The problem was expressed as a MIP, and four valid inequalities were presented to improve the model with a view to minimizing total cost was examined by Rahmati, R. et al. (2022) [9]. Ratli et al. (2022) examined the capacitated single-allocation HLRP with the distinct pickup and delivery consideration aiming to minimize the total fixed and transportation costs [20].

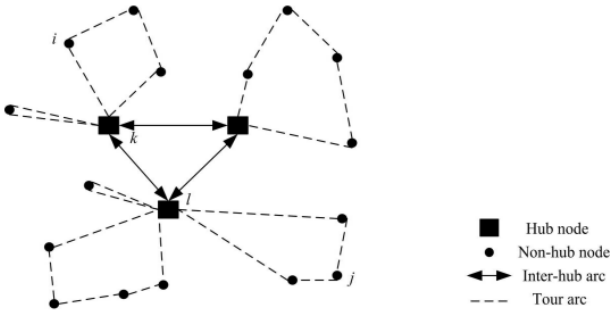


Figure 2.6 A schematic of single-allocation HLP

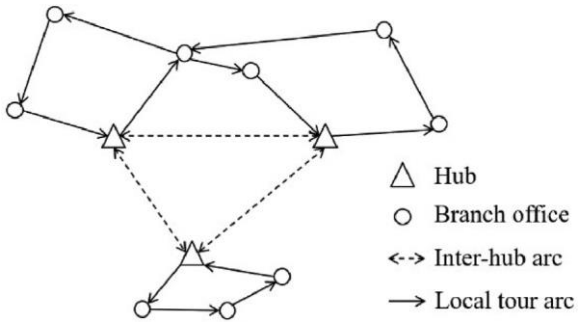


Figure 2.7 A schematic of multiple-allocation HLP

Studies on the MAHLRP are even scarcer than on the SAHLRP with only a few papers (Wu, Y. et al., 2022) [10]. Wasner and Zapfel (2004) carried out a case study of a medium-sized Austrian parcel distribution company [11]. They assumed the network had two types of facilities: central warehouses (hubs) and depots (non-hub nodes). Under various practical constraints, the goal was to assign postal zones to depots and interchange flows between depots via central warehouses or direct connections. Formally, the MAHLRP was initially explored by Çetiner et al. (2010), who did not take hub capacity or vehicle capacity into account [12]. Furthermore, they simply proposed the MAHLRP concept along with an iterative heuristic method but did not provide the mathematical formulation of the MAHLRP. Catanzaro et al. (2011) proposed the partitioning-hub-location-routing problem (PHLRP), which combined a graph partitioning and a routing problem [13]. The goal was to divide the nodes into areas and select where the hubs should be located within these areas in order to implement the flow exchange. Furthermore, in their work, non-hub nodes were permitted to be supplied by many hubs in their area, resulting in a multi-allocation dilemma. Basirati et al. (2019) addressed an MAHLRP with time windows in which both hubs and cars were assumed to be capacitated and distance balancing was addressed using a bi-objective framework [14]. In their model, the pickup and delivery operations were distinct (not concurrent), yet the pickup and delivery trips visited the same non-hub nodes. All of these studies did not examine crossdocking because it is typically prohibited in intracity express mail systems. Flow exchange at non-hub

nodes, on the other hand, is possible in ship cargo systems. Fontes and Goncalves (2021), for example, investigated an HLRP with sub-hubs of a liner shipping transportation system [15]. These sub-hubs functioned similarly to nonhub nodes assigned to multiple hubs in the manner of MAHLRP, with the exception that flows could be transferred at these sub-hubs. In the study of Demir et al. (2019), a novel model was introduced to address the multi-objective multiple allocation hub network design and routing problem [19]. The hubs that have been chosen are not presumed to be completely interconnected, and every node and arc within the network has limitations on their capacity. The problem encompasses multiple objectives including the total cost and maximum travel time required for routing minimization. Wu, Y. et al. (2022) examined a capacitated MAHLRP in the context of designing an intra-city express service system facilitating the exchange of mail and parcels between branch offices of the service provider through the utilization of local tours and hubs [10].

2.2.2 Applied techniques

The relatively large number of papers devoted to develop the approximation methods and the heuristic approaches can be classified into three categories such as clustering-based, iterative, and hierarchical heuristics, respectively. Clustering-based methods begin by partitioning the customer set into clusters, one cluster per potential depot or one per vehicle route. Then, they locate a depot in each cluster and then solve a VRP for each cluster [21–24]. Iterative heuristics decompose the problem into its two subproblems. Then, the methods iteratively solve the subproblems, feeding information from one phase to the other [25–28]. Hierarchical heuristics considers the location problem as the main problem and routing as a subordinate problem where main algorithm is devoted to solving the location problem and refers in each step to a subroutine that solves the routing problem [29, 30]. To solve the problem, the HLRP is formulated as a 0-1 mixed integer programming model. The integer programming model can provide the optimal solutions for small sized problem instances, but it is not practical for problems of large size in a reasonable computational time. Various methodologies are attempted to develop through many research papers for modeling the hub location-allocation problems while considering different scenarios to ensure

effective coordination with multiple required constraints. Because of the SAHLRP and MAHLRP's difficulty, most related studies have used heuristic methods, with just a few attempts to solve the problem exactly.

Rodrguez-Martn et al (2014) developed the HLRP based on a local route cycle, in which each hub has just one vehicle that visits all nodes assigned to it cyclically [17]. A branch-and-cut structure was used to solve the problem after several valid inequalities were provided. The researcher proposed an ant colony optimization (ACO) algorithm to address the problem of solving randomly generated instances consisting of 100 and 200 non-hub nodes (Sun, 2013) [5]. The use of a memetic algorithm (MA) was suggested to address the SAHLRP at hand by Bostel et al. (2015) [6]. The performance of the MA was then compared to that of CPLEX on instances involving up to 100 non-hub nodes. The Endosymbiotic Evolutionary Algorithm (EEA) examined by Ratli et al. (2022) effectively addressed the resolution of hub location and vehicle routing [20].

Wasner and Zapfel (2004) involved the integration of a MDVRP and HLP through the use of mixed integer nonlinear programming (MILP) and subsequently solved using a heuristic algorithm [11]. The Multi-Agent Hybrid Location Routing Problem (MAHLRP) was introduced in addition to presenting an iterative heuristic algorithm (Çetiner et al., 2010) [12]. However, they did not provide a mathematical formulation for the MAHLRP. Catanzaro et al. (2011) treated the MAHLRP as a MIP and used a branch-and-cut technique to handle instances with up to 20 nodes [13]. To handle the bi-objective problem on big instances, they developed a multi-objective imperialist competitive algorithm (MOICA) (Basirati et al., 2019) [14]. Fontes and Goncalves (2021) posed the problem as a MIP and solved it using a variable neighborhood decomposition search (VNDS) algorithm [15]. The paper of Demir et al. (2019) proposed the utilization of NSGA-II, a widely recognized multi-objective evolutionary algorithm, as a meta-heuristic solution using a data set from the Turkish postal system to assess the effectiveness of the heuristic approach [19]. A novel mixed integer programming formulation and Adaptive Large Neighborhood Decomposition Search was employed to address the CMAHLRP variant by (Wu, Y. et al., 2022) [10].

Benders decomposition (BD) algorithm (Benders, 1962) has been utilized successfully to address large-scale HLPs. de Camargo et al. (2013) offered a new formulation of the SAHLRP that may be divided into two parts: transportation and feasibility [16]. To precisely tackle the problem, a benders decomposition algorithm integrated into a branch-and-bound framework was presented. Most of the research of Ghaffarinasab, N., et al utilized the BD algorithm in tackling the uncapacitated multiple-allocation HLPs while obtaining the Lagrangian Relaxation Algorithm (LR) in solving the single-allocation HLPs in 2019 and 2022 [31, 32]. de Sá, M. et al. (2018), Mokhtar, H. et al. (2019), and Najy, W. et al. (2020) used the BD algorithm in solving the HLPs with total cost minimization while satisfying specific requirements [33 – 35]. The BD algorithm was also applied in the multi-period HLP with serial demands for cost minimization and the capacitated HLP with multiple demand classes for profit maximization but advanced with Pareto-optimal-cuts by Taherkhani, G. et al. (2020) [36].

Tabu-search based metaheuristic (TS) is another widely applied technique for effective solution performance conducted by Silva, M. R. (2017) for uncapacitated single-allocation p-hub maximal covering problem, Karimi, H. (2018) for capacitated single-allocation HCLRP [37, 18]. Ghaffarinasab, N. et al. (2020) used the TS-based matheuristic for the robust single-allocation p-hub median HLP with polyhedral demand uncertainty correspondingly [38].

Simulated Annealing (SA) algorithm has been typically introduced in lots of papers regarding HLPs. In 2018, Alumur, S. A. et al. investigated the multiple-allocation HLPS in which the market is assumed to be a duopoly with a view to maximizing the market share by utilizing the SA algorithm for high solution quality [39]. The multiple-allocation p-hub median, maximal covering, and center problems with intentional disruptions were also studied using the SA algorithm implemented by Ghaffarinasab, N. et al. (2018) [40]. Karta et al. (2017) proposed a multi-start simulated annealing (MSSA) algorithm and an ant colony optimization (ACO) algorithm to address the problem [7]. These algorithms demonstrated the capability to

obtain satisfactory solutions for instances with up to 200 nodes within a reasonable computational timeframe.

2.2.3 Key reference

“Adaptive large neighborhood decomposition search algorithm for multi-allocation hub location routing problem” (Wu, Y. Et al., 2022) paper concerned about a certain sort of hub network topology known as hub location-routing, in which the paths between hub-assigned nodes constitute a tour in this topology [10]. Subject to hub capacity, vehicle capacity, and distinct pickups and deliveries, the model reduces the overall cost of hub location and vehicle routing. For the multiple allocation class of problems, a mixed integer programming formulation was presented. In addition, an ALNDS meta-heuristic was proposed for determining the hub location and vehicle routes at the same time. The performance of MIP by CPLEX and ALNDS by PYTHON is then evaluated using a set of computational tests. The findings demonstrate that the proposed ALNDS algorithm was effective in efficiently solving the MAHLRP for all generated instances. Indeed. In the case of small instances, the ALNDS algorithm produced the most optimal solutions. Furthermore, for medium instances, the ALNDS algorithm outperformed the CPLEX solver in terms of both solution quality and computational time. Since the investigated model and problem’s functional objective is comparatively similar to the current needed issue in redesigning the delivery network of VTPost, the paper is selected to be the key reference for further comprehension and application.

Compared to the key paper, in the present research, there are two major differences in terms of allocation type decision and number of predefined hubs:

- The simultaneous pickup and delivery are applied instead of the studied separate pickup and delivery routes in the key reference. It attempts to consolidate collection and delivery activities in a single route, potentially saving time and making better use of vehicles. Simultaneous routes tend to make better use of vehicle capacity, which can result in cost savings, higher operational efficiency, lower carbon emissions, and environmental impact.

- The fixed number of established hubs is required, and the three existing hubs are set to always open in this research to better reflect the current system of Viettel Post.

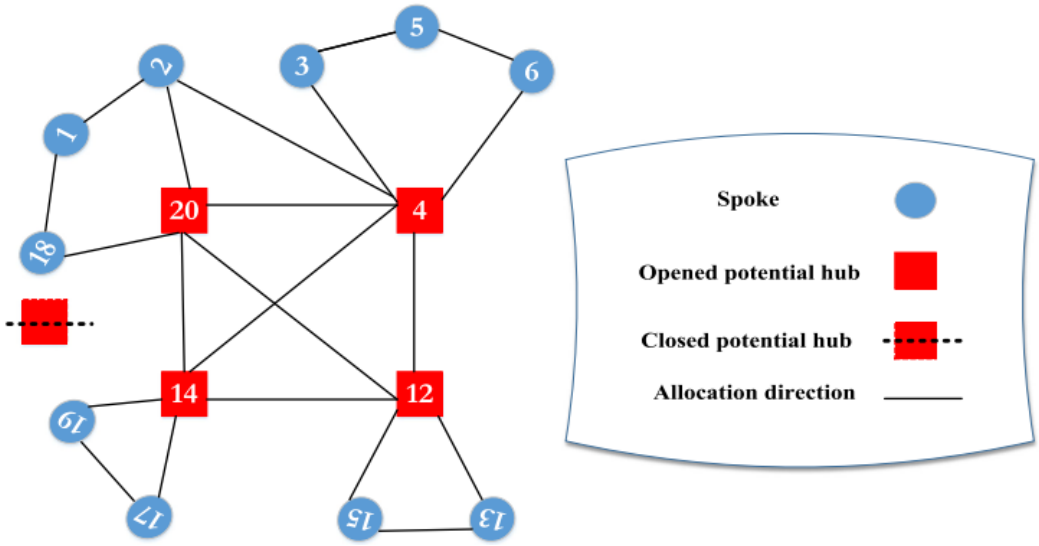


Figure 2.8 Sample result demonstration for optimizing MAHLRP with simultaneous pickup and delivery

2.3 Candidate Solution Methods

With the aim to solve and optimize the proposed problems, mathematical model formulation and corresponding programming execution are highly recommended for solution solving. The suggested programming applications are CPLEX to build a program tackling this complex problem in small to medium size instances and PYTHON to adopt other technique for large-scale instance solution.

Based on the characteristics and previous studies, the five appropriate approaches for the above-suggested improvements are defined: the Simulated Annealing (SA), Bender Decomposition (BD), Tabu Search (TS), Adaptive Large Neighborhood Search (ALNS) and Adaptive Large Neighborhood Decomposition Search (ALNDS). These techniques are able to flexibly adapt to the wide and various problem cases with short computational time which were successfully investigated in many kinds of research.

- SA

The Simulated Annealing algorithm is a metaheuristic optimization technique inspired by the metallurgical annealing process. Kirkpatrick, Gelatt, and Vecchi invented it in 1983, and it is widely used to address combinatorial optimization issues.

The SA algorithm assesses the new solution at each iteration and decides whether to accept or reject it based on a probabilistic acceptance criterion. The acceptance probability is calculated by comparing the new solution's objective function value to that of the present solution and the current "temperature" parameter. The temperature is initially set to a high value, allowing the algorithm to accept fewer desirable results and explore the solution space more broadly. The temperature is gradually decreased according to a cooling schedule as the optimization advances.

The cooling schedule governs how quickly the temperature drops between iterations. Lower temperatures result in a more deterministic search, which makes it less probable that the algorithm would accept less desirable answers as the optimization process converges towards a more promising part of the solution space. The cooling schedule is crucial for balancing exploration and exploitation and eventually arriving at a high-quality solution.

- BD

The Bender Decomposition algorithm is a well-known technique for solving large-scale optimization problems with a mix of integer and continuous variables. Bender first used it in 1964. It works by breaking down a large optimization challenge into smaller, more manageable subproblems. These subproblems are usually easier to answer on their own because they only involve a subset of the original variables.

The approach begins by attempting to solve a relaxed form of the issue, known as the master problem, in which some or all of the integer variables are represented as continuous. The relaxed problem solution provides an initial lower bound for the best goal value. Following that, a subproblem known as the pricing problem is defined to provide additional limitations known as the Bender's cuts. These cuts are applied to the master problem to reinforce the relaxation and improve the solution quality. The

subproblem entails setting some or all of the integer variables from the relaxed solution to their fractional values and solving a constrained version of the original problem. The goal of the subproblem is to generate restrictions that are violated in the master problem's existing solution. The Bender's cuts from the subproblem are then applied to the master problem, and the procedure is repeated until a practical or optimal solution is found.

- TS

Tabu Search is a metaheuristic optimization approach for solving combinatorial optimization problem developed in the 1980s by Fred Glover and is well renowned for its ability to effectively explore the solution space in search of high-quality solutions.

The Tabu Search algorithm keeps a list of candidate solutions and navigates the search space using movements or transformations. It keeps track of previously visited solutions in a tabu list, preventing the algorithm from returning to them right away. This strategy keeps the algorithm from being stuck in local optima and pushes it to explore various regions of the search space. The TS does a local search at each iteration by exploring surrounding solutions to the current solution. It assesses the quality of these neighbors and chooses the best one as the new current solution, considering both improvements in the objective function and avoidance of previously visited solutions.

Tabu Search also employs intensification and diversification tactics. Diversification stimulates investigation of less studied regions, whereas intensification focuses on exploiting promising sections of the search space. These tactics seek to achieve a balance between exploitation and exploration in order to increase the likelihood of discovering near-optimal solutions. The tabu list is updated throughout the process, allowing previously visited solutions to become eligible for revisiting after a set period of time.

- ALNS

The ALNS algorithm belongs to the family of adaptive search algorithms, was introduced in 2006 by Ropke and Pisinger. ALNS is intended for use in problems where the search space is large and typical accurate approaches are computationally infeasible. It seeks high-quality solutions by exploring different areas of the search space iteratively and adjusting the search based on input obtained during the optimization phase.

The approach improves an existing answer iteratively through a series of neighborhood search procedures. A neighborhood is a collection of transformations or changes that can be performed to the present solution to produce a new one. These alterations can include inserting, deleting, replacing, or modifying certain solution pieces. ALNS picks a subset of neighborhoods to be considered at each iteration, driven by a score or fitness function that represents their previous performance. Neighborhoods with higher rankings have previously demonstrated effectiveness in developing solutions and are more likely to be used. A perturbation technique is occasionally used after selecting a neighborhood to diversify the search and escape from local optima. Making random or disruptive modifications to the current solution is what perturbation entails.

To determine if the new solution should replace the present solution, ALNS employs an acceptance criterion. The acceptance criterion often takes into account factors such as the objective function value and a probability calculation based on the difference in the objective values of the old and new solutions. ALNS evolves throughout the optimization process by updating the scores of the neighborhoods based on their performance. Neighborhoods that regularly increase solution quality are rewarded with better ratings, while those that have not demonstrated efficacy may have their rankings reduced.

- ALNDS

Similar to the ALNS algorithm, however, the decomposition feature in the method enabling it to divide the original problem into subproblems and use a selection process to identify intriguing subproblems to investigate. During the search, it uses an

acceptance criterion to determine if a solution should be accepted or rejected. The acceptance criterion is based on a cost function that assesses the solution quality. ALNDS stores information about previously investigated subproblems and their related costs in a memory system. This memory enables the algorithm to direct the search to more promising parts of the search space. It is a useful tool for tackling combinatorial optimization problems, balancing search space exploration and exploitation of potential solutions.

Chapter 3: METHODOLOGY

3.1 Approaches Comparison and Selection

There are numerous modeling methods associated with distinct HLPs in the supply chain delivery network. Various procedures may be more efficient with different fundamental needs, which may necessitate certain numerous evaluations as a prerequisite for applying to a certain scenario. Because of the selectivity of each mathematical technique and the degree of complexity of this model, the following table highlights the competitive advantages and drawbacks of the five solution approaches discussed above.

Table 3.1 Strengths and weakness of approaches

TECHNIQUE	STRENGTHS	WEAKNESSES
Simulated Annealing	- Is effective at escaping local optima. Accepting bad alternatives with a certain probability broadens the solution space, increasing the chances of finding a superior solution, perhaps the global optimum. - Is easy to adopt and does not necessitate sophisticated problem-specific adjustments. - Is accessible to a wide range of problems. - Allow for adaptation throughout the optimization process by modifying the temperature parameter. - Is applicable for problems with not differentiating objective function or when computing derivatives is difficult. - May be customized to handle restricted optimization situations.	- Have slower convergence rate compared to other optimization methods. - Require a large number of iterations and a well-tuned cooling schedule. - Need tunable parameters. - Might be affected by the starting solution and the random number generator employed. - Is important to create an efficient cooling plan for the success of SA. Insufficient cooling schedule might lead to poor performance. - Is not guaranteed that the global optimum will be found. As the algorithm cools down, the likelihood of accepting inferior answers decreases, perhaps leading to premature convergence.
Bender Decomposition	- Is useful for large-scale MILP. - Take advantage of the variables' separability leading to faster convergence and higher solution quality. - Is possible to parallelization on many processors or computer nodes to	- Is difficult to implement, especially for problems with complicated constraints or non-linearities. - Is not guaranteed to find the global optimum. - Might be difficult to solve or cause numerical

	dramatically accelerate the optimization process. - Allow to reuse the base or first solution of the master problem, which helps speed up the total optimization process.	instability with dual subproblems. - Is dependent on the presence of variables that can be efficiently separated resulting in unconsiderable performance advantages. - Can be affected by the quality of the initial solution as well as the tactics used to update the master problem and solve subproblems.
Tabu Search	- Break free from local optima and explore various parts of the solution space, increasing the likelihood of finding superior solutions. - Can solve a broad variety of combinatorial optimization issues. - Support the incorporation of problem-specific heuristics and neighborhood structures to improve the algorithm's performance. - Is useful for applications where computing derivatives are difficult or impossible. - Store information about recent moves in a short-term memory, making it memory efficient.	- Need tunable parameters. - Can be computationally expensive, especially for large-scale problems. - Have complex implementation. - May be affected by how the solutions are displayed. - Is not guaranteed that it will locate the global optimum. - Incur additional expense and can occasionally result in premature convergence or exploration restrictions.
Adaptive Large Neighborhood Search	- Is adaptable to various combinatorial optimization problems. - Combine parts of local and global search to	- Need tunable parameters. - Is not certain that the global optimum will be

	efficiently explore the solution space and swiftly locate high-quality solutions. - Dynamically modify search tactics to fine-tune the search process and increase the rate of convergence. - Balance exploiting promising locations with exploring new areas to prevent early convergence and encourage more thorough investigation of the solution space. - Can be developed using a modular structure that separates problem-specific and algorithmic components.	found for complex issues. - Need large computational resources and execution time depending on the problem complexity and solution quality criteria. Especially for ALNDS, higher level complexity is required for decomposing the problem, developing appropriate neighborhood structures, and coordinating the parallel execution of subproblems. - May require problem-specific adjustments. Designing proper neighborhood structures and defining acceptable acceptance criteria can increase the algorithm's efficiency.
Adaptive Large Neighborhood Decompostion Search	- Tackle large-scale combinatorial optimization issues with the decomposition. - Is highly flexible because it does not rely on specific problem structures or assumptions. - Modify the decomposition and neighborhood search strategies dynamically to locate high-quality solutions. - Enable to the use of parallel since the subproblems can be solved independently allowing for faster exploration of the search space and perhaps better solution quality.	

A qualitative comparison is implemented with four criteria serving as the determining elements for the initial approach decision including time complexity, problem size and type, and parallelization.

Table 3.2 Approaches comparison

Method	Time Complexity	Size of Problem	Flexibility	Parallelization	Type of Problem
Simulated Annealing	Best-case	Medium to Large	Yes	N/A	Non-linear Combinatorial optimization
Bender Decomposition	Worst-case	Large	N/A	Yes	Combinatorial optimization
TS-based metaheuristic	Worst-case	Medium to Large	Yes	Yes	Non-linear Combinatorial optimization
Adaptive Large Neighborhood Search	Average-case	Medium to Large	Yes	Yes	Combinatorial optimization
Adaptive Large Neighborhood Decompostion Search	Average-case	Large	Yes	Yes	Combinatorial optimization

Based on the advantages and disadvantages and the above qualitative comparison, the ALNDS algorithm is selected to apply for this MAHLRP due to the following reasons:

- Scalability: Because of the necessity to establish the ideal position of hubs for effective routing, the HLRP frequently entails complicated and large-scale optimization problems. ALNDS excels at solving such challenges by breaking them down into smaller subproblems and optimizing each one separately. When compared to other techniques such as ALNS, SA, BD, or TS, this decomposition allows for more efficient handling and optimization of huge instances.
- Flexibility: ALNDS's adaptive nature and flexibility to alter the search strategy based on the issue instance and solution quality make it well-suited for solving the HLRP's special criteria. The method may change the decomposition and neighborhood search strategies dynamically to investigate potential portions of the search space, allowing for successful hub routing and locating. SA, BD, and TS on the other hand, may not be as adaptable and flexible.
- Technique Integration: ALNDS offers a powerful combination of decomposition methods and large neighborhood search, making it well-suited for handling the HLRP's multi-objective features. ALNDS can simultaneously optimize the location of hubs and routing decisions by decomposing the problem into subproblems, considering aspects like cost, distance, and service levels. When compared to other algorithms, this integration of optimization techniques can provide a more complete and successful approach.
- Solution quality: ALNDS strives for high-quality solutions by leveraging decomposition and neighborhood search strategies. This means that, in the framework of the HLRP, ALNDS can experiment with different hub locations and routing configurations, evaluating trade-offs between parameters such as transportation costs, service quality, and customer happiness. This thorough investigation may result in higher-quality solutions than ALNS, SA, BD, or TS.

3.2 Proposed System Design

The three fundamental components of the system design are input, process, and output. Inputs are the resources used to create a model system comprising clear and suitable data, specific requirements, and problem objectives. To begin, we require the procedure of data collection and process in order to remove redundant data and get a meaningful dataset for the studied topic. The system design for the investigated problem is illustrated below:

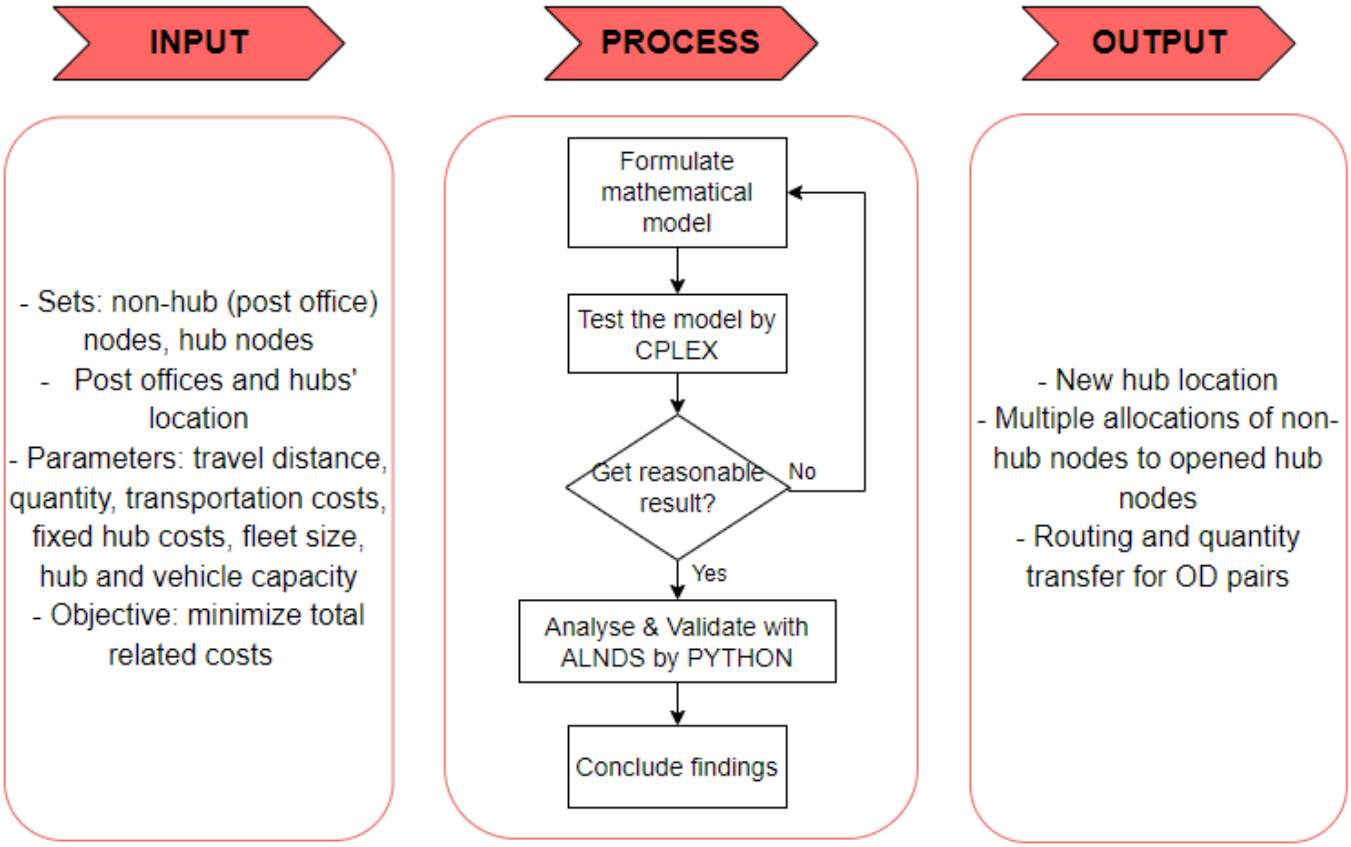


Figure 3.1 The system design of the current study

- Input

The first step for the research is to gather accurate insights, practical system operation, core problems needed to solve, desired outcomes, and essential information that reflect the real business activities. In this paper, the historical demand data of one latest year was used. Once the data collection step is finished, it is necessary to analyze, evaluate, and finally translate them into a suitable data set in solving the problem.

- Process

Modeling mathematical programming in conjunction with key objectives and constraints embedded with the ALNDS is the next step to help efficiently deliver the near-optimal solution for the allocation-routing problem to obtain the non-hubs allocations to multiple hubs and vehicle routing correspondingly. The algorithm is applied the mutual datasets with MILP to compare performance of the two techniques from small to medium size. Then, the largest case in the dataset will then be executed by the algorithm to prove its efficiency but not MILP since it is uncapable of handling such a big case.

Next, the model implementation, validation will be respectively deployed to ensure propose the most comprehensive solutions. Additionally, with sensitivity analysis, the feasibility of the proposed solution would be examined by changing some parameters, which ultimately helps to increase the result's credibility.

- Output

For the objectives of minimizing the operating costs from hubs to post offices in the delivery network, allocation of the non-hubs to hubs, and vehicle routing are the desired outcome in this investigated paper satisfying the restrictions within the limit time requirement.

Generally, this system design offers an efficient way to both analyze and document the critical aspects of model implementation, facilitating the whole process of problem-solving to develop in the right way in a reasonable time.

Chapter 4: MATHEMATICAL MODEL

4.1 Problem description

This study concerns the capacitated hubs and vehicles used to serve non-hub nodes set representing post offices. Each post office must meet both pickup and delivery requirements. Instead of directly supplying demand service between each pair of post offices, which is generally quite expensive, local tours and hubs are employed to interchange flows among these non-hub nodes. The flows are initiated at the origin post offices, routed through one or two hubs, and distributed at the destination post offices. Local tours departing from hubs help meet these demands with collection and delivery operations handled simultaneously by the same vehicle. The goal of this problem is to design the distribution network from hub to post office through local and inter-hub routes while incurring minimized operational cost, which includes local tour costs and inter-hub transportation costs. Each vehicle departs from precisely one hub and travels across the non-hub nodes, distributes parcels collected in the prior period (e.g., one day before) and picking up parcels to be delivered in the following period (e.g., the next day), which is assumed to be a warmed-up transportation system.

In the described case, five setting-up locations for Hubs are assumed to be in Lam Dong, Bac Lieu, Ben Tre, Kien Giang, and Binh Phuoc along with the three existing Hubs which are in Ho Chi Minh, Dong Nai, and Can Tho. The previously distributed post offices are then reallocated tailored to the new three established Hubs.

4.2 Prototype Solution

The Mixed-Interger Linear Programming model (MILP) for the Capacitated Multiple Hub Allocation-Routing problem is then presented. The following assumptions are illustrated to further narrow the investigated study:

1. The set of nodes is predefined.
2. The number of opening hub is fixed.
3. Each non-hub node can be assigned to at least one hub.
4. The flow between each pair of non-hub nodes must not be directly served.

5. The total pickup and total delivery load of nodes assigned to a hub do not exceed the hub's capacity.
6. One vehicle serves each hub node and is used at most by one tour. At no point throughout the tour, the total vehicle load exceeds the vehicle capacity.
7. The pickup and delivery flows are assumed to utilize hub capacity.
8. The inter-hub connections (hub-level network) are completely connected.
9. Each node must satisfy the flow balance requirement. The quantity of flow entering a node must be the same as the amount of flow leaving it.
10. Inter-hub transportation has no capacity constraints and no fixed costs for connecting hubs.
11. The company has their own trucks; thus, no employed vehicle fixed cost is considered.

4.2.1 Notations

Table 4.1 Sets, parameters and decision variables

Symbol	Description
Set	
$N = C \cup H$	Set of nodes
C	Set of non-hub nodes
H	Set of hub nodes
Indices	
$i, j, t \in C$	Non-hub nodes
$m, n \in H$	Hub nodes
Parameters	
α	Unit inter-hub transportation cost ($/km.kg)
β	Unit local tour cost ($/km)
w_{ij}	Given amount of flows between non-hub nodes i and j

c_{mn}	Distance between hub nodes k and m
c_{ij}	Distance between non-hub nodes i and j
Qh_m	Capacity of hub m
Qv	Capacity of vehicle
F_m	Fixed cost of opening hub m
K	A limited fleet size of homogeneous vehicle
M	Big number
Decision Variables	
u_{im}	Total amount of flow delivered on the route related to hub at node m before serving node i
v_{im}	Total amount of flow picked up on the route related to hub at node m that includes up to node i
z_{ijmn}	The flow fraction from origin i to destination j transferring through hubs m and n
x_{ijm}	A binary variable that takes 1 if a vehicle for the hub at node m moves directly from node i to node j and 0 otherwise
y_m	A binary variable that takes 1 if a potential hub m ∈ H is opened and 0 otherwise

4.2.2 Mathematical Model

- ***Objective***

The objective function is to minimizes the network system cost, including local tour, hub installation, and inter-hub transportation costs:

$$Min\ TC = \alpha \sum_{i \in N} \sum_{j \in N} \sum_{m \in H} c_{ij}\, x_{ijm} + \sum_{m \in H} F_m\, y_m + \beta \sum_{m \in H} \sum_{n \in H} \sum_{i \in C} \sum_{j \in C} w_{ij}\, z_{ijmn} c_{mn}$$

- ***Constraints***

Table 4.2 Constraints of MILP capacitated multiple hub allocation-routing problem

$\sum_{m \in H} \sum_{n \in H} z_{ijmn} = 1$	$\forall i, j \in$ C	(1)
$\sum_{i \in C} \sum_{j \in C} \sum_{n \in H} w_{ij}\, z_{ijjmn} + \sum_{i \in C} \sum_{j \in C} \sum_{n \neq m \in H} w_{ij}\, z_{ijnm} \leq Q h_m\, y_m$	$\forall$m $\in$ H	(2)
$\sum_{j \in N} x_{mjm} \leq M y_m$	$\forall$m $\in$ H	(3)
$\sum_{j \neq i \in N} \sum_{m \in H} x_{ijm} \geq 1$	$\forall$i $\in$ C	(4)
$\sum_{j \neq i \in N} x_{ijm} \leq 1$	$\forall$i $\in$ C, m $\in$ H	(5)
$\sum_{j \in N} x_{ijm} = \sum_{j \in N} x_{jim}$	$\forall$i $\in$ N, m $\in$ H	(6)
$\sum_{j \in N} \sum_{n \neq m \in H} x_{njm} = 0$	$\forall$m $\in$ H	(7)
$M \sum_{j \in N} x_{ijm} \geq \sum_{j \in C} \sum_{n \in H} z_{ijmn} + \sum_{j \in C} \sum_{n \in H} z_{jinm}$	$\forall$i $\in$ C, m $\in$ H	(8)
$u_{im} + \sum_{t \in C} \sum_{n \in H} d_{jt}\, z_{jtmn} - M(1 - x_{ijm}) \leq u_{jm}$	$\forall$i $\in$ N, j $\neq$ i $\in$ C, m $\in$ H	(9)
$v_{im} - \sum_{t \in C} \sum_{n \in H} d_{ti}\, z_{tinm} + M(1 - x_{ijm}) \geq v_{jm}$	$\forall$i $\in$ C, j $\neq$ i $\in$ N, m $\in$ H	(10)
$v_{im} \leq Qv$	$\forall$i $\in$ C, m $\in$ H	(11)
$u_{im} + v_{im} - \sum_{t \in C} \sum_{n \in H} d_{ti}\, z_{tinm} \leq Qv$	$\forall$i $\in$ C, m $\in$ H	(12)
$\sum_{j \in N} \sum_{m \in H} x_{mjm} \leq K$		(13)
$y_m = 1$	$\forall$m $\in$ {1,2,3}	(14)

$\sum_{m \in H} y_m = 4$		(15)
$x_{ijm}, y_m \in \{0, 1\}$	∀i, j ∈ N, m ∈ H	(16)
$u_{im}, v_{im} \geq 0$	i ∈ N, m ∈ H	(17)
$z_{ijmn} \geq 0$	∀i, j ∈ C, m, n ∈ H	(18)

- Constraint (1) guarantees each non-hub node can be assigned to at least hub.
- Constraint (2) ensures that all the non-hub nodes are served only by open hubs and imposes hub capacity restriction.
- Constraint (3) indicates vehicles can only depart from an established hub.
- Constraint (4) ensures that each non-hub node must be visited at least once if it is allocated to a hub to implement the pickup and delivery process.
- Constraint (5) limits each pair of non-hub & hub must be linked by more than one local tour.
- Constraint (6) concerns vehicle flow conservation.
- Constraint (7) ensures each local tour visits at most one hub.
- Constraint (8) claims if any flow coming from or intended for non-hub node i is allocated to hub m, there must be exactly one local tour connecting it to hub m.
- Constraints (9) and (10) demonstrate load on vehicles for delivery and pickup shipment accordingly.
- Constraint (11) limits the vehicle loads when leaving the hub.
- Constraints (12) limits the vehicle loads after having served non-hub node i.
- Constraint (13) limits the vehicle numbers not over the fleet size.
- Constraint (14) requires three hubs 1, 2, 3 to always open in all scenarios to follow strictly to the current system of Viettel Post.
- Constraint (15) restrict the number of hubs to be opened is 4 meaning one new hub is considered to be established besides the existing hubs.
- Constraints (16) – (18) are variable domains.

4.3 Data collection

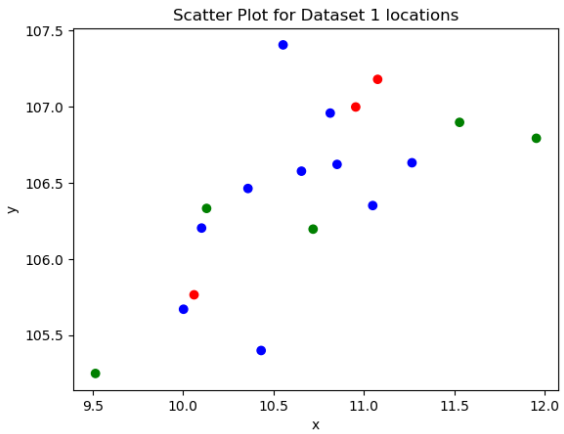
4.3.1 Data assumptions

The goods are packed in a carton box with the dimension of 40 x 40 x 20 cm; however, the box volume will not be taken into account in this model except for its weight. Each package is 5kg in total and the demand is evaluated in a month with all the involved post offices or non-hub nodes. The quantity demand matrix is not necessarily symmetrical signifying that the quantity demand $w_{ij} \neq w_{ji}$. The pickup load from a specific node is not required to be equal to the delivery load to that node.

There are 6 datasets with different sizes in non-hub nodes to assess and validate between the exact and metaheuristic method which will be mentioned in Section 5.2 of Chapter 5. The instances will be classified in terms of size:

- Small-sized instances: 10 non-hub nodes.
- Medium-sized instances: 20, 30, 40 non-hub nodes.
- Large-sized instances: from 50 non-hub nodes.

In this paper, the instance of 10, 20, 30, 40, 50, 70, and 115 non-hub nodes are considered to objectively investigate. And based on the logic of the model, the indices of hubs and non-hub nodes will followingly continue and not be separately distinct, meaning the 8 hub indices are from index 1 to index 8 and 10 customers for instance are from index 9 to index 18.



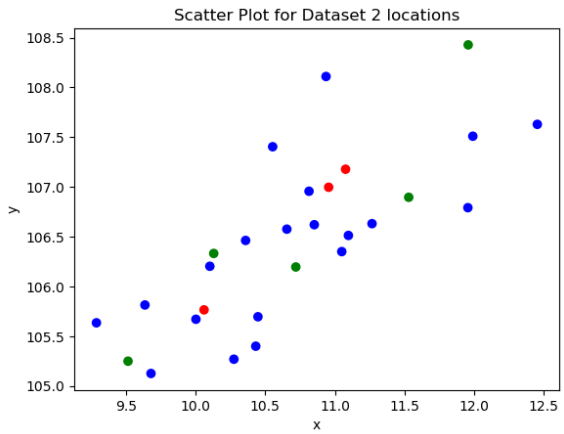
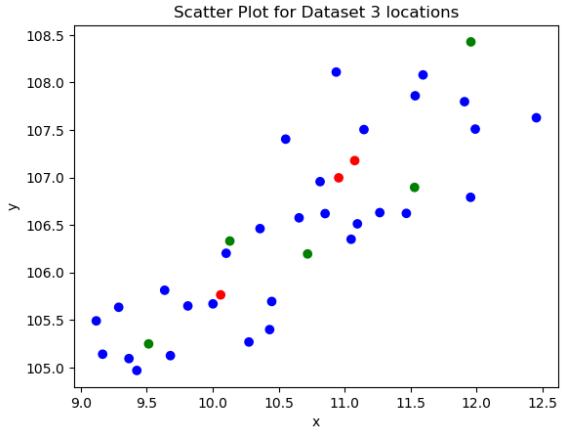


Figure 4.1 Scatter plot for 10 non-hub nodes

Figure 4.2 Scatter plot for 20 non-hub nodes



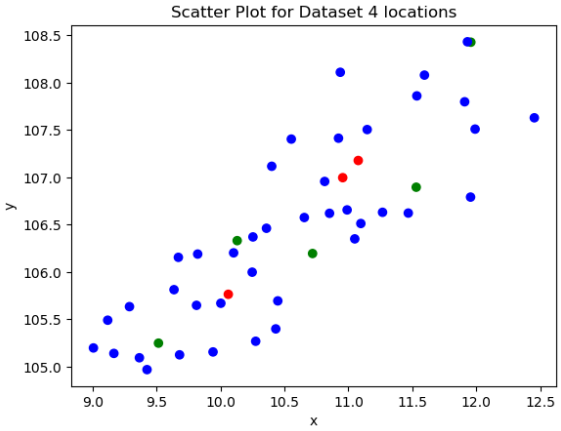
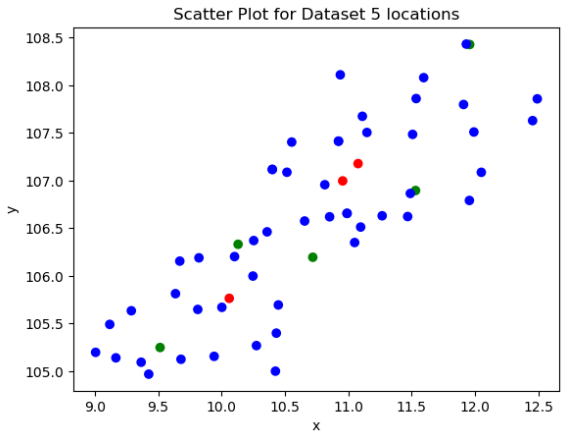


Figure 4.3 Scatter plot for 30 non-hub nodes

Figure 4.4 Scatter plot for 40 non-hub nodes



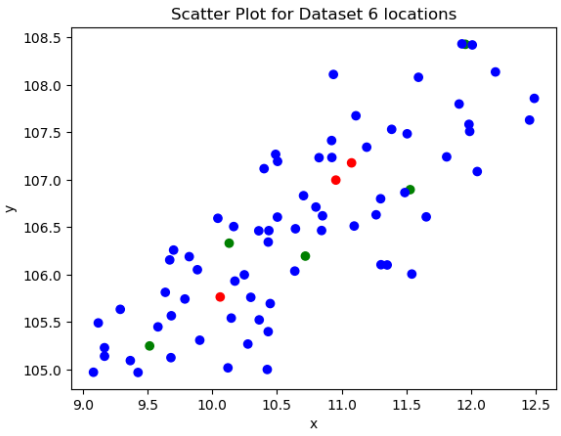


Figure 4.5 Scatter plot for 50 non-hub nodes

Figure 4.6 Scatter plot for 70 non-hub nodes

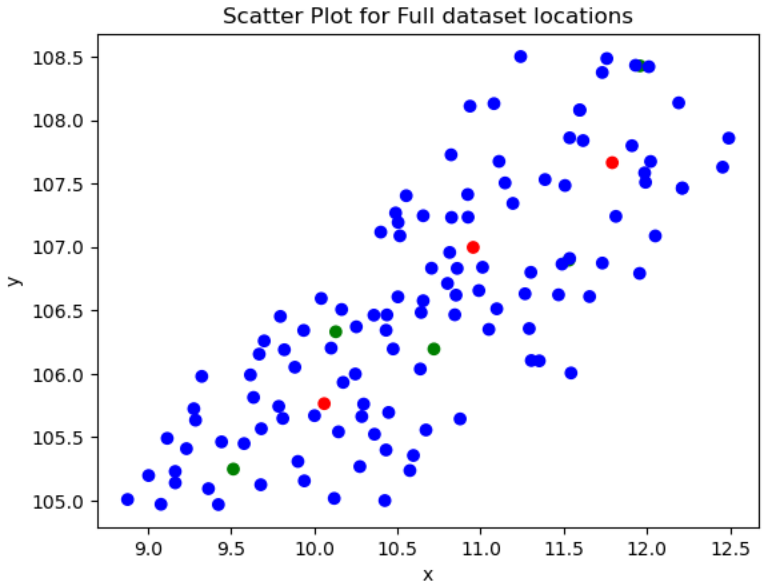


Figure 4.7 Scatter plot for 115 non-hub nodes

Since the hub capacity is fixed in different instance sizes, it must be heightened to large amount to satisfy the large-sized instance to avoid conflicts in constraints among all the datasets. The total considered hubs' capacity must meet the needs of holding both the sum of the total pickup and deliver loads since the pickup and deliver loads can be simultaneously stored at hub: $Qh_m \geq \sum_{i \in C} O_i + \sum_{j \in C} D_j$.

Besides the required restriction in the hub capacity, the vehicle capacity must ensure enough weight to load both the total pickup and delivery of each route which entails all the non-hub nodes due to the possibility of one local trip result: $Qv \geq Maximum\ (O_i + D_i)$. Furthermore, the homogeneous vehicles' fleet size is also considered to obtain the total vehicle capacity not violated with the logic constraints. The fleet size is different for each instance to help restrict the utilization of vehicle number.

4.3.2 Location

4.3.2.1 Hub location

Assuming the research and development department has delivered a list of potential hubs to evaluate the capability to open considering a local tour, inter-hub, and hub

establishment costs minimization. These hub locations are narrowed down resulted from the set covering method, and multi-criteria decisional making including surrounding infrastructure conditions (accessibility to the hub, diverse transportation systems connection, electricity, telecommunication, etc), goods storage conditions with the environment and area assessments, distance and travel time to post offices and other hubs, manpower quality, so on and so forth. Here are the hub location lists with indices 1 to 3 are the existing hubs and the remaining are the potential ones, which are marked as red and green correspondingly in the following map demonstration.

Table 4.3 Considered Hub nodes' location

Hub Location			
No.	**Address**	**x coordinate**	**y coordinate**
1	Khu đất Z11, phường Trung Mỹ Tây, Quận 12, TP Hồ Chí Minh	10.852247	106.620359
2	23 Cách Mạng Tháng 8, Bình Thủy, Cần Thơ	10.06052	105.76485
3	Kho Long Sơn, Đường số 1, KCN Long Bình, Tp Biên Hoà, Đồng Nai	10.9554259	106.996952
4	QL20, Liên Nghĩa, Đức Trọng, Lâm Đồng	11.076832	107.178291
5	Khu Phố Vĩnh Đông 2, Vĩnh Thuận, Kiên Giang	9.5146	105.248902
6	QL60, TT. Mỏ Cày, Mỏ Cày Nam, Bến Tre	10.129992	106.331674
7	Phú Riềng Đỏ, Tân Xuân, Đồng Xoài, Bình Phước	11.53037	106.896116
8	Trần Văn Trà, khu phố 3, Thạnh Hóa, Long An	10.719501	106.195714

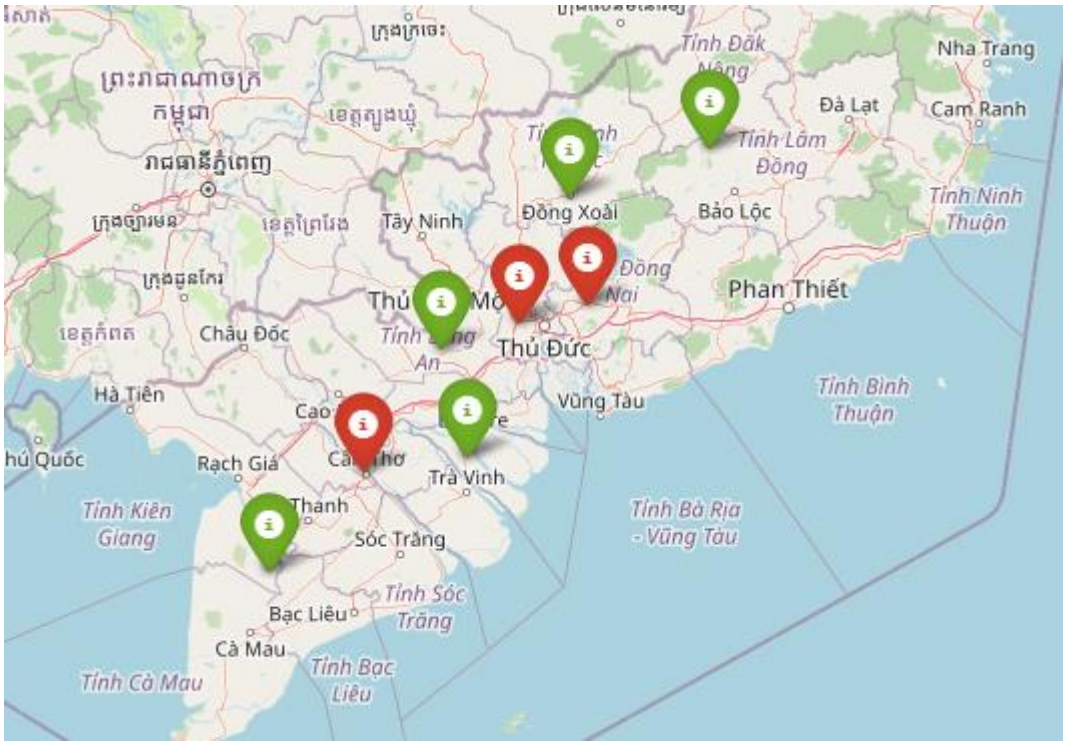


Figure 4.8 Illustration location of hub set

4.3.2.2 Post office location

The post offices are taken from Lam Dong to Ca Mau to help better evaluate the hub opening decision. The dataset 1, 2, 3, 4, and 5 investigate 10, 20, 30, 40, 50 post offices respectively. The dataset 6 is the largest instance with 115 post offices consideration. The map below shows the spread of the non-hub and hub nodes in which red marks, green marks, and blue marks refer to the existing hubs, potential hubs, and post offices correspondingly.

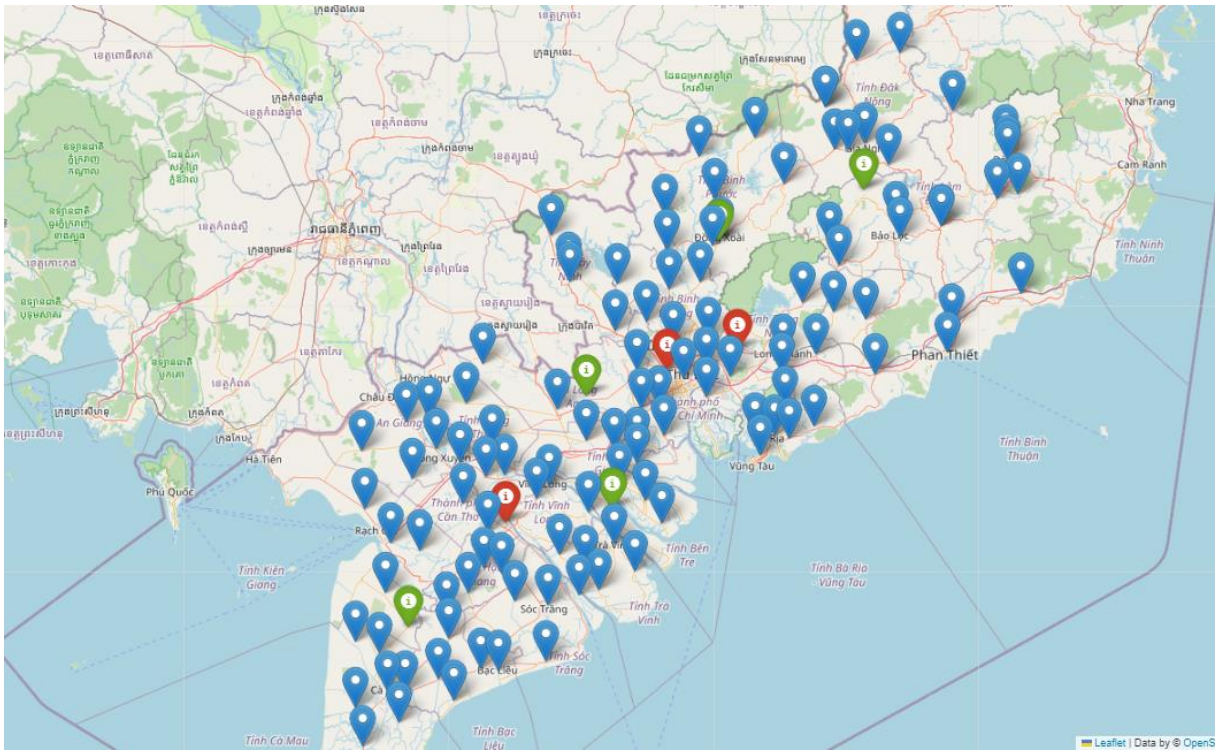


Figure 4.9 Hub and non-hub nodes location illustration

4.3.3 Cost and distance

For the acquisition of locations and corresponding distances among hub and non-hub nodes, Google Map Distance Matrix API is utilized with PYTHON language to create the distance matrix. The post offices' locations are taken scatterly from Lam Dong to Ben Tre to perform the application of the model development. The below figure depicts those nodes in which the existing hub nodes refer to red marks, the potential hub nodes refer to green marks and post offices refer to blue ones.

The objective value of the model is computed based on the hub fixed cost ($), unit local tour cost ($/km), and unit inter-hub transportation cost ($/km.kg). The unit local tour cost can be obtained from expenses for delivery divided by the average distance among the nodes daily. The unit inter-hub transportation cost can be computed as above with weight consideration along with the traveled distance. However, a discount factor is applied in this hub linking cost for economy scale reflection to help the local tour and hub transferring costs comparable. The setting unit local tour cost and inter-

hub transportation cost for objective function computation are \$0.8 and \$0.05 accordingly.

4.4 Solution development

Since the MAHLRP is a NP-hard problem composing of three problems: hub location, non-hub nodes assignment, and vehicle routing, the solution space is significantly large and ineffective to solve a practical instance size with MIP. Therefore, the Adaptive large neighborhood decomposition search (ALNDS) metaheuristic is applied in this problem to decompose an entire solution space into three subproblems as listed to optimally solve the problem in a shorter computational time. The metaheuristic's result is then compared to the Exact method's result for validation.

4.4.1 Method description

Below is the pseudocode overview of the ALNDS:

```
Generate initial solution S0
Set current solution S' = S0
Set current best solution Sbest = S0
While terminal condition unsatisfied:
    Select a subproblem
    Select destroy method and destroy the solution of the subproblem
    Obtain the new solution S''
    If S'' better than Sbest
        Sbest = S''
        S' = S''
    Else
        If S'' better than S'
            S' = S''
        Else
            Accept solution with SA mechanism
        Endif
    Endif
    Update scores of corresponding subprobem and operators
    Update weights of subproblems, destroy methods and repair methods if needed
End While
Output Sbest
```

The suggested ALNDS first builds an initial solution and then iteratively improves it. At each iteration, one of the subproblems is chosen, and the existing solution of the chosen subproblem is destroyed using a destroy method and repaired using a repair technique to yield a new solution. A set of destroy and repair procedures are used for

each subproblem. Each subproblem, destroy operator and repair operator is assigned a weight and is chosen by a roulette mechanism based on that weight. The weights are adaptively modified based on previous iterations. The new solution is accepted and utilized as input for the following iteration if it outperforms the present one. Otherwise, a simulated annealing (SA) technique is used to decide whether or not it is acceptable. When the halting condition is satisfied, the procedure is terminated.

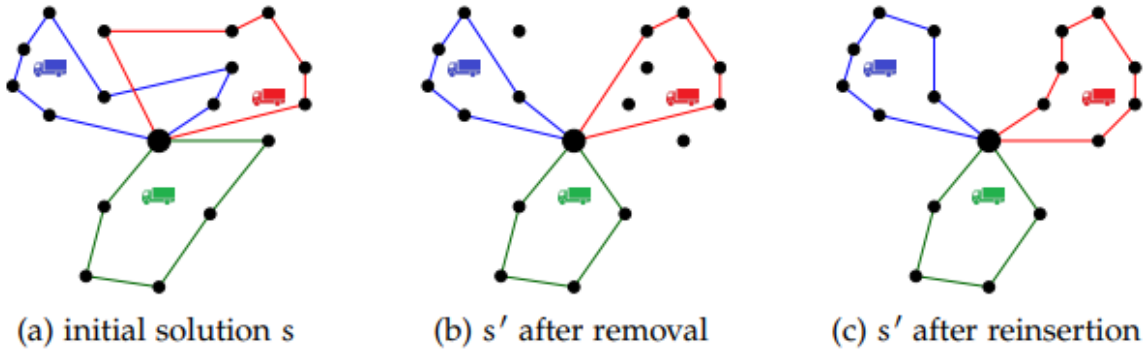


Figure 4.10 Sample of random removal and insertion operator on non-hub nodes in the generated route

4.4.2 ALNDS algorithm

4.4.2.1 Objective function

The algorithm does not limit the search to only feasible solutions. Instead, it accepts breaches of the vehicle capacity and hub capacity limitations. To account for these infractions, the ALNDS employ a weighted penalty mechanism. The following objective function is adapted:

$$\boldsymbol{f(s) = c(s) + \mu v(s) + \varphi h(s)}$$

Where:

c(s): system's operation cost (local route, inter-hub transportation, hub fixed costs)

v(s): vehicle capacity constraint violation

h(s): hub capacity constraint violation

μ, φ: corresponding weights

The Algorithm 2 obtained below is applied to estimate $f(s)$ for computational time savings, and the subproblem is solved exactly if a new best solution is found. It seeks to solve the problem by iteratively assigning Origin-Destination (OD) pairs to paths between multiple-allocation nodes in a way that meets capacity limitations and reduces costs.

```
Input solution s
Obtain the network of s
Assign OD (i, j) to hubs and vehicles, both i and j are single-allocation nodes
Update the remaining capacities of hubs and vehicles
While unassigned ODs remained:
    Select unassigned OD (I, j) with maximum dij
    If the capacity of path p is larger than dij
        Assign OD (i, j) to path p
        Set dij to 0
        Update the remaining capacities of hubs and vehicles of path p
    Else
        Assign OD (i, j) to path p
        Update the remaining capacities of corresponding hubs and vehicles of path p
        Set dij to the difference between dij and the capacity of path p
    Endif
End While
Calculate f(s)
Output f(s)
```

- Input: A s solution made up of assigned OD pairs is generated based on the below Algorithm 3.
- Obtain the solution s network: The algorithm starts by examining the solution s to build a network that reflects the movement of commodities from hubs to vehicles and from vehicles to non-hub nodes.
- Assign Hubs and Vehicles: The algorithm iterates through each OD (i, j) in the solution s. The OD (i, j) is assigned to hubs and vehicles based on their current capacity.
- Update capacity: After assigning each OD pair, the algorithm modifies the remaining capacities of the respective hubs and vehicles to reflect the new assignments.
- OD Pair Assignment Loop: The algorithm loops until all OD pairs are assigned to pathways connecting multiple-allocation nodes.

- Choose an OD pair with the highest load: In each iteration, the algorithm chooses an unassigned OD (i, j) with the maximum d_{ij}.
- Find the Lowest-Cost Path: The method then discovers the lowest-cost path p in the network from node i to j.
- Capacity Check: The algorithm determines whether or not the capacity of path p is greater than the load d_{ij}. If satisfy, the OD (i, j) can be allocated to path p without going over the capacity limit. It updates the remaining capacities of path p's corresponding hubs and vehicles. If not satisfy, the algorithm assigns OD to path p and updates the remaining capacities of path p's relevant hubs and vehicles and sets d_{ij} to the difference between d_{ij} and path p's capacity.

The algorithm repeats this process until all OD pairs are assigned to pathways and then computes the approximated objective function f(s) based on the OD pairs assigned and their trajectories.

Output: The algorithm returns the calculated approximated objective function 'f(s)', which represents the quality of the solution's' obtained using this approximation method.

4.4.2.2 Initial solution generation

The below is the pseudocode of two-stage construction heuristic:

```
Set all potential hubs closed
While unassigned non-hub nodes remained:
    Select unassigned non-hub node I with maximum (Oi + Di)
    Assign non-hub node i to the nearest open hub that has enough capacity
    If such hub does not exist
        Set the closed hub nearest to non-hub node i open
    Endif
    Update remaining capacity of the related hub
End While
For each open hub m:
    While unlinked non-hub nodes assigned to hub m remained:
        Set current node v = m
        Set current pickup capacity p = Qv
        Set current delivery capacity d = Qv
        While feasible unlinked non-hub node remained:
            Select unlinked non-hub node I that is nearest to v
            v = i
```

p = p – O_i, d = min (p – O_i, d – D_i)
End While
End While
End For
Output generated initial solution

Denote:

$O_i = \sum_{j \in C} w_{ij}$: The total input flows originated from node i

$D_i = \sum_{j \in C} w_{ji}$: The total output flows destined to node i ∈ C

This algorithm seeks to efficiently allocate non-hub nodes to open hubs and build feasible routes for each hub while taking their capacity into account.

- **Stage 1**: Allocation of Hubs
  - At first, all prospective hubs are closed, which means they are not used to serve clients.
  - Iteratively, the algorithm assigns non-hub nodes to open hubs.
  - The method selects an unassigned non-hub node i with the maximum value of (O_i + D_i) in each iteration.
  - The chosen non-hub node i is assigned to the nearest open hub with sufficient capacity to service it. If no such hub exists (i.e., all hubs are already filled), the algorithm reopens the closed hub closest to node i to meet its requests.
  - After node i is assigned, the remaining capacity of the connected hub is updated.
- **Stage 2**: Route Design for Each Hub

The algorithm creates routes to serve the assigned non-hub nodes for each open hub m. The procedure iterates until all non-hub nodes assigned to hub m are linked to the hub m via feasible paths.

- The algorithm creates a current node v as the hub for each hub m.
- It additionally configures two current capacities: current pickup capacity p and current delivery capacity d, both of which are initially set to the total capacity of hub m (Qv).

- The algorithm finds an unlinked non-hub node i that is closest to the current node v in each iteration.
- The current node v is changed to node i and the pickup and delivery capacities p and d are also changed.

The algorithm repeats this process until a workable route for hub m is constructed, taking into account the pickup and delivery requests of the nodes assigned to the hu

After completing both stages, the algorithm returns the developed starting solution.

Overall, this two-stage construction heuristic approach attempts to identify an initial solution to the CMHLRP by allocating non-hub nodes to hubs and designing feasible routes for each hub while taking capacity limitations and minimizing distances between nodes into account. The resulting answer can then be further optimized and improved utilizing additional approaches and algorithms.

4.4.2.3 Destroy operators

Flows routing among non-hub nodes can be affected by a new non-hub node insertion in the MAHLRP. As a result, depending on the hub number that each non-hub node is allocated, the algorithm decomposes it into one or more sub-nodes. Besides, the destroy operators only delete sub-nodes rather than whole non-hub nodes. The number of sub-nodes to delete, represented by q, is chosen at random from the range [0, ρ] of the total sub-nodes number in the solution in which ρ is the removal fraction.

Each sub-node transfers flows with just one hub and is associated with the flows demand departured and destined to the parent non-hub node, which travels via the specific hub in the present flow assignment. In other words, the flow between each non-hub pair is transferred through their sub-nodes. For instance, if non-hub node i is assigned to 2 hubs m and n, it is decomposed into two sub-nodes, in which one transfers flows with hub m and the other with hub n. To decrease computing time, the flow assignment remains fixed during the destroy and repair phase.

In hub location subproblem, hubs are closed and opened in order to find a better solution. Because to the change in the hub backbone, non-hub node allocation and vehicle routing would also be affected. Regarding the non-hub node allocation subproblem, vehicle routing will be adjusted at the same time to connect the selected sub-nodes to the new hubs to which they have been assigned. Because no hub is closed or opened throughout these processes, the network's backbone remains constant. In vehicle routing subproblem, the allocation of hubs and non-hub nodes, as well as the hub backbone, is fixed in the search space of the vehicle routing subproblem.

The following presents the destroy operators used in each subproblem.

Table 4.4 Destroy operators description

DESTROY OPERATORS		
SUBPROBLEM	**OPERATOR NAME**	**PROCESS**
Hub Location	*Random Hub Removal (RHR)*	One random hub in the opening hub set is selected to close. All sub-nodes assigned to the closed hub are removed and placed in the sub-node pool. In addition, all routes that originate at that hub are eliminated.
	Worst Usage Hub Removal (WUHR)	This operator eliminates the open hub with the lowest capacity utilization ratio. All sub-nodes currently allocated to this hub are removed and placed in the sub-node pool, as are all routes that begin there.
	Random Hub Opening (RHO)	One of the closed hubs is randomly chosen and opened. The q sub-nodes are then chosen at random, removed from their previously allocated hubs, and added to the sub-node pool.
Non-hub node Allocation	*Random Allocation Change (RAC)*	RAC randomly alters the allocation connection between some sub-nodes and hubs. The chosen q sub-nodes are removed from the routes and placed in the sub-node pool.

	Worst Allocation Removal (WAR)	The operator eliminates sub-nodes that have the largest removal gain. The gain is the distance between the sub-node and the hub to which it is currently allocated. To prevent selecting the same sub-nodes again, the gain is normalized. This operator's goal is to keep sub-nodes from being assigned to hubs that are too far apart.
	Duplication Operator (DO)	Certain non-hub nodes are chosen at random, added to the sub-node pool, and then introduced in the solution using a Greedy Insertion (described in the next section). They would not be taken out from their present routes in this operator while doing so. The goal is to employ the multi-allocation process.
	Deduplication Operator (DDO)	This operator selects certain multi-allocation non-hub nodes at random and removes them from their present routes in order to minimize the visiting number to them. In other words, it eliminates unwanted non-hub node accesses.
Vehicle Routing	*Random Removal (RR)*	This operator chooses q sub-nodes at random from all the sub-nodes and places them in the sub-node pool.

	Worst Cost Removal (WCR)	Similar to WAR, it also takes out sub-nodes with the biggest removal gain. Nonetheless, the gain in this operator is the local tour cost difference between when the considered sub-node remains in the solution and when it is removed.
	Shaw Removal (SR)	This tries to eliminate sub-nodes that are identical to each other. The relatedness between non-hub nodes i and j is calculated as follow: $R_{ij} = \gamma_1 d_{ij} + \gamma_2 c_{ij} + \gamma_3 l_{ij}$. Where: $\gamma_1 - \gamma_3$ are Shaw parameters, d_{ij} is the flow quantity, c_{ij} is the travel distance, $l_{ij} = -1$ when non-hub nodes I and j are served by same vehicle.
	Random Route Removal (RRR)	- Eliminate a random route and add the appropriate sub-nodes to the sub-node pool.

4.4.2.4 Repair operators

When applied to different subproblems, there are some slight alterations:

- For the hub location subproblem, the number of hubs is firstly evaluated whether it is sufficient. If not, RHO is applied first, followed by repair operators.
- Regarding the vehicle routing subproblem, sub-nodes can only be placed into routes that initiate from the relevant hub they are allocated.

Table 4.5 Repair operators description

REPAIR OPERATORS	
OPERATOR NAME	**PROCESS**
Greedy Insertion (GI)	- Sub-nodes are inserted one after the other in a random order into the place that minimizes the insertion cost, including the operation and penalty costs.
Regret Insertion (RI)	- Anticipate by computing a regret equal to the total cost difference between the best and second-best insertion for each unserved sub-node. - Due to the unavailability of some certain insertion places, the regrets must be recalculated after each insertion.

4.2.1.5 Operators and subproblems selection

A roulette system chooses subproblems and operators depending on their weights. The insertion operators are chosen independently of the destroy operators. The adaptive weight control approach alters the weights based on their prior performance. The whole search is separated into segments, and each segment comprises several iterations of the ALNDS with specified weights. The weights are modified at the end of each segment based on the performance of the operators. To evaluate the

performance of the operators, each one is assigned a score, which is set to zero at the start of the segment and obtained increasing scores based on their solution quality throughout execution with $\sigma_1 > \sigma_2 > \sigma_3$ or 0.

$$\boldsymbol{w_{o,seg+1} = (1-\eta)w_{o,seg} + \eta \frac{score_o}{time_o}}$$

Where:

$w_{o,seg+1}$ and $w_{o,seg}$: operator o's weight at the initiation and end of a segment

$score_o$: operator o's score in the segment

$time_o$: operator o's execution times in the segment

$\eta \in [0,1]$: prior weight impact reflection on the newly obtained weight.

4.4.2.5 Acceptance criteria

SA mechanism is used to accept degraded solutions. The following formula is used to determine the probability of a deteriorating solution s' ($f(s') > f(s_{curent})$):

$$\boldsymbol{p = e^{-(f(s') - f(s_{current}))/T}}$$

T is referred as the temperature and is set as T_{start} in the beginning to ensure that a solution 30% poorer than the initial one is accepted with the possibility of τ is lowered at each iteration by multiplying a cooling factor $\theta \in [0, 1]$ to keep the search stable. The below figure is parameter tuning applied in ALNDS.

Parameter tuning results.

Symbol	Role	Value
μ	Hub capacity violation penalty cost	10,000
φ	Vehicle capacity violation penalty cost	100
δ	Penalty cost factor	1.1
ρ	Removal fraction	0.4
$\gamma_1, \gamma_2, \gamma_3$	Shaw parameters	1, 1, 100
$\sigma_1, \sigma_2, \sigma_3$	Scores	5, 2, 0.5
η	Score reaction factor	0.9
τ	To set initial temperature	0.2
θ	Cooling factor	0.9995

Figure 4.11 Parameter tuning used in ALNDS

Chapter 5: RESULT ANALYSIS

5.1 Result Illustration and Explanation

5.1.1 Exact method

Since the instance size of 60 non-hub nodes are impossible to execute in CPLEX version of 12.10 due to its out-of-memory storage, the testing dataset will stop at 50 non-hub nodes.

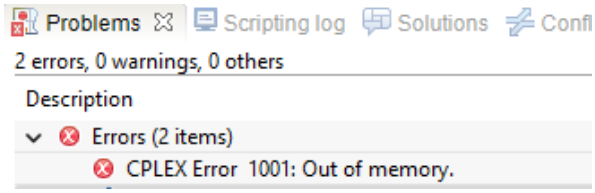


Figure 5.1 Out-of-memory storage of 60 non-hub node instance size

- **Dataset 1**

For the 10 non-hub nodes execution with 5 tons of vehicle capacity and 4 vehicles, the objective value is $96,650.824.

```
// solution (integer optimal, tolerance) with objective 96650.824
// Quality Incumbent solution:
// MILP objective                                  9,6650824000e+04
// MILP solution norm |x| (Total, Max)             2,45705e+05  1,20000e+04
// MILP solution error (Ax=b) (Total, Max)         5,06589e-10  2,27374e-12
// MILP x bound error (Total, Max)                 0,00000e+00  0,00000e+00
// MILP x integrality error (Total, Max)           0,00000e+00  0,00000e+00
// MILP slack bound error (Total, Max)             1,09139e-11  1,81899e-12
//
```

Figure 5.2 Objective value of 10 non-hub nodes instance

Table 5.1 Hub opening decision variable

y_m								
Hub m	1	2	3	4	5	6	7	8
Value	1	1	1	1	0	0	0	0

Since the hubs 1, 2 and 3 are always set to open, the result always yields the value of 1 to these hubs. However, some non-hub nodes are unnecessarily assigned to these always-opening hubs due to instance size and distances of the involved nodes.

The flow fraction is a well-known variable for demand flow routes identification with demand split consideration. If the value is equal to 1, the pickup or deliver quantity is fully loaded and transferred via hubs. When variable's value is not integer, the quantity is fractionally split with different vehicles or different routes. This variable also denote allocations, however, to shorten the solution presentation, the allocations are not illustrated in the following table. To sum up, all the 10 non-hub nodes are allocated to only hub 4 (i.e. $z_{9,9,4,4} = 1$ and so on).

Table 5.2 Flow fraction from non-hub nodes i to j passing through hubs m and n (i – m – n – j)

z_{ijmn}					z_{ijmn}				
Node i	**Node j**	**Hub m**	**Hub n**	**Value**	**Node i**	**Node j**	**Hub m**	**Hub n**	**Value**
18	17	4	4	1	13	18	4	4	1
	16	4	4	1		17	4	4	1
	15	4	4	1		16	4	4	1
	14	4	4	1		15	4	4	1
	13	4	4	1		14	4	4	1
	12	4	4	1		12	4	4	1
	11	4	4	1		11	4	4	1
	10	4	4	1		10	4	4	1
	9	4	4	1		9	4	4	1
17	18	4	4	1	12	18	4	4	1
	16	4	4	1		17	4	4	1
	15	4	4	1		16	4	4	1
	14	4	4	1		15	4	4	1
	13	4	4	1		14	4	4	1
	12	4	4	1		13	4	4	1
	11	4	4	1		11	4	4	1
	10	4	4	1		10	4	4	1
	9	4	4	1		9	4	4	1
16	18	4	4	1	11	18	4	4	1
	17	4	4	1		17	4	4	1
	15	4	4	1		16	4	4	1
	14	4	4	1		15	4	4	1

	13	4	4	1			14	4	4	1
	12	4	4	1			13	4	4	1
	11	4	4	1			12	4	4	1
	10	4	4	1			10	4	4	1
	9	4	4	1			9	4	4	1
15	18	4	4	1		10	18	4	4	1
	17	4	4	1			17	4	4	1
	16	4	4	1			16	4	4	1
	14	4	4	1			15	4	4	1
	13	4	4	1			14	4	4	1
	12	4	4	1			13	4	4	1
	11	4	4	1			12	4	4	1
	10	4	4	1			11	4	4	1
	9	4	4	1			9	4	4	1
14	18	4	4	1		9	18	4	4	1
	17	4	4	1			17	4	4	1
	16	4	4	1			16	4	4	1
	15	4	4	1			15	4	4	1
	13	4	4	1			14	4	4	1
	12	4	4	1			13	4	4	1
	11	4	4	1			12	4	4	1
	10	4	4	1			11	4	4	1
	9	4	4	1			10	4	4	1

According to the flow fraction result, the quantity transfer among OD pairs is processed at only hub 4 aiming cost minimization. For instance, $z_{18,17,4,4} = 1$ meaning nodes 18 and 17 are assigned to hub 4, and demand is fully loaded from node 18 to transfer to node 17 through hub 4.

N (size 18)	N (size 18)	H (size 8)	Value
18	16	4	1
17	18	4	1
16	15	4	1
15	14	4	1
14	11	4	1
13	17	4	1
12	4	4	1
11	12	4	1
10	13	4	1
9	10	4	1
4	9	4	1

Figure 5.3 Local route variable result of 10 non-hub nodes instance

Table 5.3 Local routes for dataset 1

x_{ijm}											
Node i	18	17	16	15	14	13	12	11	10	9	**4**
Node j	16	18	15	14	11	17	**4**	12	13	10	9
Hub m	4	4	4	4	4	4	4	4	4	4	4
Value	1	1	1	1	1	1	1	1	1	1	1

With the first index *i*, the vehicle starts at hub 4 in the final column according to the above table and turns back to hub 4 in the second index *j* to form the local route for collecting and delivering demand. The route is as follow: 4 – 9 – 10 – 13 – 17 – 18 – 16 – 15 – 14 – 11 – 12 – 4.

Next, the u_{im} and v_{im} present the pickup and delivery loads correspondingly which combine quantity of all the non-hub nodes allocated to the specific hubs based on the above fractional flow.

Table 5.4 Pickup load beginning from hub m after serving node i

u_{im}	**Hub m**							
Non-hub i	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**
9	0	0	0	570	0	0	0	0
10	0	0	0	730	0	0	0	0
11	0	0	0	840	0	0	0	0

12	0	0	0	480	0	0	0	0
13	0	0	0	1125	0	0	0	0
14	0	0	0	940	0	0	0	0
15	0	0	0	770	0	0	0	0
16	0	0	0	615	0	0	0	0
17	0	0	0	625	0	0	0	0
18	0	0	0	495	0	0	0	0

Table 5.5 Delivery load beginning from hub m before serving node i

v_{im}	**Hub node**							
Non-hub node	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**
9	0	0	0	840	0	0	0	0
10	0	0	0	460	0	0	0	0
11	0	0	0	755	0	0	0	0
12	0	0	0	625	0	0	0	0
13	0	0	0	735	0	0	0	0
14	0	0	0	785	0	0	0	0
15	0	0	0	780	0	0	0	0
16	0	0	0	690	0	0	0	0
17	0	0	0	675	0	0	0	0
18	0	0	0	845	0	0	0	0

Similar to dataset 1, the below table sums up results from the datasets 2, 3, 4 and 5 including local routes, opened hubs, and objective values:

Table 5.6 Results of dataset 2, 3, 4, 5

Instance **C – Qv (ton) – K**	**Local Route**	**Hub open**	**Objective value ($)**	**Solving time (s)**
20 - 8 - 10	3 – 16 – 12 – 9 – 11 – 10 – 13 – 14 – 3 3 – 17 – 18 – 20 – 3 3 – 21 – 22 – 25 – 27 – 28 – 26 – 24 – 23 – 19 – 15 – 3 7 – 18 – 19 – 24 – 23 – 21 – 25 – 20 – 22 – 15 – 14 – 7	1, 2, 3, 7	100,863.248	453.65
30 - 10 - 17	1 – 16 – 20 – 24 – 21 – 15 – 20 – 1 1 – 23 – 24 – 21 – 15 – 20 – 1 1 – 30 – 26 – 29 – 28 – 33 – 34 – 32 – 27 – 19 – 11 – 1 2 – 29 – 28 – 33 – 34 – 32 – 2 2 – 30 – 27 – 25 – 2 2 – 37 – 38 – 34 – 35 – 2 3 – 9 – 12 – 13 – 23 – 18 – 37 – 36 – 31 – 3 3 – 16 – 20 – 24 – 25 – 26 – 3 4 – 21 – 20 – 23 – 19 – 17 – 22 – 25 – 24 – 26 – 28 – 29 – 30 – 32 – 31 – 36 – 34 – 37 – 38 – 35 – 27 – 16 – 4	1, 2, 3, 4	131,688.8305	904.6
40 - 15 - 20	1 – 19 – 32 – 18 – 1 1 – 21 – 20 – 15 – 13 – 16 – 17 – 10 – 9 – 36 – 11 – 12 – 30 – 29 – 42 – 34 – 1 1 – 28 – 24 – 1 1 – 31 – 27 – 25 – 26 – 22 – 1 1 – 39 – 38 – 40 – 41 – 48 – 46 – 45 – 14 – 1	1, 2, 3, 4	149,036.48	7200

	1 – 44 – 43 – 1 2 – 27 – 25 – 26 – 29 – 30 – 2 2 – 34 – 42 – 43 – 12 – 14 – 47 – 45 – 36 – 31 – 17 – 18 – 33 – 22 – 2 2 – 37 – 28 – 23 – 2 2 – 38 – 9 – 13 – 15 – 21 – 10 – 16 – 39 – 25 – 29 – 30 – 2 2 – 40 – 41 – 44 – 35 – 20 – 2 2 – 46 – 32 – 11 – 12 – 2 3 – 16 – 17 – 22 – 28 – 10 – 42 – 37 – 40 – 44 – 43 – 27 – 19 – 36 – 9 – 3 3 – 26 – 11 – 23 – 24 – 3 4 – 33 – 4 4 – 38 – 48 – 46 – 14 – 21 – 20 – 3 4 – 47 – 34 – 35 – 31 – 18 – 36 – 9 – 4			
50 - 18 - 30	1 – 11 – 21 – 22 – 18 – 57 – 26 – 33 – 35 – 43 – 1 1 – 17 – 1 1 – 23 – 9 – 45 – 51 – 1 1 – 31 – 54 – 56 – 52 – 25 – 42 – 39 – 12 – 1 1 – 41 – 1 2 – 21 – 22 – 18 – 57 – 26 – 33 – 35 – 43 – 1 2 – 26 – 33 – 35 – 43 – 1 2 – 29 – 56 – 49 – 2 2 – 45 – 51 – 2 2 – 47 – 41 – 51 – 50 – 2	1, 2, 3, 4	164,682.58	7200

	2 – 45 – 14 – 47 – 48 – 2 2 – 57 – 58 – 25 – 32 – 55 – 42 – 40 – 2 3 – 36 – 34 – 13 – 15 – 3 3 – 44 – 40 – 39 – 3 3 – 43 – 52 – 58 – 41 – 3 3 – 44 – 23 – 3 3 – 48 – 11 – 30 – 45 – 10 – 9 – 51 – 50 – 53 – 3 3 – 55 – 54 – 18 – 21 – 12 – 20 – 57 – 25 – 3 4 – 9 – 33 – 20 – 57 – 52 – 12 – 4 4 – 11 – 22 – 50 – 4 4 – 16 – 17 – 43 – 39 – 54 – 38 – 32 – 55 – 4 4 – 25 – 4 4 – 35 – 58 – 29 – 21 – 4 4 – 48 – 26 – 4 4 – 23 – 30 – 20 – 13 – 11 – 21 – 27 – 19 – 34 – 54 – 12 – 4			

** Note: C: Number of non-hub nodes, Qv: Vehicle capacity, K: Fleet size*

Comment:

As the instance size increases in the number of non-hub nodes, the objective value and the computational time are evidently enhanced due to the large amount of data and the solution space. The first and second instances can be solved under 8 and 15 minutes, meanwhile, the other three instances required considerable solving time which needs to be set time limit in 3 hours to obtain the results. In instance 1, 3, 4, and 5, the new hub is located in Lam Dong, while the opening hub in Binh Phuoc is obtained in instance 2.

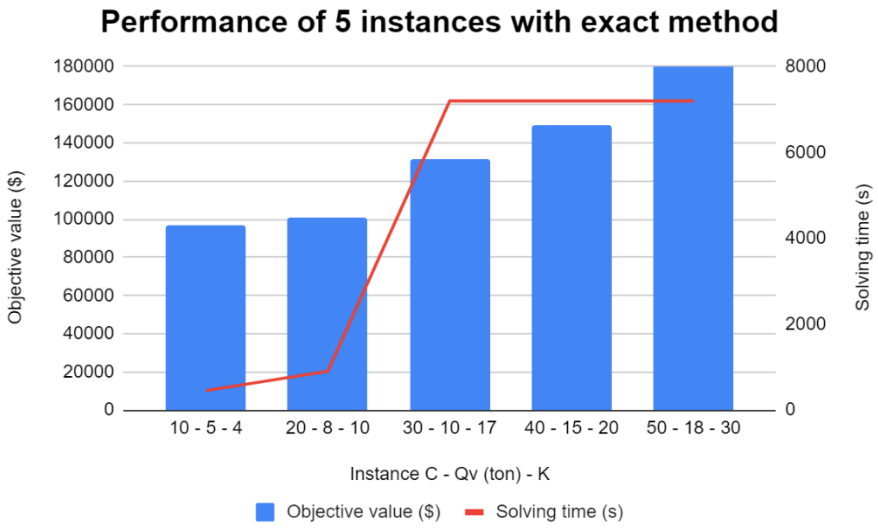


Figure 5.4 Performance of 5 instances with Exact method regarding result and time

5.1.2 Metaheuristic - ALNDS

For the implementation of the ALNDS algorithm, the following table presents the summary results of 7 datasets since the metaheuristic is able to solve large-sized problem:

Table 5.7 Summary results of 7 instances

Instance C – Qv (ton) – K	Hub open	Objective value ($)	Solving time (s)
10 - 5 - 4	1, 2, 3, 4	98,016.34	1.21
20 - 8 - 10	1, 2, 3, 7	105,427.735	3.04
30 - 10 - 17	1, 2, 3, 4	117,418.54	14.76
40 - 15 - 20	1, 2, 3, 5	130,613.62	60.15
50 - 18 - 30	1, 2, 3, 4	141,507.84	121.17
70 - 20 - 34	1, 2, 3, 7	158,721.26	302.81
115 - 30 - 54	1, 2, 3, 7	191,329.102	2,580.34

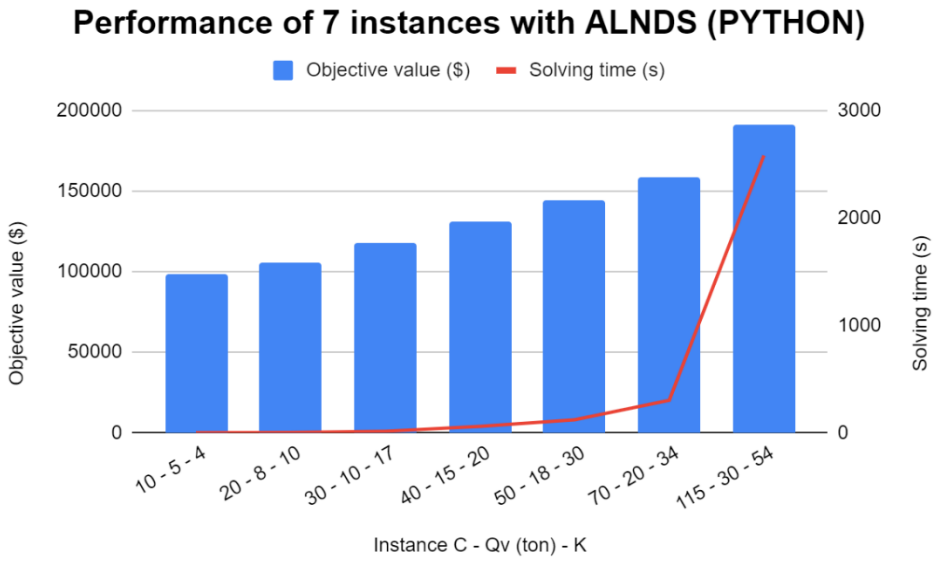


Figure 5.5 Performance of 7 instances with ALNDS regarding result and time

The detailed result of local routes in the largest instance with 115 non-hub nodes is presented as the following figures:

```
HUB | ID: 1 | Status: open
        Route 1: [96, 99, 100, 101, 93, 94, 95]
        Route 2: [97, 98, 76, 77, 78, 79, 80, 81, 83, 84, 86]
        Route 3: [102, 103, 104, 105, 106, 107, 108, 109, 110, 111, 112, 113, 114, 115, 116, 117, 118, 119, 120, 121, 122]
        Route 4: [12, 13, 20, 24, 25, 26, 27, 29, 32, 33, 34, 35, 36, 37, 38, 44, 47, 49, 54, 55, 56, 60, 61, 65, 81]
        Route 5: [13, 19, 20, 21, 23]
        Route 6: [11, 12, 13, 28, 29, 30]
        Route 7: [39, 40, 43, 44, 45, 46, 47, 25, 26]
---------------------------------
HUB | ID: 2 | Status: open
        Route 1: [88, 89, 66, 87]
        Route 2: [68]
        Route 3: [89, 90, 91, 92]
        Route 4: [89, 75, 9, 10, 11, 12, 13, 18]
        Route 5: [64, 67, 69, 71, 74, 43, 44, 82, 89, 59, 60, 62, 63]
        Route 6: [72, 73, 42, 13, 47, 85, 21, 91, 60]
        Route 7: [43, 44, 46, 50, 55, 56, 57]
---------------------------------
HUB | ID: 3 | Status: open
        Route 1: [48, 49, 52, 53, 54, 55, 56, 57, 58]
        Route 2: [45]
        Route 3: [64, 68, 69, 70, 44, 76, 83, 84]
        Route 4: [112, 113, 122, 117]
        Route 5: [99, 100, 101, 102, 79, 48, 80, 116]
        Route 6: [42, 43, 44, 50, 86, 87, 90, 92, 94, 95, 96, 103, 104, 106, 107, 111, 114, 115, 118, 119]
        Route 7: [97, 39, 40, 73, 41, 21, 22, 23, 58, 59, 62]
        Route 8: [18, 26, 29, 20]
        Route 9: [53]
---------------------------------
HUB | ID: 4 | Status: close
---------------------------------
HUB | ID: 5 | Status: close
---------------------------------
HUB | ID: 6 | Status: close
---------------------------------
HUB | ID: 7 | Status: open
        Route 1: [61, 64, 65, 67, 71, 72, 55, 58, 76, 84, 89, 92, 95, 94, 91, 90, 102]
        Route 2: [32, 33, 34, 35, 36, 37, 38, 39, 40, 41, 31]
        Route 3: [65, 68, 69, 85, 46, 45, 27, 21, 22, 15, 53, 55, 51, 50]
        Route 4: [9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 22, 23, 24, 25, 26, 27, 28, 29, 30, 31, 46, 50, 51, 70]
        Route 5: [36, 13, 15]
        Route 6: [34, 36, 74, 43, 24, 25]
        Route 7: [18, 19, 20, 21, 22]
        Route 8: [33, 42, 43, 44, 45, 46, 47]
---------------------------------
HUB | ID: 8 | Status: close
---------------------------------
Total Cost: 191329.102
```

Figure 5.6 Detailed local routes for instance of 115 non-hub nodes

5.2 Validation

In this section, the results of the exact and ALNDS methods are taken into evaluation to better evaluate their performance. The below table illustrates the summary results and corresponding gap of the objective value and the solving time on the generated instances:

Table 5.8 Comparison between CPLEX and ALNDS

Instance C - Qv (ton) - K	Exact Method			ALNDS			% Gap in Objective value	% Gap in Solving time
	Objective value ($)	Hub open	Solving time (s)	Objective value ($)	Hub open	Solving time (s)		
10 - 5 - 4	96650.824	1, 2, 3, 4	453.65	98016.34	1, 2, 3, 4	1.21	1.39	99.73
20 - 8 - 10	100863.248	1, 2, 3, 7	904.6	105427.735	1, 2, 3, 7	3.04	4.33	99.66
30 - 10 - 17	131688.83	1, 2, 3, 4	7200	117418.54	1, 2, 3, 4	14.76	12.15	99.80
40 - 15 - 20	149,036.48	1, 2, 3, 4	7200	130613.62	1, 2, 3, 5	60.15	14.10	99.16
50 - 18 - 30	184,682.58	1, 2, 3, 4	7200	144507.84	1, 2, 3, 4	121.17	21.75	98.32
70 - 20 - 34	-	-	-	158721.26	1, 2, 3, 7	302.81	-	-
115 - 30 - 54	-	-	-	191329.102	1, 2, 3, 7	2580.34	-	-

In terms of the opened hub solution, the three first and fifth instances have the same result using both techniques. In the fourth instance, hub 4 is selected while hub 5 is opened using Exact and ALNDS method accordingly. This implies comparative consistency in the solution of the two methods. In the large instances of 70 and 115 non-hub nodes, the results are obtained by only ALNNDS since the Exact method cannot solve these large datasets. And the hub opening in those two instances are hub 7 which is located in Binh Phuoc province. Nonetheless, it is not firmly to conclude the company should open new hub in Binh Phuoc because there is another resulted potential new hub in Lam Dong in the three smaller instances. Thus, the researchers and executives should consider internal and external factors to objectively yield the result in the favour of Viettel Post's current system.

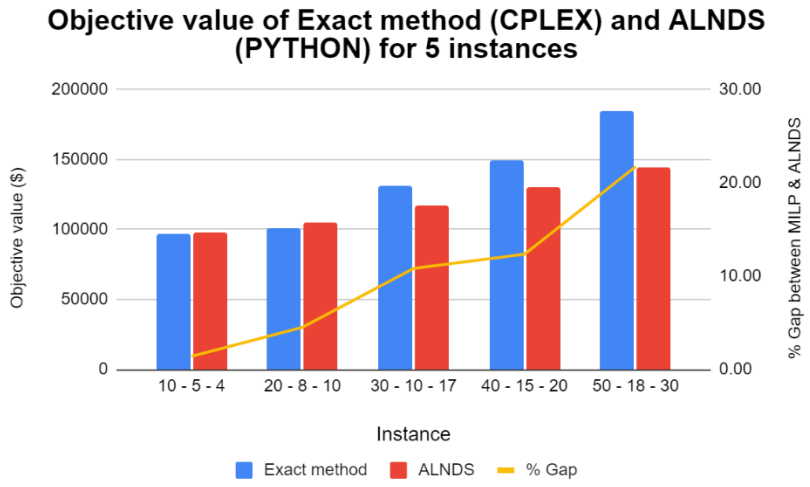


Figure 5.7 Mixed chart for two methods' performance in objective value

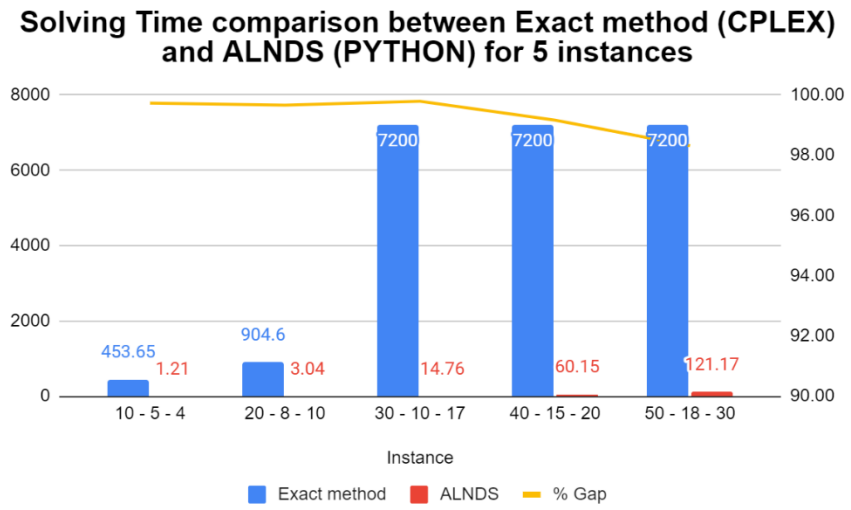


Figure 5.8 Mixed chart for two methods' performance in computational time

Regarding the computational time, the ALNDS using PYTHON has strongly outperformed Exact method using CPLEX in all instances, meanwhile, CPLEX has solved from 7 to 15 minutes for small scale and taken hours to yield result in medium to large-sized instances. Both techniques' solving time enhances as the instance size increases, however, there is a big gap between the computational time in Exact method and ALNDS. Thus, the metaheuristic gets a strong advantage in time consideration with finding near-optimal solutions while Exact method obtains exactly optimal solutions that takes considerable time.

Considering the overall costs for each dataset, the CPLEX performed slightly better than the metaheuristic technique with 1.39% and 4.33% in objective value gaps for small size instances (10 and 20 non-hub nodes). On the contrary, the solution quality for larger instances (30, 40, 50 non-hub nodes) obtained from Exact method using CPLEX declined since the problem is a complex and hard problem to exactly solve under reasonable time. For the metaheuristic algorithm, it yielded higher quality solutions with smaller costs for the MAHLRP within a much shorter computing time in a bigger numerical size with 12.15%, 14.1%, and 21.75% respectively in the case of 30, 40, 50 non-hub nodes. The larger the size of the instance, the higher the gap in objective value between exact and metaheuristic solutions.

Therefore, the ALNDS can be highly sufficient under the limited time case or various techniques comparison study to improve in much higher and feasible under more particular restraints acquisition. Nonetheless, the hub location and non-hub assignment are considered as a long-term decision, which is not strictly time sensitive. Thus, the researchers need to smartly select and combine feasible solution methods for adapting practical cases.

5.3 Result Analysis

5.3.1 Sensitivity Analysis

Dataset 1 is taken into consideration due to shorter computational time for sensitivity analysis to clearly comprehend the adjustment pattern of the objective values.

5.3.1.1 Changes in total vehicle capacity

Since the objective function cost does not include the fixed cost of the vehicle, the total homogeneous vehicle capacity is investigated to adjust for the flow quantity cost calculation.

Table 5.9 Total vehicle capacity (kg) sensitivity analysis

Vehicle capacity (kg)	3000	4000	5000	4000	5000
Fleet size	**5**	**4**	**4**	**6**	**5**
Total vehicle capacity (kg)	**15000**	**16000**	**20000**	**24000**	**25000**
Obj value ($)	99173.411	98632.166	**96650.824**	95278.382	95326.708
% Gap with the original optimal value	2.61%	2.05%	0.00%	1.42%	1.37%

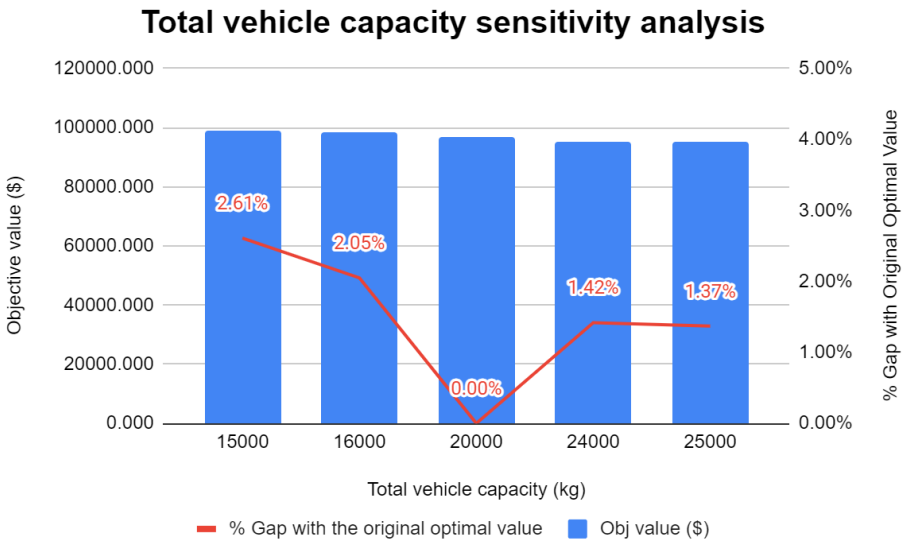


Figure 5.9 Mixed chart for fleet size sensitivity analysis

The total capacity of the vehicle is calculated by the multiplication of each vehicle's capacity and the fleet size. Based on the chart, the total operation cost declines when the total vehicle capacity is enlarged because the local routes will either insert more non-hub nodes until the vehicle capacity is fully loaded or split into multiple trips for shorter travel distances leading to lower transportation costs.

5.3.1.2 Changes in local tour and inter-hub costs

The unit local tour cost β and unit inter-hub transportation cost α are evaluated simultaneously to assess the alteration of the total cost.

Table 5.10 Unit Local tour and Inter-hub costs sensitivity analysis

α	**0.6**			**0.8**			**1**		
β	0.03	0.05	0.07	0.03	0.05	0.07	0.03	0.05	0.07
Objective value ($)	93161.15	96432.39	99499.94	93392.71	**96650.824**	99703.15	93625.06	96868.3	99907.1
% Gap with Original Optimal Value	3.75%	0.23%	2.86%	3.49%	0.00%	3.06%	3.23%	0.22%	3.26%

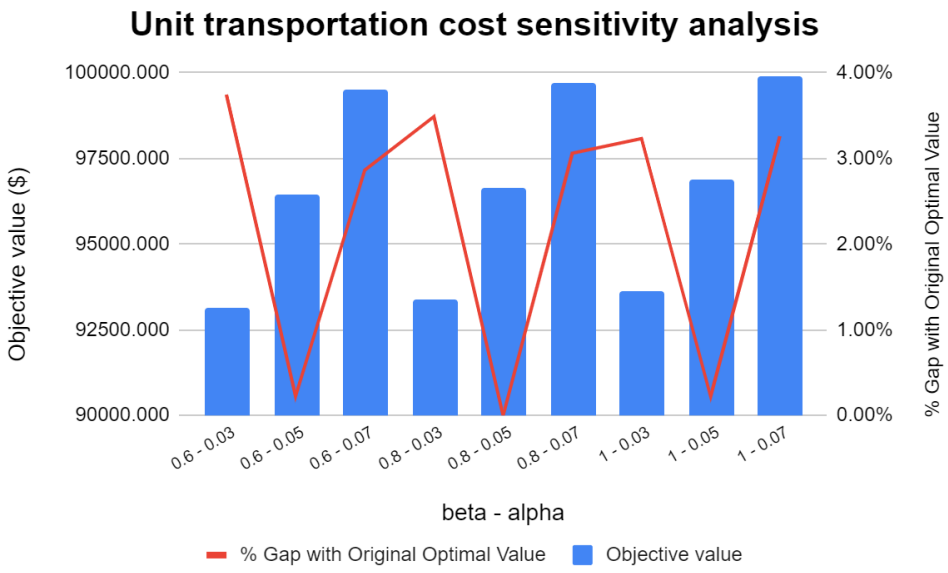


Figure 5.10 Mixed chart for $\alpha - \beta$ sensitivity analysis

Increasing the α while keep β constant in three scenarios of β modification delivers higher objective values. Similarly, enhancing the β with the consideration of remaining α fixed also results a slightly bigger total cost.

To sum up, the total vehicle capacity has a lesser impact on the objective value compared to the unit transportation cost. The whole cost will eventually rise in accordance with the same pattern, and vice versa when modifying the unit local tour and inter-hub transportation costs. On the contrary, the total cost is gradually decreased when increasing the total vehicle capacity. Besides, the hub capacity and its fixed cost can play a significant role in the final cost. As a result, managers or stakeholders should look for ways to reduce the associated costs as much as feasible and improve the capacity in order to provide the best total cost.

5.3.1 Environmental, social, and economic impacts

- Environmental impact

- Transportation Emissions: The use of automobiles during the routing phase can result in greenhouse gas emissions and air pollution. However, route optimization can assist in reducing the distance traveled and, as a result, emissions and the carbon footprint. Efficient allocation and routing decisions

can help to reduce energy consumption, especially when considering electric or hybrid vehicles, which emit less pollution and use less energy than typical fossil fuel vehicles.

- Land usage: The establishment of hub facilities and their locations can have an impact on land usage, potentially resulting in the conversion of green spaces or agricultural land. Natural habitats can be negatively impacted by improper planning and location choices.

- Social impact

- Accessibility: The location of hubs and the allocation of resources can have a substantial impact on customer access to goods and services. Well-planned hub locations, particularly in rural or disadvantaged areas, can boost accessibility.
- Job Opportunities: The formation of hubs can lead to the creation of new jobs in the local community, particularly in logistics and warehouse management.
- Congestion on Roads: Efficient routing can alleviate traffic congestion on roads, benefiting the community by reducing travel time and increasing overall traffic flow.

- Economic impact

- Cost Savings: By reducing transportation expenses, inventory holding costs, and facility running costs, organizations can save money by optimizing hub placement and routing decisions.
- Improved allocation and routing can improve supply chain efficiency, resulting in faster delivery times and better customer service.
- Revenue Generation: Strategically placed hubs and efficient routing can help businesses reach more customers and perhaps enhance revenue streams.

It is critical to recognize that the environmental, social, and economic impacts are inextricably linked. For example, more ecologically friendly routing selections may result in lower operational costs and, as a result, a favorable economic impact. Likewise, greater accessibility and transportation services can have a good social influence on communities.

To address these effects, decision-makers can adopt multi-objective optimization systems that optimize for environmental, social, and economic goals at the same time. Incorporating sustainability criteria into the optimization model can also help balance these implications and promote long-term decision-making.

Chapter 6: CONCLUSION

6.1 Result Discussion and Implication

The research presents the problem of Viettel Post case study regarding the delivery rate, which refers to the hub location and routing problem with the multi-allocation mechanism. The southern region of Vietnam is taken into investigation to narrow the scope problem with 23 provinces from Lam Dong to Ca Mau. To attach to practical scenarios, three existing hubs at Ho Chi Minh, Dong Nai, Can Tho are also considered to better evaluate the result performance with taking the value of hub installment equal to 1 in mathematical model formulation and metaheuristic algorithm in all scenarios. The five datasets from 10 to 50 non-hub nodes are executed using MIP by CPLEX and the ALNDS by PYTHON to objectively assess the objective function performance in each case. The results show that the MIP yields more optimal solution in small cases with 10 and 20 non-hub nodes, meanwhile, the ALNDS obtains the higher quality solution in medium to large cases with lower computing time compared to MILP. However, CPLEX can only obtain solutions up to 50 non-hub nodes instance. With 60 non-hub nodes and more, CPLEX is unavailable to implement with such big data since it is out of memory. Therefore, another large dataset of 115 non-hub nodes has been taken into consideration for the algorithm to measure its effectiveness on the incredibly large case. Under 30 minutes of the computational time, the metaheuristic has optimally solved with the opening hub number is 8 with a total of 35 local routes. The study shows that the metaheuristic efficiently performed in tackling this NP-hard case. Some parameters adjustment has been operated to assess its impact on the total cost value including unit costs and total vehicle capacity, which results in higher costs when increasing and lower costs when enhancing respectively.

6.2 Limitation of the Study and Recommendations for Future Research

The study about hub location and routing still has several limitations:

- Moderate research about this problem, especially for multi allocations; thus, there requires more development on adaptable constraints and robust optimization technique to real world scenarios.
- No direct connections between non-hub nodes are allowed due to the characteristics of the warmed-up transportations system studied for this type of problem.
- One echelon is taken into consideration from Hub to Post Offices.
- Data size is not large enough to tackle for practical cases.
- External conditions are not studied such as time windows, traffic conditions, surrounding environment to establish new hub, vehicle availability, etc.
- Lacking real-world logistics data to conduct in-depth case studies and empirical analyses to test the proposed algorithms and evaluate their effectiveness in practical scenarios.

The location and split delivery vehicle routing problems are well-known; however, combine these two in one model can require more time for investment and development with specific traits such as multi allocations. Some suggestions can be made:

- Robust optimization techniques: are recommended to consider for addressing uncertainty in input data, such as demand variations, transportation costs, and so on to make solutions more resilient to changes in the real-world environment.
- Adaptation in Real-Time: Create dynamic and adaptive algorithms that can swiftly react to changing demands, traffic circumstances, and vehicle availability in real-time or near real-time settings.
- Green and Sustainable Solutions: Incorporate environmental concerns into the MAHLRP by reducing carbon emissions, energy consumption, and encouraging environmentally friendly transportation alternatives.
- Multi-echelon Hub Networks: Extend the MAHLRP to consider multi-echelon hub networks, where hubs may have varying levels of capacity and capabilities, in order to more realistically mimic complex logistics networks.

- Benchmark Instance Development: Create standardized benchmark instances and datasets for the MAHLRP to allow for fair comparisons of different methods and to promote additional research in this area.

References

[1]. Đinh Thu Phương, Giải pháp cho hoạt động giao hàng chặng cuối trong logistics, Kỷ yếu hội thảo quốc tế Thương mại và phân phối lần 1 năm 2018, Đại học Bà Rịa – Vũng Tàu, 2018.

[2]. Trang Chủ. Viettel Post. (n.d.). Retrieved March 5, 2023, from https://viettelpost.com.vn/

[3]. Taherkhani, G., Alumur, S. A., & Hosseini, M. (2020). Benders decomposition for the profit maximizing capacitated hub location problem with multiple demand classes. Transportation Science, 54(6), 1446-1470.

[4]. Nagy, G., & Salhi, S. (1998). The many-to-many location-routing problem. Top, 6, 261-275.

[5]. Sun, J. U. (2013). An integrated hub location and multi-depot vehicle routing problem. Applied Mechanics and Materials, 409, 1188-1192.

[5]. Basirati, M., Akbari Jokar, M. R., & Hassannayebi, E. (2020). Bi-objective optimization approaches to many-to-many hub location routing with distance balancing and hard time window. Neural Computing and Applications, 32, 13267-13288.

[6]. Bostel, N., Dejax, P., & Zhang, M. (2015). A model and a metaheuristic method for the hub location routing problem and application to postal services. In 2015 International Conference on Industrial Engineering and Systems Management (pp. 1383–1389).

[7]. Kartal, Z., Hasgul, S., & Ernst, A. T. (2017). Single allocation p-hub median location and routing problem with simultaneous pick-up and delivery. Transportation Research Part E: Logistics and Transportation Review, 108, 141–159.

[8]. Karimi, H., & Setak, M. (2018). A bi-objective incomplete hub location-routing problem with flow shipment scheduling. Applied Mathematical Modelling, 57, 406-431.

[9]. Ratli, M., Urošević, D., El Cadi, A. A., Brimberg, J., Mladenović, N., & Todosijević, R. (2022). An efficient heuristic for a hub location routing problem. Optimization Letters, 1-20.

[10]. Wu, Y., Qureshi, A. G., & Yamada, T. (2022). Adaptive large neighborhood decomposition search algorithm for multi-allocation hub location routing problem. European Journal of Operational Research, 302(3), 1113-1127.

[11]. Wasner, M., & Zäpfel, G. (2004). An integrated multi-depot hub-location vehicle routing model for network planning of parcel service. International journal of production economics, 90(3), 403-419.

[12]. Çetiner, S., Sepil, C., & Süral, H. (2010). Hubbing and routing in postal delivery systems. Annals of Operations research, 181, 109-124.

[13]. Catanzaro, D., Gourdin, E., Labbé, M., & Özsoy, F. A. (2011). A branch-and-cut algorithm for the partitioning-hub location-routing problem. Computers & Operations Research, 38(2), 539–549.

[14]. Basirati, M., Jokar, M. R. A., & Hassannayebi, E. (2019). Bi-objective optimization approaches to many-to-many hub location routing with distance balancing and hard time window. Neural Computing and Applications, 1–22.

[15]. Fontes, F. F. D. C., & Goncalves, G. (2021). A variable neighbourhood decomposition search approach applied to a global liner shipping network using a hub-andspoke with sub-hub structure. International Journal of Production Research, 59(1), 30–46.

[16]. de Camargo, R. S., de Miranda, G., & Løkketangen, A. (2013). A new formulation and an exact approach for the many-to-many hub location-routing problem. Applied Mathematical Modelling, 37(12-13), 7465-7480.

[17]. Rodríguez-Martín, I., Salazar-González, J. J., & Yaman, H. (2014). A branch-and-cut algorithm for the hub location and routing problem. Computers & Operations Research, 50, 161–174.

[18]. Karimi, H. (2018). The capacitated hub covering location-routing problem for simultaneous pickup and delivery systems. Computers & Industrial Engineering, 116, 47-58.

[19]. Demır, İ., Ergın, F. C., & Kıraz, B. (2019). A new model for the multi-objective multiple allocation hub network design and routing problem. IEEE Access, 7, 90678-90689.

[20]. Rahmati, R., Bashiri, M., Nikzad, E., & Siadat, A. (2022). A two-stage robust hub location problem with accelerated Benders decomposition algorithm. International Journal of Production Research, 60(17), 5235-5257.

[21]. Nambiar, J. M., Gelders, L. F., & Van Wassenhove, L. N. (1981). A large scale location-allocation problem in the natural rubber industry. European Journal of Operational Research, 6(2), 183-189.

[22]. Branco, I. M., & Coelho, J. D. (1990). The Hamiltonian p-median problem. European Journal of Operational Research, 47(1), 86-95.

[23]. Min, H. (1996). Consolidation terminal location-allocation and consolidated routing problems. Journal of Business Logistics, 17(2), 235.

[24]. Billionnet, A., Elloumi, S., & Djerbi, L. G. (2005). Designing radio-mobile access networks based on synchronous digital hierarchy rings. Computers & operations research, 32(2), 379-394.

[25]. Perl, J., & Daskin, M. S. (1985). A warehouse location-routing problem. Transportation Research Part B: Methodological, 19(5), 381-396.

[26]. Salhi, S., & Fraser, M. (1996). An intergrated heuristic approach for the combined location vehicle fleet mix problem. Studies in Locational Analysis, 8, 3-22.

[27]. Wu, T. H., Low, C., & Bai, J. W. (2002). Heuristic solutions to multi-depot location-routing problems. Computers & Operations Research, 29(10), 1393-1415.

[28]. Schwardt, M., & Dethloff, J. (2005). Solving a continuous location-routing problem by use of a self-organizing map. International Journal of Physical Distribution & Logistics Management, 35(6), 390-408.

[29]. Albareda-Sambola, M., Díaz, J. A., & Fernández, E. (2005). A compact model and tight bounds for a combined location-routing problem. Computers & operations research, 32(3), 407-428.

[30]. Melechovský, J., Prins, C., & Calvo, R. W. (2005). A metaheuristic to solve a location-routing problem with non-linear costs. Journal of Heuristics, 11, 375-391.

[31]. Ghaffarinasab, N., & Kara, B. Y. (2019). Benders decomposition algorithms for two variants of the single allocation hub location problem. Networks and spatial economics, 19, 83-108.

[32]. Ghaffarinasab, N., & Kara, B. Y. (2022). A conditional β-mean approach to risk-averse stochastic multiple allocation hub location problems. Transportation Research Part E: Logistics and Transportation Review, 158, 102602.

[33]. de Sá, E. M., Morabito, R., & de Camargo, R. S. (2018). Efficient Benders decomposition algorithms for the robust multiple allocation incomplete hub location problem with service time requirements. Expert Systems with Applications, 93, 50-61.

[34]. Mokhtar, H., Krishnamoorthy, M., & Ernst, A. T. (2019). The 2-allocation p-hub median problem and a modified Benders decomposition method for solving hub location problems. Computers & Operations Research, 104, 375-393.

[35]. Najy, W., & Diabat, A. (2020). Benders decomposition for multiple-allocation hub-and-spoke network design with economies of scale and node congestion. Transportation research part b: methodological, 133, 62-84.

[36]. Taherkhani, G., Alumur, S. A., & Hosseini, M. (2020). Benders decomposition for the profit maximizing capacitated hub location problem with multiple demand classes. Transportation Science, 54(6), 1446-1470.

[37]. Silva, M. R., & Cunha, C. B. (2017). A tabu search heuristic for the uncapacitated single allocation p-hub maximal covering problem. European Journal of Operational Research, 262(3), 954-965.

[38]. Ghaffarinasab, N., Zare Andaryan, A., & Ebadi Torkayesh, A. (2020). Robust single allocation p-hub median problem under hose and hybrid demand uncertainties: models and algorithms. International Journal of Management Science and Engineering Management, 15(3), 184-195.

[39]. Alumur, S. A., Campbell, J. F., Contreras, I., Kara, B. Y., Marianov, V., & O'Kelly, M. E. (2021). Perspectives on modeling hub location problems. European Journal of Operational Research, 291(1), 1-17.

[40]. Ghaffarinasab, N., Motallebzadeh, A., Jabarzadeh, Y., & Kara, B. Y. (2018). Efficient simulated annealing based solution approaches to the competitive single and multiple allocation hub location problems. Computers & Operations Research, 90, 173-192.

Appendices

Appendix A DATA COLLECTION

Some notices for data presentation:

- The 8 hub node indices are represented from 1 – 8. Followingly, non-hub node indices will continue from 9 to so on.
- The full dataset entails 115 customers and other datasets take 10, 20, 30, 40, 50 non-hub nodes correspondingly from that full one with remaining non-hub nodes data from smaller datasets. For instance, 10 post offices' locations in dataset 1 are kept constant in datasets 2, 3, 4, 5 and add in respective post offices number corresponding to that dataset size. The datasets 2, 3, and 4 are similar.
- The locations from smaller datasets will be equally taken from the full dataset to better evaluate and yield more practical solution.
- The hub capacity and fixed cost remain constant in all datasets.
- The hub 1, 2, and 3 are assigned in each dataset to be always opened.
- Vehicle capacity: 5000 kg.
- Fleet size: 4 homogeneous vehicles.

Table A.1 Capacity and fixed cost of hub nodes

Hub	1	2	3	4	5	6	7	8
Capacity (kg)	250000	196000	153000	169000	182400	221000	205600	187000
Fixed cost ($)	0	0	0	96087	114747.11	168500	147054.55	121152.89

Table A.2 Location of Viettel Post Offices

No.	Address	x	y
1	QL28 Thôn 3, Lộc Bảo, Bảo Lâm, Lâm Đồng	11.790815	107.664755
2	Khu Phố Vĩnh Đông 2, Vĩnh Thuận, Kiên Giang	9.514600	105.248902
3	270 QL60, TT. Mỏ Cày, Mỏ Cày Nam, Bến Tre	10.129992	106.331674
4	818 Phú Riềng Đỏ, Tân Xuân, Đồng Xoài, Bình Phước	11.53036995	106.8961164
5	3 Trần Văn Trà, khu phố 3, Thạnh Hóa, Long An	10.71950095	106.1957144
6	158 Ngô Quyền, Phường 6, Đà Lạt, Lâm Đồng	11.95753449	108.427861
7	32 Triệu Việt Vương, Phường 4, TP Đà Lạt	11.93074114	108.4322518
8	66 Thống Nhất Liên Nghĩa Đức Trọng Lâm Đồng	11.73075667	108.3756147
9	717 Đ. Hùng Vương, Di Linh, Lâm Đồng	11.59465771	108.0801722
10	87 Trần Phú, Madagoil, Đạ Huoai, Lâm Đồng	11.38715628	107.531658
11	100 Đ. 30/4, Đạ Tẻh, Đạ The, Lâm Đồng	11.50706072	107.4848279
12	250 QL20, Lộc Nga, Bảo Lộc, Lâm Đồng	11.53506654	107.8607326
13	717 Đ. Hùng Vương, Di Linh, Lâm Đồng	11.59667198	108.078799
14	Thôn Phi Có, Đam Rông, Lâm Đồng	12.19006428	108.1364852
15	22 Thống Nhất, Thị trấn Lạc Dương, Lạc Dương, Lâm Đồng	12.01086763	108.4211952
16	18 Trần Hưng Đạo,Thạnh Mỹ, Đơn Dương, Lâm Đồng	11.75820514	108.4856317
17	110 Thôn 4, xã Quảng Khê, Đắk Glong, Đắk Nông	11.90912612	107.7985006
18	168 Nguyễn Tất Thành, Tổ 2, Gia Nghĩa, Đắk Nông	12.02117873	107.6740598
19	118 Nguyễn Tất Thành, Đắk Mâm, Krông Nô, Đắk Nông	12.4908024	107.8579419
20	ĐT681, Đắk Bút So, Tuy Đức, Đắk Nông	12.21055312	107.4624719
21	QL14, Nhân Cơ, Đắk R'Lấp, Đắk Nông	11.98536883	107.5843744
22	126 NGUYỄN TẤT THÀNH, THỊ TRẤN KIẾN ĐỨC, H.ĐẮK RLẤP, Đăk	11.99075764	107.5099724

	Nông		
23	Thôn 2, Tuy Đức, Đắk Nông	12.2121356	107.4654546
24	80 Đắc Phúc, Đắk Lao, Đắk Mil, Đắk Nông	12.45395301	107.6292369
25	319 Đường 8 Tháng 4, thị trấn Ma Lâm, Tp. Phan Thiết, Bình Thuận	10.93650613	108.1101311
26	65 ĐƯỜNG 25 THÁNG 13, Lạc Tánh, Phan Thiết, Bình Thuận	11.11075907	107.6747872
27	289 QL28, Ma Lâm, Hàm Thuận Bắc, Bình Thuận	11.08017596	108.1303169
28	332 Đường 3/2, ĐứcTài, Đức Linh, Bình Thuận	11.14682927	107.505043
29	37 Cách Mạng Tháng Tám, Tân Nghĩa, Hàm Tân, Bình Thuận	10.82269857	107.7268458
30	190 Nguyễn Tất Thành, Chợ Lầu, Bắc Bình, Bình Thuận	11.24063478	108.5013052
31	4 Trần Quang Khải, Phường Tân Thiện, Đồng Xoài, Bình Phước	11.53480192	106.9096873
32	Tổ 6 kp tân an, Tân Phú, Đồng Phú, Bình Phước	11.4889968	106.8652564
33	QL14, Đức Lập, Bù Đăng, Bình Phước	11.81258887	107.2414704
34	QL13, Chơn Thành, Bình Phước	11.46814482	106.6229736
35	Thôn tân hiệp xã, Bu Nho, huyện phú riềng, Bình Phước	11.73130154	106.8730781
36	220, Thị trấn Thanh Bình, Bù Đốp, Bình Phước	11.95523947	106.7916822
37	ĐT741, Đăk Ơ, Bù Gia Mập, Bình Phước	12.0493627	107.0870615
38	220 Nguyễn Huệ, P. An Lộc, Tx. Bình Long, Bình Phước	11.6546401	106.6091297
39	358 ĐT741, Khu Phố 5, Phú Giáo, Bình Dương	11.30173576	106.8001624
40	55 Đ. NC, Lai Uyên, Bến Cát, Bình Dương	11.26683864	106.6310755
41	139 Độc Lập, kp1, Dầu Tiếng, Bình Dương	11.29201675	106.3567913
42	228 Đ. Phạm Ngũ Lão, Hiệp Thành, Thủ Dầu Một, Bình Dương	10.98974772	106.6563386
43	1823E, Tổ 8, Ấp 5, Xã Thạnh Phú, Huyện Vĩnh Cửu, Tỉnh Đồng Nai	11.0118584	106.8400116
44	94 Hùng Vương, TT. Lộc Thắng, Bảo Lâm, Lâm Đồng	11.6155116	107.8387422
45	151 Trần Phú, TT. Gia Ray, Xuân Lộc, Đồng Nai	10.9225085	107.4137288
46	142 A, Phú Thạnh, Nhơn Trạch, Đồng Nai	10.7052997	106.8325601
47	473 QL1A, Xuân Hoà, Long Khánh, Đồng Nai	10.9245372	107.2349139

48	685 đường Trường Chinh, Tổ 26 khu Phước Hải, TT. Long Thành, Long Thành, Đồng Nai	10.8143218	106.9569097
49	269 Khu 5, TT. Định Quán, Định Quán, Đồng Nai	11.1939372	107.3435744
50	QL56, Long Giao, Cẩm Mỹ, Đồng Nai	10.8254054	107.2329809
51	207 KP TƯỜNG THÀNH, THỊ TRẤN ĐẤT ĐỎ, TP. Bà Rịa - Vũng Tàu	10.48945462	107.2681427
52	673 Đường 30/4, Phường Rạch Dừa, TP. Bà Rịa - Vũng Tàu	10.4002000	107.1172815
53	110 27 tháng 4, TT. Phước Bửu, Xuyên Mộc, Bà Rịa - Vũng Tàu	10.55329896	107.4042834
54	7 Đ. Bình Giã, TT. Ngãi Giao, Châu Đức, Bà Rịa - Vũng Tàu	10.65511056	107.246139
55	QL51, Tân Hoà, Tân Thành, Bà Rịa - Vũng Tàu	10.51544934	107.087723
56	198 Võ Văn Kiệt, Phường Long Tâm, Thành Phố Bà Rịa - Vũng Tàu	10.50452057	107.1937755
57	250,ĐƯỜNG NGUYỄN CHÍ THANH,KHU PHỐ 5,PHƯỜNG 3,THÀNH PHỐ TÂY NINH,TỈNH TÂY NINH	11.35243763	106.1022277
58	179 Xuyên Á, khu phố Tân Lộ, Trảng Bàng, Tây Ninh	11.04956231	106.3503873
59	34 Nguyễn Chí Thanh, Tân Biên, Tây Ninh	11.54315417	106.006565
60	Khu phố 4, Phường 3, Tp. Tây Ninh, Tây Ninh	11.30454885	106.104856
61	361 Nguyễn Văn Tăng, Long Thạnh Mỹ, Quận 9, Thành phố Hồ Chí Minh	10.85919854	106.8311895
62	398 Tỉnh lộ 7 Ấp Chợ Cũ 2, Củ Chi, Thành phố Hồ Chí Minh	11.09714306	106.5126746
63	488a Điện Biên Phủ, Phường 21, Bình Thạnh, Thành phố Hồ Chí Minh	10.80115789	106.7126167
64	47 Trần Văn Giàu, TT. Tầm Vu, Châu Thành, Long An	10.43712277	106.46433
65	40 Đ. Nguyễn Hữu Thọ, TT. Bến Lức, Bến Lức, Long An	10.6427668	106.4837065
66	62 ấp Thuận Tây, Thuận Thành, Cần Giuộc, Long An	10.6550349	106.575912
67	147 đường Trần Văn Ngà, khu phố 1, Tân Thạnh, Long An	10.63780779	106.0379321
68	666 ấp Bình Hữu 1, Đức Hòa Thượng, Đức Hòa, Long An	10.84494621	106.4655514
69	36 KHU 2 TT CẦN ĐƯỚC, THỊ TRẤN CẦN ĐƯỚC, Long An	10.50324481	106.6067421
70	120 đường 3/2, TT. Tân Hưng, Tân Hưng, Long An	10.87690339	105.6444747
71	541 Trần Văn Ưng, TT. Chợ Gao, Chợ Gạo, Tiền Giang	10.3590094	106.4624127
72	QL1A Ấp cữu hòa, Thân Cửu Nghĩa, Châu Thành, Tiền Giang	10.43224985	106.3435967

73	73 ĐT867, TT. Mỹ Phước, Tân Phước, Tiền Giang	10.47475973	106.1961939
74	300 Đoàn Hoàng Minh, P. Phú Khương, Bến Tre	10.2533191	106.3708455
75	41A1 Trần Hưng Đạo, khu phố 2, Ba Tri, Bến Tre	10.04252952	106.5942143
76	340 Kp3, TT. Giồng Trôm, Giồng Trôm, Bến Tre	10.16510531	106.5071721
77	05 Nguyễn Minh Trí, khóm Mỹ Thuận, Cao Lãnh, Đồng Tháp	10.44764195	105.6953568
78	304C QL80, Khóm Bình Phú Quới, Lấp Vò, Đồng Tháp	10.36249042	105.5236292
79	789 Trần Hưng Đạo, Khóm 2, Tam Nông, Đồng Tháp	10.67179976	105.5572191
80	630 QL80, TT. Lai Vung, Lai Vung, Đồng Tháp	10.28618662	105.6632727
81	7 Đường Nguyễn Tất Thành, Phường 1, Sa Đéc, Đồng Tháp	10.29731747	105.7613081
82	173 Đường Số 1, Thanh Đức, Long Hồ, Vĩnh Long	10.2473163	105.9985241
83	Số 30/9, Ấp Phước Hòa, Long Hồ, Vĩnh Long	10.1742826	105.9323544
84	9B Khóm 2, Vũng Liêm, Vĩnh Long	10.10214899	106.2028058
85	53 Hùng Vương, Phường 3, Trà Vinh	9.937272078	106.342243
86	Khóm Minh Thuận B, Cầu Ngang, Trà Vinh	9.797194756	106.4525907
87	10 Đường Hai Bà Trưng, TT. Tiểu Cần, Tiểu Cần, Trà Vinh	9.821095616	106.1898047
88	249 ĐƯỜNG 3/2, TT TRÀ CÚ, Trà Vinh	9.699968192	106.2592411
89	102 30 Tháng 4, TT. Cầu Kè, Cầu Kè, Trà Vinh	9.884192407	106.0521302
90	253 Lê Lợi, TT. An Châu, Huyện Châu Thành, An Giang	10.43188587	105.3995703
91	232 Đường Nguyễn Huệ, TT. Núi Sập, Thoại Sơn, An Giang	10.27479743	105.2688771
92	95 Đ. Chu Văn An, TT. Phú Mỹ, Phú Tân, An Giang	10.59632195	105.3564033
93	160 Trần Hưng Đạo, TT. Tri Tôn, Tri Tôn, An Giang	10.42471503	105.0006958
94	327 QL91, TT. Cái Dầu, Châu Phú, An Giang	10.57528072	105.2374515
95	ẤP THỚI THUẬN A, Thới Lai, Cần Thơ	10.14626483	105.5425058
96	90 Phan Văn Trị, Thị trấn Phong Điền, Phong Điền, Cần Thơ	10.00215576	105.6702843
97	Khu phố Thị Tứ, huyện Hòn Đất. Kiên Giang	10.12056725	105.0179311
98	603 QL61, TT. Minh Lương, Châu Thành, Kiên Giang	9.941101267	105.1560279

99	38 Khu Phố 3, TT. Giồng Riềng, Giồng Riềng, Kiên Giang	9.902608523	105.3092484
100	Ấp Đặng Văn Do, Thạnh Yên, U Minh Thượng, Kiên Giang	9.679449957	105.1254665
101	80 đường 30 tháng Tư, phường Thuận An, Long Mỹ, Hậu Giang	9.682628033	105.5670221
102	960 ấp Cầu Xáng, Tân Bình, Phụng Hiệp, Hậu Giang	9.81190384	105.6484476
103	72 đường Hùng Vương, ấp Mỹ Quới, Phụng Hiệp, Hậu Giang	9.787509395	105.7435486
104	269 Trần Hưng Đạo, khóm 1, Sóc Trăng	9.3249241	105.9799137
105	193 đường Triệu Nương, TT. Mỹ Xuyên, Mỹ Xuyên, Sóc Trăng	9.617377419	105.9914367
106	159 Đường Hùng Vương, TT. Huỳnh Hữu Nghĩa, Sóc Trăng	9.63597111	105.8137633
107	103 Đoàn Thế Trung, TT. Cù Lao Dung, Cù Lao Dung, Sóc Trăng	9.670069441	106.1552614
108	Ấp Nội Ô, Ngan Dừa, Hồng Dân, Bạc Liêu	9.579136216	105.4498838
109	224 QL1A, Ấp thị trấn A, Hoà Bình, Bạc Liêu	9.2876330	105.6345973
110	346 Ninh Bình, Phường 2, Bạc Liêu	9.277277463	105.7246616
111	153 Khóm 2 Phường HỘ PHÒNG Thị xã, Giá Rai, Bạc Liêu	9.233044832	105.4099523
112	11 Diêm điền, Điền Hải, Đông Hải, Bạc Liêu	9.117357664	105.491335
113	Ấp long thành, Phước Long, Bạc Liêu	9.443937372	105.4635318
114	Khóm 8, TT. Năm Căn, Cà Mau	8.87925117	105.0091826
115	Cà Phê, Cạnh, Khóm 1, U Minh, Cà Mau	9.424950175	104.9692406

Table A.3 Distance among hub and non-hub nodes

Distance (km)	1	2	3	4	5	6	…	65	66	67	68	69	70	71	…	118	119	120	121	122	123
1	0.0	128.46	42.69	65.83	211.31	86.29	…	29.66	11.57	49.21	27.67	22.46	67.95	16.93	…	240.52	257.72	235.03	267.3	241.56	247.84
2	128.46	0.0	167.47	191.42	82.95	62.53	…	141.31	132.38	87.24	101.86	110.64	70.8	116.1	…	112.24	132.73	106.65	139.49	115.5	120.77
3	42.69	167.47	0.0	23.96	249.51	117.1	…	55.15	35.47	81.9	65.96	56.84	110.54	59.31	…	279.65	292.91	273.17	304.43	277.54	284.45
4	65.83	191.42	23.96	0.0	273.37	140.17	…	72.67	59.37	105.56	89.91	80.79	133.76	81.97	…	303.58	316.28	297.01	328.13	301.06	308.08
5	211.31	82.95	249.51	273.37	0.0	136.95	…	223.78	214.78	168.05	184.42	192.84	151.86	199.03	…	32.25	56.85	23.72	57.13	38.88	40.58
6	86.29	62.53	117.1	140.17	136.95	0.0	…	109.35	85.47	37.11	59.39	64.2	64.97	80.84	…	168.63	176.28	160.04	189.4	161.5	168.96
…	…	…	…	…	…	…	…	…	…	…	…	…	…	…	…	…	…	…	…	…	…
65	29.66	141.31	55.15	72.67	223.78	109.35	…	0.0	39.49	73.58	50.62	49.64	72.78	28.51	…	251.18	273.48	247.32	280.68	256.6	262.07
66	11.57	132.38	35.47	59.37	214.78	85.47	…	39.49	0.0	48.73	30.59	22.07	75.92	27.42	…	244.61	259.56	238.48	270.18	243.79	250.42
67	49.21	87.24	81.9	105.56	168.05	37.11	…	73.58	48.73	0.0	22.96	27.13	51.68	45.35	…	198.7	211.16	191.62	222.58	195.65	202.55
68	27.67	101.86	65.96	89.91	184.42	59.39	…	50.62	30.59	22.96	0.0	10.17	48.72	22.57	…	214.09	230.15	208.14	240.1	214.09	220.5
69	22.46	110.64	56.84	80.79	192.84	64.2	…	49.64	22.07	27.13	10.17	0.0	58.82	24.32	…	222.83	237.52	216.52	248.13	221.72	228.35
70	67.95	70.8	110.54	133.76	151.86	64.97	…	72.78	75.92	51.68	48.72	58.82	0.0	52.09	…	178.55	203.46	175.2	208.95	186.1	190.94
71	16.93	116.1	59.31	81.97	199.03	80.84	…	28.51	27.42	45.35	22.57	24.32	52.09	0.0	…	227.5	247.17	222.7	255.53	230.59	236.47
…	…	…	…	…	…	…	…	…	…	…	…	…	…	…	…	…	…	…	…	…	…
118	240.52	112.24	279.65	303.58	32.25	168.63	…	251.18	244.61	198.7	214.09	222.83	178.55	227.5	…	0.0	52.95	15.26	38.37	40.66	34.41
119	257.72	132.73	292.91	316.28	56.85	176.28	…	273.48	259.56	211.16	230.15	237.52	203.46	247.17	…	52.95	0.0	41.56	26.23	18.1	18.88
120	235.03	106.65	273.17	297.01	23.72	160.04	…	247.32	238.48	191.62	208.14	216.52	175.2	222.7	…	15.26	41.56	0.0	34.49	26.76	22.76
121	267.3	139.49	304.43	328.13	57.13	189.4	…	280.68	270.18	222.58	240.1	248.13	208.95	255.53	…	38.37	26.23	34.49	0.0	29.97	20.87
122	241.56	115.5	277.54	301.06	38.88	161.5	…	256.6	243.79	195.65	214.09	221.72	186.1	230.59	…	40.66	18.1	26.76	29.97	0.0	9.86
123	247.84	120.77	284.45	308.08	40.58	168.96	…	262.07	250.42	202.55	220.5	228.35	190.94	236.47	…	34.41	18.88	22.76	20.87	9.86	0.0

Table A.4 Flow among non-hub nodes

Quantity (kg)	**9**	**10**	**11**	**12**	**13**	**14**	**…**	**65**	**66**	**67**	**68**	**69**	**70**	**71**	**…**	**118**	**119**	**120**	**121**	**122**	**123**
9	0	100	125	115	160	55	**…**	45	145	135	110	0	15	170	**…**	90	165	75	170	95	60
10	95	0	40	175	60	45	**…**	85	35	175	165	15	150	10	**…**	25	155	125	115	105	15
11	50	120	0	85	115	15	**…**	15	5	30	150	65	45	55	**…**	130	75	85	30	55	120
12	60	30	165	0	105	165	**…**	110	125	135	105	90	65	10	**…**	35	120	35	160	60	135
13	120	120	30	5	0	100	**…**	30	75	115	155	15	30	85	**…**	160	175	20	45	25	50
14	165	80	155	160	75	0	**…**	105	105	120	60	40	10	170	**…**	60	150	155	90	20	125
…	**…**	**…**	**…**	**…**	**…**	**…**	**…**	**…**	**…**	**…**	**…**	**…**	**…**	**…**	**…**	**…**	**…**	**…**	**…**	**…**	**…**
65	150	150	170	70	105	30	**…**	0	20	20	110	30	0	120	**…**	60	130	0	160	65	15
66	95	50	85	10	10	85	**…**	110	0	110	20	150	95	50	**…**	50	45	5	160	125	150
67	90	20	125	45	145	165	**…**	170	120	0	80	135	20	105	**…**	70	110	70	55	25	140
68	85	130	30	125	95	30	**…**	70	125	0	0	95	105	150	**…**	20	85	100	80	130	105
69	105	175	125	85	30	80	**…**	35	15	100	125	0	25	130	**…**	120	45	95	135	20	100
70	85	130	175	60	165	10	**…**	20	25	120	45	160	0	0	**…**	0	155	175	40	65	105
71	80	55	75	45	70	75	**…**	135	5	10	50	20	130	0	**…**	105	5	90	15	60	40
…	**…**	**…**	**…**	**…**	**…**	**…**	**…**	**…**	**…**	**…**	**…**	**…**	**…**	**…**	**…**	**…**	**…**	**…**	**…**	**…**	**…**
118	55	105	145	20	25	80	**…**	130	55	135	85	110	145	35	**…**	0	120	65	110	165	85
119	40	170	0	125	170	105	**…**	160	70	35	165	25	15	105	**…**	65	0	25	45	35	40
120	30	90	155	110	60	20	**…**	160	135	20	130	140	5	95	**…**	175	0	0	155	135	120
121	30	5	145	35	80	40	**…**	125	145	95	75	55	95	0	**…**	105	90	55	0	80	145
122	10	15	10	15	45	115	**…**	165	65	110	5	100	70	60	**…**	160	175	110	90	0	20
123	55	70	45	85	70	55	**…**	150	110	10	130	150	90	100	**…**	30	45	115	45	175	0

Appendix B CODE

B.1 CPLEX

- File mod

```
int numH = 8; // define number of H
int numC = 50; // define number of C
int M = 100000;
int p = 4;

range H = 1 .. numH; //range of Non-hub nodes
range C = numH+1 .. numH+numC; //range of Hub nodes
range N = 1 .. numH+numC; // range combination

//Parameter
float Qv = ...; //Capacity of vehicles
float Qh = ...; //Capacity of Hubs
float cn[numH+1..numH+numC, numH+1..numH+numC] = ...; //Distance between non-hub
nodes
float ch[1..numH, 1..numH] = ...; //Distance between hub nodes
float w[numH+1..numH+numC, numH+1..numH+numC] = ...; // Package quantity
int K = ...; // Limited fleet size of homogeneous vehicle
float F[H] = ...; //Fixed cost of hub installation

//Decision Variables
dvar int+ u[N][H]; //Total flows related to hub m before serving node i
dvar int+ v[N][H]; //Total flows related to hub m including up to node i
dvar float+ z[C][C][H][H]; //Vehicle moves from i to j departing from hub m
dvar boolean x[N][N][H]; //Allocation decision
dvar boolean y[H]; //Hub installment

//Objective function
minimize
    1 * sum (i, j in N, m in H) cn[i][j] * x[i][j][m] + sum (m in H) F[m] * y[m]
    + 0.004 * sum (m, n in H, i, j in C) w[i][j] * ch[m][n] * z[i][j][m][n];

//Constraints
subject to {
  // Existing hubs
  y[1] == 1;
  y[2] == 1;
  y[3] == 1;
  sum (m in H) y[m] >= p;
  //const 1:
  forall (i, j in C) {
  sum (m,n in H) z[i][j][m][n] == 1;
  }
  //const 2:
  forall (m in H) {
```

```
  sum (i, j in C, n in H) w[i][j] * z[i][j][m][n] + sum (i, j in C, n in H: n!=m)
  w[i][j] * z[i][j][n][m] <= Qh[m] * y[m];
  }
  //const 3:
  forall (m in H) {
  sum (j in N) x[m][j][m] <= M * y[m];
  }
  //const 4:
  forall (i in C) {
  sum (j in N, m in H: j!=i) x[i][j][m] >= 1;
  }
  //const 5:
  forall (i in C, m in H) {
  sum (j in N: i!=j) x[i][j][m] <= 1;
  }
  //const 6:
  forall (i in N, m in H) {
  sum (j in N) x[i][j][m] == sum (j in N) x[j][i][m];
  }
  //const 7:
  forall (m in H) {
  sum (j in N, n in H: n!=m) x[n][j][m] == 0;
  }
  //const 8:
  forall (i in C, m in H) {
  M * sum (j in N) x[i][j][m] >= sum (j in C, n in H) z[i][j][m][n] + sum(j in C,
  n in H) z[j][i][n][m];
  }
  //const 9:
  forall (i in N, j in C, m in H: j!=i) {
  u[i][m] + sum (t in C, n in H) w[j][t] * z[j][t][m][n] - M  *(1 - x[i][j][m])
  <= u[j][m];
  }
  //const 10:
  forall (i in C, j in N, m in H: i!=j) {
  v[i][m] - sum (t in C, n in H) w[t][i] * z[t][i][n][m] + M * (1 - x[i][j][m])
  >= v[j][m];
  }
  //const 11:
  forall (i in C, m in H) {
  v[i][m] <= Qv;
  }
  //const 12:
  forall (i in C, m in H) {
  u[i][m] + v[i][m] - sum (t in C, n in H) w[t][i] * z[t][i][n][m] <= Qv;
  }
  //const 13:
  sum (j in N, m in H) x[m][j][m] <= K;
}
```

- File dat

```
SheetConnection file("Dataset 5.xlsx");
```

```
Qv from SheetRead(file, "Hub!E14"); // Capacity of vehicles
Qh from SheetRead(file, "Hub!E3:E10"); // Capacity of Hubs
K from SheetRead(file, "Hub!E15"); // Limited fleet size of homogeneous vehicle
cn from SheetRead(file, "Distance!B2:BG59"); // Distance between nodes
ch from SheetRead(file, "Distance!B2:I9"); // Distance between hub nodes
w from SheetRead(file, "Quantity!B2:AY51"); // Quantity demand
F from SheetRead(file, "Hub!F3:F10"); // Fixed cost
```

B.2 PYTHON

B.2.1 Map Illustration

```
import gspread
import pandas as pd
import math
import numpy as np
import folium
import os
import googlemaps

sheet_id = '1R3Xl9bBWmjV9SsEYy3807JnFWn-wKYQe'
xls =
pd.ExcelFile(f"https://docs.google.com/spreadsheets/d/{sheet_id}/export?format=xls
x")

# read the data from the Excel file into a Pandas dataframe
df = pd.read_excel(xls, sheet_name = 'Coor', usecols=['x', 'y'])
df.index = range(1, df.shape[0] + 1)
print(df)

# create a map object centered on the first row of the dataframe
my_map = folium.Map(location=[df.iloc[0]['x'], df.iloc[0]['y']], zoom_start=7)

# iterate over the rows of the dataframe, creating a marker for each row
for index, row in df.iterrows()
        # Existing hub marks
        if index <= 3:
              folium.Marker(location=[row['x'], row['y']],
icon=folium.Icon(color="red")).add_to(my_map)
        # Potential hub marks
```

```
        if 3 < index <= 8:
             folium.Marker(location=[row['x'], row['y']],
icon=folium.Icon(color="green")).add_to(my_map)
        #Post office marks
        if 8 < index:
             folium.Marker(location=[row['x'], row['y']], tooltip='My
Marker').add_to(my_map)

# save the map to an HTML file
my_map.save('my_map.html')
my_map
```

B.2.2 Distance Matrix

```
from haversine import haversine, Unit
from scipy.spatial import distance_matrix
from math import radians, sin, cos, sqrt, atan2
sheet_id = '1R3Xl9bBWmjV9SsEYy3807JnFWn-wKYQe'
xls =
pd.ExcelFile(f"https://docs.google.com/spreadsheets/d/{sheet_id}/export?format=xls
x")
# read the data from the Excel file into a Pandas dataframe
df = pd.read_excel(xls, sheet_name = 'Coor', usecols=['x', 'y'])
coordinates = df[['x', 'y']]
def haversine(coor1, coor2)
         x1, y1 = coor1
         x2, y2 = coor2
         # Convert degrees to radians
         x1, y1, x2, y2 = map(radians, [x1, y1, x2, y2])
         # Haversine formula
         dx = x2 - x1
         dy = y2 - y1
         a = sin(dx / 2) ** 2 + cos(x1) * cos(x2) * sin(dy / 2) ** 2
         c = 2 * atan2(sqrt(a), sqrt(1 - a))
         km = 6371 * c  # Earth's radius in kilometers
         return km
     # calculate the distance matrix using a nested loop
     distances = np.zeros((len(coordinates), len(coordinates)))
     for i in range(len(coordinates))
         for j in range(len(coordinates))
```

```
            distances[i][j] = round(haversine(coordinates.iloc[i],
coordinates.iloc[j]), 2)
    # create a dataframe from the distances array
    dist_matrix = pd.DataFrame(distances, index=df.index, columns=df.index)
    print(dist_matrix)
    # Export to CSV file (which can be imported into Google Sheets)
    csv_file = 'distance_matrix.csv'
    dist_matrix.to_csv(csv_file, index=True)  # Set index=True to include
row/column labels
```

B.2.3 Scatter plot

```
from sklearn.cluster import KMeans
import pandas as pd
from matplotlib import pyplot as plt
sheet_id = '1R3Xl9bBWmjV9SsEYy3807JnFWn-wKYQe'
xls =
pd.ExcelFile(f"https://docs.google.com/spreadsheets/d/{sheet_id}/export?format=xls
x")
df = pd.read_excel(xls, sheet_name='Coor', usecols=['x', 'y'])
# Assign default colors and shapes
df['color'] = 'blue'
df['shape'] = 'o'
# Loop to set colors and shapes based on conditions
for i in range(len(df)):
    if 1 <= i <= 3:
        df.loc[i, ['color', 'shape']] = 'red', 'o'
    elif 3 < i < 9:
        df.loc[i, ['color', 'shape']] = 'green', 'o'
# Scatter plot with colors and shapes
plt.scatter(df['x'], df['y'], c=df['color'])
plt.xlabel('x')
plt.ylabel('y')
plt.title('Scatter Plot for Full dataset locations')
plt.show()
```

B.2.4 ALNDS

```
import copy
import math
import random
```

```
import numpy as np
import pandas as pd

class Parameters():
     # Filename and sheet data
     file_path = "Dataset 1.xlsx"
     hub_data = ["Hub", "A:F", 2, 10]
     distance_data = ["Distance", "B:S", 2, 19]
     quantity_data = ["Quantity", "B:K", 2, 11]
     Q_vehicle = 12000
     always_open_hubs = [1, 2, 3]

    GAMMA_1, GAMMA_2, GAMMA_3 = 1, 1, 100

class Utils():
    @staticmethod
    def read_data(file_path, sheet_name, usecols, start_row, end_row):
        data = pd.read_excel(file_path, sheet_name=sheet_name, usecols=usecols,
                             skiprows=range(0, start_row-1), nrows=end_row-
start_row)
        return data
    @staticmethod
    def calculate_O(quantity_matrix, i):
        O = 0
        for j, v in enumerate(quantity_matrix[i]):
            if i != j: O += v
        return O
    @staticmethod
    def calculate_D(quantity_matrix, i):
        rows = len(quantity_matrix)
        cols = len(quantity_matrix[0])
        D = 0
        for r in range(rows):
            for c in range(cols):
                if i == c:
                    D += quantity_matrix[r][c]
        return D

class Hub():
    def __init__(self, _id, capacity, fixed_cost, Q_vehicle):
        self.id = _id
        self.capacity = capacity
        self.fixed_cost = fixed_cost
        self.Q_vehicle = Q_vehicle

        if _id in params.always_open_hubs: self.status = 'open'
        else: self.status = 'close'
        self.routes = []

    def total_load(self):
        total_load = 0
```

```
        if not self.routes: return total_load
        if isinstance(self.routes[0], list):
            for route in self.routes:
                total_load += sum(node.demand for node in route)
        else:
            total_load += sum(node.demand for node in self.routes)
        return total_load

    def info(self):
        print("[INFO] HUB | ID = {}".format(self.id))
        for route in self.routes:
            _route_ids = [node.id for node in route]
            if _route_ids:
                print("\t{}".format(_route_ids))
                print("\t\tRoute load: {}".format(sum(node.demand for node in
route)))

class NonHub():
    def __init__(self, _id, O, D):
        self.id = _id
        self.status = "unassigned"
        self.demand = D
        self.D = D
        self.O = O
        self.O_add_D = O + D

class Problem():
    def __init__(self, file_path: str) -> None:
        self.file_path = file_path
        self.hub_df = Utils.read_data(
            file_path=self.file_path,
            sheet_name=params.hub_data[0],
            usecols=params.hub_data[1],
            start_row=params.hub_data[2],
            end_row=params.hub_data[3]
        )

        Q_vehicle = params.Q_vehicle
        hubs = []
        for idx, row in self.hub_df.iterrows():
            hubs.append(Hub(
                _id=idx+1, capacity=row['Capacity (kg)'], fixed_cost=row['Fixed
cost'], Q_vehicle=Q_vehicle
            ))
        self.hubs = hubs
        self.number_of_hub = len(hubs)

        self.distance_df = Utils.read_data(
            file_path=self.file_path,
            sheet_name=params.distance_data[0],
            usecols=params.distance_data[1],
```

```
            start_row=params.distance_data[2],
            end_row=params.distance_data[3]
        )
        self.distance_matrix = self.distance_df.values.tolist()

        self.quantity_df = Utils.read_data(
            file_path=self.file_path,
            sheet_name=params.quantity_data[0],
            usecols=params.quantity_data[1],
            start_row=params.quantity_data[2],
            end_row=params.quantity_data[3]
        )
        self.quantity_matrix = self.quantity_df.values.tolist()


        self.number_of_non_hub = len(self.quantity_matrix)
        self.non_hubs = []

        for i in range(self.number_of_non_hub):
            self.non_hubs.append(NonHub(
                _id=i+self.number_of_hub+1,
O=Utils.calculate_O(self.quantity_matrix, i),
D=Utils.calculate_D(self.quantity_matrix, i)
            ))

class TwoStageHeuristic():
    def __init__(self, problem: Problem):
        self.problem = problem
        self.hubs = problem.hubs.copy()
        self.non_hubs = problem.non_hubs.copy()
        self.opened_hubs = [hub for hub in self.hubs if hub.status=='open']
        _ids = [j for j, hub in enumerate(self.hubs) if hub.status=='close']
        random.seed(SEED)
        _selected_hub = random.choice(_ids)
        self.hubs[_selected_hub].status = 'open'
        params.always_open_hubs.append(self.hubs[_selected_hub].id)

    def select_non_hub_node(self):
        selected_non_hubs = [non_hub for non_hub in self.non_hubs
            if non_hub.status == "unassigned"]
        selected_non_hubs = sorted(selected_non_hubs,
            key= lambda x: x.O_add_D, reverse=True)
        return selected_non_hubs[0]

    def assign_to_nearest_hub(self, node):
        node_id = node.id
        self.opened_hubs = [hub for hub in self.hubs if hub.status=='open']

        hubs_and_dist = []
        for hub in self.opened_hubs:
            hub_id = hub.id
```

```
            distance = self.problem.distance_matrix[node_id-1][hub_id-1]
            hubs_and_dist.append((hub, distance))

        hubs_and_dist = sorted(hubs_and_dist, key= lambda x: x[1])
        for selected_hub, distance in hubs_and_dist:
            if selected_hub.total_load() + node.demand <= selected_hub.capacity:
                self.hubs[selected_hub.id-1].routes.append(node)
                self.non_hubs[node_id-self.problem.number_of_hub-1].status =
'assigned'
                return True
        return False

    def set_nearest_hub_open(self, node):
        node_id = node.id
        min_distance = float('inf')
        selected_hub = None
        close_hubs = [hub for hub in self.hubs if hub.status == 'close']
        if not len(close_hubs):
            return None
        for hub in close_hubs:
            hub_id = hub.id
            distance = self.problem.distance_matrix[node_id-1][hub_id-1]
            if distance < min_distance:
                min_distance, selected_hub = distance, hub
        self.hubs[selected_hub.id-1].status = 'open'
        self.hubs[selected_hub.id-1].routes.append(node)
        self.non_hubs[node_id-self.problem.number_of_hub-1].status = 'assigned'

    def find_nearest_node(self, node, linked_nodes):
        i = node.id
        node_and_dist = []
        for non_hub in linked_nodes:
            j = non_hub.id
            if i!=j:
                dist = self.problem.distance_matrix[i-1][j-1]
                node_and_dist.append((non_hub, dist))
        node_and_dist = sorted(node_and_dist, key=lambda x: x[1])
        return node_and_dist[0][0]

    def init(self):
        unassigned_nodes_remained = True
        while unassigned_nodes_remained:
            # Chọn nonhubs chưa được gán có O+D lớn nhất
            selected_non_hub = self.select_non_hub_node()
            # Add cho hubs gần nhất có đủ sức chứa
            if not self.assign_to_nearest_hub(node=selected_non_hub):
                # Nếu không thỏa điều kiện thì mở hubs gần nhất và add vào
                self.set_nearest_hub_open(node=selected_non_hub)
            # Kiểm tra lại hết tất cả các nút đã được add chưa
            unassigned_nodes_remained = len([node for node in self.non_hubs if
node.status=='unassigned'])
```

```
        # Tạo routes
        for hub in self.hubs:
            # Khởi tạo routes rỗng
            routes = [[]]
            if hub.status == 'open':
                linked_nodes = hub.routes.copy()
                while len(linked_nodes):
                    current_node_v = hub
                    current_pickup_capacity = current_node_v.Q_vehicle
                    current_delivery_capacity = current_node_v.Q_vehicle
                    while len(linked_nodes):
                        nearest_node = self.find_nearest_node(current_node_v,
linked_nodes)
                        current_node_v = nearest_node
                        current_pickup_capacity = current_pickup_capacity -
current_node_v.O
                        current_delivery_capacity = min(
                            current_pickup_capacity, current_delivery_capacity -
current_node_v.D
                        )
                        if current_delivery_capacity > 0 and
current_pickup_capacity > 0:
                            routes[-1].append(current_node_v)
                            # Update
                            linked_nodes = [
                                node for node in linked_nodes if node.id !=
current_node_v.id
                            ]
                        else:
                            routes.append([])
                            break
            # Update routes cho hubs
            hub.routes = routes
        for i, hub in enumerate(self.hubs):
            length_route = hub.capacity // hub.Q_vehicle
            if len(hub.routes) < length_route:
                for _ in range(length_route -len(hub.routes)):
                    self.hubs[i].routes.append([])
        return self.hubs

class SubMethod():
    name = None
    weight = 0
    time = 0
    score = 0

class SubProblem():
    name = None
    weight = 0
    time = 0
    score = 0
```

```
    destropy_methods = None

class Method():
    @staticmethod
    def worst_usage_hub_removal(solution):
        # Loại bỏ hubs có tỷ lệ sử dụng công suất ít nhất
        opened_hubs = [hub for hub in solution if hub.status == 'open' and hub.id
not in params.always_open_hubs]
        new_solution = [copy.deepcopy(hub) for hub in solution]
        if len(opened_hubs) == 0:
            return solution, []

        hub_and_efficiency = [
            (index, hub, hub.capacity - hub.total_load())
            for index, hub in enumerate(opened_hubs)
        ]
        hub_and_efficiency = sorted(hub_and_efficiency, key=lambda x: x[-1],
reverse=True)

        selected_hub_id = hub_and_efficiency[0][1].id
        selected_hub = new_solution[selected_hub_id-1]
        selected_routes = selected_hub.routes
        selected_nonhub_nodes = []
        for route in selected_routes:
            selected_nonhub_nodes.extend(route)

        selected_hub.status = 'close'
        selected_hub.routes = [[]]

        new_solution[selected_hub_id-1] = selected_hub
        return new_solution, selected_nonhub_nodes

    @staticmethod
    def random_hub_opening(solution):
        # Chọn mở một hubs đang đóng và random xóa một số node
        new_solution = [copy.deepcopy(hub) for hub in solution]
        closed_hub = [hub for hub in new_solution if hub.status == 'close']

        selected_nonhub_nodes = []
        for i, hub in enumerate(new_solution):
            if hub.status == "open":
                try:
                    random.seed(SEED)
                    j = random.choice([_ for _ in range(len(hub.routes))])
                    random.seed(SEED)
                    k = random.choice([_ for _ in range(len(hub.routes[j]))])
                except:
                    continue
                selected_nonhub_nodes.append(new_solution[i].routes[j][k])

        if len(closed_hub):
```

```
            random.seed(SEED)
            selected_hub_id = random.choice(closed_hub).id
            new_solution[selected_hub_id-1].status = "open"

        return new_solution, selected_nonhub_nodes

    @staticmethod
    def random_allocation_change(solution):
        # Random xóa một số node
        new_solution = [copy.deepcopy(hub) for hub in solution]
        selected_nonhub_nodes = []
        for i, hub in enumerate(new_solution):
            if hub.status == "open":
                try:
                    random.seed(SEED)
                    j = random.choice([_ for _ in range(len(hub.routes))])
                    random.seed(SEED)
                    k = random.choice([_ for _ in range(len(hub.routes[j]))])
                except:
                    continue
                selected_nonhub_nodes.append(new_solution[i].routes[j][k])
                new_solution[i].routes[j].pop(k)
        return new_solution, selected_nonhub_nodes

    @staticmethod
    def worst_allocation_removal(solution, distance_matrix):
        # Phá bỏ subnode dựa vào khoảng cách đến hub
        new_solution = [copy.deepcopy(hub) for hub in solution]
        selected_nonhub_nodes = []

        for i, hub in enumerate(new_solution):
            if hub.status == 'open':
                node_and_dist = []
                for j, route in enumerate(hub.routes):
                    for k, node in enumerate(route):
                        distance = distance_matrix[hub.id-1][node.id-1]
                        node_and_dist.append([(j, k), distance])

                node_and_dist = sorted(node_and_dist, key=lambda x: x[-1],
reverse=True)
                if not len(node_and_dist):
                    continue

                selected_node_info = node_and_dist[0][0]
                j, k = selected_node_info
                selected_nonhub_nodes.append(new_solution[i].routes[j][k])
                new_solution[i].routes[j].pop(k)

        return new_solution, selected_nonhub_nodes

    @staticmethod
```

```
    def worst_cost_removal(solution, distance_matrix):
        # Phá bỏ subnode dựa vào tổng cost khoảng cách của route
        new_solution = [copy.deepcopy(hub) for hub in solution]
        selected_nonhub_nodes = []

        for i, hub in enumerate(new_solution):
            if hub.status == 'open':
                node_and_dist = []
                for j, route in enumerate(hub.routes):
                    for k, node in enumerate(route):
                        distance_cost = 0
                        route_ids = [n.id for n in route if n.id != node.id]
                        route_id_merge = [hub.id] + route_ids + [hub.id]
                        for z in range(len(route_id_merge[:-1])):
                            a = route_id_merge[z]
                            b = route_id_merge[z+1]
                            distance_cost += distance_matrix[a-1][b-1]
                        node_and_dist.append([(j, k), distance_cost])

                node_and_dist = sorted(node_and_dist, key=lambda x: x[-1],
reverse=True)
                if not len(node_and_dist):
                    continue

                selected_node_info = node_and_dist[0][0]
                j, k = selected_node_info
                selected_nonhub_nodes.append(new_solution[i].routes[j][k])
                new_solution[i].routes[j].pop(k)

        return new_solution, selected_nonhub_nodes

    @staticmethod
    def shaw_removal(solution, distance_matrix, quantity_matrix):
        # GAMMA_1, GAMMA_2, GAMMA_3 = 1, 1, 100
        new_solution = [copy.deepcopy(hub) for hub in solution]
        route_info = []
        for i, hub in enumerate(new_solution):
            if hub.status == "open":
                for j, route in enumerate(hub.routes):
                    for k, node in enumerate(route):
                        route_info.append((i, j, k))
        if not route_info:
            return new_solution, []

        relatedness = []
        for node_a in route_info:
            for node_b in route_info:
                i1, j1, k1 = node_a
                i2, j2, k2 = node_b
                node_a_id = new_solution[i1].routes[j1][k1].id
                node_b_id = new_solution[i2].routes[j2][k2].id
```

```
                lij = -1 if (i1==i2) and (j1==j2) else 1
                cij = distance_matrix[node_a_id-1][node_b_id-1]
                dij = quantity_matrix[node_a_id-len(quantity_matrix)-1][node_b_id-
len(quantity_matrix)]

                rij = params.GAMMA_1*dij + params.GAMMA_2*cij + params.GAMMA_3*lij

                relatedness.append(
                    [(i1, j1, k1), (i2, j2, k2), rij]
                )

        relatedness = sorted(relatedness, key=lambda x: x[-1], reverse=True)
        selected = relatedness[0]
        i, j, k = selected[0]
        selected_nonhub_nodes = [new_solution[i].routes[j][k]]
        new_solution[i].routes[j].pop(k)
        return new_solution, selected_nonhub_nodes

    @staticmethod
    def random_removal(solution):
        current_solution = [copy.deepcopy(hub) for hub in solution]
        nonhub_nodes = []
        for i, hub in enumerate(current_solution):
            for j, route in enumerate(hub.routes):
                for k, node in enumerate(route):
                    nonhub_nodes.append((i,j,k))

        random.seed(SEED)
        q = random.choice([i+1 for i in range(len(nonhub_nodes)//2)])
        random.seed(SEED)
        selected_nodes = random.sample(nonhub_nodes, q)

        selected_nonhubs = []
        for node in selected_nodes:
            i, j, k = node
            try:
                selected_nonhubs.append(current_solution[i].routes[j][k])
                current_solution[i].routes[j].pop(k)
            except:
                continue

        return current_solution, selected_nonhubs

    @staticmethod
    def greedy_insertion(hubs, selected_nonhub_nodes, problem):
        current_solution = [copy.deepcopy(hub) for hub in hubs]
        for non_hub in selected_nonhub_nodes:
            case_info = []
            for i, hub in enumerate(current_solution):
                if hub.status == 'open':
```

```
                    for j, route in enumerate(hub.routes):
                        for k, node in enumerate(route):
                            case_info.append([i, j, k])
            index_and_cost = []
            for case in case_info:
                i, j, k = case
                new_solution = [copy.deepcopy(hub) for hub in current_solution]
                new_solution[i].routes[j].insert(k, non_hub)
                cost = CostFunction.caculate_cost(new_solution, problem)
                index_and_cost.append([(i,j,k), cost])

            index_and_cost = sorted(index_and_cost, key= lambda x: x[-1])
            _solution_id = index_and_cost[0]
            i, j, k = _solution_id[0]
            current_solution[i].routes[j].insert(k, non_hub)
        return current_solution

    @staticmethod
    def regret_insertion(hubs, nonhubs, problem):
        current_solution = [copy.deepcopy(hub) for hub in hubs]
        selected_nonhub_nodes = [copy.deepcopy(nonhub) for nonhub in nonhubs]
        while len(selected_nonhub_nodes):
            regreted_cost = []
            for non_hub_id, non_hub in enumerate(selected_nonhub_nodes):
                case_info = []
                for i, hub in enumerate(current_solution):
                    if hub.status == 'open':
                        for j, route in enumerate(hub.routes):
                            for k, node in enumerate(route):
                                case_info.append([i, j, k])
                index_and_cost = []
                for case in case_info:
                    i, j, k = case
                    new_solution = [copy.deepcopy(hub) for hub in
current_solution]
                    new_solution[i].routes[j].insert(k, non_hub)
                    cost = CostFunction.caculate_cost(new_solution, problem)
                    index_and_cost.append([(i,j,k), cost])

                index_and_cost = sorted(index_and_cost, key= lambda x: x[-1])
                regreted = index_and_cost[1][1] - index_and_cost[0][1]
                regreted_cost.append(
                    [non_hub_id, index_and_cost[0][0], regreted]
                )

            regreted_cost = sorted(regreted_cost, key=lambda x: x[-1])
            _selected_nonhub_id, _solution_id, _ = regreted_cost[0]
            i, j, k = _solution_id
            current_solution[i].routes[j].insert(k,
selected_nonhub_nodes[_selected_nonhub_id])
            selected_nonhub_nodes.pop(_selected_nonhub_id)
```

```
        return current_solution

    @staticmethod
    def duplication_operator(solution):
        current_solution = [copy.deepcopy(hub) for hub in solution]
        nonhub_nodes = []
        for i, hub in enumerate(current_solution):
            for j, route in enumerate(hub.routes):
                for k, node in enumerate(route):
                    nonhub_nodes.append((i,j,k))

        random.seed(SEED)
        node = random.choice(nonhub_nodes)
        i, j, k = node
        while True:
            i2, j2, k2 = random.choice(nonhub_nodes)
            if sum([i, j, k]) != sum([i2, j2, k2]):
                break
        current_solution[i2].routes[j2][k2].demand =
current_solution[i2].routes[j2][k2].demand / 2
        current_solution[i].routes[j].insert(k,
current_solution[i2].routes[j2][k2])

        return current_solution, []

    @staticmethod
    def deduplication_operator(solution):
        current_solution = [copy.deepcopy(hub) for hub in solution]
        nonhub_nodes = []
        for i, hub in enumerate(current_solution):
            for j, route in enumerate(hub.routes):
                for k, node in enumerate(route):
                    nonhub_nodes.append((i,j,k))

        selected_nonhubs = []
        p = 1
        step = 1
        while p > 0.3:
            random.seed(SEED)
            node = random.choice(nonhub_nodes)
            i, j, k = node
            current_solution[i].routes[j][k].demand =
current_solution[i].routes[j][k].demand/2
            selected_nonhubs.append(current_solution[i].routes[j][k])
            random.seed(SEED+step)
            step += 1
            p = random.random()

        return current_solution, selected_nonhubs
```

```
    @staticmethod
    def random_hub_removal(solution):
        current_solution = [copy.deepcopy(hub) for hub in solution]
        random.seed(SEED)
        selected_hub_id = [i for i in range(len(current_solution))]
        selected_hub_id = [_id for _id in selected_hub_id if _id+1 not in
params.always_open_hubs]
        selected_hub_id = random.choice(selected_hub_id)

        selected_node_ids = []
        for route in  current_solution[selected_hub_id].routes:
            selected_node_ids.extend(route)

        current_solution[selected_hub_id].routes = [[]]
        current_solution[selected_hub_id].status = 'close'

        return current_solution, selected_node_ids

    @staticmethod
    def random_route_removal(solution):
        current_solution = [copy.deepcopy(hub) for hub in solution]
        route_info = []
        for i, hub in enumerate(current_solution):
            if i+1 in params.always_open_hubs:
                continue
            for j, route in enumerate(hub.routes):
                route_info.append((i, j))

        random.seed(SEED)
        route = random.choice(route_info)
        i, j = route
        selected_nodes = current_solution[i].routes[j]
        current_solution[i].routes.pop(j)
        if len(current_solution[i].routes) == 0:
            current_solution[i].routes = [[]]
        return current_solution, selected_nodes

class CostFunction():
    @staticmethod
    def caculate_cost(solution, problem, final=False):
        total = 0
        for hub in solution:
            total += CostFunction.caculate_cost_hub(hub, problem, final)
        return total

    @staticmethod
    def caculate_cost_hub(hub, problem, final=False):
        ALPHA, MEW, PHI = 0.004, 10000, 100
        distance_matrix = problem.distance_matrix
        quantity_matrix = problem.quantity_matrix
        # Fixed cost
```

```
        if final:
            fixed_cost = 0
            for route in hub.routes:
                if len(route):
                    fixed_cost += hub.fixed_cost
                    break
        else:
            fixed_cost = hub.fixed_cost
        # Distance_cost
        node_visited = []
        distance_cost = 0
        for route in hub.routes:
            route_ids = [node.id for node in route]
            route_merge_ids = [hub.id] + route_ids + [hub.id]
            for k in range(len(route_merge_ids[:-1])):
                i = route_merge_ids[k]
                j = route_merge_ids[k+1]
                if j in node_visited:
                    continue
                else:
                    node_visited.append(j)
                distance_cost += distance_matrix[i-1][j-1]
        distance_cost = ALPHA*distance_cost
        # total load
        total_demand_load = 0
        for route in hub.routes:
            for node in route:
                total_demand_load += node.demand
            demand_remain = hub.capacity - total_demand_load
        penalty_cost = 0
        if demand_remain < 0:
            penalty_cost += MEW*abs(demand_remain)

        node_visited = []
        for route in hub.routes:
            total_quantity_load = 0
            route_id = [node.id for node in route]
            for k in range(len(route_id[:-1])):
                i = route_id[k]
                j = route_id[k+1]
                if j in node_visited:
                    continue
                else:
                    node_visited.append(j)
                total_quantity_load += quantity_matrix[i-len(quantity_matrix)-
1][j-len(quantity_matrix)-1]
            quantity_remain = hub.Q_vehicle - total_quantity_load
            if quantity_remain < 0:
                penalty_cost += PHI*abs(quantity_remain)

        total_cost = fixed_cost + distance_cost + penalty_cost
```

```
        return total_cost

class ALNDS:
    def __init__(self, initial_solution, max_iter=100, problem=None):
        self.current_solution = [copy.deepcopy(hub) for hub in initial_solution]
        self.best_solution = [copy.deepcopy(hub) for hub in initial_solution]
        self.max_iter = max_iter
        self.problem = problem
        self.T = 0.2

        # Khởi tạo biến lưu trữ các tham số của subproblems
        sub_problem_names = ["hub_location", "nonhub_allocation",
"vehicle_routing"]
        sub_problems = dict()
        for problem in sub_problem_names:
            sub_problems[problem] = SubProblem()
            sub_problems[problem].name = problem
            if problem == "hub_location":
                destropy_method_names = ["WUHR", "RHO", "RHR"]
            if problem == "nonhub_allocation":
                destropy_method_names = ["RAC", "WAR", "DO", "DDO"]
            if problem == "vehicle_routing":
                destropy_method_names = ["WCR", "SR", "RR", "RRR"]

            # Khởi tạo biến lưu trữ các tham số cho các problem tương ứng với từng
subproblem
            destropy_methods = dict()
            for method in destropy_method_names:
                destropy_methods[method] = SubMethod()
                destropy_methods[method].name = method
            sub_problems[problem].destropy_methods = destropy_methods

        self.sub_problems = sub_problems

        # Khởi tạo biến lưu trữ các tham số cho các thuật toán của repair
        repair_method_names = ["GI", "RI"]
        repair_methods = dict()
        for method in repair_method_names:
            repair_methods[method] = SubMethod()
            repair_methods[method].name = method
        self.repair_methods = repair_methods

    def select_with_probs(self, sublist):
        _weights = [sublist[name].weight for name in sublist]
        probs = []
        if all(w==0 for w in _weights):
            inverted_weights = [1/len(sublist)]*len(sublist)
        else:
            probs = [w/sum(_weights) for w in _weights]
            inverted_weights = [1/(w+0.00001) for w in probs]
```

```
        random.seed(SEED)

        keys = list(sublist.keys())
        selected = random.choices(keys, inverted_weights)[0]
        return selected

    def select_subproblem(self) -> str:
        return self.select_with_probs(sublist=self.sub_problems)
    def select_destroymethod(self, subproblem) -> str:
        return
self.select_with_probs(sublist=self.sub_problems[subproblem].destropy_methods)
    def select_repairmethod(self) -> str:
        return self.select_with_probs(sublist=self.repair_methods)

    def destroy(self, destroymethod):
        try:
            if destroymethod == "RHR":
                result = Method.random_hub_removal(solution=self.current_solution)
            if destroymethod == "WUHR":
                result =
Method.worst_usage_hub_removal(solution=self.current_solution)
            if destroymethod == "RHO":
                result = Method.random_hub_opening(solution=self.current_solution)
            if destroymethod == "RAC":
                result =
Method.random_allocation_change(solution=self.current_solution)
            if destroymethod == "WAR":
                result =
Method.worst_allocation_removal(solution=self.current_solution,
distance_matrix=self.problem.distance_matrix)
            if destroymethod == "DO":
                result =
Method.duplication_operator(solution=self.current_solution)
            if destroymethod == "DDO":
                result =
Method.deduplication_operator(solution=self.current_solution)
            if destroymethod == "WCR":
                result = Method.worst_cost_removal(solution=self.current_solution,
distance_matrix=self.problem.distance_matrix)
            if destroymethod == "SR":
                result = Method.shaw_removal(solution=self.current_solution,
distance_matrix=self.problem.distance_matrix,
quantity_matrix=self.problem.quantity_matrix)
            if destroymethod == "RR":
                result = Method.random_removal(solution=self.current_solution)
            if destroymethod == "RRR":
                result =
Method.random_route_removal(solution=self.current_solution)
        except Exception as err:
            return None
        return result
```

```
    def repair(self, repairmethod, hubs, selected_nonhub_nodes):
        if repairmethod == "GI":
            return Method.greedy_insertion(hubs, selected_nonhub_nodes,
self.problem)
        if repairmethod == "RI":
            return Method.regret_insertion(hubs, selected_nonhub_nodes,
self.problem)

    def update_scores(self, subproblem, destroymethod, repairmethod):
        self.time_subproblem[subproblem] += 1
        self.time_repair[repairmethod] += 1
        self.time_destroy[subproblem][destroymethod] += 1

        THETA1, THETA2, THETA3 = 5, 2, 0.5
        random.seed(SEED)
        score = random.choice([THETA1, THETA2, THETA3, 0])
        self.score_subproblem[subproblem] += score
        self.score_repair[repairmethod] += score
        self.score_destroy[subproblem][destroymethod] += score

    def update_weights(self, subproblem, destroymethod, repairmethod):
        ETA = 0.9
        THETA1, THETA2, THETA3 = 5, 2, 0.5

        self.sub_problems[subproblem].time += 1
        self.sub_problems[subproblem].destropy_methods[destroymethod].time += 1
        self.repair_methods[repairmethod].time += 1
        random.seed(SEED)
        score = random.choice([THETA1, THETA2, THETA3, 0])
        self.sub_problems[subproblem].score += score
        self.sub_problems[subproblem].destropy_methods[destroymethod].score +=
score
        self.repair_methods[repairmethod].score += score

        self.sub_problems[subproblem].weight = \
            (1 - ETA) * self.sub_problems[subproblem].weight \
                + ETA * self.sub_problems[subproblem].score /
self.sub_problems[subproblem].time

        self.sub_problems[subproblem].destropy_methods[destroymethod].weight = \
            (1 - ETA) *
self.sub_problems[subproblem].destropy_methods[destroymethod].weight \
                + ETA *
self.sub_problems[subproblem].destropy_methods[destroymethod].score /
self.sub_problems[subproblem].destropy_methods[destroymethod].time

        self.repair_methods[repairmethod].weight = \
            (1 - ETA) * self.repair_methods[repairmethod].weight \
                + ETA * self.repair_methods[repairmethod].score /
self.repair_methods[repairmethod].time
```

```
    def SA_mechanism(self, new_solution):
        THETA = 0.9995
        self.T = THETA*self.T

        cost_current = CostFunction.caculate_cost(self.current_solution,
problem=self.problem)
        cost_new = CostFunction.caculate_cost(new_solution, problem=self.problem)
        p = math.e**(-(cost_new-cost_current)/self.T)
        random.seed(SEED)
        q = random.random()
        return q < p

    def better_than_best(self, new_solution):
        return CostFunction.caculate_cost(new_solution, problem=self.problem) \
            <= CostFunction.caculate_cost(self.best_solution,
problem=self.problem)

    def better_than_current(self, new_solution):
        return CostFunction.caculate_cost(new_solution, problem=self.problem) \
            <= CostFunction.caculate_cost(self.current_solution,
problem=self.problem)

    def solve(self):
        f_logger = open("log.txt", "w")
        global SEED
        iter_count = 0
        new_solution= self.current_solution.copy()
        best_cost = 0

        # Vòng lặp thuật toán
        while iter_count < self.max_iter:
            # Chọn subproblem, destroy method and repair method
            subproblem = self.select_subproblem()
            destroymethod = self.select_destroymethod(subproblem=subproblem)
            repairmethod = self.select_repairmethod()
            print(f"Method: {subproblem} - {destroymethod} - {repairmethod}")

            destroied_result = self.destroy(destroymethod)
            if destroied_result:
                f_logger.write(f"Method: {subproblem} - {destroymethod} -
{repairmethod}\n")
                destroyed_hubs, selected_nonhub_nodes = destroied_result
                new_solution = self.repair(repairmethod=repairmethod,
hubs=destroyed_hubs, selected_nonhub_nodes=selected_nonhub_nodes)

                if self.better_than_best(new_solution):
                    self.best_solution = new_solution
                    best_cost = CostFunction.caculate_cost(new_solution,
problem=self.problem)
                    self.current_solution = new_solution
```

```
                else:
                    if self.better_than_current(new_solution):
                        self.current_solution = new_solution
                    else:
                        if self.SA_mechanism(new_solution):
                            self.current_solution = new_solution

                self.update_weights(subproblem, destroymethod, repairmethod)
            for i in range(len(self.current_solution)):
                if len(self.current_solution[i].routes[0]) == 0 and i+1 not in
params.always_open_hubs:
                    self.current_solution[i].status = "close"
            iter_count += 1
            SEED += 1
            print(f"[INFO] Step {iter_count}/{self.max_iter}", "Cost: ",
CostFunction.caculate_cost(self.current_solution, self.problem))
            f_logger.write(f"[INFO] Step {iter_count}/{self.max_iter} | Cost:
{CostFunction.caculate_cost(self.current_solution, self.problem)}\n")

        for i, hub in enumerate(self.best_solution):
            for j, route in enumerate(hub.routes):
                self.best_solution[i].routes[j] = list(set(route))

        return self.best_solution, CostFunction.caculate_cost(self.best_solution,
self.problem, True)

#Result display
def process():
    global params
    # Khởi tạo probelm
    problem = Problem(file_path=params.file_path)
    inital_solution = TwoStageHeuristic(problem=problem).init()
    print("[INFO] Initial Solution")

    alnds = ALNDS(initial_solution=inital_solution, max_iter=100, problem=problem)
    solution, cost = alnds.solve()

    # [hub.info() for hub in solution]
    print("[INFO] Final Solution - Cost: ", cost)

    # Save solution
    with open("solution.txt", "w") as f:
        for hub in solution:
            if sum(len(route) for route in hub.routes) > 0:
                hub.status = "open"
        for hub in solution:
            i = 0
            f.write(f"HUB | ID: {hub.id} | Status: {hub.status}\n")
            for _, route in enumerate(hub.routes):
                if len(route):
```

```
                    f.write(f"\tRoute {i+1}: {list(set([node.id for node in
route]))}\n")
                    f.write("\t\tRoute load: {}\n".format(sum(node.demand for node
in route)))
                    i += 1
            f.write("---------------------------------\n")
        f.write(f"Total Cost: {CostFunction.caculate_cost(solution, problem,
True)}")

if __name__=="__main__":
    SEED = 99
    params = Parameters()
    process()
```