TECHNICAL UNIVERSITY OF CRETE | SCHOOL OF PRODUCTION ENGINEERING AND MANAGEMENT

Cuckoo Search Algorithm for the Green Location Routing Problem

Αλγόριθμος Αναζήτησης του Κούκου για το Πράσινο Πρόβλημα Χωροθέτησης Εγκαταστάσεων και Δρομολόγησης Οχημάτων

Author:

Pappas Anastasios

Supervisor:

Dr. Ioannis Marinakis

ACKNOWLEDGEMENTS

I would like to express my deepest gratitude to Dr. Ioannis Marinakis, my supervisor, for his invaluable guidance, encouragement, and support throughout the completion of this thesis. I am profoundly grateful for the opportunity he provided me to complete my studies.

I'm also deeply thankful to my family, particularly my parents and brother for their unwavering faith and trust in me during this challenging journey. Their encouragement and support kept my spirits high and motivated me to persevere.

Finally, I wish to extend my heartfelt appreciation to my close friends and colleagues, both on and off campus, for their patience, companionship, and continued support throughout this endeavour.

ΕΥΧΑΡΙΣΤΙΕΣ

Θα ήθελα να εκφράσω την πιο βαθιά μου ευγνωμοσύνη στον Δρ. Ιωάννη Μαρινάκη, επιβλέποντα καθηγητή μου, για την ανεκτίμητη καθοδήγηση , ενθάρρυνση και υποστήριξή του καθ΄ όλη τη διάρκεια της εκπόνησης αυτής της διπλωματικής εργασίας. Είμαι βαθύτατα ευγνώμων για την ευκαιρία που μου έδωσε να ολοκληρώσω τις σπουδές μου.

Επίσης, εκφράζω την ειλικρινή μου ευγνωμοσύνη στην οικογένεια μου, ιδιαίτερα στους γονείς μου και στον αδελφό μου, για την αδιάκοπη πίστη και εμπιστοσύνη τους σε εμένα κατά την διάρκεια αυτής της δύσκολης πορείας. Η ενθάρρυνση και η υποστήριξή τους κράτησαν το ηθικό μου υψηλό και με ενέπνευσαν να συνεχίσω.

Τέλος, θα ήθελα να απευθύνω τις θερμές μου ευχαριστίες στους στενούς μου φίλους και συναδέλφους, τόσο εντός και εκτός πανεπιστημίου, για την υπομονή, τη συντροφικότητα και τη συνεχή τους υποστήριξη κατά τη διάρκεια αυτής της προσπάθειας.

ABSTRACT

This thesis explores the application of the Cuckoo Search algorithm to the Location Routing Problem, a complex optimization challenge in logistics. The Location Routing Problem involves determining the optimal placement of facilities, such as warehouses or distribution centers, and designing the most efficient routes for capacitated vehicles to serve customers. The goal is to minimize operational costs, including the fixed costs of facility locations and the variable costs.

The Cuckoo Search algorithm, inspired by the brood parasitism behavior of cuckoo birds, is a metaheuristic optimization technique known for efficiently exploring large and complex search spaces. In the context of the Location Routing Problem, the algorithm generates and iteratively improves candidate solutions by mimicking the natural process of laying eggs in the nests of other birds, allowing for a robust search for optimal solutions.

This research implements the Cuckoo Search algorithm across 18 different datasets, each representing a unique logistics scenario, to demonstrate its effectiveness in solving the Location Routing Problem. This thesis provides insights into how the Cuckoo Search algorithm can be applied to optimize logistics operations.

ΠΕΡΙΛΗΨΗ

Η παρούσα διπλωματική εργασία εξετάζει την εφαρμογή του Αλγορίθμου Αναζήτησης Κούκου (Cuckoo Search) στο Πρόβλημα Χωροθέτησης Εγκαταστάσεων και Δρομολόγησης Οχημάτων (Location Routing Problem), ένα σύνθετο πρόβλημα βελτιστοποίησης στη διοίκηση εφοδιαστικής αλυσίδας. Το Πρόβλημα Χωροθέτησης Εγκαταστάσεων και Δρομολόγησης Οχημάτων (Location Routing Problem) αφορά τον προσδιορισμό των βέλτιστων θέσεων εγκαταστάσεων, όπως αποθήκες ή κέντρα διανομής, και τον σχεδιασμό των πιο αποδοτικών δρομολογίων για οχήματα με περιορισμένη χωρητικότητα, προκειμένου να εξυπηρετηθούν οι πελάτες. Ο στόχος είναι η ελαχιστοποίηση του συνολικού λειτουργικού κόστους, το οποίο περιλαμβάνει τόσο τα πάγια κόστη των εγκαταστάσεων όσο και τα μεταβλητά κόστη.

Ο αλγόριθμος Αναζήτησης Κούκου, εμπνευσμένος από τη συμπεριφορά παρασιτισμού των αυγών των πουλιών κούκου, είναι μια μεθευρετική τεχνική βελτιστοποίησης, γνωστή για την ικανότητά της να εξερευνά μεγάλους και πολύπλοκους χώρους αναζήτησης. Στο πλαίσιο του Προβλήματος Χωροθέτησης Εγκαταστάσεων και Δρομολόγησης Οχημάτων,

ο αλγόριθμος Αναζήτησης Κούκου παράγει και βελτιώνει επαναληπτικά υποψήφιες λύσεις, μιμούμενος τη φυσική διαδικασία τοποθέτησης των αυγών στη φωλιά άλλων πουλιών, επιτρέποντας έτσι μια ισχυρή αναζήτηση για βέλτιστες λύσεις.

Η παρούσα έρευνα εφαρμόζει τον αλγόριθμο Αναζήτησης Κούκου σε 18 διαφορετικά σύνολα δεδομένων, καθένα από τα οποία αντιπροσωπεύει ένα μοναδικό σενάριο, προκειμένου να καταδείξει την αποτελεσματικότητά του στην επίλυση του Προβλήματος Χωροθέτησης Εγκαταστάσεων και Δρομολόγησης Οχημάτων. Η διπλωματική αυτή παρέχει πληροφορίες σχετικά με τον τρόπο με τον οποίο ο αλγόριθμος Αναζήτησης Κούκου μπορεί να εφαρμοστεί για τη βελτιστοποίηση των λειτουργιών της εφοδιαστικής αλυσίδας.

TABLE OF CONTENTS

CHAPTER 1: INTRODUCTION

1.1 LOGISTICS AND SUPPLY CHAIN MANAGEMENT

Logistics, derived from the ancient Greek word “λόγος” (logos) meaning reason or calculation, originally referred to the management of resources such as arms and supplies for military operations. Over time, logistics became a very important component of modern commerce, focusing on the efficient transportation of goods from one point to another, warehousing, inventory management, and order processing and fulfillment. **[1]**

Essentially, logistics adds “place utility” to goods by guaranteeing that products are delivered from their point of origin to their destination in the most cost-effective manner. This process includes the coordination of five key elements (Fig.1):

- **Transportation:** Aspects of transport management include managing the shipment of products across several modes of transport (such as road, rail, air, or sea), transport infrastructure, type of delivery, routing and scheduling.
- **Warehousing:** Storing goods at strategically chosen and optimized locations as to balance inventory size, meet demand, and optimize delivery times and routes.
- **Inventory Management:** Determining the right quantity of products and goods to stock at the right locations. Not to be confused with warehouse management; inventory management deals with the amount of stock of the product and goods whereas warehouse management is about the storage of this stock.
- **Packaging and Unitization:** The process ensuring that all goods are packaged, based on their type and value, for protection and efficient handling.
- **Information Management:** The utilization of advanced systems to track goods, process orders and deliveries, and coordinate activities across the supply chain.

In modern business, Logistics is important for achieving operational efficiency and strategic success in domestic and global markets. It guarantees that products are delivered to the right place at the right time while minimizing costs. Therefore, companies worldwide rely on effective logistics to gain a competitive edge and refine their operations to respond to market demands.

Furthermore, logistics is essential to realizing broader business objectives, such as sustainability and innovation. Green logistics practices, such as optimizing vehicle routes and using clean-energy vehicles to reduce carbon emissions, help companies meet environmental targets while also achieving their financial goals.

It is evident that logistics is no longer just a supporting function but a strategic and essential enabler of growth, innovation, and sustainability for a company. Its

combination with other supply chain management components ensures alignment with organizational goals and, therefore, success.

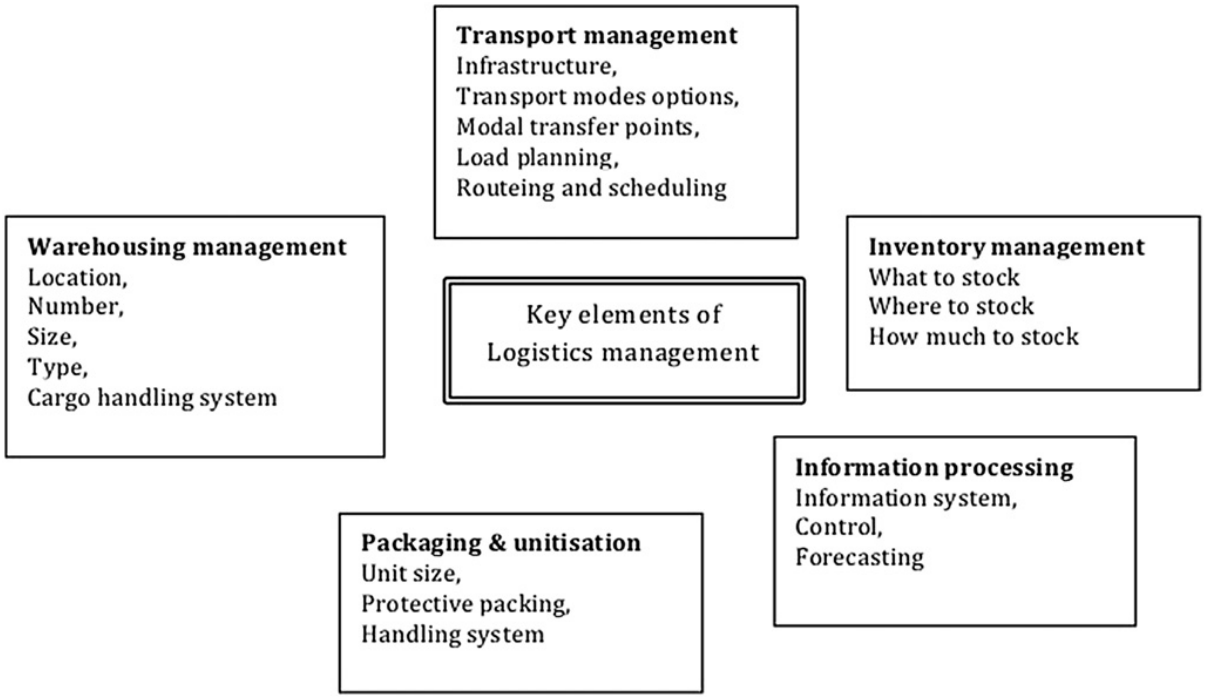

Fig.1: The key elements of logistics management

1.1.2 Supply Chain Management (SCM)

At the most fundamental level, Supply Chain Management (SCM) is management of the flow of goods, data, and finances related to a product or service, from the procurement of raw materials to the delivery of the product at its final destination. Supply Chain Management is divided into planning and execution. Planning involves processes such as the prediction of needed goods and the coordination of production and delivery. The execution focuses on the results of planning, by utilizing methods such as the control of inventory and production and managing transportation and delivery. This ensures customers are provided with products and services timely and with the desired quality, while simultaneously a high level of efficiency is maintained.

The key components of Supply Chain Management are:

- **Procurement:** A vital operation of Supply chain Management which involves the acquisition of required raw materials and goods from suppliers to ensure the smooth continuation of production processes.
- **Production:** Transforming raw materials and goods into finished products that can be forwarded to customers through manufacturing and assembly processes.
- **Transportation:** The modes of transport used to deliver materials from the supplier to production sites and from there to warehouses. Transportation adds up to a significant cost and, therefore, demands careful planning.
- **Warehousing:** Encompasses the safe and effective storage of raw materials, worked-on goods, and final products up until they are needed for production or distribution.
- **Distribution:** Delivery of the finished products from production or warehouses to the customer via effective transportation and logistics strategies.
- **Customer Relationship Management:** It is crucial for businesses to maintain excellent communication with customers to meet their needs and expectations effectively.

Logistics serves as the operational foundation for these Supply Chain Management operations by providing the infrastructure and strategies required to move goods efficiently and meet supply chain objectives.

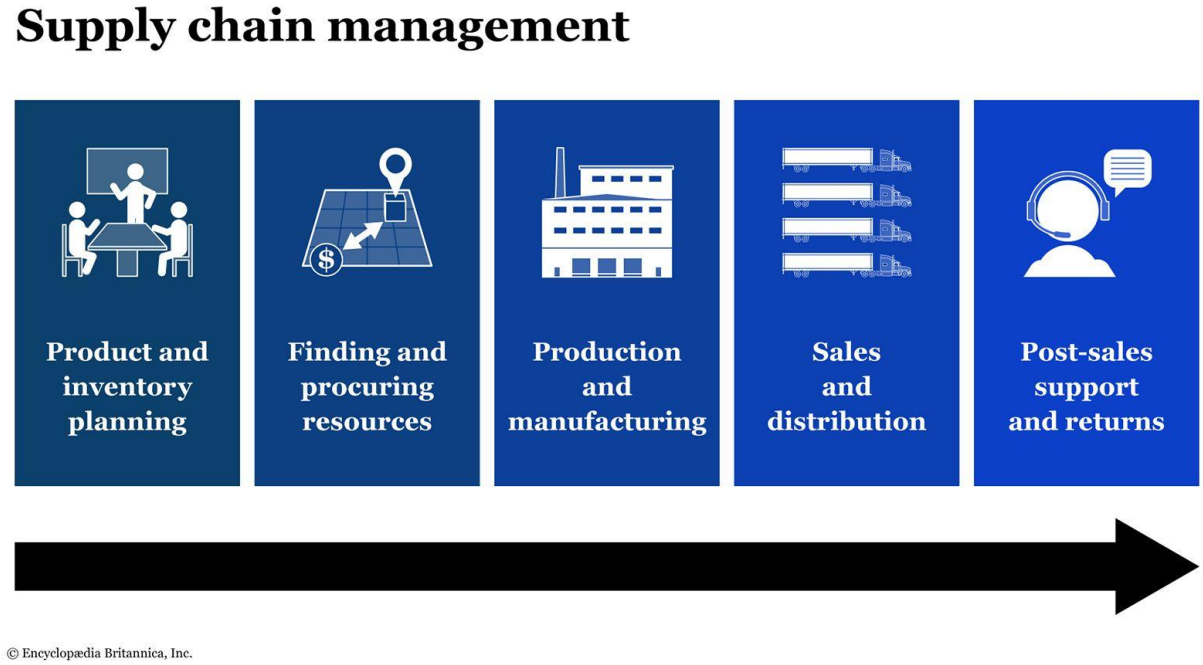

Fig.2: Five general phases of Supply Chain Management

1.2 Importance of Location Routing Problems (LRPs)

The Location Routing Problem (LRP) is a foundational challenge in logistics and supply chain management, integrating two interdependent decisions: where is the optimal location of arbitrary types of facilities (depots, warehouses, etc.) and which are the most efficient routes for vehicles to serve customers. Making these decisions separately from each other may lead to highly suboptimal planning results (Salhi & Rand, 1989). **[2]**

Facility Location and Vehicle Routing:

- **Facility Location:** The choice of locations for depots or warehouses is a crucial strategic decision with long-term effects on operational efficiency. Choosing a suboptimal location can lead to increased transportation costs and disruption across the supply chain.
- **Vehicle Routing:** Once the locations of facilities are established, a decision must be made about which vehicle routes should be built to deliver goods to customers. Efficient routing minimizes distribution costs, reduces fuel consumption, and guarantees timely delivery.

By addressing these decisions simultaneously, the LRP provides a more consolidated and cost-effective approach than solving them separately.

Furthermore, LRPs assist businesses with the following:

1. **Cost Optimization:** The LRP minimizes costs by optimizing facility and routing decisions in unison. This includes fixed costs, variable costs and distance attributes. The integration of these decisions prevents suboptimal solutions that arise when they are made independently
2. **Service-Level Improvements:** Solving the LRP guarantees that customers are served timely and efficiently, meeting any parameters set by the business (delivery time windows, customer demand, etc.) and maintaining customer satisfaction.
3. **Scalability:** Modern, complex supply chains involve a vast network of suppliers, manufacturers, distributors, and customers. LRPs help scale operations effectively by laying out a structured framework for making facility and routing decisions in large, multi-echelon networks.
4. **Sustainability:** By optimizing routes and, therefore, minimizing travel distance, LRPs reduce fuel consumption and greenhouse gas emissions, supporting sustainability. This is becoming increasingly important as companies prioritize environmentally friendly practices.

In conclusion, the growing complexity of global supply chains and the increasing use of customer-centric business models have made Location Routing Problems an essential tool for any business that aims to remain competitive and sustainable.

1.3 Objective of the Thesis

This thesis aims to construct an effective optimization approach to solving the Location Routing Problem, a complex optimization problem involving facility location and vehicle routing decisions. By simultaneously addressing these mutually dependent problems, this research aims to minimize operational costs while improving the efficiency and reliability of logistics operations. Specifically, the focus is on minimizing the total distance traveled by vehicles departing from opened depots to serve customers.

To achieve this, the thesis employs a two-step approach which will be expanded upon in subsequent chapters:

1. **Initial Solutions via Nearest Neighbor Algorithm:** The Nearest Neighbor Algorithm is a heuristic algorithm used to generate initial solutions for the LRP. By altering the combination of opened depots, 20 different solutions are generated and tested the 18 datasets.
2. **Optimization using the Cuckoo Search Algorithm:** The Cuckoo Search Algorithm (CSA) is a powerful metaheuristic algorithm inspired by the brood parasitism of some cuckoo bird species. It is applied 100 times to each solution to iteratively refine the routes and minimize total vehicle travel distances.

Unlike traditional comparative studies, this research does not evaluate the Cuckoo Search Algorithm's performance against other algorithms. Instead, it focuses on the ability of the CSA to improve the initial solutions generated by the Nearest Neighbor Algorithm. By analyzing the results, this study aims to highlight the effectiveness of the Cuckoo Search Algorithm as an optimization tool for solving Location Routing Problems. In addition, it explores how influential depot selection and routing decisions are in reducing the overall distance traveled. The findings of this research contribute to the broader understanding of metaheuristic algorithms and their applications in logistics and supply chain optimization.

1.4 Structure of the Thesis

This thesis is organized into seven chapters, each addressing an important aspect of the research.

Chapter 1 introduces the thesis topic, highlighting the importance of logistics and supply chain management, the significance of the Location Routing Problem (LRP), the objective of the research, and the structure of the thesis itself.

Chapter 2 presents the foundational routing problems, including the Traveling Salesman Problem (TSP) and the Vehicle Routing Problem (VRP), examining their mathematical formulations and solution methods. This chapter concludes by introducing the Location Routing Problem (LRP) and its connection to the field of logistics and routing problems.

Chapter 3 investigates the definition, significance, and challenges of Location Routing Problems. It includes a brief literature review, a detailed mathematical formulation, and a discussion on the complexities of solving these Problems.

Chapter 4 examines optimization algorithms, presenting heuristic, greedy, evolutionary, and metaheuristic methods. The Nearest Neighbor Algorithm, used in this thesis for generating initial solutions for the Location Routing Problem, is introduced in this chapter along with its strengths and limitations. The Cuckoo Search Algorithm, a metaheuristic method utilized to optimize these solutions, is also detailed.

Chapter 5 explains the research methodology, focusing on the Cuckoo Search Algorithm and its application to the Location Routing Problem. This chapter describes the algorithm's steps, its adaptation for this thesis, and the process of optimizing the initial solutions generated by the Nearest Neighbor Algorithm.

Chapter 6 presents and describes the research results, including the performance analysis of the Cuckoo Search Algorithm across the 18 datasets. The discussion incorporates insights into the effects of selecting the opened depots and routing decisions on minimizing the total distance, supported by tables and graphs.

Chapter 7 concludes the thesis by summarizing the study findings and highlighting the contribution of this research in logistics and supply chain management optimization.

CHAPTER 2: ROUTING PROBLEMS

Overview of Routing Problems

Routing problems are crucial challenges in logistics and supply chain management, focusing on the optimization of vehicle routing while minimizing costs such as distance traveled, fuel consumption, or time. Two of the most researched routing problems are the Traveling Salesman Problem (TSP) and the Vehicle Routing Problem (VRP), both of which form the basis for comprehending more complex challenges like the Location Routing Problem (LRP).

2.1 The Traveling Salesman Problem (TSP)

The Traveling Salesman Problem is one of the most well-known and extensively researched combinatorial optimization problems. Formulating the Traveling Salesman Problem is deceptively simple: Given a set of cities and the distances between them, the objective of the problem is to establish the shortest possible route that visits each city exactly once and returns to the starting point. Despite its simplicity, the Traveling Salesman Problem is computationally difficult and when the theory of NP-completeness developed, the TSP was one of the first problems to be proven NP-hard by **[3]** Karp in 1972. Meaning that the required time to solve the problem grows exponentially with the number of cities.

The Traveling Salesman Problem (TSP) was first mathematically formulated in the 1930s. Due to its theoretical complexity and wide-ranging applications, it became prominent in combinatorial optimization. The TSP has been significant in the development of exact algorithms, heuristics, and metaheuristics, impacting the study of many other optimization problems, such as the Vehicle Routing Problem (VRP) and the Location Routing Problem (LRP).

The TSP belongs to the class of NP-hard problems (Nondeterministic Polynomial), making it impractical to solve optimally for large instances using brute-force approaches. The running time for this approach lies within a polynomial factor of O(n!), the factorial of

the number of cities, so this solution becomes infeasible even for only 20 cities. This necessitates the use of advanced algorithms, including exact operations such as the branch-and-bound and branch-and-cut algorithms for small to moderate-sized problems and heuristic and metaheuristic approaches, like simulated annealing, genetic algorithms, and ant colony optimization, for larger instances. **[4]**

The Traveling Salesman Problem can be mathematically formulated as follows:

- N = {1, 2, ..., n} the set of cities
- c_{ij} represents the distance or cost of traveling from city i to city j.
- x_{ij} stands for a binary decision variable, where:

$$x = \begin{cases} 1, if\ the\ vehicle\ travels\ from\ city\ i\ to\ cityj \\ 0, otherwise, \forall i, \forall j, i \neq j \end{cases}$$

The objective is to minimize the total distance the vehicle has covered. This can be formulated mathematically like this:

$$min \sum_{i=1}^{n} \sum_{j=1}^{n} c_{ij} x_{ij}$$

Subject to the following constraints:

$$\sum_{j=1}^{n} x_{ij} = 2, i = 1, \dots, n$$

$$\sum_{i \in C} \sum_{j \in \bar{C}} x_{ij} \geq 1, \forall C\ \text{C}\{1, \dots., n\}, C \neq 0$$

$$x_{ij} \in \{0,1\}, \forall i, \forall j, i \neq j$$

2.1.2 Example of a Traveling Salesman Problem

A Traveling Salesman Problem with 5 cities (A, B, C, D, and E), where the distance between each pair of cities is given in the table below:

From/To	A	B	C	D	E
A	0	10	15	20	25
B	10	0	35	25	30
C	15	35	0	30	20
D	20	25	30	0	15
E	25	30	20	15	0

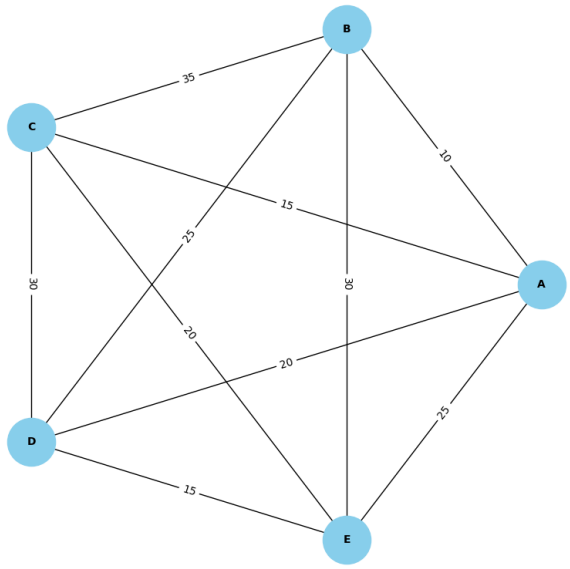

Fig.3: TSP Example Graph

The objective is to find the shortest possible route that visits all cities once and returns to the starting point. For example, a potential route might be:

$$A \rightarrow B \rightarrow D \rightarrow E \rightarrow C \rightarrow A$$

The total distance for this route is calculated as follows:

$$10(A \rightarrow B) + 25(B \rightarrow D) + 15(D \rightarrow E) + 20(E \rightarrow C) + 15(C \rightarrow A) = 85$$

Optimization methods can identify the shortest route and calculate its total distance, as in the example above.

2.1.3 Solution Methods for the Traveling Salesman Problem

Several solution methods have been developed over decades of research to address the Traveling Salesman Problem and its computational challenges. These methods can be broadly categorized into exact methods, heuristics, and metaheuristics.

- **Exact Methods**
  Exact Methods ensure the optimal solution but are computationally expensive, limiting their application to small and medium-sized optimization problems. The most notable exact methods are:
  - **Dynamic Programming:** A mathematical optimization method developed by Richard Bellman in the 1950s. Simplifies a complicated problem by breaking it down into simpler sub-problems, solving them recursively.
  - **Branch-and-Bound:** This algorithm systematically explores the solution space, pruning branches that cannot yield better results. **[5]**
  - **Branch-and-Cut:** Makes use of linear programming relaxations and iterative addition of constraints to optimize the feasible solution region. **[5]**

- **Heuristic Algorithms**
  Heuristics effectively provide near-optimal solutions, although they often sacrifice some accuracy for speed. A frequently used Heuristic algorithm is the Nearest Neighbor Algorithm, which constructs a tour by repeatedly visiting the nearest unvisited city. This Algorithm will be detailed later in this thesis.

- **Metaheuristic Algorithms**
  Metaheuristics are used to explore larger solution spaces and are appropriate for larger instances of the Traveling Salesman Problem. Some noteworthy metaheuristic algorithms are:
  - **Genetic Algorithms:** Using the principles of evolution, such as selection and crossover, they improve solutions iteratively.
  - **Ant Colony Optimization:** Inspired by the foraging behavior of ants, this method utilizes pheromone trails to guide the search for the best routes.

- **Cuckoo Search Algorithm:** Inspired by the brood parasitism behavior of cuckoo birds, the Cuckoo Search algorithm generates and iteratively improves candidate solutions by mimicking the natural process of laying eggs in the nests of other birds, allowing for a robust search for optimal solutions.

In this thesis, the Traveling Salesman Problem provides a foundational framework for tackling the Location Routing Problem (LRP). The LRP extends the Traveling Salesman Problem by including facility location decisions and multiple vehicle routing, introducing additional layers of complexity. The Nearest Neighbor Algorithm, frequently used to solve the Traveling Salesman Problem, is used in this thesis to generate initial solutions for the LRPs.

2.2 The Vehicle Routing Problem (VRP)

The Vehicle Routing Problem (VRP) is crucial to logistics and operations research, addressing the challenge of optimizing routes for a fleet of vehicles serving a set of geographically dispersed customers. The first record in the Vehicle Routing Problem literature is the paper by Dantzig, Fulkerson, and Johnson (1954) **[6]** which studied a relatively large-scale Traveling Salesman Problem and proposed a solution method. Following this study, Dantzig and Ramser (1959) **[7]** extended the TSP by including multiple vehicles and depot constraints in their research. This problem was generalized five years later to a linear optimization problem by Clarke and Wright (1964) **[8]**. This became known as the Vehicle Routing Problem.

The objective of the Vehicle Routing Problem is to minimize costs such as total distance traveled, time, or fuel consumption while satisfying constraints such as vehicle capacity and customer demands. Over the decades, its significance has grown exponentially due to its applicability in various industries, such as transportation, e-commerce, and urban logistics.

The Vehicle Routing Problem is comprised of several fundamental elements that define its structure and practical challenges:

- **Depots:** Central facilities from which vehicles depart and return to after completing their routes and serve the customers. The number and location of depots significantly influence routing efficiency.
- **Vehicle Fleet:** A set of vehicles used for deliveries. The Fleet may be homogeneous (identical vehicles) or heterogeneous (vehicles with different capacities, costs, or other attributes).

- **Customers:** Geographically distributed locations, each with specific demands for products or services.
- **Routes:** Paths assigned to vehicles to serve customers. The primary objective of the VRP is to optimize these routes.

In addition to these crucial components, the Vehicle Routing Problem involves constraints that demonstrate real-world problems:

- **Capacity Constraints:** Each vehicle has a maximum load capacity that cannot be exceeded.
- **Route Length Limits:** Vehicles may have limits on total distance traveled or time spent on a route.
- **Time Windows:** There are some VRP variants where there is a need of deliveries to occur within discrete timeframes.
- **Customer Requirements:** Each customer must be served exactly once, and their specific demands must be met.
- **Depot Constraints:** In multi-depot VRPs, each vehicle must begin and end its route at the same depot.

By addressing these constraints with its core components, the Vehicle Routing Problem presents a flexible framework for modeling and solving complex logistics and supply chain management problems.

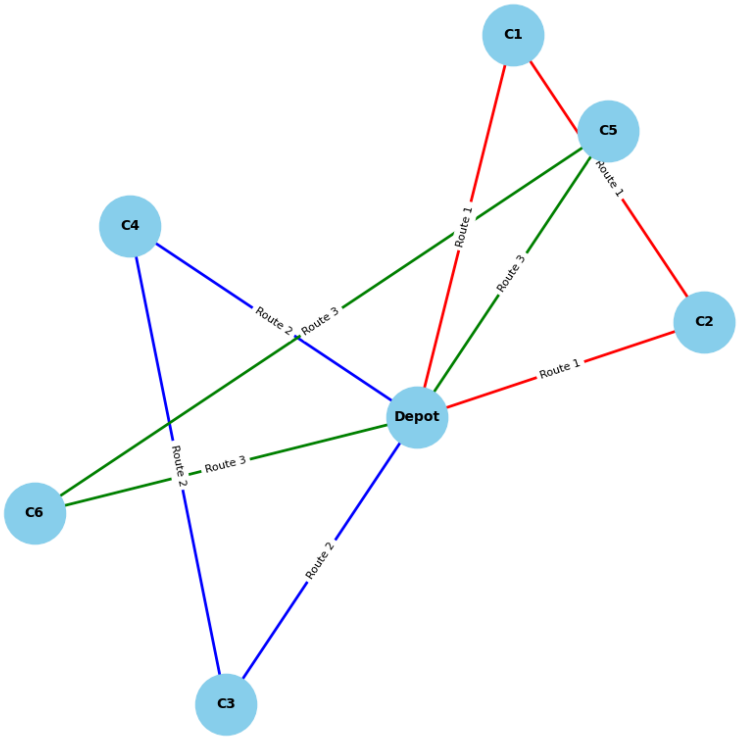

Fig.4: Illustration of a VRP

The Vehicle Routing Problem can be mathematically formulated as follows:

- V = {0, 1, 2, ..., n}: Set of nodes, where 0 represents the depot and {1, 2, ..., n} represent customer locations.
- K: Set of vehicles available at the depot.
- c_{ij}: Cost (distance or time) of traveling from node i to node j.
- Q: Capacity of each vehicle.
- q_i: Demand of customer i.
- x^k_{ij}: A binary variable indicating whether vehicle k travels from node i to node j.

$$x_{ij}^k = \begin{cases} 1, if\ vehicle\ k\ travels\ from\ node\ i\ to\ node\ j \\ 0, otherwise \end{cases}$$

The objective function/ the function to minimize the total cost:

$$min \sum_{k \in K} \sum_{i \in V} \sum_{j \in V, i \neq j} c_{ij} x_{ij}^k$$

Subject to the following constraints:

- Each customer is visited exactly once by a vehicle:

$$\sum_{k \in K} \sum_{j \in V, j \neq i} x_{ij}^k = 1, \forall i \in V\{0\}$$

- Each vehicle must leave and return to the depot:

$$\sum_{j \in V, j \neq 0} x_{0j}^k = \sum_{i \in V, i \neq 0} x_{i0}^k = 1, \forall k \in K$$

- The total customer demand on a route must not exceed the vehicle's capacity:

$$\sum_{i \in V} q_i \sum_{j \in V, j \neq i} x_{ij}^k \leq Q, \forall k \in K$$

- Binary Decision Variables:

$$x_{ij}^k \in \{0,1\}, \forall i, j \in V, k \in K$$

2.2.2 Variants of the Vehicle Routing Problem and Solution Methods

The basic Vehicle Routing Problem involves a single depot, a homogeneous fleet, and non-fluctuating customer demands. However, real-world logistics often involve more complex requirements, leading to several VRP variants. Based on these different constraints and requirements, different variants are designed to address specific problems. **[9]**

1. **Capacitated VRP (CVRP)**

The Capacitated Vehicle Routing Problem includes constraints for each vehicle, making certain that the total customer demand on a route does not exceed the vehicle's capacity. This is one of the most studied VRP variants and is foundational to many other extensions. The CVRP replicates real-world scenarios, such as delivery trucks transporting products, where overloading could lead to safety hazards and insufficiencies.

2. **Multi-Depot VRP (MDVRP)**

The Multi-Depot Vehicle Routing Problem extends the VRP to multiple depots, with vehicles starting and ending their routes at assigned depots. The MDVRP variant demonstrates the complexity of large-scale logistics networks. This variant involves both depot assignment and route optimization, making it computationally challenging.

3. **Time Window VRP (TWVRP)**

In the Time Window Vehicle Routing Problem, each customer is assigned with a specific time window during which the delivery must be made. TWVRPs are important in applications where timely delivery is crucial, like food delivery or medical logistics. This VRP extension adds complexity as routes not only minimize costs but also must comply with time window constraint.

4. **VRP with Pickup and Delivery (VRPPD)**

The Vehicle Routing Problem with Pickup and Delivery entails picking up products from determined locations and delivering them to designated destinations. The challenge of this variant lies in guaranteeing that the sequence of pickups and deliveries adhere to the constraints while optimizing routes. These specific origin-destination constraints make the VRPPD suitable for ride-sharing applications and courier services.

5. **Open VRP (OVRP)**

In the Open Vehicle Routing Problem, vehicles may not return to the depot. The flexibility of this variant reduces the total travel distance and operational costs, reflecting real-world scenarios like service industries or one-way delivery networks.

6. **Green VRP (GVRP)**

The Green Vehicle Routing Problem introduces environmental objectives into the routing process, focusing on reducing fuel consumption, and minimizing emissions. The GVRP gained significant attention as sustainability is becoming a priority for business. For example, companies pursue to minimize CO2 emissions while also meeting customer demands.

The methods for solving the Vehicle Routing Problem (VRP) significantly coincide with those used for the Traveling Salesman Problem. Both problems are NP-hard and a mix of exact, heuristic, and metaheuristic approaches is required for practical solutions. However, the VRP introduces additional complexities, like multiple vehicles, and capacity constraints, which call for modification of these methods. Therefore, while VRP solution techniques build upon TSP methods, their successful application depends on the ability to efficiently adapt to the Vehicle Routing Problem's constraints and challenges. Metaheuristic and hybrid methods are the most competent for handling the difficulties of large-scale, real-world VRPs.

The Vehicle Routing Problem is crucial for understanding and addressing more complex logistics challenges, such as the Location Routing Problem (LRP). While the VRP focuses on optimizing vehicle routes to efficiently serve customers, the LRP extends this by combining facility location decisions. This thesis develops the Vehicle Routing Problem's established methodologies to address the interdependent challenges of depot placement and vehicle route optimization in the Location Routing Problem.

CHAPTER 3: LOCATION ROUTING PROBLEMS

3.1 Introduction to Location Routing Problems (LRP)

The Location Routing Problem (LRP) is a crucial challenge in logistics, supply chain management, and operations research. It integrates two critical interdependent optimization problems: facility location and vehicle routing. Traditionally, these problems were focused on independently, which often led to suboptimal solutions. The LRP combines these challenges into a unified framework, providing a more holistic approach to optimization. This combination ensures resources are allocated efficiently, costs are reduced, and service levels across the supply chain are enhanced.

The Location Routing Problem focuses on making three crucial decisions simultaneously:

1. **Facility Location:** Choosing which facilities, such as depots or warehouses, to open from a set of possible locations. Facility selection considers fixed opening costs, operational efficiency, and geographical distribution.
2. **Customer Assignment:** Assigning customers to the opened facilities considering factors like proximity, demand, and transportation costs.
3. **Vehicle Routing:** Designing the optimal routes for vehicles departing from each open facility to serve assigned customers while complying with constraints like vehicle capacity and delivery time.

The objective of the LRP is to minimize the total cost, which includes the fixed costs of opening facilities, transportation costs associated with routing vehicles (distance traveled), and any other possible variable cost.

The Location Routing Problem is fundamental in shaping effective and cost-efficient distribution networks. Research has shown that making decisions about facility locations and vehicle routing simultaneously, often leads to more optimal outcomes, such as reduced transportation costs and better utilization of vehicles and facilities. Optimizing facility locations and vehicle routes improves service quality and guarantees timely delivery while keeping costs substantially low. These capabilities make the LRP a versatile tool for designing effective supply chains.

Despite its importance, the Location Routing Problem poses significant computational challenges. Classified as an NP-hard problem, it becomes exponentially difficult to find optimal solutions as the problem size grows. The interdependence of facility location and routing decisions, combined with additional constraints like facility and vehicle capacities, and customer demands, are responsible for the complexity of the LRP. Advanced optimization methods, specifically metaheuristic algorithms, are often used to overcome these challenges. These methods efficiently generate high-quality

solutions, demonstrating the significance of the Location Routing Problem in logistics and supply chain management.

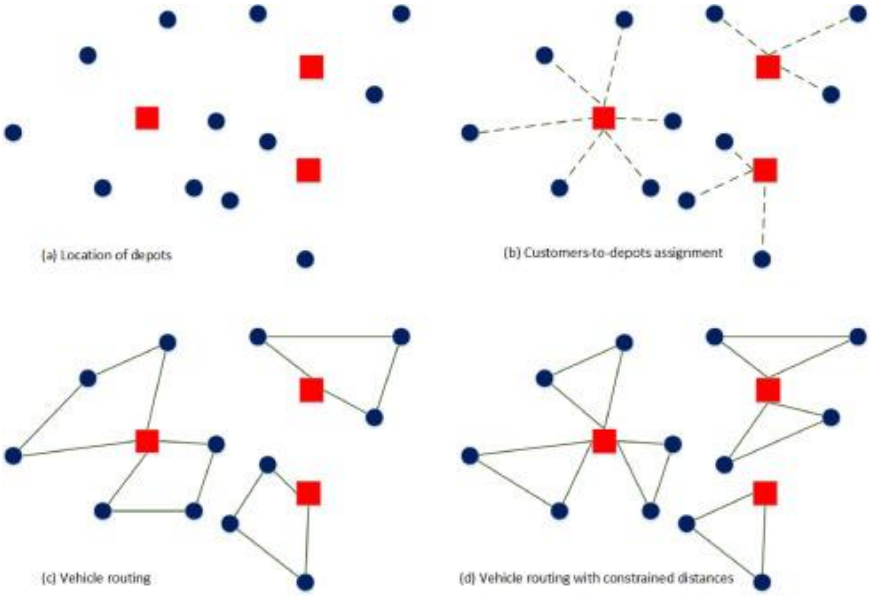

Fig.5: Graphing an LRP

3.2 Literature Review on Location Routing Problems

Due to its practical applications, the location routing problem (LRP) has become pivotal in logistics and operations research. As an extension of facility location and vehicle routing problems, the LRP combines these two problems, offering a cohesive framework for optimizing supply chains. This section examines the evolution of LRP research, its primary variants, solution methods, and emerging trends and advancements.

Early research in the 1960s and 1970s was primarily focused on Facility Location Problems and Vehicle Routing Problems as independent areas. Facility Location Problems aimed to minimize the costs of placing facilities like depots or warehouses to fulfil a given demand, while Vehicle Routing Problems focused on optimizing the vehicle routes for the efficient delivery of goods to customers.

Salhi and Rand (1989) **[2]** quantified the potential benefits of including vehicle routing decisions in depot location decisions for the first time in the 1980s. For example, placing depots without considering routing costs could result in longer travel distances and higher operational expenses, while optimizing routes independently of facility location could lead to underutilized facilities or poorly located depots.

This seminal study inspired a growing stream of research that introduced the Location Routing Problem as a unified problem. This research emphasized the benefits of

integrating the two aspects, demonstrating that the combination of location and routing decisions could significantly reduce costs and improve efficiency. The introductory models were simple and focused on deterministic scenarios, but they laid the foundation for more sophisticated approaches.

As computational capabilities improved, researchers began developing more complex Location Routing Problems that involve real-world constraints, like vehicle and facility capacities, customer demand, time windows, and multi-echelon networks. The rise of metaheuristic algorithms, such as Genetic Algorithms, Anto Colony Optimization, Cuckoo Search, and more, allowed for the practical solution of larger and more complicated Location Routing Problems. In the last years, the Location Routing Problem research community has been very active, and its research area has numerous applications in industries like retail, logistics, and urban planning.

3.2.2 Location Routing Problem Variants

The Location Routing Problem evolves along with the growing complexity of logistics networks and the need for adaptable and flexible optimization frameworks. As research advanced, it became obvious that different logistics contexts require modified approaches to tackle unique challenges, like vehicle and depot capacity limitations, multi-echelon supply chains, and environmental concerns. These needs resulted in several variants of the Location Routing Problem, each designed to address specific real-world scenarios more effectively. In this section, the key variants of the Location Routing Problem will be explored, highlighting their distinctive features, applications, and significance to modern logistics.

- **Capacitated Location Routing Problem (CLRP)**

One of the most prominent variants is the Capacitated Location Routing Problem, which like the name suggests, introduces capacity constraints on both vehicles and facilities. In this variant, depots have a maximum limit on the demand they can handle, while vehicles have a maximum carrying capacity. These constraints ensure that the assignment of customers to depots and the routing of vehicles remain possible within operational limits. The Capacitated Location Routing Problem can be found in industries like retail and pharmaceuticals, where storage and transportation resources need to be managed efficiently. For instance, pharmaceutical logistics rely on this variant to manage the distribution of temperature-sensitive medicines while complying with strict capacity constraints. **[10]**

- **Multi-Echelon Location Routing Problem (MELRP)**

The Multi-Echelon Location Routing Problem, the most important LRP modeling extension of the last decade, extends the standard Location Routing Problem to involve multi-tiered distribution systems. In this variant, goods are transported through multiple levels, like central depots, regional hubs, and local delivery depots, to optimize the flow of goods from origin to end customers. This structure adds complexity but allows for more realistic representation of large-scale logistics networks. The MELRP integrates decision-making across multiple layers of the supply chain, requiring simultaneous optimization of:

1. Facility Location: Finding the optimal locations for depots and distribution centers at different levels of the supply chain
2. Inter-Echelon Routing: Deciding the most efficient vehicle routes for goods to flow between central warehouses, regional hubs, and local depots
3. Last-Mile Delivery: Determining vehicle routes from the final tier of facilities to customers

The Multi-Echelon Location Routing Problem is crucial for global supply chains and healthcare logistics, where the flow of goods and inventory must be carefully coordinated across hierarchical networks. For example, international logistics systems require the MELRP to manage inventory distribution from manufacturing centers to regional distribution hubs to retail stores and finally to the customers. **[11]**

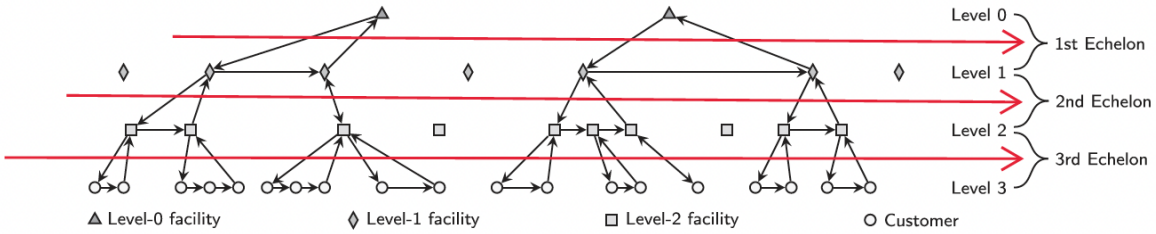

Fig.6: Example of a three-echelon routing problem

- **Dynamic and Stochastic Location Routing Problems**

These advanced variants address the inherent variability and uncertainty in real-world logistics operations. The Dynamic and Stochastic LRPs extend the classical Location Routing Problem by introducing methods to handle real-time changes and probabilistic processes, making them vital for businesses where conditions are dynamic or uncertain. The Dynamic Location Routing Problem (D-LRP) is suitable for scenarios where key parameters, like vehicle capacity, customer demand, or travel times, change over time. Dynamic LRPs continuously adapt to new conditions unlike static LRPs, whose inputs remain fixed. The flexibility of the Dynamic LRP makes it valuable in time-sensitive and

on-demand logistics operations, like courier services where routes need to be optimized in response to incoming orders. On the other hand, Stochastic Location Routing Problems (S-LRP) address uncertainty by representing parameters like customer demand, travel times, or vehicle capacity as random variables. This probabilistic process ensures that solutions are optimized and feasible even when conditions stray from initial estimates. An example where the Stochastic Location Routing Problem is applied, is in public transportation where passenger demand changes daily or seasonally. **[12]**

- **Open Location Routing Problem (OLRP)**

In the Open Location Routing Problem vehicles are not required to return to the depot after competing their delivery routes. This flexibility is useful in scenarios where the end point of a vehicle's route is different to the starting depot, allowing for greater efficiency and reducing operational costs. The elimination of the need for vehicles to return to their starting depot, reduces total travel distances and fuel consumption, making it suitable for many logistics operations. For instance, the OLRP is highly applicable in waste management operations where waste collection vehicles usually conclude their routes at disposal facilities, like landfills or recycling plants, rather than returning to the initial depot. The OLRP is used in this instance to optimize these routes, therefore minimizing travel distances and operational costs. **[13]**

- **Inventory-Based Location Routing Problem (ILRP)**

The Inventory-Based Location Routing Problem (ILRP) integrates inventory management decisions with facility location and vehicle routing optimization, addressing the interdependencies between stock levels, storage, and distribution logistics. This variant simultaneously optimizes inventory levels at facilities, facility locations, and delivery routes, guaranteeing customer demands are met without excess stock or shortages. By combining these decisions, the ILRP minimizes total logistics costs, including storage, replenishment, and transportation costs. The Inventory-Based LRP is particularly prevalent in industries where inventory availability is crucial for maintaining service levels. For example, a lot of big retail chains use ILRP to balance stock across regional distribution centers and stores, ensuring high-demand products are available while avoiding overstocking. **[14]**

- **Multi-Period Location Routing Problem (MPLRP)**

The Multi-Period Location Routing Problem (MPLRP) extends the classical LRP by including a temporal dimension, allowing for logistics operations decisions to be optimized across multiple time periods. The MPLRP considers dynamic changes in customer demands, resource availability, and operational constraints over time, laying out a flexible system for balancing short-term operational efficiency and long-term

strategic goals. In this variant, decisions about facility locations and vehicle routing are integrated and evolve as conditions change across time periods. This model enables changes such as opening or closing facilities and adjusting vehicle routes to adapt to varying demands, making it prevalent in scenarios with fluctuating or seasonal requirements. The Multi-Period Location Routing Problem is applied in several industries where logistics operations must adapt to time-related variations. For instance, construction projects use MPLRP to manage material deliveries and temporary facility locations, ensuring stock availability and efficient deliveries in accordance with project phases. **[15]**

- **Pickup-and-Delivery Location Routing Problem (PDLRP)**

In the Pickup-and-Delivery Location Routing Problem facility location decisions are combined with routing assignments that include both pickup and delivery operations. In contrast to traditional Location Routing Problems focused solely on distribution, the PDLRP variant involves vehicles transporting goods or passengers between specified pickup and delivery points, making the optimization process more complex. In this variant, vehicles must handle simultaneous pickup and delivery assignments, often requiring them to revisit facilities or change routes dynamically to adapt to shifting demands. The problem is further complicated by constraints like vehicle capacities, time windows for pickup and deliveries, and customer requirements. Some variants of the PDLRP include simultaneous pickup and delivery, where both assignments occur during the same visit, and many-to-many pickup and delivery, where products are picked up from one set of locations and transported to another. Pickup-and-Delivery Location Routing Problems are often applied in industries like postal services and reverse logistics, where pickup and delivery activities need efficient coordination. For example, postal services rely on PDLRP methods to optimize routes that guarantee timely pickups from customers and subsequent deliveries to assigned locations. **[16]**

- **Green Location Routing Problem (GLRP)**

The Green Location Routing Problem incorporates environmental sustainability into the traditional Location Routing Problem. Like the traditional LRP, the first step is locating depots on a subset of a discrete set of points, from where vehicles with a limited capacity will be dispatched to serve a number of customers with service requirements. Secondly, vehicles are routed by determining the order of customers served by each vehicle. Finally, the speed of the vehicles on each stage of the journey is set so that customers are served within their respective time windows. The Green Location Routing Problem extends this by aiming to achieve ecological goals, minimizing fuel consumption and CO_2 emissions. **[17]**

3.3 Mathematical Formulation of the Location Routing Problem

The Location Routing Problem (LRP) combines two important logistics challenges: figuring out the best spots for depots and determining how to assign customers to these depots, all while ensuring that the routes are as efficient as possible. To illustrate this, think of it as a complete directed graph, $G = (V, A)$, where V represents the nodes, which include both depot and customer locations, and A represents the connections, or arcs, that link these nodes together.

Objective Function

The objective is to minimize the total cost C, which includes:

1. Depot Opening Costs: Fixed costs associated with opening and operating depots.

2. Routing Costs: Costs related to traveling between nodes.

$$Minimize \quad C = \sum_{i \in I} O_i y_i + \sum_{k \in K} \sum_{(i,j) \in A} c_{ij} x_{ij}^k$$

Where:

- O_i: Cost of opening depot i
- y_i: Binary variable indicating whether depot is opened ($y_i = 1$) or closed ($y_i = 0$)
- c_{ij}: Traveling cost between nodes i and j
- x_{ij}^k: Binary variable indicating whether arc (i, j) is traveled by vehicle k

Constraints:

1. Depot Capacity: The total demand assigned to a depot must not exceed its capacity

$$\sum_{j \in J} d_i x_{ij} \leq Q_i y_i, \qquad \forall i \in I$$

Where Q_i is the capacity of depot i and d_j is the demand of customer j

2. Customer Assignment: Each customer must be assigned to exactly one vehicle.

$$\sum_{k \in K} \sum_{j \in J} x_{ij}^{k} = 1, \forall j \in J$$

3. Route Continuity: By balancing the flow of vehicles at each location, their routes can remain consistent and logical.

$$\sum_{j \in J} x_{ij}^{k} = \sum_{j \in J} x_{ij}^{k}, \forall i \in J, \forall k \in K$$

4. Vehicle Capacity: The total demand served on any vehicle route cannot exceed its capacity.

$$\sum_{j \in J} d_i x_{ij}^{k} \leq C_k, \qquad \forall k \in K$$

Where C_k is the capacity of vehicle k

5. Depot Availability: A depot needs to be open in order to serve the customers effectively.

$$x_{ij}^{k} \leq y_i, \qquad \forall i \in I, \forall j \in J, \forall k \in K$$

6. Fleet Limitations: The overall fleet size determines the maximum number of vehicles available at each depot.

This formulation combines the decisions made about locations and routes, highlighting the computational challenges and their relevance to real-world situations. It serves as a solid basis for investigating various solution methods, including exact, heuristic, and metaheuristic approaches.

3.4 Challenges in Solving Location Routing Problems

The Location Routing Problem (LRP) is crucial in logistics and supply chain optimization. However, due to its complexity and real-world constraints, finding effective solutions can be challenging. In this section, the key difficulties involved in tackling LRPs will be explored.

The Location Routing Problem (LRP) is recognized as an NP-hard problem. This difficulty arises from its two main components: the Facility Location Problem (FLP) and the Vehicle Routing Problem (VRP), both of which are also NP-hard. When these two challenges are combined, the potential solutions grow exponentially. For example, a scenario with 50 customers and 10 possible depot locations can lead to millions of different combinations to explore. Such complexity often makes it hard to use exact optimization methods in real-world situations.

The scalability issue significantly complicates the task of tackling Location Routing Problems (LRPs). In real-world logistics, you often encounter hundreds or even thousands of customers, numerous potential depots, and various vehicles, each with different capacities. Many heuristic and metaheuristic methods for solving large-scale problems struggle to achieve a balance between solution quality and computational efficiency. Consequently, finding top-notch solutions within a reasonable time frame continues to be a major hurdle when scaling LRPs for practical use.

One distinctive aspect of the LRP is the close relationship between where facilities are located and how vehicles are routed. The choice of depot locations can significantly shape routing strategies, while the costs associated with routing can influence depot placement. When these two decisions are not optimized simultaneously, it often leads to suboptimal outcomes because the connection between location and routing is overlooked. For example, if depots are chosen based solely on their location costs without considering the routing implications, overall transportation costs could rise unexpectedly. To tackle this problem effectively, it is important to adopt integrated approaches that simultaneously optimize both location and routing.

Factoring real-world constraints into LRP models adds yet another layer of complexity to the optimization process. Aspects like time windows, vehicle capacities, customer-specific demands, and depot limitations must all be considered during the optimization. Time windows, for instance, require careful scheduling to ensure that deliveries occur within set timeframes, complicating routing decisions further. Similarly, restrictions on

depot and vehicle capacity can limit how customers are assigned, and routes are determined, requiring advanced algorithms to find feasible solutions.

Some variants of LRP also incorporate dynamic and stochastic elements that create additional challenges. Dynamic LRPs (D-LRPs) need to respond to changing factors, such as fluctuating demands or real-time traffic conditions, and thus require adaptive algorithms that can recalibrate solutions as new information emerges. Stochastic LRPs (S-LRPs), on the other hand, deal with uncertainties, such as unpredictable travel times or customer demands, and require robust optimization techniques to ensure the solutions remain effective under varying circumstances. Such dynamic and unpredictable settings are particularly prevalent in fields like on-demand delivery and emergency logistics, where the ability to adapt in real-time is critical.

Many practical applications of LRPs have multiple objectives, such as minimizing costs, maximizing service levels, and achieving environmental sustainability. Balancing these often-conflicting goals adds further complexity to the challenge, requiring multi-objective optimization techniques that can account for trade-offs. For instance, cutting transportation costs might mean having fewer depots, which could, in turn, affect customer service quality. Approaches based on Pareto optimization and other advanced multi-objective methods are frequently employed to navigate these issues, but they also increase computational demands and require specialized methods for finding solutions.

Despite these challenges, the LRP is a vital area of exploration and practice, presenting significant opportunities for innovation. Emerging technologies like machine learning and big data analytics are utilized to enhance LRP solutions by improving demand forecasting, facilitating real-time adjustments, and optimizing resource allocation. Additionally, integrating sustainability goals, such as reducing carbon emissions, is becoming increasingly important as industries work towards greener supply chain practices. Addressing these challenges is crucial for developing efficient, scalable, and sustainable logistics networks that meet modern supply chains' evolving demands.

CHAPTER 4: OPTIMIZATION ALGORITHMS

4.1 Overview of Optimization Algorithms

Optimization algorithms are essential tools in tackling challenges within logistics and supply chain management. They work by seeking the best or nearly the best solutions to intricate issues like the Location Routing Problem (LRP), often maneuvering through large and complex solution landscapes. Given the combinatorial complexity of these problems, optimization approaches are divided into several categories, including exact methods, heuristics, metaheuristics, and hybrid strategies. Each type is designed to address particular challenges and the scale of the problem at hand.

Exact Methods

Exact optimization methods are designed to find the best possible solution by thoroughly examining every potential option in the solution space. Common techniques used for this purpose include mixed-integer linear programming (MILP), branch-and-bound, and branch-and-cut methods. These approaches utilize detailed mathematical formulations, which allow them to deliver definitive results, especially for small to medium-sized location routing problems.

For instance, branch-and-bound methods work by systematically eliminating solutions that are either infeasible or suboptimal, thereby streamlining the search process and ensuring that an optimal solution is reached. Branch-and-cut takes this a step further by adding extra constraints that further refine the solution space.

However, despite their accuracy, exact methods have significant limitations as the size of the problem increases. The computational demands can rise exponentially with the number of customers, depots, and vehicles, making these methods less feasible for larger, real-world applications. For example, attempting to solve an LRP involving hundreds of customers and multiple depots using exact methods can become incredibly time-consuming, often taking longer than what is considered practical.

Heuristic Methods

Heuristics are methods that help solve optimization problems by streamlining the decision-making process and providing approximate solutions. They are specifically designed to quickly produce workable solutions by using rules tailored to the problem or intuitive strategies. This approach often prioritizes speed and efficiency over finding the perfect solution. **[18]**

Some well-known heuristic methods include:

- Nearest Neighbor Algorithm: This method builds routes by repeatedly choosing the closest unvisited location. It offers a straightforward yet effective solution for challenges like the Traveling Salesman Problem (TSP) and Vehicle Routing Problem (VRP).
- Savings Algorithm: This technique clusters customers based on the cost savings achieved by merging routes, which helps in lowering the overall travel distance.
- Clustering Methods: These methods group customers by their proximity or compatibility in terms of demand before optimizing the routes within each group.

While heuristics are excellent for quickly generating solutions, they often serve as a milestone toward more sophisticated approaches like metaheuristics. For example, a heuristic solution can be used as the starting point for metaheuristic algorithms, which then work to enhance and improve that initial solution.

Metaheuristic Methods

Metaheuristics enhance traditional heuristics by incorporating strategies that allow for a more comprehensive exploration of potential solutions. They strike a balance between exploring new areas within the solution and refining existing promising solutions, which makes them particularly effective for tackling large and complex optimization problems like Location Routing Problems (LRPs) **[19]**

Some of the key metaheuristic methods include:

- Genetic Algorithms (GAs): These algorithms are inspired by natural selection. They work with groups of solutions that evolve over time through operations like crossover, mutation, and selection.
- Simulated Annealing (SA): This technique mimics the annealing process used in metallurgy, gradually reducing the likelihood of accepting suboptimal solutions to hone in on the best possible outcome.
- Ant Colony Optimization (ACO): ACO is based on the behavior of ants foraging for food. The method utilizes pheromone trails to guide the search for the most efficient routes.
- Particle Swarm Optimization (PSO): This approach simulates the social dynamics of birds or fish. In this model, individual particles (representing potential solutions) adjust their positions based on their own experiences as well as those of their neighbors.
- Cuckoo Search Algorithm (CSA): The Cuckoo Search Algorithm draws inspiration from the unique nesting habits of cuckoo birds, which lay their eggs in the nests of other birds. This algorithm employs Levy flights to navigate through the search space effectively. Thanks to its adaptability and efficiency, CSA has become a favored option for tackling complicated logistics optimization challenges.

Metaheuristics are highly versatile, making them well-suited for addressing LRPs that involve dynamic and uncertain elements. For instance, ACO is particularly effective in scenarios where customer demand or travel times fluctuate, while PSO is adept at handling optimization problems with multiple objectives.

Hybrid Methods

Hybrid optimization algorithms blend different techniques, including exact methods, heuristics, and metaheuristics, to take advantage of their unique strengths. For example, a hybrid strategy might begin with a heuristic algorithm like the Nearest Neighbor Algorithm to obtain an initial solution and then improve that solution using a metaheuristic method such as the Cuckoo Search Algorithm or Genetic Algorithms.

Recently, there has been a growing trend to incorporate machine learning models into these hybrid approaches. This allows for better predictions of demand patterns and the dynamic adjustment of algorithm parameters. Such methods boost scalability and adaptability, making them especially effective for tackling real-world Location Routing Problems that have intricate constraints and objectives. **[20]**

Relevance to the LRP

The close relationship between location and routing decisions in Location Routing Problems (LRPs) creates distinct challenges that demand optimization algorithms capable of navigating both interdependencies and constraints. While exact methods can deliver precise results, they often struggle with scalability. On the other hand, heuristic and metaheuristic approaches offer practical, high-quality solutions for larger, more complex scenarios. Hybrid methods improve efficiency even further by combining the strengths of different strategies, effectively bridging the gap between theoretical models and real-world applications.

As a result, optimization algorithms are essential for tackling the computational hurdles associated with LRPs, helping to design efficient, scalable, and sustainable logistics networks. The upcoming sections explore popular optimization methods and recent advancements in detail, emphasizing their relevance and application to logistics challenges such as LRPs.

4.2 Popular Optimization Algorithms

Optimization algorithms have become essential for solving logistics challenges like the Location Routing Problem (LRP). Their efficiency in navigating large solution spaces makes them invaluable for addressing the intricacies of logistics and supply chain management. While exact methods can deliver precise results, their computational limitations often render them ineffective for larger problems, especially those involving dynamic or uncertain factors. This reality has led to a growing preference for heuristic and

metaheuristic approaches, which strike a balance between solution quality and computational efficiency.

Heuristic methods, such as the Nearest Neighbor Algorithm (NNA), are popular for generating initial solutions, offering a solid foundation for further refinement. Their simplicity and speed make them particularly appealing for real-world scenarios where quick decisions are crucial. In contrast, metaheuristic algorithms like Genetic Algorithms, Simulated Annealing, and the Cuckoo Search Algorithm enhance heuristic methods by thoroughly exploring the solution space and escaping local optima.

This section will explore some of the most widely used optimization algorithms, with a special emphasis on the Nearest Neighbor Algorithm, which is applied in this thesis to generate initial solutions for the LRP.

4.2.2 Nearest Neighbor Algorithm

The Nearest Neighbor Algorithm is a straightforward and effective approach commonly applied to routing challenges like the Traveling Salesman Problem (TSP) and the Vehicle Routing Problem (VRP). The main idea behind this algorithm is quite simple: it begins at a starting point and then repeatedly chooses and visits the nearest unvisited location until every destination has been covered. This method helps create a route that meets the essential criteria of the problem, such as ensuring all customer nodes are visited with the least amount of distance traveled.

One of the reasons the Nearest Neighbor Algorithm (NNA) is so popular is its efficiency and ease of use, which make it great for generating initial solutions in various optimization scenarios. However, because the algorithm chooses the next step based solely on immediate distance, it can sometimes overlook the bigger picture, leading to less-than-ideal routes. Even with this drawback, the NNA remains a key method in logistics, serving as a foundation for more sophisticated optimization methods. **[21]**

Algorithm Steps

The Nearest Neighbor Algorithm builds a route by repeatedly choosing the closest unvisited location, focusing on ease of use and quick calculations. Here's a detailed, step-by-step breakdown of how the algorithm works:

1. Initialization:

- Choose a starting node, typically the depot in problems like the LRP or VRP. If no depot is specified, any node can be designated as the starting point.
- Create a list or array to track visited nodes, marking the starting node as visited.
- Set the current node as the starting node, which will serve as the reference for selecting the next node.

Example: In a network consisting of nodes {A, B, C, D, E, F}, with A serving as the depot, the algorithm starts its journey at A. At the outset, the list of visited nodes contains just [A], and A is designated as the current node.

2. Node Selection:

- Identify all unvisited nodes in the network.
- Compute the distance (or cost) from the current node to each unvisited node using a predefined metric, such as Euclidean distance.
- Select the node with the shortest distance to the current node as the next node to visit.

  Example: If the current node is A and the unvisited nodes are {B, C, D, E, F}, the distances are calculated:

  - d(A, B) = 10
  - d(A, C) = 12
  - d(A, D) = 8
  - d(A, E) = 15
  - d(A, F) = 9

  The nearest unvisited depot is D, with a distance of 8.

3. Update:

- Move the selected node and add it to the visited list.
- Set this node as the new current node

Example: After visiting D, the visited list becomes [A, D], and D is now the current node.

4. Repeat:

- Repeat the Node Selection and Update processes until all nodes have been visited.

Example: Starting from D, the algorithm calculates the distances to the remaining unvisited nodes {B, C, E, F}, selecting the nearest node and updating the visited list. This process continues until all nodes are visited.

5. Completion:

- Once all nodes have been visited, return to the starting node to complete the route.
- Record the resulting route as the final solution.

Example: After visiting nodes in the order [A, D, F, B, C, E], the algorithm returns to A, completing the route $A \rightarrow D \rightarrow F \rightarrow B \rightarrow C \rightarrow E \rightarrow A$.

Application of the Nearest Neighbor Algorithm in This Thesis

In this study, the Nearest Neighbor Algorithm (NNA) is utilized as a key starting point to create initial solutions for the Location Routing Problem (LRP). This algorithm plays an important role in matching customers with depots and planning vehicle routes. It helps improve computational efficiency while keeping travel distances as short as possible.

1. Calculating Distances

The first step involves calculating and storing the distances between all relevant nodes:

- Customer-Depot Distances: The Euclidean distance between each customer and every depot is calculated and stored in the **depot_distances** list of each customer.
- Customer-Customer Distances: Similarly, the distances between every pair of customers are calculated and stored in **customer_distances** list.

These precalculated distances allow for efficient comparisons during the execution of the Nearest Neighbor Algorithm.

2. Assigning Customers to Depots

The **assign_depots** function applies the Nearest Neighbor Algorithm to assign each customer to the nearest feasible depot:

- Sorting by proximity: The list of depot distances is sorted by proximity for each customer, prioritizing the closest depots.
- Greedy Assignment: The algorithm iterates through the sorted list and assigns the customer to the nearest depot that meets the following criteria:
    - The depot is either already open or can be opened without exceeding capacity.
    - The customer's demand can be accommodated by the depot.
- Depot Management: If a closed depot is selected, it is opened, incurring a fixed cost. The depot's capacity is then updated to reflect the assigned customer's demand.

This process ensures that each customer is assigned to the nearest depot that can serve their demand, following the core principle of the Nearest Neighbor Algorithm.

```
Function assign_depots(customers, depots, total_cost):
    For each customer in customers:
        Sort depots by distance to the customer (ascending order)
        For each depot in sorted depots:
            If depot is open AND depot.capacity >= customer.demand:
                Assign customer to depot
                Add customer to depot's assigned_customers list
                Reduce depot.capacity by customer.demand
                Mark depot as the assigned_depot for the customer
                Break out of loop
            Else If depot is closed AND depot.capacity >= customer.demand:
                Open the depot
                Add depot.fixed_cost to total_cost
                Assign customer to depot
                Add customer to depot's assigned_customers list
                Reduce depot.capacity by customer.demand
                Mark depot as the assigned_depot for the customer
                Break out of loop
            Else If depot.capacity < customer.demand:
                Continue to the next depot
    Return total_cost
```

Fig.7: Pseudocode for **assign_depots**

3. Defining Vehicle Routes

   After customers are assigned to depots, vehicle routes are defined using the Nearest Neighbor Algorithm:

   - Initialization: Each depot initializes its vehicle fleet. The first vehicle begins by selecting the closest customer to the depot.
   - Iterative Customer Assignment: The algorithm iteratively adds the next closest unvisited customer to the vehicle route until the vehicle's capacity is exhausted. Then, a new vehicle is initialized, and the process is repeated until all customers assigned to the depot are routed.
   - Completion: The resulting vehicle routes minimize travel distances from each depot to its assigned customers.

```
Function define_vehicle_routes(depots):
    For each depot in depots:
        Initialize vehicle_number = 0
        While depot has assigned customers:
            Create a new vehicle for the depot
            Assign the closest customer to the vehicle
            Remove the assigned customer from depot's assigned_customers list
            Reduce vehicle.capacity by customer.demand
            While vehicle has remaining capacity AND there are assigned customers:
                Find the closest customer to the last customer in the vehicle
                If vehicle.capacity >= customer.demand:
                    Assign customer to the vehicle
                    Remove the assigned customer from depot's assigned_customers list
                    Reduce vehicle.capacity by customer.demand
                Else:
                    Break out of loop
        Calculate the total distance for each vehicle's route

```

Fig. 8: Pseudocode for **define_vehicle_routes**

4. Generating Diverse Solutions

To explore multiple solutions to the LRP, the **create_combinations** function generates different depot combinations:

- Depot Subsets: The function systematically creates subsets of depots, reducing the number of open depots in each combination.
- Iterative Application of the Nearest Neighbor Algorithm: For each depot combination, the **assign_depots** and **define_vehicle_routes** functions are executed to produce a complete solution. This approach enables the evaluation of diverse depot combinations while maintaining computational efficiency.

The incorporation of the Nearest Neighbor Algorithm (NNA) into the solution process offers a strong foundation for optimization. Despite being a greedy algorithm that does not always find the best possible solution, its quickness and ease of use make it ideal for producing initial solutions. These preliminary solutions are subsequently improved using the Cuckoo Search Algorithm, which combines the advantages of both techniques to effectively tackle the complexities of the Location Routing Problem (LRP).

4.2.3 Metaheuristic Methods

Genetic Algorithms (GAs)

Genetic Algorithms (GAs) take inspiration from the natural selection process to improve a group of potential solutions over time. Each generation undergoes crossover, mutation, and selection, which collectively work to refine and enhance the quality of the outcomes.

One of the key features of GAs is their ability to maintain a diverse pool of solutions, enabling a broader exploration of the solution space. They are particularly effective for multi-objective optimization problems, showcasing their adaptability in complex scenarios. However, GAs can be computationally intensive when managing large populations of solutions, and they require careful tuning of parameters like mutation rates and crossover probabilities to optimize their performance effectively. **[22]**

Simulated Annealing (SA)

Simulated Annealing (SA) is inspired by the annealing process used in metallurgy, which allows it to occasionally accept suboptimal solutions. This flexibility enables the algorithm to avoid becoming trapped in local optima and to gradually steer itself toward high-quality solutions. It is especially effective for single-objective optimization challenges that involve complex constraints. As the algorithm advances, it employs a controlled exploration strategy that gradually diminishes the chances of accepting inferior solutions, facilitating a more focused search for the optimal outcome. SA is widely applied in variants of Vehicle Routing Problems (VRP) and Location Routing Problems (LRP), particularly in scenarios that involve dynamic or uncertain factors, making it a versatile tool in these fields. **[23]**

Ant Colony Optimization (ACO)

Ant Colony Optimization (ACO) is inspired by the foraging behavior of ants, which use pheromone trails to communicate and guide each other toward the most efficient paths. The algorithm constructs solutions in a probabilistic manner, relying on the strength of these pheromone trails along with additional heuristic information to make decisions. Pheromones naturally evaporate over time, promoting the exploration of diverse and potentially more effective routes. ACO is particularly well-suited for tackling various routing challenges, such as the Traveling Salesman Problem (TSP), Vehicle Routing Problem (VRP), and Location Routing Problem (LRP). Additionally, the algorithm's design allows for parallel processing, making it computationally efficient and adaptable to complex logistics scenarios. **[24]**

Cuckoo Search Algorithm (CSA)

The Cuckoo Search Algorithm (CSA) is a metaheuristic inspired by the unique nesting behavior of cuckoo birds, which are known for laying their eggs in the nests of other bird species. This algorithm employs a technique called Lévy flights to explore potential solutions, balancing wide searches for new possibilities with the refinement of existing ones. In this thesis, the CSA is used to improve the initial solutions generated by the Nearest Neighbor Algorithm (NNA). By addressing the limitations of the NNA, CSA effectively contributes to optimizing the final outcomes. This two-step approach combines the strengths of both algorithms to address the complexities of the Location Routing Problem (LRP).

4.3 Advancements and Hybrid Approaches

The field of optimization algorithms has made remarkable strides in recent years, largely due to the growing complexity of real-world logistics challenges, such as the Location Routing Problem (LRP). This section explores the latest trends and hybrid strategies that have improved the performance of optimization methods, especially in large-scale and ever-changing environments.

Emerging Trends in Optimization Algorithms

- Machine Learning Integration

Machine learning techniques are increasingly being integrated into optimization algorithms to boost their effectiveness. For instance, reinforcement learning models can adjust parameters on the fly, allowing algorithms to become more adaptable in real-time situations. In addition, predictive models help anticipate customer demands or traffic trends, enabling optimization methods to respond effectively to shifting conditions. This integration is especially beneficial for dynamic and stochastic variants of the Location Routing Problem (LRP), where conventional methods might face challenges.

- Multi-Objective Optimization

Numerous real-world challenges feature conflicting objectives, such as minimizing costs while simultaneously maximizing service quality or reducing environmental impacts. Recent advancements in multi-objective optimization have facilitated the development of algorithms capable of identifying Pareto-optimal solutions, thereby effectively balancing these trade-offs. For instance, genetic algorithms with multi-objective capabilities have been utilized in the context of green logistics, enabling the simultaneous optimization of fuel consumption and delivery efficiency.

Hybrid Approaches

1. Combination of Heuristics and Metaheuristics

Hybrid methods that blend heuristics and metaheuristics harness the advantages of both strategies. For example, heuristics like the Nearest Neighbor Algorithm (NNA) can quickly generate initial solutions, which can then be improved upon using metaheuristics such as Simulated Annealing or Genetic Algorithms. This two-step approach strikes a balance between processing speed and the quality of the final solution.

2. Exact and Metaheuristic Methods

The integration of exact methods with metaheuristics effectively addresses the inherent limitations of each approach. For instance, branch-and-bound techniques can be employed to refine the solution space, while Ant Colony Optimization can subsequently explore this constrained space to identify high-quality solutions. This hybrid

methodology proves particularly advantageous for small to medium-sized problems, where the application of exact methods remains feasible without incurring excessive computational demands.

3. Metaheuristic-Metaheuristic Hybrids

The integration of multiple metaheuristic algorithms has demonstrated significant efficacy in addressing complex optimization problems. For instance, a hybrid approach that amalgamates the exploration capabilities inherent in the Cuckoo Search Algorithm with the solution diversity fostered by Genetic Algorithms can yield superior outcomes. This synergistic method establishes a robust equilibrium between global exploration and local exploitation, thereby enhancing convergence rates and improving the quality of the resultant solutions.

Recent advancements in optimization algorithms and the integration of hybrid methodologies have substantially enhanced the capability to address complex Location Routing Problems (LRPs). These sophisticated approaches are particularly adept at managing dynamic and stochastic LRPs, where conditions evolve over time, necessitating real-time adaptability of the algorithms. Moreover, they demonstrate significant efficacy in multi-echelon LRPs, which involve hierarchical distribution systems requiring concurrent optimization across multiple tiers. Furthermore, these methods are well-aligned with the principles of green logistics, focusing on the reduction of environmental impact while simultaneously maintaining logistics efficiency. By harnessing these cutting-edge techniques, researchers and practitioners have effectively navigated the multifaceted challenges associated with LRPs, thereby advancing the potential and applicability of optimization algorithms in this field.

CHAPTER 5: THE CUCKOO SEARCH ALGORITHM FOR THE LRP

5.1 Cuckoo Search Algorithm

The Cuckoo Search Algorithm (CSA), introduced by Yang and Deb in 2009 **[25]**, is a pioneering optimization method inspired by the intriguing behavior of cuckoo birds. These birds are known for laying their eggs in the nests of other birds, thereby relying on the hosts to care for their chicks. This clever strategy is mirrored in the CSA, where candidate solutions compete for survival based on their effectiveness, similar to how cuckoo eggs can displace those of the host.

The algorithm operates on a few fundamental principles. First, each cuckoo lays one egg, representing a candidate solution, in a randomly chosen host nest. The solutions that prove to be the most effective, akin to the nests with the best conditions, are preserved for the next generation. Additionally, a portion of the nests is abandoned based on a certain probability, pa, and replaced with new solutions generated at random.

A distinguishing characteristic of the CSA is its use of Lévy flights to explore the solution space. This method incorporates a random walk pattern consisting of frequent short steps combined with occasional longer jumps. This approach strikes a balance between global exploration and local refinement, allowing the algorithm to break free from local optima and efficiently converge on high-quality solutions.

5.1.2 Mathematical Formulation of CSA

The Cuckoo Search Algorithm operates iteratively as follows:

1. **Initialization:** A population of n solutions $\{X_1, X_2, \dots, X_n\}$ is randomly generated within the search space.
2. **Generate New Solutions:** New candidate solutions are generated using Lévy flights:

   $$X_{new} = X_{current} + \alpha * \ Lévy(\lambda)$$

   Here, α is the step size scaling factor, and $Lévy(\lambda)$ represents the step length drawn from Lévy distribution.
3. **Fitness Evaluation:** The fitness $f(X)$ of each solution is evaluated based on the optimization objective. Better solutions replace inferior ones in the population.

4. **Abandonment and Replacement:** A fraction pa of solutions are abandoned and replaced by new randomly generated solutions. This maintains diversity in the solution pool and prevents premature convergence.
5. **Stopping Criteria:** The algorithm iterates until it meets a predefined stopping criterion, such as a maximum number of iterations or achieving a target solution quality.

5.1.3 Applications of CSA and Relevance to This Thesis

The Cuckoo Search Algorithm (CSA) has proven effective in tackling a range of optimization challenges. In Capacitated Vehicle Routing Problems (CVRP), it enhances vehicle routes to minimize costs while ensuring that capacity limits are respected. Additionally, CSA efficiently manages resource allocation over time in scheduling problems, such as job-shop scheduling. Furthermore, the algorithm addresses logistics optimization by effectively resolving issues related to facility locations and planning the most efficient routes.

In this research, the Cuckoo Search Algorithm (CSA) is employed to enhance the initial solutions generated by the Nearest Neighbor Algorithm (NNA) for the Location Routing Problem (LRP). While the NNA serves as an effective method for producing efficient initial solutions, the CSA further improves these outcomes through iterative refinement of depot placements and vehicle routing strategies. This approach capitalizes on the global search capabilities of the CSA to systematically minimize the total distance traveled while concurrently adhering to the constraints imposed by depot capacities and customer demand requirements.

5.2 Adaptation for the Location Routing Problem

The Location Routing Problem (LRP) poses a complex optimization challenge that combines elements of facility location with vehicle routing. To address this complexity, this thesis customizes the Cuckoo Search Algorithm (CSA) to improve depot configurations and optimize vehicle routing in line with critical constraints, including capacity limits and the minimization of total travel distances. This tailored adaptation builds on the foundational solutions generated by the Nearest Neighbor Algorithm (NNA), effectively utilizing CSA's strengths in global exploration and local exploitation to refine outcomes and enhance overall optimization results.

The optimization process begins by generating initial solutions through the Nearest Neighbor Algorithm (NNA). This algorithm assigns customers to the nearest available depots based on precomputed Euclidean distances while respecting capacity limitations. Depots are opened as needed to meet demand, resulting in fixed costs.

Following this, vehicle routes are created by iteratively selecting the nearest unvisited customer until capacity limits are reached. These initial solutions provide a strong foundation for the Cuckoo Search Algorithm (CSA), allowing it to focus more on refining the routes rather than constructing them from the ground up, which improves computational efficiency.

Custom Fitness Function and Optimization Objective

To adapt the Cuckoo Search Algorithm (CSA) for the Location Routing Problem (LRP), a fitness function was formulated to systematically evaluate and enhance the generated solutions. This fitness function is aimed at minimizing:

$$f(X) = Total\ Distance + \sum(Fixed\ Cost\ of\ Opened\ Depots)$$

- The total distance accounts for the cumulative travel distance of all vehicles in a solution.
- The fixed cost of opened depots penalizes solutions with excessive depot openings, encouraging efficient resource utilization.

  This balanced formulation aligns with the LRP's goals of cost-efficiency and feasibility.

Important adjustments were made to the core functions of the Cuckoo Search Algorithm (CSA) to meet the unique needs of the Location Routing Problem (LRP):

1. **Lévy Flights for Exploration and Refinement:**
   Lévy flights are known for their unique blend of both long exploratory leaps and shorter, more precise adjustments. These movements were designed to dynamically change depot setups and routing choices. The larger jumps allowed for global changes, like swapping customers between different depots, while the smaller steps helped refine customer assignments and optimize vehicle routes.

2. **Abandonment Strategy for Diversity:**
   In each generation, a portion of the solutions, based on the parameter pa, was discarded and replaced with new randomly generated configurations. This approach introduced a variety of depot arrangements and customer assignments, helping to avoid early convergence and ensuring a thorough exploration of the solution space.

3. **Parameter Settings:**
   - **Population Size:** A population of 20 solutions was maintained, providing sufficient diversity without overwhelming computational resources.

- **Abandonment Probability (pa):** Solutions were checked for abandonment with a probability equal to 0.25, introducing controlled diversity.

- **Lévy Flight Step (S):** $S = \frac{u}{|v|^{1/\beta}}$

  Where:
  - $u \sim N(0, \sigma_u^2)$ and $v \sim N(0, \sigma_v^2)$ are random variables drawn from a normal distribution.
  - $\beta = 1.5$
  - $\sigma_u = \left(\frac{\Gamma(1+\beta)\sin\left(\pi^{\beta}/_{2}\right)}{\Gamma\left(\frac{1+\beta}{2}\right)\beta 2^{\beta-1/_{2}}} \right)$ and $\sigma_v = 1$

- **Calculate Threshold for Lévy Flight Step (S):** In this thesis, the threshold for the Lévy Flight Step (S) was determined by calculating S over 50 iterations, yielding an average value. This threshold plays a crucial role in guiding the optimization strategy used throughout the research. When evaluating each newly generated S, the new S is compared to the established threshold. If the new S is found to be below the threshold, a local search strategy is implemented, as the smaller step size allows for more precise refinements of the solution space. Conversely, if the new S exceeds the threshold, a global search approach is utilized, enabling extensive exploration of the search landscape. This dual strategy effectively balances exploration and exploitation, thereby enhancing the overall effectiveness of the optimization process within the study.

- **Step Size (α):** Calibrated for effective Lévy flight scaling. (0.01, 0.05, 0.1)

- **Scaled Lévy Flight Step (Scaled S):** In the context of the Scaled Lévy Flight Step, each computed step size S is multiplied by a predefined step size parameter (a). This process produces the Scaled S, which quantitatively indicates the extent of changes to be applied in generating the new solution. By adjusting the step size, the Scaled S enables a systematic exploration of the solution space, thereby improving the algorithm's effectiveness in navigating complex landscapes and optimizing outcomes.

- **Stopping Criterion:** Iterations are limited to a maximum of 100 generations.

The implementation of the Cuckoo Search Algorithm (CSA) followed a systematic workflow designed for optimizing depot configurations and vehicle routes with efficiency. The process began with initializing the population, during which initial solutions were generated using the Nearest Neighbor Algorithm (NNA). These preliminary solutions laid a foundation by connecting customers to depots and outlining initial vehicle routes. The combination of CSA with NNA illustrates a synergistic approach to solving the Location Routing Problem (LRP). While NNA provides effective initial solutions, CSA enhances these through iterative refinement, leveraging its capabilities for both global exploration and local exploitation. This dual-method strategy offers a scalable and efficient framework for addressing large-scale logistics challenges, enabling optimal decisions in depot selection and routing while adhering to practical constraints.

Next, the algorithm engaged in iterative refinement, in which new solutions were progressively generated in each generation using Lévy flights. These flights introduced changes to both depot configurations and customer routes, ensuring a thorough exploration of the solution space. Each solution was evaluated based on a fitness function, retaining the most effective solutions for subsequent generations. An abandonment strategy was applied to maintain diversity within the solution pool, where some of the existing solutions were replaced with newly generated random solutions. This iterative process continued until the stopping criterion was satisfied, achieved by reaching a maximum number of iterations. Ultimately, the algorithm produced the optimal solution, detailing the improved depot configurations, vehicle routes, and total travel distance, thus successfully addressing the objectives of the LRP.

5.3 Pseudocode and Logic

The method used to solve the Location Routing Problem (LRP) in this thesis consists of two main stages. First, initial solutions are generated using the Nearest Neighbor Algorithm (NNA). These solutions are then enhanced with the Cuckoo Search Algorithm (CSA). This section outlines the logical workflow and provides the pseudocode for the key components of the implementation.

The algorithm is designed to enhance depot setups and vehicle routes while minimizing total travel distance and adhering to specific constraints. It begins with input parsing, where customer and depot information is gathered and processed to establish the initial nodes. Following this, an initial solution is generated by assigning customers to depots and creating vehicle routes using the Nearest Neighbor Algorithm (NNA). To further improve these solutions, an iterative refinement process is employed through the Cuckoo Search Algorithm (CSA), which utilizes Lévy flights for effective exploration and implements strategies to maintain diversity. The solutions are then evaluated by calculating their fitness based on total travel distance and fixed depot costs. Ultimately, the algorithm outputs the best solution, accompanied by detailed information about the depot configurations.

A. Initialization

The Cuckoo Search Algorithm (CSA) begins by leveraging solutions generated by the Nearest Neighbor Algorithm (NNA). These initial solutions consist of practical depot configurations and vehicle routes, which provide a solid foundation for further development. The process starts with importing customer and depot data from files, followed by precomputing the distances between depots and customers to enable quick assessments. Customers are then assigned to depots based on their proximity and capacity constraints, laying the groundwork for a robust solution set. This approach significantly reduces the computational workload for the CSA, allowing it to focus on refining these initial solutions rather than constructing everything from scratch.

```
FUNCTION GenerateInitialSolutions(customers, depots, num_solutions):
    combinations ← CreateDepotCombinations(depots, num_solutions)
    solutions ← []
    FOR EACH combination IN combinations:
        solution ← CreateSolution(customers, depots)
        AssignDepots(customers, depots, solution)
        DefineVehicleRoutes(depots)
        CalculateTotalDistance(solution)
        solutions.ADD(solution)
    RETURN solutions
```

Fig. 9: Pseudocode for Generating Initial Solutions Using NNA

B. Lévy Flights for Exploration and Threshold Mechanic

In the Lévy Step Calculation, the size of each step is generated using the **levy_flight_step** function, which depends on parameters such as the Lévy distribution parameter to influence the distribution of these steps. At the onset of the Cuckoo Search Algorithm (CSA), a dynamic threshold is established based on a sample of steps; this threshold functions as a flexible benchmark that differentiates between various levels of search intensity. When the global search operates above this threshold, it introduces significant changes to the solution, such as reassigning depots or redistributing customers across different depots. Conversely, when operating below the threshold, the local search fine-tunes the solutions by optimizing routes within depots, thereby enhancing overall efficiency.

```
FUNCTION levy_flight_step(beta):
    v ← RandomNormalValue()  // Generate random normal variable
    u ← LevyStepValue(beta)  // Generate step based on Lévy distributio
    step ← u / (ABS(v))^(1/beta)
    RETURN ABS(step)

FUNCTION levy_threshold(beta, num_samples):
    steps ← []
    FOR i ← 1 TO num_samples:
        step ← LevyFlightStep(beta)
        steps.ADD(ABS(step))
    threshold ← Mean(steps)
    RETURN threshold
```

Fig. 10: Pseudocode for **levy_flight_step** and **levy_threshold**

```python
def levy_flight_step():  2 usages
    v = np.random.normal( loc: 0,  scale: 1)
    sigma_u = (gamma(1 + beta) * np.sin(np.pi * beta / 2) /
               (gamma((1 + beta) / 2) * beta * 2**((beta - 1) / 2)))**(1 / beta)

    u = np.random.normal( loc: 0, sigma_u)
    return u / abs(v)**(1 / beta)

def levy_threshold():  1 usage
    levy_steps = []
    for i in range(levy_steps_number):
        levy_steps.append(abs(levy_flight_step()))
    return sum(levy_steps) / len(levy_steps)
```

Fig. 10.1: Python Implementation of **levy_flight_step** and **levy_threshold**

C. Global Search

Global Search emphasizes making changes at the depot level to uncover substantial variations in the solution landscape. The process begins by randomly selecting two depots from the existing solution. Next, customers are exchanged between these selected depots while ensuring that capacity constraints are respected. After making these adjustments, the fitness of the modified solution is evaluated. This approach introduces alternative depot configurations, enabling a broader exploration of possible solutions within the search space.

```
FUNCTION global_search(solution, scaled_step):
    FOR i ← 1 TO scaled_step:
        depot1, depot2 ← SelectRandomDepots(solution)
        WHILE depot1 HAS customers AND depot2 HAS capacity:
            customer ← SelectRandomCustomer(depot1)
            IF CustomerCanFitInDepot(customer, depot2):
                AssignCustomerToDepot(customer, depot2)
                RemoveCustomerFromDepot(customer, depot1)
    RETURN solution
```

Fig. 11: Pseudocode for **global_search**

```
def global_search(solution: Solution, scaled_s):  1 usage
    # Require more than 1 depot
    if len(solution.depots) == 1:
        return
    depot_1 = random.choice(solution.depots)
    depot_2 = random.choice(solution.depots)
    if len(depot_2.vehicles) == 0:
        return
    vehicle_1 = random.choice(depot_1.vehicles)
    vehicle_2 = random.choice(depot_2.vehicles)
    for i in range(scaled_s):
        if len(vehicle_2.customers) == 0:
            break
        customer_2 = random.choice(vehicle_2.customers)
        if (vehicle_1.capacity - customer_2.demand > 0
            and depot_1.capacity - customer_2.demand > 0
            and len(vehicle_1.customers) > 0):
            new_index = random.choice(range(len(vehicle_1.customers)))
            vehicle_1.customers.insert(new_index, customer_2)
            vehicle_2.customers.remove(customer_2)
        if len(vehicle_2.customers) == 0:
            break
```

Fig. 11.1: Python Implementation of **global_search**

D. Local Search

Local Search optimizes customer assignments and vehicle routing originating from a depot. The process begins by selecting a random depot from the existing solution. The algorithm then rearranges or swaps customer stops within a vehicle's route to minimize overall travel distance. Based on the new assignments, it also dynamically adjusts vehicle capacities to ensure efficiency. This approach enhances the efficiency of the routes and maintains the integrity of the depot assignments, resulting in a more streamlined operation.

```
1  FUNCTION local_search(solution, scaled_step):
2      FOR i ← 1 TO scaled_step:
3          depot ← SelectRandomDepot(solution)
4          vehicle ← SelectRandomVehicle(depot)
5          WHILE vehicle HAS customers:
6              customer1, customer2 ← SelectTwoRandomCustomers(vehicle)
7              IF SwappingImprovesRoute(customer1, customer2, vehicle):
8                  SwapCustomersInRoute(customer1, customer2, vehicle)
9      RETURN solution
```

Fig. 12: Pseudocode for **local_search**

```
def local_search(solution: Solution, scaled_s):  1 usage
    for i in range(scaled_s):
        depot = random.choice(solution.depots)
        vehicle = random.choice(depot.vehicles)
        if len(vehicle.customers) == 0:
            continue
        customer_1 = random.choice(vehicle.customers)
        customer_2 = random.choice(vehicle.customers)
        if len(vehicle.customers) == 1:
            continue
        # Do not accept same customer
        while customer_1 == customer_2:
            customer_1 = random.choice(vehicle.customers)
            customer_2 = random.choice(vehicle.customers)
        index_1 = vehicle.customers.index(customer_1)
        index_2 = vehicle.customers.index(customer_2)
        vehicle.customers[index_1] = customer_2
        vehicle.customers[index_2] = customer_1
```

Fig. 12.1: Python Implementation of **local_search**

E. Abandonment Strategy

The abandonment mechanism is pivotal in maintaining diversity within the solution pool, which is vital for preventing premature convergence. By replacing a certain percentage of the current solutions, referred to as (pa), with new random configurations, the algorithm introduces new possibilities. These new configurations can include entirely different depot setups alongside randomized customer assignments, contributing to a richer solution landscape. This variety helps the algorithm escape local optima by exploring regions that have yet to be visited. In the context of Location Routing Problems (LRPs), this strategy enables the Cuckoo Search Algorithm (CSA) to discover depot and route configurations that might not emerge from straightforward iterative improvements alone, thus enhancing its overall effectiveness in finding optimal solutions.

```
ApplyAbandonment(solution, probability):
    IF RandomValue() < probability:
        solution ← GenerateRandomSolution(solution)
    RETURN solution

FUNCTION create_random_solution(original_solution):
    // Initialize and reset the solution
    new_solution ← DeepCopy(original_solution)
    customer_pool ← Copy(original_solution.customers)

    // Clear all depot and vehicle assignments
    FOR EACH depot IN new_solution.depots:
        depot.assigned_customers ← []
        depot.vehicles ← []

    // Assign at least one customer to each depot
    FOR EACH depot IN new_solution.depots:
        random_customer ← SelectRandomCustomer(customer_pool)
        AssignCustomerToDepot(random_customer, depot)
        customer_pool.REMOVE(random_customer)

    // Assign remaining customers randomly to depots
    FOR EACH customer IN customer_pool:
        WHILE TRUE:
            random_depot ← SelectRandomDepot(new_solution.depots)
            IF DepotHasCapacity(random_depot, customer):
                AssignCustomerToDepot(customer, random_depot)
                BREAK

    // Define vehicle routes for the new solution
    FOR EACH depot IN new_solution.depots:
        CreateVehicleRoutes(depot)

    // Calculate the fitness of the new solution
    CalculateDepotAndRouteDistances(new_solution)
    RETURN new_solution
```

Fig. 13: Pseudocode for Abandonment Strategy and the **create_random_solution** function

```python
def create_random_solution(solution: Solution):  1 usage
    solution = copy.deepcopy(solution)
    customer_pool = copy.copy(solution.customers)
    if len(customer_pool) == 0:
        pass
    for depot in solution.depots:
        depot.assigned_customers = []
        depot.vehicles = []
    # Ensure all depots have at least 1 customer
    for depot in solution.depots:
        random_customer = random.choice(customer_pool)
        depot.assigned_customers.append(random_customer)
        customer_pool.remove(random_customer)
    # Assign customers
    for customer in customer_pool:
        random_depot = random.choice(solution.depots)
        if random_depot.capacity - customer.demand > 0:
            random_depot.assigned_customers.append(customer)
        else:
            while True:
                random_depot = random.choice(solution.depots)
                if random_depot.capacity - customer.demand > 0:
                    random_depot.assigned_customers.append(customer)
                    break
    for depot in solution.depots:
        vehicle_number = 0
        vehicle = VehicleRoute(vehicle_number, depot)
        depot.vehicles.append(vehicle)
        for customer in depot.assigned_customers:
            if vehicle.capacity - customer.demand > 0:
                vehicle.customers.append(customer)
                depot.assigned_customers.remove(customer)
            else:
                vehicle_number += 1
                vehicle = VehicleRoute(vehicle_number, depot)
                vehicle.customers.append(customer)
                depot.vehicles.append(vehicle)
                depot.assigned_customers.remove(customer)
    for depot in solution.depots:
        for vehicle in depot.vehicles:
            vehicle.create_vector()
            vehicle.calculate_vehicle_distance(reset=True)
    solution.calculate_total_distance(reset=True)
    return solution
```

Fig. 13.1: Python Implementation of **create_random_solution**

Main Cuckoo Search Algorithm Workflow

```
FUNCTION apply_cuckoo(solutions, max_iterations, beta, pa):
    threshold ← CalculateThreshold(beta, num_samples)  // Compute Lévy threshold
    FOR iteration ← 1 TO max_iterations:
        FOR EACH solution IN solutions:
            step ← LevyFlightStep(beta)
            scaled_step ← ScaleStep(step)
            IF step ≥ threshold:  // Perform global search
                solution ← GlobalSearch(solution, scaled_step)
            ELSE:  // Perform local search
                solution ← LocalSearch(solution, scaled_step)
            solution ← ApplyAbandonment(solution, pa)
        best_solution ← FindBestSolution(solutions)
    RETURN best_solution
```

Fig. 14: Pseudocode for the Cuckoo Search Algorithm

```python
def apply_cuckoo(solution, threshold, minimum_solution):  1 usage
    s = abs(levy_flight_step())
    scaled_s = ceil(alpha * s)
    temp_solution = copy.deepcopy(solution)
    if s >= threshold:
        global_search(temp_solution, scaled_s)
    else:
        local_search(temp_solution, scaled_s)
    for depot in temp_solution.depots:
        for vehicle in depot.vehicles:
            vehicle.calculate_vehicle_distance(reset=True)
    temp_solution.calculate_total_distance(reset=True)
    if solution.total_distance > temp_solution.total_distance:
        solution = temp_solution
    # Check abandonment
    random_number = random.random()
    if random_number < pa:
        solution = create_random_solution(solution)
    if minimum_solution.total_distance > solution.total_distance:
        minimum_solution = solution
    return solution, minimum_solution
```

Fig. 14.1: Python Implementation of the Cuckoo Search Algorithm

5.4 Challenges and Solutions

The implementation of the Cuckoo Search Algorithm (CSA) to address the Location Routing Problem (LRP) necessitated a comprehensive approach to tackle various challenges associated with computational efficiency, logical coherence, and practical applicability. This section delineates the key issues encountered during the development phase and the methodologies employed to effectively address these challenges.

One of the main challenges encountered was managing large and complex datasets, particularly regarding memory usage and computational efficiency. Some datasets contained a significant number of depots and customers, which led to considerable memory demands for storing distances and the various states of solutions. Furthermore, the practice of using deep copies to create separate instances of solutions further drained resources, sometimes resulting in runtime errors. To tackle these issues, strategies to optimize memory usage were implemented by avoiding redundant deep copies. This was achieved by carefully isolating mutable components, like distance metrics, while reusing immutable data structures. Additionally, the solution initialization process was streamlined to minimize unnecessary operations, thus reducing the overall computational overhead linked to the algorithm.

The success of the Cuckoo Search Algorithm (CSA) depends on achieving the right balance between exploring the entire solution space to prevent getting trapped in local optima and refining the best available solutions. However, adjusting the parameters for Lévy flights, along with the abandonment probability (pa), has proven to be quite challenging. If these parameters are not set correctly, the search process can either stall or fail to converge effectively. To address this issue, a sensitivity analysis was conducted to optimize these parameters. Through multiple test runs across various datasets, optimal ranges were identified, ensuring consistent performance while preventing overfitting to specific instances.

The iterative nature of the Cuckoo Search Algorithm (CSA), combined with the large number of solutions and the size of the dataset, led to significant computational demands. The challenge arose from generating numerous initial solutions and refining them through both global and local searches, which placed a considerable strain on processing power. Additionally, the requirement for real-time distance calculations added another layer of complexity. To address these challenges, a strategy of precomputing and caching the distances between customers and depots was formed, which significantly reduced redundant calculations. Furthermore, parallel processing was utilized where feasible, distributing tasks across multiple processing cores to enhance overall performance and improve runtime efficiency.

To ensure the accuracy of the Cuckoo Search Algorithm (CSA), thorough debugging and validation was implemented. Several challenges were encountered, particularly with

logical errors related to the deep copy mechanism and route adjustments, which proved difficult to trace in large datasets. Minor mistakes in customer assignments or depot capacities often escalated, leading to misleading outcomes. To resolve these issues, debugging tools were used to track the intermediate states of our solutions, allowing for step-by-step validation throughout the process. Additionally, unit tests for critical functions such as **assign_depots** and **define_vehicle_routes** were developed, which helped confirm their accuracy and reliability. This systematic approach enabled us to enhance the effectiveness of the algorithm and ensure robust results.

Finally, incorporating realistic logistics constraints added complexity to the algorithm. Real-world logistics frequently include additional elements such as time restrictions, various types of vehicles, and fluctuating demands that were not addressed in this implementation. Although these factors were excluded, the modular structure of the Cuckoo Search Algorithm (CSA) permits future enhancements to manage these constraints. This existing framework provides a flexible foundation for addressing more complex logistics challenges as they arise.

CHAPTER 6: RESULTS

6.1 Datasets and Experimental Setup

For this research, a variety of datasets focusing on Location-Routing Problems (LRP) were utilized, sourced from well-established benchmarks in the literature. These datasets provide essential information formatted for effective computational analysis, including customer files that list the customer number, X-coordinate, Y-coordinate, and demand, as well as depot files that detail the depot number, X-coordinate, Y-coordinate, capacity, and fixed cost. The datasets were adapted from foundational sources like Christofides et al. (1969), Daskin (1995), and Gaskell (1967), each containing unique configurations that range from smaller problems with a limited number of depots and customers to more complex scenarios featuring higher dimensionality and varying constraints. Notable examples of the datasets included in the analysis are Christofides69-50x5, which consists of 50 customers and 5 depots and emphasizes regional delivery planning; Daskin95-150x10, a larger problem with 150 customers and 10 depots suitable for testing scalability; and Perl83-318x4, a high-dimensional dataset with 318 customers and 4 depots, often used to evaluate algorithmic efficiency under significant constraints. These diverse datasets allow for robust testing of the initial Nearest Neighbor Algorithm (NNA) and the Cuckoo Search Algorithm (CSA), providing real-world scenarios that comprehensively evaluate performance metrics, including total travel distance and depot utilization.

Instance		Veh. Cap.
1	Christofides69-50x5	160
2	Christofides69-75x10	140
3	Christofides69-100x10	200
4	Daskin95-88x8	9000000
5	Daskin95-150x10	8000000
6	Gaskell67-21x5	6000
7	Gaskell67-22x5	4500
8	Gaskell67-29x5	4500
9	Gaskell67-32x5	8000
10	Gaskell67-32x5	11000
11	Gaskell67-36x5	250
12	Min92-27x5	2500
13	Min92-134x8	850
14	Perl83-12x2	140
15	Perl83-55x15	120
16	Perl83-85x7	160
17	Perl83-318x4	25000
18	Perl83-318x4	8000
19	Or76-117x14	150

Fig. 15: LRP Instances

The experiments utilized a structured methodology to assess the performance of the Cuckoo Search Algorithm (CSA). Initially, the Nearest Neighbor Algorithm (NNA) generated starting solutions for each dataset, resulting in 20 unique initial solutions per dataset. This was accomplished by varying the number of open depots, which provided a diverse range of starting points for the CSA. The optimization process involved applying the CSA 100 times to each initial solution, gradually refining depot and route configurations. Parameters were fine-tuned based on preliminary tests to achieve an optimal balance between exploration and exploitation. For evaluation, the focus was on

the total distance traveled by vehicles, accounting for both the travel distance and fixed costs associated with the depots.

The experiments were conducted on an Apple MacBook Air equipped with an Apple M2 processor, which features an 8-core CPU and 8GB of Unified RAM. The system operated on macOS, and the programming was executed using Python 3.9. Several Python libraries were employed in the implementation, including NumPy for numerical calculations, Matplotlib for visualizing results, Copy for managing deep copies of solution states, Random for adding randomness to the algorithm, Itertools for efficient iteration and combination generation, and Typing for clear type hinting. This configuration provided a robust platform for executing the iterative processes of the Cuckoo Search Algorithm and effectively analyzing the resulting solutions across various datasets.

6.2 Performance Analysis

1. **Dataset: Christofides69-50x5**

This case involves 50 customer nodes and 5 depot nodes. The depots are evenly spread throughout the area, while customer demands vary, presenting a moderate challenge in optimizing routing and depot selection.

After applying the Cuckoo Search Algorithm (CSA), the following depots were selected as open: ['Depot 3', 'Depot 4', 'Depot 5'] (Fig. 16.1). The total distance covered by all vehicles, including fixed costs for the selected depots, was 631.5848730381724 units. Interestingly, in this case, the Cuckoo Search Algorithm (CSA) did not outperform the solution produced by the Nearest Neighbor Algorithm (NNA). This result underscores how the performance can vary depending on the specific characteristics of the dataset and how important it is to fine-tune the parameters for optimization. **[26]**

```
Solution Total Distance: 631.5848730381724
Active Depots:
['Depot 3', 'Depot 4', 'Depot 5']
Depot ID: 3
    Vehicle ID: 0
        Vehicle 0 goes through [13, 41, 19, 42, 17, 4, 18, 47, 14, 25, 24, 43, 40]
Total Vehicle distance: 177.364958197381
Depot ID: 4
    Vehicle ID: 0
        Vehicle 0 goes through [10, 49, 9, 50, 34, 30, 39, 33, 45, 15, 44, 37, 5]
Total Vehicle distance: 113.63104753843953
Depot ID: 5
    Vehicle ID: 0
        Vehicle 0 goes through [22, 1, 32, 11, 38, 16, 2, 29, 21, 20, 35, 36, 3, 28, 31, 8, 26, 7, 23, 48, 27, 6, 46, 12]
Total Vehicle distance: 220.58886730235184
```

Fig. 16: Solution for Christofides69-50x5

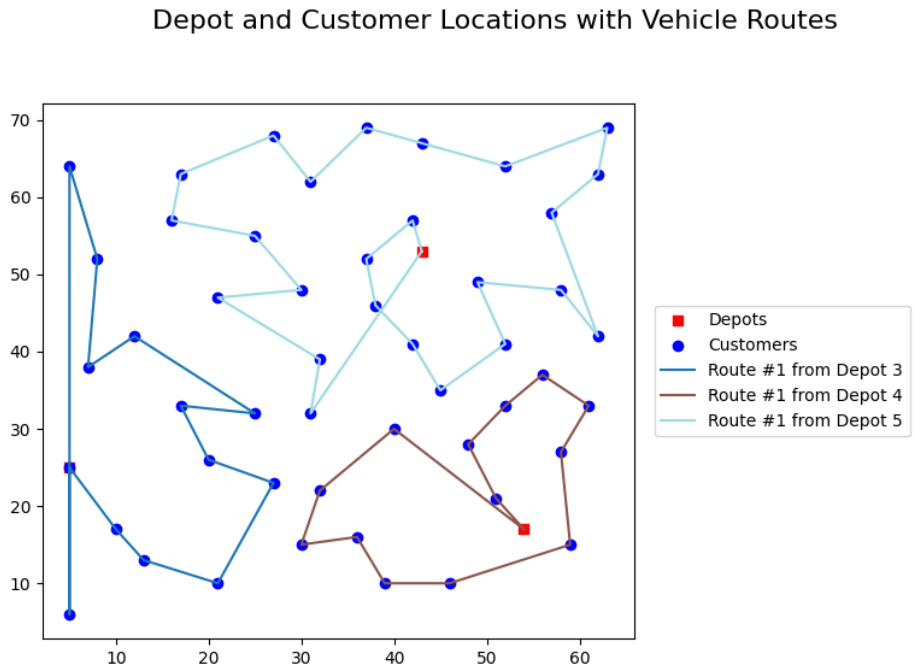

Fig. 16.1: Graph for Christofides69-50x5

2. Christofides69-75x10

This dataset includes 75 customer locations and 10 depot locations spread across the region. The depots are strategically positioned to meet a wide range of customer needs. As the number of depots and customers increases, the optimization challenge becomes more complex, requiring a thoughtful approach to depot selection and effective vehicle route planning.

After applying the Cuckoo Search Algorithm (CSA), the following depots were selected as open: ['Depot 1', 'Depot 2', 'Depot 3', 'Depot 4', 'Depot 5', 'Depot 7', 'Depot 8', 'Depot 9'] (Fig. 17.1). The total distance covered by all vehicles, including fixed costs for the selected depots, was 946.3761140258465 units. In this case, CSA showed improvement over the original Nearest Neighbor Algorithm (NNA) solution. This highlights the algorithm's effectiveness in optimizing depot and route configurations, particularly when handling larger datasets. **[26]**

```
Solution Total Distance: 946.3761140258465
Active Depots:
['Depot 1', 'Depot 2', 'Depot 3', 'Depot 4', 'Depot 5', 'Depot 7', 'Depot 8', 'Depot 9']
Depot ID: 1
    Vehicle ID: 0
        Vehicle 0 goes through [67, 34, 46, 8, 35, 7, 26, 12, 40, 17, 51, 6, 68, 75, 4, 45, 29, 27, 52, 30, 2, 73, 33, 63, 16, 3, 44, 32, 24, 49, 23, 56, 1]
Total Vehicle distance: 253.27877950198382
Depot ID: 2
    Vehicle ID: 0
        Vehicle 0 goes through [71, 47, 48, 36]
Total Vehicle distance: 26.065621957406403
Depot ID: 3
    Vehicle ID: 0
        Vehicle 0 goes through [10, 58, 72, 39, 9, 25, 50, 18, 55, 31]
Total Vehicle distance: 90.38980333509583
Depot ID: 4
    Vehicle ID: 0
        Vehicle 0 goes through [37, 20, 70, 60, 5]
Total Vehicle distance: 43.065392793884044
Depot ID: 5
    Vehicle ID: 0
        Vehicle 0 goes through [69, 21, 74, 28, 62, 22, 42, 41, 43, 64, 61]
Total Vehicle distance: 97.92334776675432
Depot ID: 7
    Vehicle ID: 0
        Vehicle 0 goes through [57, 15, 13, 54]
Total Vehicle distance: 44.3183470878823
Depot ID: 8
    Vehicle ID: 0
        Vehicle 0 goes through [59, 14, 19]
Total Vehicle distance: 26.927100720401782
Depot ID: 9
    Vehicle ID: 0
        Vehicle 0 goes through [65, 38, 11, 66, 53]
Total Vehicle distance: 44.40772086243801
```

Fig. 17: Solution for Christofides69-75x10

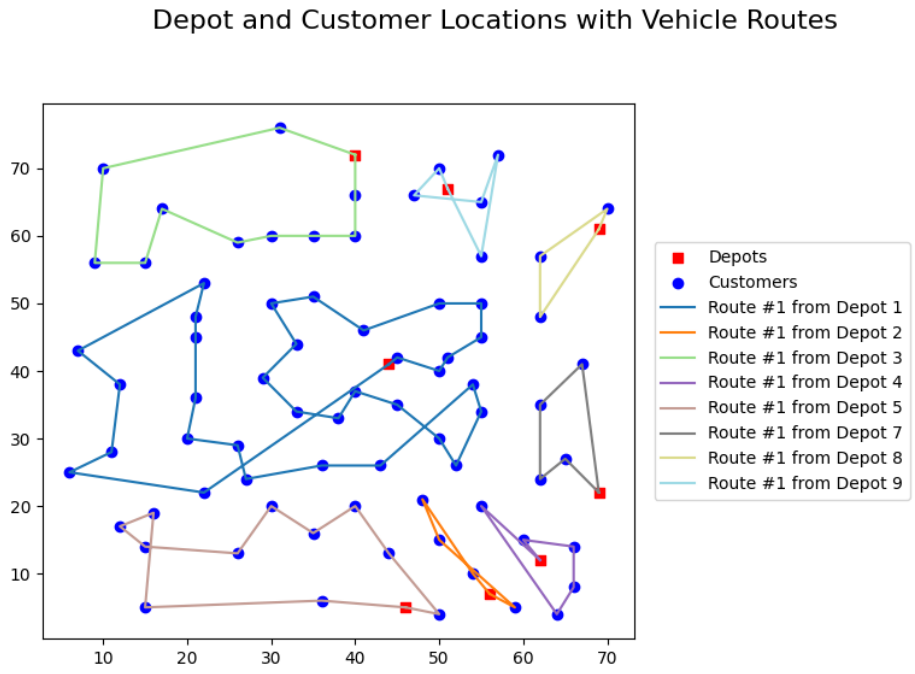

Fig. 17.1: Graph for Christofides69-75x10

3. Christofides69-100x10

This dataset includes 100 customer locations and 10 depot locations, creating a moderately large and complex logistics challenge. The depots are spread throughout the area, and customers' varying demands necessitate efficient strategies for choosing depots and routing vehicles effectively.

After applying the Cuckoo Search Algorithm, the following depots were selected as open: ['Depot 1', 'Depot 2', 'Depot 3', 'Depot 4', 'Depot 5', 'Depot 7', 'Depot 8', 'Depot 9'] (Fig 18.1). The total distance covered by all vehicles, including fixed costs for the selected depots, was 1102.3914979904373 units. In this case, CSA showed improvement over the original Nearest Neighbor Algorithm (NNA) solution. This highlights the algorithm's effectiveness in optimizing depot and route configurations, particularly when handling larger datasets. **[26]**

```
Solution Total Distance: 1102.3914979904373
Active Depots:
['Depot 1', 'Depot 2', 'Depot 3', 'Depot 4', 'Depot 5', 'Depot 7', 'Depot 8', 'Depot 9']
Depot ID: 1
    Vehicle ID: 0
        Vehicle 0 goes through [48, 82, 7, 62, 11, 19, 47, 36, 49, 64, 46]
Total Vehicle distance: 112.4181312389595
Depot ID: 2
    Vehicle ID: 0
        Vehicle 0 goes through [52, 60, 18]
Total Vehicle distance: 14.342617547667329
Depot ID: 3
    Vehicle ID: 0
        Vehicle 0 goes through [45, 8, 83, 84, 5, 61, 16, 91, 85, 93, 100, 44, 14, 38, 86, 17]
Total Vehicle distance: 104.83937003684167
Depot ID: 4
    Vehicle ID: 0
        Vehicle 0 goes through [76, 77, 3, 79, 33, 81, 78, 34, 68, 50, 28]
Total Vehicle distance: 77.8415042711998
Depot ID: 5
    Vehicle ID: 0
        Vehicle 0 goes through [89, 6, 94, 95, 97, 92, 59, 99, 96, 98, 37, 87, 2, 57, 15, 43, 42, 13, 58, 40, 53, 27]
Total Vehicle distance: 124.84284995602677
Depot ID: 7
    Vehicle ID: 0
        Vehicle 0 goes through [66, 20, 51, 9, 71, 35, 65, 32]
Total Vehicle distance: 89.25830152923496
Depot ID: 8
    Vehicle ID: 0
        Vehicle 0 goes through [54, 80, 12, 26, 21, 73, 72, 74, 22, 75, 56, 39, 23, 67, 25, 55, 4, 24, 29, 41]
Total Vehicle distance: 195.79334059029264
Depot ID: 9
    Vehicle ID: 0
        Vehicle 0 goes through [31, 88, 10, 90, 63, 30, 70, 1, 69]
Total Vehicle distance: 63.05538282021469
```

Fig. 18: Solution for Christofides69-100x10

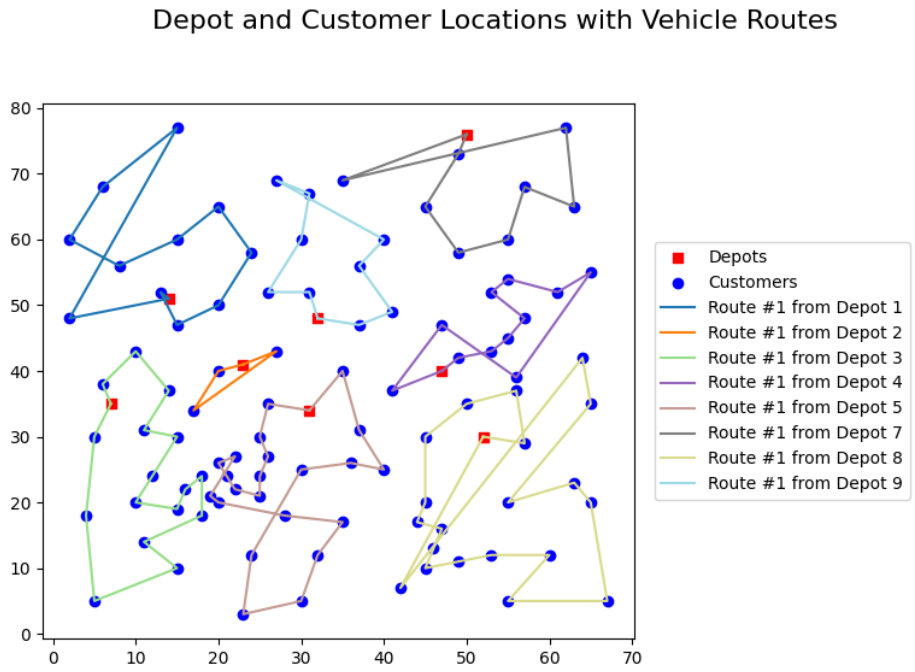

Fig. 18.1: Graph for Christofides69-100x10

4. Daskin95-88x8

This dataset consists of 88 customer locations and 8 depot locations, presenting a moderately large and complex logistics challenge. The depots are distributed across the area, and customers' varying demands require efficient strategies for selecting depots and routing vehicles effectively.

After applying the Cuckoo Search Algorithm, the following depots were selected to remain open: ['Depot 1', 'Depot 2', 'Depot 3', 'Depot 4', 'Depot 5', 'Depot 7'] (Fig 19.1). The total distance covered by all vehicles, including the fixed costs for the selected depots, was 971.4671734606933 units. In this instance, the Cuckoo Search Algorithm (CSA) did not surpass the solution yielded by the Nearest Neighbor Algorithm (NNA). **[27]**

```
Solution Total Distance: 971.4671734606933
Active Depots:
['Depot 1', 'Depot 2', 'Depot 3', 'Depot 4', 'Depot 5', 'Depot 7']
Depot ID: 1
    Vehicle ID: 0
        Vehicle 0 goes through [1, 63, 70, 88, 79, 20, 61, 86]
Total Vehicle distance: 12.89292222699217
Depot ID: 2
    Vehicle ID: 0
        Vehicle 0 goes through [2, 32, 6, 46, 11, 39, 14, 41, 78, 65, 62, 85, 76, 74, 38, 33, 9, 68, 30, 81, 21]
Total Vehicle distance: 87.55303381510247
Depot ID: 3
    Vehicle ID: 0
        Vehicle 0 goes through [3, 17, 58, 69, 34, 80, 31, 67, 47, 57, 56, 42, 51, 87, 77, 13, 25]
Total Vehicle distance: 57.478991965701724
Depot ID: 4
    Vehicle ID: 0
        Vehicle 0 goes through [4, 27, 10, 8, 28, 29, 43, 50, 60, 18, 55, 24, 52, 59, 66, 22, 26]
Total Vehicle distance: 78.93421268300206
Depot ID: 5
    Vehicle ID: 0
        Vehicle 0 goes through [5, 72, 83, 12, 19, 82, 54, 37, 53, 75, 45]
Total Vehicle distance: 41.64673055050901
Depot ID: 7
    Vehicle ID: 0
        Vehicle 0 goes through [7, 48, 64, 16, 23, 40, 73, 84, 44, 36, 71, 35, 15, 49]
Total Vehicle distance: 47.161282219385726
```

Fig. 19: Solution for Daskin95-88x8

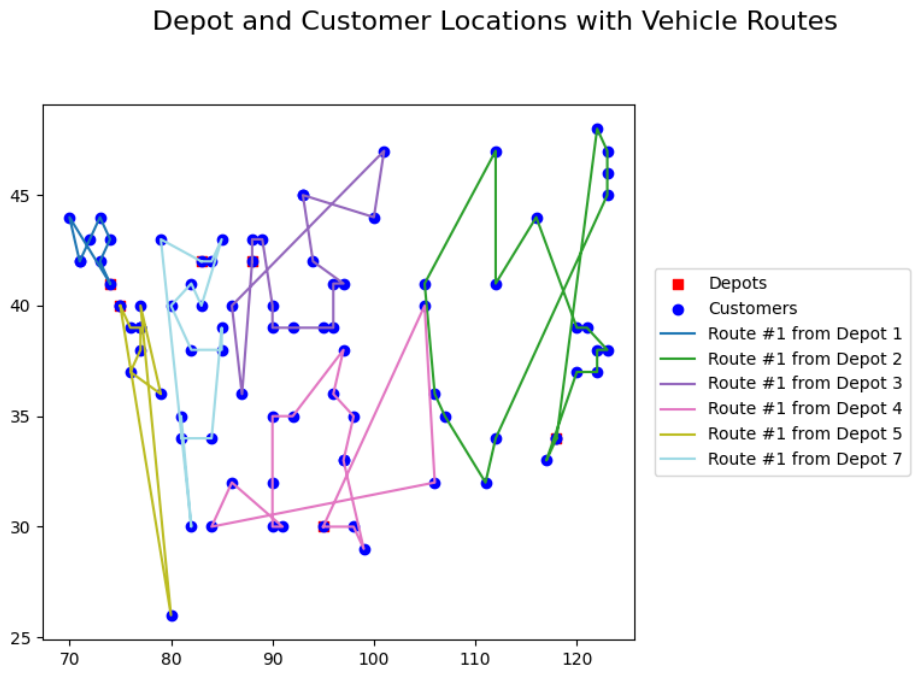

Fig. 19.1: Graph for Daskin95-88x8

5. Daskin95-150x10

This dataset consists of 150 customer locations and 10 depot locations, presenting a moderately large and complex logistics challenge. The depots are distributed across the area, and customers' varying demands require efficient strategies for selecting depots and routing vehicles effectively.

After applying the Cuckoo Search Algorithm, the following depots were selected to remain open: ['Depot 1', 'Depot 2', 'Depot 3', 'Depot 5', 'Depot 6', 'Depot 7', 'Depot 8', 'Depot 9', 'Depot 10'] (Fig 20.1). The total distance covered by all vehicles, including the fixed costs for the selected depots, was 74269.90350693598 units. In this instance, the Cuckoo Search Algorithm (CSA) did not surpass the solution yielded by the Nearest Neighbor Algorithm (NNA). **[27]**

```
Solution Total Distance: 74269.90350693598
Active Depots:
['Depot 1', 'Depot 2', 'Depot 3', 'Depot 5', 'Depot 6', 'Depot 7', 'Depot 8', 'Depot 9', 'Depot 10']
Depot ID: 1
    Vehicle ID: 0
        Vehicle 0 goes through [1, 114, 124, 51, 79, 98, 137, 93]
Total Vehicle distance: 945.1261420800324
    Vehicle ID: 1
        Vehicle 1 goes through [70, 84, 111, 67, 48, 76, 54, 87, 15, 75, 73, 149, 56, 28, 96, 13, 148, 43, 147, 145, 125]
Total Vehicle distance: 4102.68519411002
Depot ID: 2
    Vehicle ID: 0
        Vehicle 0 goes through [2, 49, 95, 50, 99, 109, 141, 12, 101, 42]
Total Vehicle distance: 1722.9492551098488
Depot ID: 3
    Vehicle ID: 0
        Vehicle 0 goes through [3, 136, 69, 132, 86, 62, 119, 143, 78, 134, 60, 100, 37, 71, 24, 150, 36, 91, 77, 103, 61, 117]
Total Vehicle distance: 6357.984198363996
Depot ID: 5
    Vehicle ID: 0
        Vehicle 0 goes through [5, 115, 17, 46, 31, 44, 23, 107, 140, 130, 104, 131, 45, 80]
Total Vehicle distance: 2205.3774584831617
Depot ID: 6
    Vehicle ID: 0
        Vehicle 0 goes through [6, 122, 108, 38, 68, 90, 32, 128, 55, 81, 30, 116, 39, 35, 106, 88, 16, 72, 59, 89, 144, 26, 139, 40, 102, 138, 118, 41, 20, 133]
Total Vehicle distance: 3856.682075307462
Depot ID: 7
    Vehicle ID: 0
        Vehicle 0 goes through [7, 113, 121, 67, 127, 52, 21, 146]
Total Vehicle distance: 2957.366704389103
Depot ID: 8
    Vehicle ID: 0
        Vehicle 0 goes through [8, 105, 126, 63, 22, 29, 53, 120, 110, 19]
Total Vehicle distance: 2266.4901475653837
Depot ID: 9
    Vehicle ID: 0
        Vehicle 0 goes through [9, 142, 94, 123, 83, 57, 58, 4, 14, 25, 64, 92, 27, 18, 65, 66, 129, 85, 82, 74, 33, 97, 11]
Total Vehicle distance: 3975.904646101435
Depot ID: 10
    Vehicle ID: 0
        Vehicle 0 goes through [10, 135, 112, 34]
```

Fig. 20: Solution for Daskin95-150x10

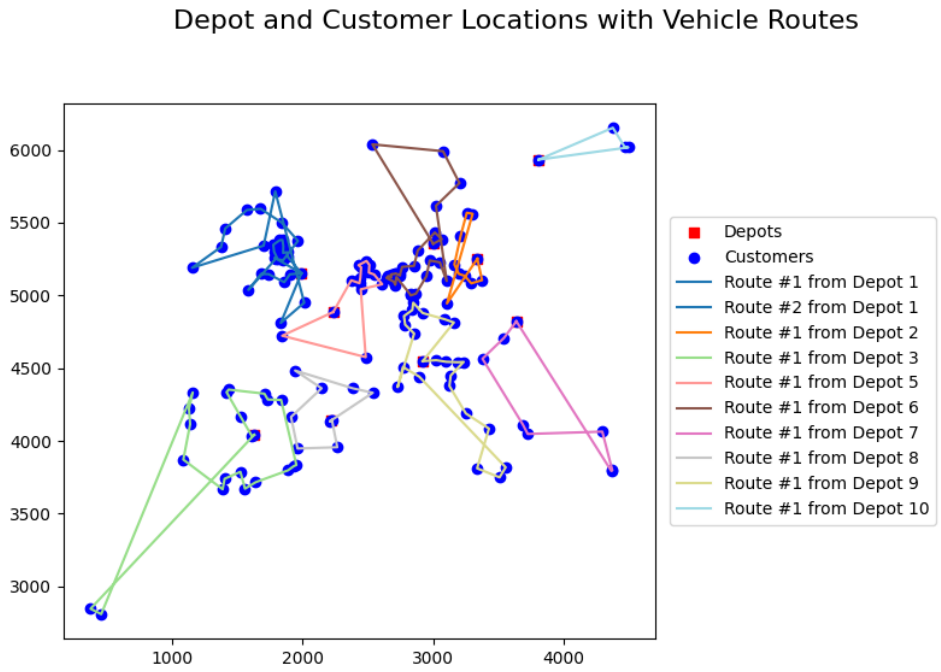

Fig. 20.1: Graph for Daskin95-150x10

6. Gaskell67-21x5

This dataset consists of 21 customer locations and 5 depot locations, presenting a moderately large and complex logistics challenge. The depots are distributed across the area, and customers' varying demands require efficient strategies for selecting depots and routing vehicles effectively.

After applying the Cuckoo Search Algorithm, the following depots were selected to remain open: ['Depot 1', 'Depot 2', 'Depot 4', 'Depot 5'] (Fig. 21.1). The total distance covered by all vehicles, including the fixed costs for the selected depots, was 438.8253409538019 units. In this case, CSA showed improvement over the original Nearest Neighbor Algorithm (NNA) solution. **[28]**

```
Solution Total Distance: 438.8253409538019
Active Depots:
['Depot 1', 'Depot 2', 'Depot 4', 'Depot 5']
Depot ID: 1
    Vehicle ID: 0
        Vehicle 0 goes through [21, 18, 20, 17]
Total Vehicle distance: 46.7699809718173
Depot ID: 2
    Vehicle ID: 0
        Vehicle 0 goes through [8, 11, 10, 9, 7, 5, 2, 1, 3, 4, 6]
Total Vehicle distance: 127.14750348509804
Depot ID: 4
    Vehicle ID: 0
        Vehicle 0 goes through [16, 15, 12, 14]
Total Vehicle distance: 39.81621238266169
Depot ID: 5
    Vehicle ID: 0
        Vehicle 0 goes through [13, 19]
Total Vehicle distance: 25.091644114224952
```

Fig. 21: Solution for Gaskell67-21x5

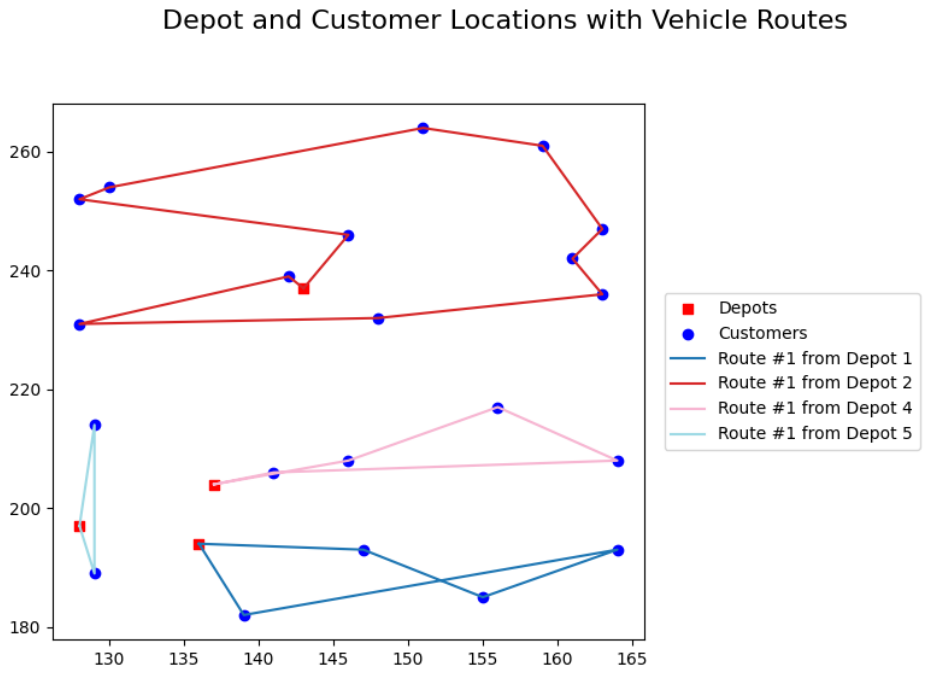

Fig. 21.1: Graph for Gaskell67-21x5

7. Gaskell67-22x5

This dataset consists of 22 customer locations and 5 depot locations, presenting a moderately large and complex logistics challenge. The depots are distributed across the area, and customers' varying demands require efficient strategies for effectively selecting depots and routing vehicles.

After applying the Cuckoo Search Algorithm, the following depots were selected to remain open: ['Depot 1', 'Depot 2', 'Depot 4', 'Depot 5'] (Fig. 22.1). The total distance covered by all vehicles, including the fixed costs for the selected depots, was 619.4042713383195 units. In this case, CSA showed improvement over the original Nearest Neighbor Algorithm (NNA) solution. This highlights the algorithm's effectiveness in optimizing depot and route configurations, particularly when handling larger datasets. **[28]**

```
Solution Total Distance: 619.4042713383195
Active Depots:
['Depot 1', 'Depot 2', 'Depot 4', 'Depot 5']
Depot ID: 1
    Vehicle ID: 0
        Vehicle 0 goes through [7, 9, 10, 13, 6, 1, 11, 12]
Total Vehicle distance: 122.3713623641716
Depot ID: 2
    Vehicle ID: 0
        Vehicle 0 goes through [18, 19]
Total Vehicle distance: 35.03876636398556
Depot ID: 4
    Vehicle ID: 0
        Vehicle 0 goes through [21, 5, 4, 8]
Total Vehicle distance: 72.19343888075522
Depot ID: 5
    Vehicle ID: 0
        Vehicle 0 goes through [17, 2, 16, 3, 15, 14, 20, 22]
Total Vehicle distance: 189.8007037294071
```

Fig. 22: Solution for Gaskell67-22x5

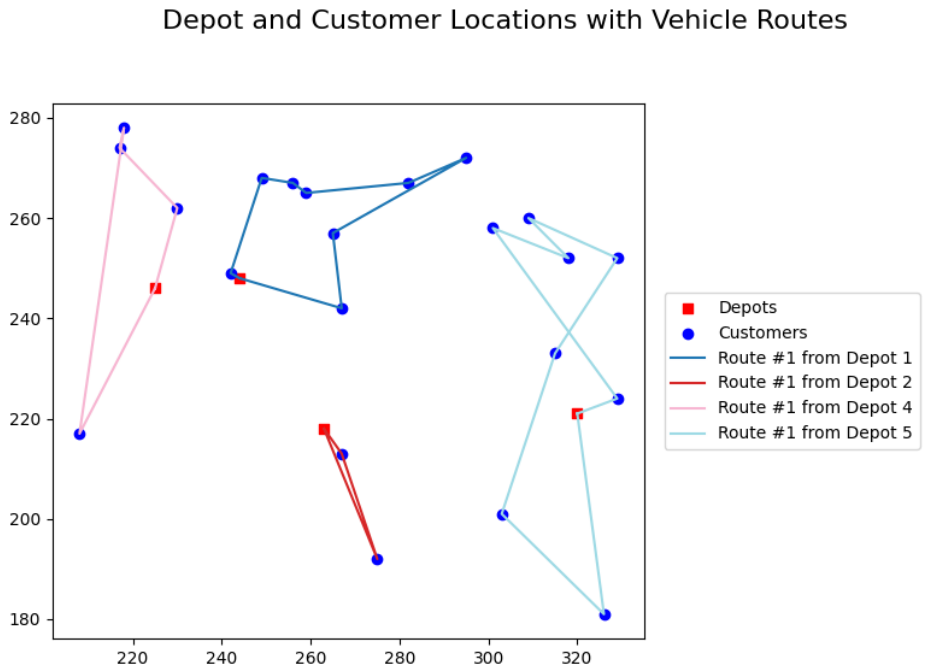

Fig. 22.1: Graph for Gaskell67-22x5

8. Gaskell67-29x5

This dataset consists of 29 customer locations and 5 depot locations, presenting a moderately large and complex logistics challenge. The depots are distributed across the area, and customers' varying demands require efficient strategies for effectively selecting depots and routing vehicles.

After applying the Cuckoo Search Algorithm, the following depots were selected to remain open: ['Depot 1', 'Depot 2', 'Depot 3', 'Depot 4'] (Fig. 23.1). The total distance covered by all vehicles, including the fixed costs for the selected depots, was 560.9411665792288 units. In this instance, the Cuckoo Search Algorithm (CSA) did not surpass the solution yielded by the Nearest Neighbor Algorithm (NNA). **[28]**

```
Solution Total Distance: 560.9411665792288
Active Depots:
['Depot 1', 'Depot 2', 'Depot 3', 'Depot 4']
Depot ID: 1
    Vehicle ID: 0
        Vehicle 0 goes through [29, 25, 24, 1, 6, 27, 28]
Total Vehicle distance: 98.66204091063456
Depot ID: 2
    Vehicle ID: 0
        Vehicle 0 goes through [7, 13, 16, 15, 10, 11, 12, 8, 14, 9, 17]
Total Vehicle distance: 75.10509495992657
Depot ID: 3
    Vehicle ID: 0
        Vehicle 0 goes through [22, 20, 19, 18, 23, 21, 3, 4, 5, 2]
Total Vehicle distance: 177.1241550875468
Depot ID: 4
    Vehicle ID: 0
        Vehicle 0 goes through [26]
Total Vehicle distance: 10.04987562112089
```

Fig. 23: Solution for Gaskell67-29x5

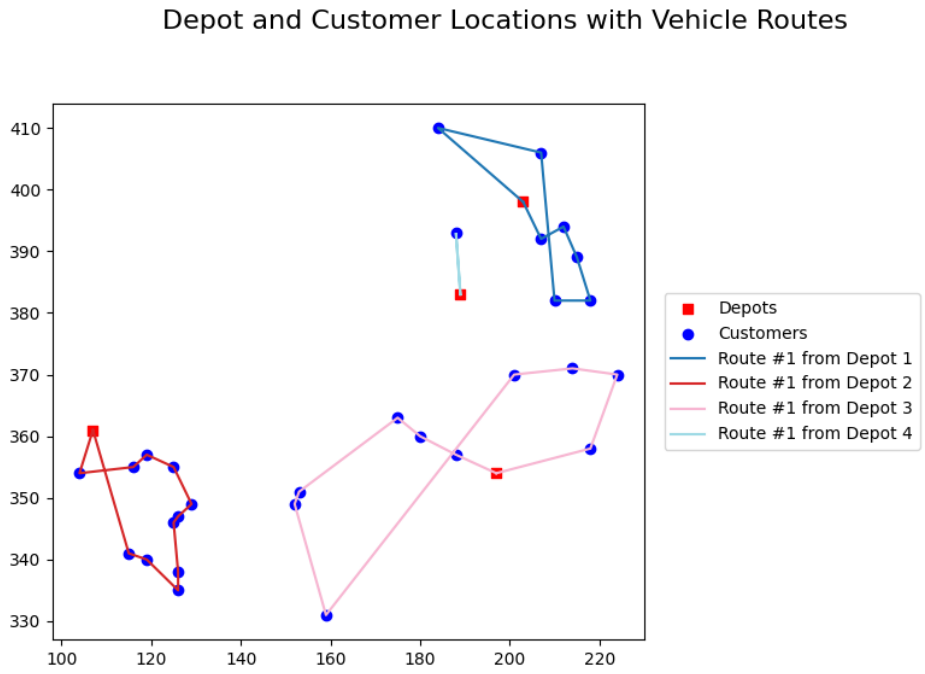

Fig. 23.1: Graph for Gaskell67-29x5

9. Gaskell67-32x5 with 8000 units vehicle capacity

This dataset consists of 32 customer locations and 5 depot locations, and the vehicles have a capacity of 8000 units, presenting a moderately large and complex logistics challenge. The depots are distributed across the area, and customers' varying demands require efficient strategies for effectively selecting depots and routing vehicles.

After applying the Cuckoo Search Algorithm, the following depots were selected to remain open: ['Depot 1', 'Depot 2', 'Depot 5'] (Fig. 24.1). The total distance covered by all vehicles, including the fixed costs for the selected depots, was 573.0818601646255 units. In this case, CSA showed improvement over the original Nearest Neighbor Algorithm (NNA) solution. **[28]**

```
Solution Total Distance: 573.0818601646255
Active Depots:
['Depot 1', 'Depot 2', 'Depot 5']
Depot ID: 1
    Vehicle ID: 0
        Vehicle 0 goes through [30, 4, 3, 31]
Total Vehicle distance: 105.11869830892579
Depot ID: 2
    Vehicle ID: 0
        Vehicle 0 goes through [18, 19, 21, 20, 22, 24, 23, 25, 17, 15, 1, 13, 11, 32, 8, 9, 10, 7, 6, 5, 12, 2]
Total Vehicle distance: 208.87701681156602
Depot ID: 5
    Vehicle ID: 0
        Vehicle 0 goes through [29, 16, 27, 26, 14, 28]
Total Vehicle distance: 109.08614504413369
```

Fig. 24: Solution for Gaskell67-32x5 with 8000 units vehicle capacity

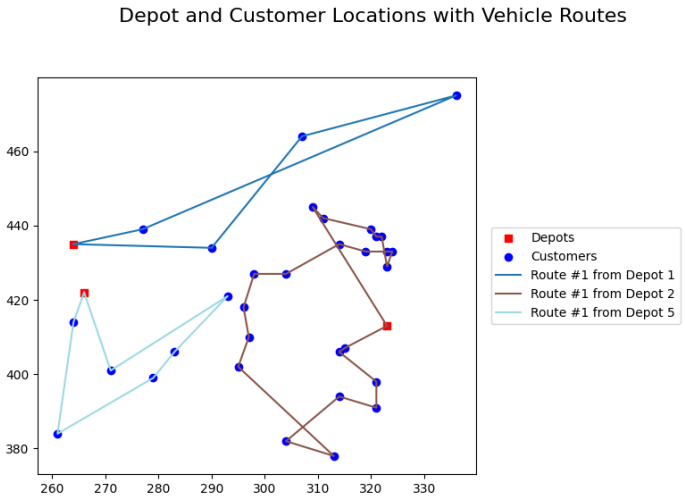

Fig. 24.1: Graph for Gaskell67-32x5 with 8000 units vehicle capacity

10. Gaskell67-32x5 with 11000 units vehicle capacity

This dataset consists of 32 customer locations and 5 depot locations. The vehicles have a capacity of 11,000 units, presenting a moderately large and complex logistics challenge. The depots are distributed across the area, and customers' varying demands require efficient strategies for effectively selecting depots and routing vehicles.

After applying the Cuckoo Search Algorithm, the following depots were selected to remain open: ['Depot 1', 'Depot 2', 'Depot 5'] (Fig. 25.1). The total distance covered by all vehicles, including the fixed costs for the selected depots, was 558.5182683801917 units. In this case, CSA showed significant improvement over the original Nearest Neighbor Algorithm (NNA) solution. **[28]**

```
Solution Total Distance: 558.5182683801917
Active Depots:
['Depot 1', 'Depot 2', 'Depot 5']
Depot ID: 1
    Vehicle ID: 0
        Vehicle 0 goes through [30, 4, 3, 31]
Total Vehicle distance: 105.11869830892579
Depot ID: 2
    Vehicle ID: 0
        Vehicle 0 goes through [18, 19, 21, 20, 22, 24, 23, 25, 17, 15, 1, 13, 11, 32, 8, 9, 10, 7, 6, 5, 12, 2]
Total Vehicle distance: 208.87701681156602
Depot ID: 5
    Vehicle ID: 0
        Vehicle 0 goes through [14, 16, 27, 26, 28, 29]
Total Vehicle distance: 94.52255325969982
```

Fig. 25: Solution for Gaskell67-32x5 with 11000 units vehicle capacity

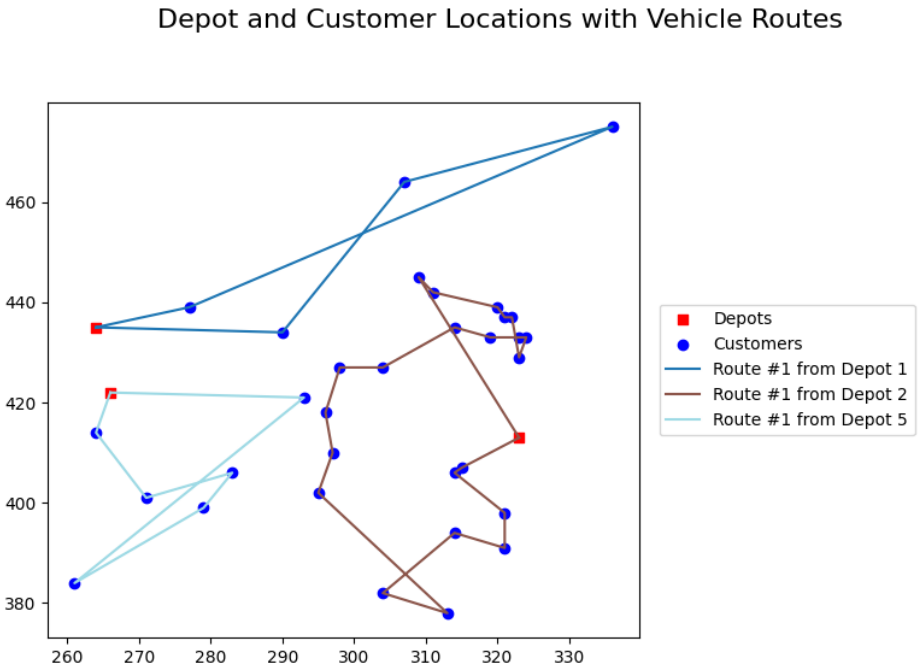

Fig. 25.1: Graph for Gaskell67-32x5 with 11000 units vehicle capacity

11. Gaskell67-36x5

This dataset consists of 32 customer locations, and5 depot locations,.aTe vehicles have a capacity of 250 units, presenting a moderately large and complex logistics challenge. The depots are distributed across the area, and customers' varying demands require efficient strategies for effectively selecting depots and routing vehicles.

After applying the Cuckoo Search Algorithm, the following depots were selected to remain open: ['Depot 1', 'Depot 3', 'Depot 4'] (Fig. 25.1). The total distance covered by all vehicles, including the fixed costs for the selected depots, was 535.9326293198819 units. In this case, CSA showed improvement over the original Nearest Neighbor Algorithm (NNA) solution. **[28]**

```
Solution Total Distance: 535.9326293198819
Active Depots:
['Depot 1', 'Depot 3', 'Depot 4']
Depot ID: 1
    Vehicle ID: 0
        Vehicle 0 goes through [10, 9, 15, 16, 17, 11, 5, 6, 12, 18, 24]
Total Vehicle distance: 125.24099870362662
Depot ID: 3
    Vehicle ID: 0
        Vehicle 0 goes through [4, 2, 1, 7, 8, 3]
Total Vehicle distance: 56.00054941671414
Depot ID: 4
    Vehicle ID: 0
        Vehicle 0 goes through [27, 21, 20, 14, 13, 19, 25, 26, 31, 32, 33, 34, 28, 22, 23, 29, 30, 36, 35]
Total Vehicle distance: 204.69108119954112
```

Fig. 25: Solution for Gaskell67-36x5

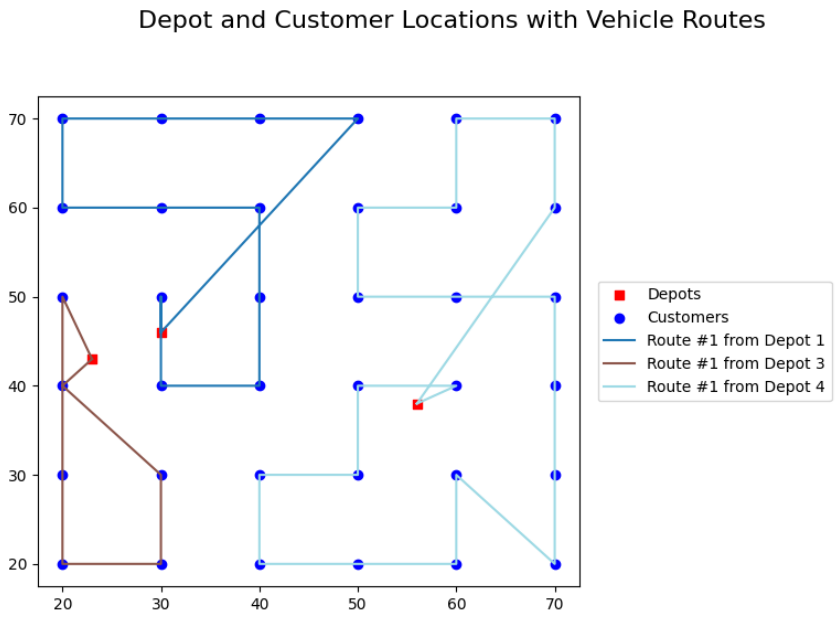

Fig. 25.1: Graph for Gaskell67-36x5

12. Min92-27x5

This dataset consists of 27 customer locations, and 5 depot locations, presenting a moderately large and complex logistics challenge. The depots are distributed across the area, and customers' varying demands require efficient strategies for effectively selecting depots and routing vehicles.

After applying the Cuckoo Search Algorithm, the following depots were selected to remain open: ['Depot 2', 'Depot 3', 'Depot 4', 'Depot 5'] (Fig. 26.1). The total distance covered by all vehicles, including the fixed costs for the selected depots, was 3171.99554109593 units. In this case, CSA showed improvement over the original Nearest Neighbor Algorithm (NNA) solution. **[29]**

```
Solution Total Distance: 3171.99554109593
Active Depots:
['Depot 2', 'Depot 3', 'Depot 4', 'Depot 5']
Depot ID: 2
    Vehicle ID: 0
        Vehicle 0 goes through [4, 6, 5, 15, 14, 13, 12, 18, 19, 20, 24, 16, 17]
Total Vehicle distance: 961.2045180449321
Depot ID: 3
    Vehicle ID: 0
        Vehicle 0 goes through [10, 11, 9, 3, 7, 8, 2, 1]
Total Vehicle distance: 401.9891790995864
Depot ID: 4
    Vehicle ID: 0
        Vehicle 0 goes through [26, 25, 27]
Total Vehicle distance: 190.45021186131774
Depot ID: 5
    Vehicle ID: 0
        Vehicle 0 goes through [21, 23, 22]
Total Vehicle distance: 530.3516320900941
```

Fig. 26: Solution for Min92-27x5

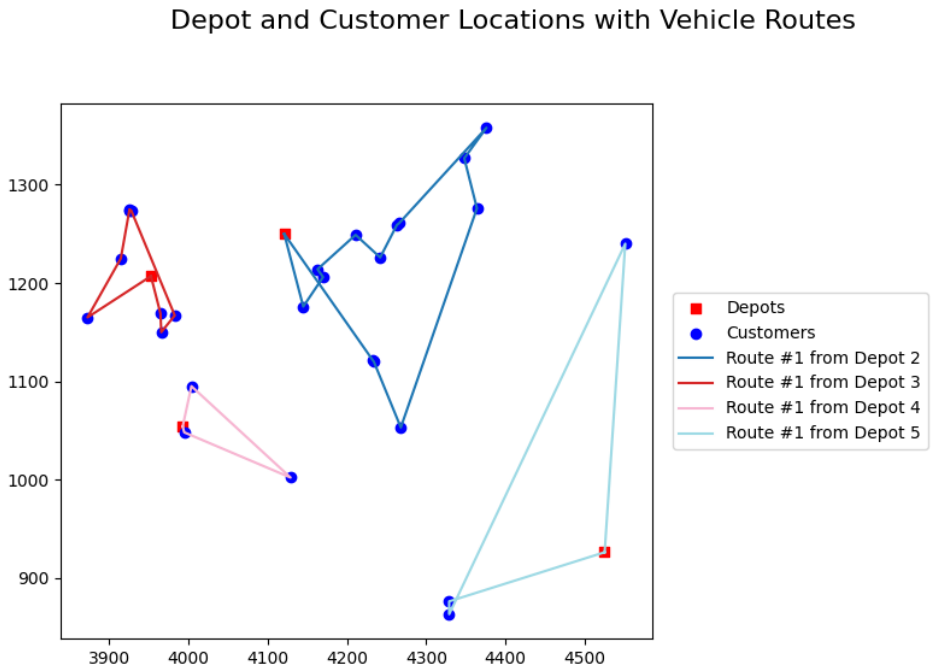

Fig. 26.1: Graph for Min92-27x5

13. Min92-134x8

This dataset comprises 134 customer locations and 8 depot locations, presenting a moderately large and complex logistics challenge. The depots are scattered throughout the area, and the varying demands of customers necessitate efficient strategies for selecting depots and routing vehicles effectively. After applying the Cuckoo Search Algorithm, the following depots were chosen to remain open: ['Depot 2', 'Depot 3', 'Depot 4', 'Depot 5', 'Depot 6', 'Depot 7', 'Depot 8'] (Fig.27.1). The total distance covered by all vehicles, including the fixed costs for the selected depots, was 6865.367882247668 units. In this case, the Cuckoo Search Algorithm (CSA) did not outperform the solution provided by the Nearest Neighbor Algorithm (NNA). **[29]**

```
Solution Total Distance: 6865.367882247668
Active Depots:
['Depot 2', 'Depot 3', 'Depot 4', 'Depot 5', 'Depot 6', 'Depot 7', 'Depot 8']
Depot ID: 2
    Vehicle ID: 0
        Vehicle 0 goes through [37, 32, 33, 26, 15, 23, 6, 4, 9, 11, 10, 27, 12, 20, 17, 29, 46, 63, 64, 54, 53, 56, 65, 62, 48, 51, 18, 5, 25, 21, 8, 22, 19, 24, 7, 16, 13, 52, 68, 67, 49, 47, 57, 39,
Total Vehicle distance: 1350.2817578547665
Depot ID: 3
    Vehicle ID: 0
        Vehicle 0 goes through [118, 111, 120, 110, 112, 130, 113, 122, 123, 125, 124, 116, 121, 1, 3, 28, 2, 14]
Total Vehicle distance: 612.0838437855958
Depot ID: 4
    Vehicle ID: 0
        Vehicle 0 goes through [86, 81, 90, 74, 91, 84, 88, 50, 61, 59, 66, 60, 58, 55, 93, 83, 107, 103]
Total Vehicle distance: 709.9735716649641
Depot ID: 5
    Vehicle ID: 0
        Vehicle 0 goes through [82, 99, 100, 101, 102, 126, 85, 41, 42, 36, 34, 38, 30, 45, 31, 87, 89, 95, 106, 94, 104, 98]
Total Vehicle distance: 860.9530417176658
Depot ID: 6
    Vehicle ID: 0
        Vehicle 0 goes through [131, 134, 128, 129, 132, 109, 119, 108, 127, 115, 133, 117, 114]
Total Vehicle distance: 463.0329808269067
Depot ID: 7
    Vehicle ID: 0
        Vehicle 0 goes through [79, 69, 71, 97, 105, 96]
Total Vehicle distance: 521.2247652078365
Depot ID: 8
    Vehicle ID: 0
        Vehicle 0 goes through [92, 70, 78, 72, 77, 76, 73, 80, 75]
Total Vehicle distance: 471.81808118993255
```

Fig. 27: Solution for Min92-134x8

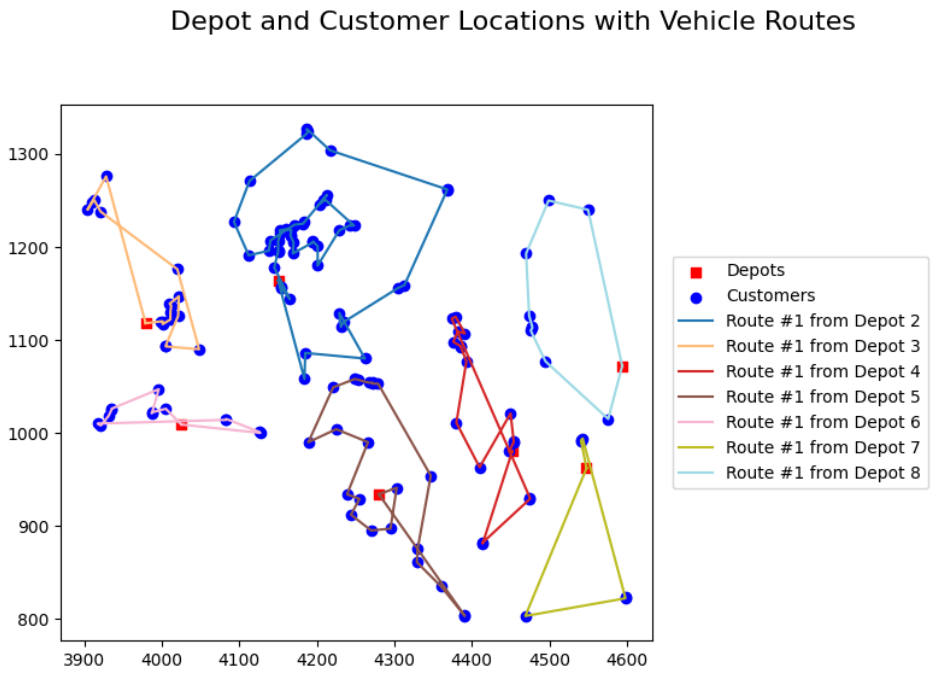

Fig. 27.1: Graph for Min92-134x8

14. Perl83-12x2

This dataset consists of 12 customer locations and 2 depot locations, presenting a moderately large and complex logistics challenge. The depots are distributed across the area, and customers' varying demands require efficient strategies for effectively selecting depots and routing vehicles.

After applying the Cuckoo Search Algorithm, the following depots were selected to remain open: ['Depot 1', 'Depot 2'] (Fig. 28.1). The total distance covered by all vehicles, including the fixed costs for the selected depots, was 296.7403193375044 units. In this case, CSA showed improvement over the original Nearest Neighbor Algorithm (NNA) solution. **[30]**

```
Solution Total Distance: 296.7403193375044
Active Depots:
['Depot 1', 'Depot 2']
Depot ID: 1
    Vehicle ID: 0
        Vehicle 0 goes through [9, 6, 1, 2, 7, 8]
Total Vehicle distance: 37.70082765359607
Depot ID: 2
    Vehicle ID: 0
        Vehicle 0 goes through [3, 4, 5, 11, 12, 10]
Total Vehicle distance: 59.03949168390835
```

Fig. 28: Solution for Perl83-12x2

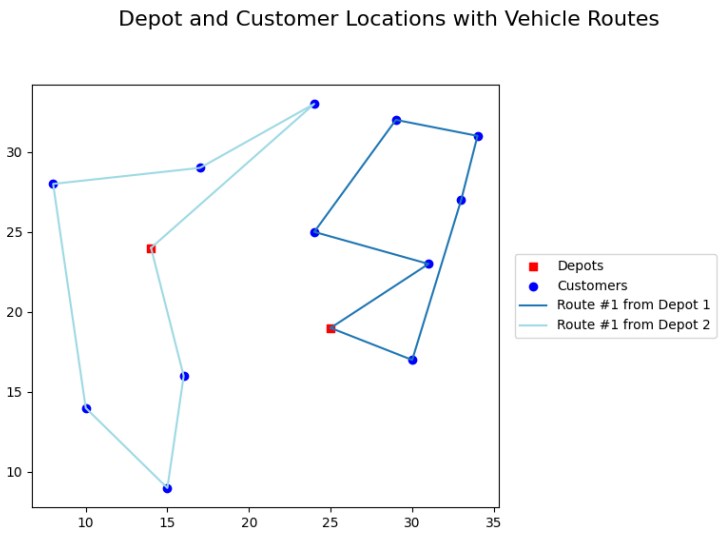

Fig. 28.1: Graph for Perl83-12x2

15. Perl83-55x15

This dataset consists of 55 customer locations and 15 depot locations, presenting a moderately large and complex logistics challenge. The depots are distributed across the

area, and customers' varying demands require efficient strategies for effectively selecting depots and routing vehicles.

After applying the Cuckoo Search Algorithm, the following depots were selected to remain open: ['Depot 1', 'Depot 2', 'Depot 3', 'Depot 4', 'Depot 5', 'Depot 6', 'Depot 7', 'Depot 8', 'Depot 9', 'Depot 10', 'Depot 11', 'Depot 13', 'Depot 14'] (Fig.29.1). The total distance covered by all vehicles, including the fixed costs for the selected depots, was 3477.5145341986627 units. In this instance, the Cuckoo Search Algorithm (CSA) did not surpass the solution yielded by the Nearest Neighbor Algorithm (NNA). **[30]**

```
Solution Total Distance: 3477.5145341986627
Active Depots:
['Depot 1', 'Depot 2', 'Depot 3', 'Depot 4', 'Depot 5', 'Depot 6', 'Depot 7', 'Depot 8', 'Depot 9', 'Depot 10', 'Depot 11', 'Depot 13', 'Depot 14']
Depot ID: 1
    Vehicle ID: 0
        Vehicle 0 goes through [14, 27]
Total Vehicle distance: 11.502143751025633
Depot ID: 2
    Vehicle ID: 0
        Vehicle 0 goes through [12, 17, 22, 16, 28]
Total Vehicle distance: 47.55188749000442
Depot ID: 3
    Vehicle ID: 0
        Vehicle 0 goes through [39]
Total Vehicle distance: 3.1622776601683795
Depot ID: 4
    Vehicle ID: 0
        Vehicle 0 goes through [38, 54, 55, 40]
Total Vehicle distance: 34.482655088826185
Depot ID: 5
    Vehicle ID: 0
        Vehicle 0 goes through [30, 33, 32, 45, 3, 31, 29, 19, 23]
Total Vehicle distance: 48.387919928717345
Depot ID: 6
    Vehicle ID: 0
        Vehicle 0 goes through [43, 46]
Total Vehicle distance: 9.60555127546399
Depot ID: 7
    Vehicle ID: 0
        Vehicle 0 goes through [8, 34]
Total Vehicle distance: 5.0
Depot ID: 8
    Vehicle ID: 0
        Vehicle 0 goes through [1, 13, 44, 2]
Total Vehicle distance: 15.242640687119286
```

```
Depot ID: 9
    Vehicle ID: 0
        Vehicle 0 goes through [42, 4, 9, 6, 41, 10, 15, 7]
Total Vehicle distance: 40.22300409530674
Depot ID: 10
    Vehicle ID: 0
        Vehicle 0 goes through [5, 11]
Total Vehicle distance: 2.0
Depot ID: 11
    Vehicle ID: 0
        Vehicle 0 goes through [36, 35, 24, 26, 18]
Total Vehicle distance: 31.711723647031185
Depot ID: 13
    Vehicle ID: 0
        Vehicle 0 goes through [47, 53, 50, 52, 37, 21]
Total Vehicle distance: 58.149381303672
Depot ID: 14
    Vehicle ID: 0
        Vehicle 0 goes through [49, 20, 25, 48, 51]
Total Vehicle distance: 50.495349271327626
```

Fig. 29: Solution for Perl85-55x15

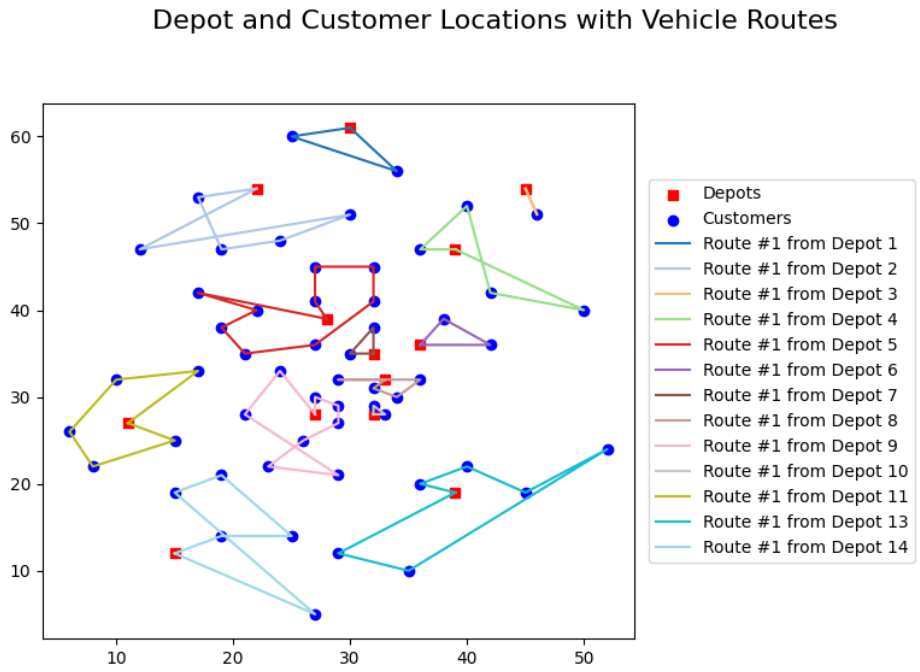

Fig. 29.1: Graph for Perl85-55x15

16. Perl83-85x7

This dataset comprises 85 customer locations and 7 depot locations, presenting a moderately large and complex logistics challenge. The depots are distributed across the area, and customers' varying demands require efficient strategies for effectively selecting depots and routing vehicles.

After applying the Cuckoo Search Algorithm, the following depots were selected to remain open: ['Depot 1', 'Depot 2', 'Depot 3', 'Depot 5', 'Depot 6'] (Fig.30.1). The total distance covered by all vehicles, including the fixed costs for the selected depots, was 2320.9927021147896 units. In this instance, the Cuckoo Search Algorithm (CSA) did not surpass the solution yielded by the Nearest Neighbor Algorithm (NNA). **[30]**

```
Solution Total Distance: 2320.9927021147896
Active Depots:
['Depot 1', 'Depot 2', 'Depot 3', 'Depot 5', 'Depot 6']
Depot ID: 1
    Vehicle ID: 0
        Vehicle 0 goes through [38, 2, 62, 54, 61, 65, 66, 68, 69, 59, 58, 39, 56, 57, 40, 55]
Total Vehicle distance: 86.31695569510117
Depot ID: 2
    Vehicle ID: 0
        Vehicle 0 goes through [33, 30, 19, 23, 75, 17, 22, 72, 63, 16, 32, 27, 14, 70, 71, 12, 73, 74, 28]
Total Vehicle distance: 109.74905035280734
Depot ID: 3
    Vehicle ID: 0
        Vehicle 0 goes through [8, 3, 7, 31, 29, 18, 84, 42, 4, 9, 5, 11, 1, 44, 34, 45, 43, 46]
Total Vehicle distance: 81.85083625686362
Depot ID: 5
    Vehicle ID: 0
        Vehicle 0 goes through [53, 47, 77, 37, 78, 79, 13, 67, 82, 50, 81, 83, 80, 52, 64]
Total Vehicle distance: 79.144406054745
Depot ID: 6
    Vehicle ID: 0
        Vehicle 0 goes through [24, 20, 21, 51, 76, 49, 48, 25, 41, 60, 10, 6, 15, 85, 36, 35, 26]
Total Vehicle distance: 103.93145375527206
```

Fig. 30: Solution for Perl83-85x7

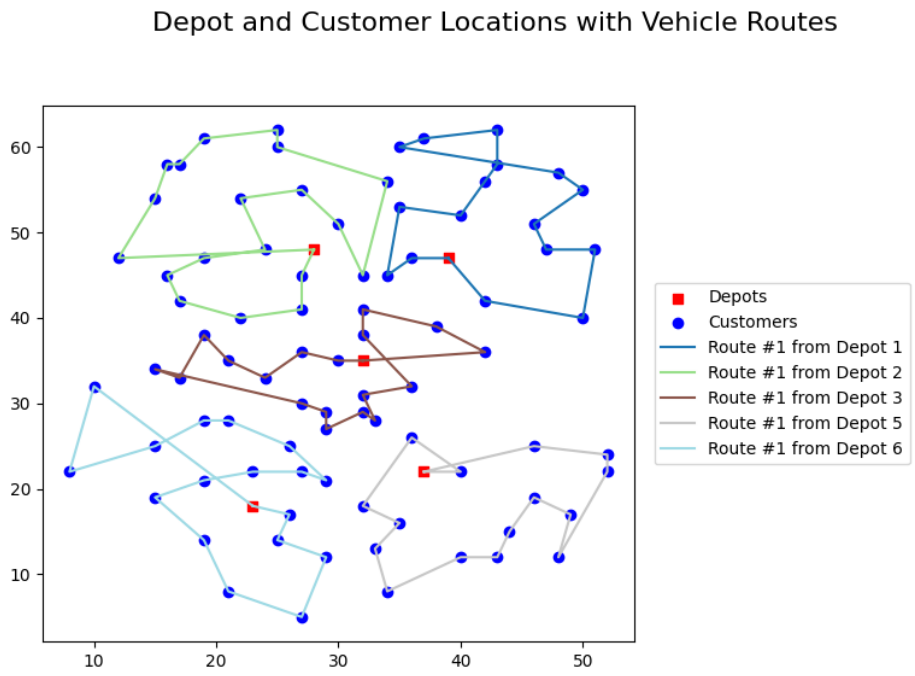

Fig. 30.1: Graph for Perl83-85x7

17. Perl83-318x4 with 25000 vehicle capacity

This dataset includes 318 customer locations and 4 depot locations. The vehicles have a capacity of 25000 units, presenting a moderately large and complex logistics challenge. The depots are spread throughout the area, and the varying demands of customers necessitate efficient strategies for selecting depots and routing vehicles effectively.

After applying the Cuckoo Search Algorithm, the following depots were chosen to remain open: ['Depot 1', 'Depot 2', 'Depot 4'] (Fig. 31.1). The total distance covered by all vehicles, including the fixed costs for the selected depots, amounted to 681618.746709361 units. In this instance, the Cuckoo Search Algorithm (CSA) did not surpass the solution yielded by the Nearest Neighbor Algorithm (NNA). **[30]**

```
Solution Total Distance: 681618.746709361
Active Depots:
['Depot 1', 'Depot 2', 'Depot 4']
Depot ID: 1
    Vehicle ID: 0
        Vehicle 0 goes through [95, 108, 192, 176, 172, 181, 173, 170, 190, 188, 189, 171, 180, 178, 195, 183, 185, 193, 186, 191, 184, 182, 174, 175, 179, 159, 211, 58, 60, 209, 210, 4, 69, 70,
Total Vehicle distance: 194636.9408489691
Depot ID: 2
    Vehicle ID: 0
        Vehicle 0 goes through [21, 79, 130, 129, 134, 137, 132, 133, 136, 138, 135, 131, 40, 89, 88, 48, 44, 57, 80, 109, 110, 82, 74, 27, 75, 25, 38, 49, 103, 42, 43, 45, 124, 47, 90, 99, 113,
Total Vehicle distance: 289384.77160353726
Depot ID: 4
    Vehicle ID: 0
        Vehicle 0 goes through [298, 310, 312, 315, 311, 314, 313, 316, 317, 244, 297, 270, 271, 308, 302, 293, 305, 261, 254, 267, 8, 19, 168, 169, 167, 202, 200, 153, 147, 318, 295, 237, 238,
Total Vehicle distance: 192957.05425685455
```

Fig. 31: Solution for Perl83-318x4 with 25000 vehicle capacity

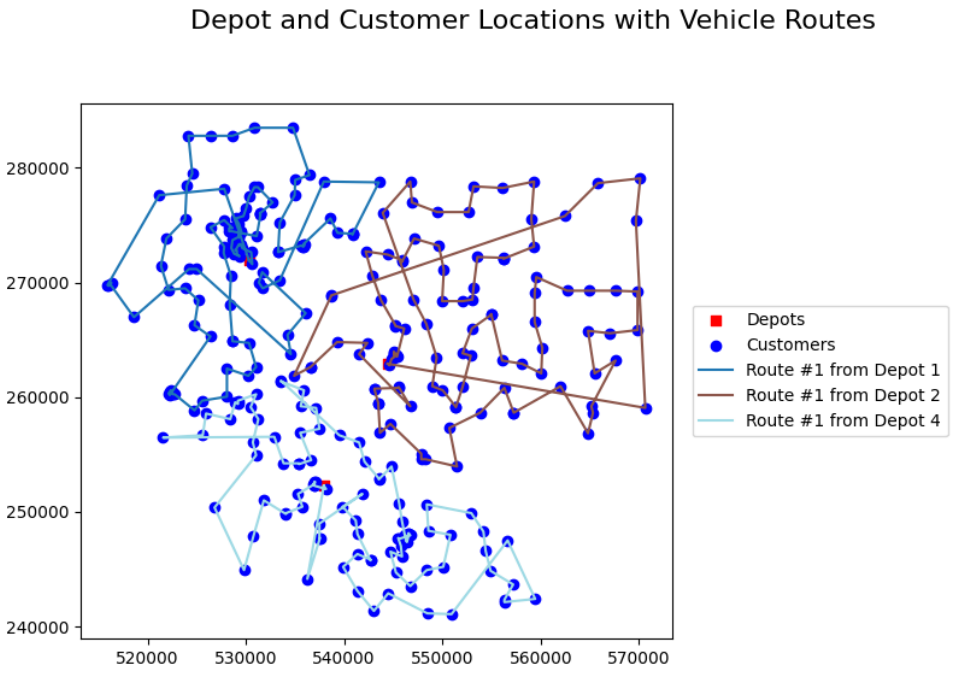

Fig. 31.1: Graph for Perl83-318x4 with 25000 vehicle capacity

18. Perl83-318x4 with 8000 vehicle capacity

This dataset includes 318 customer locations and 4 depot locations. The vehicles' capacity is 8,000 units, presenting a moderately large and complex logistics challenge. The depots are spread throughout the area, and customers' varying demands necessitate efficient strategies for selecting depots and routing vehicles effectively.

After applying the Cuckoo Search Algorithm, the following depots were chosen to remain open: ['Depot 1', 'Depot 3', 'Depot 4'] (Fig. 32.1). The total distance covered by all vehicles, including the fixed costs for the selected depots, amounted to 763214.8838060184 units. **[30]**

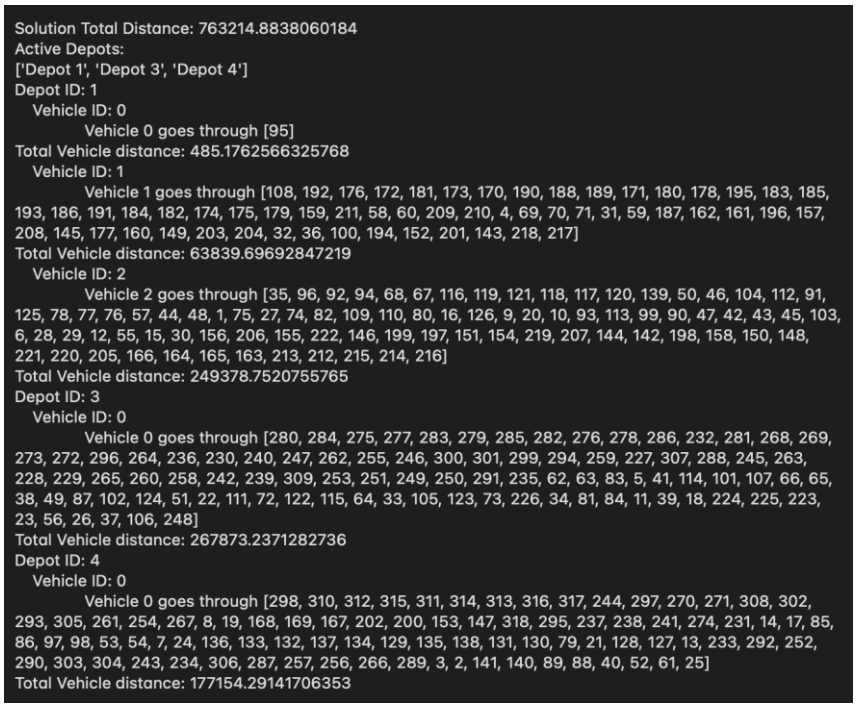

Fig. 32: Solution for Perl83-318x4 with 8000 vehicle capacity

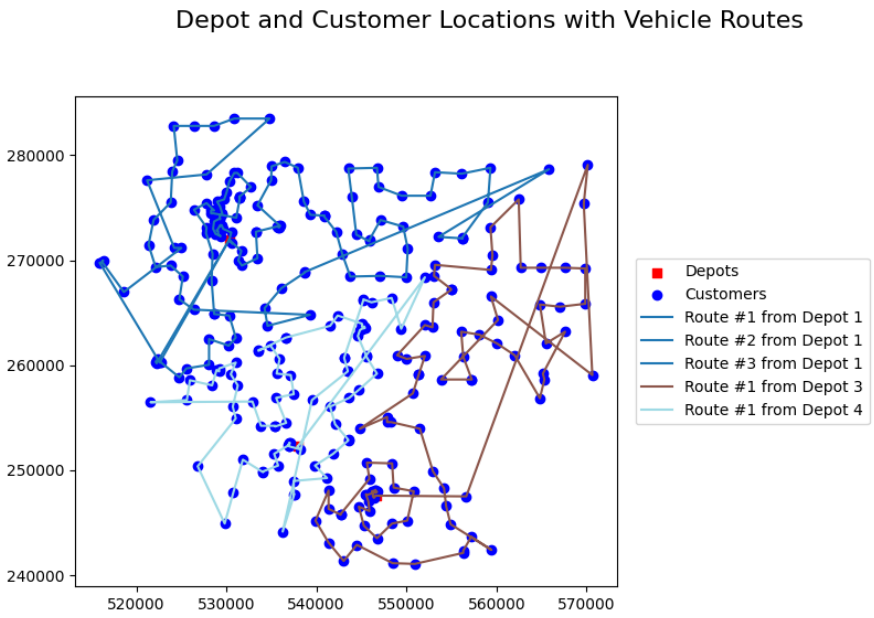

Fig. 32.1: Graph for Perl83-318x4 with 8000 vehicle capacity

19. Or76-117x14

This dataset consists of 117 customer locations and 14 depot locations, presenting a moderately large and complex logistics challenge. The depots are distributed across the area, and customers' varying demands require efficient strategies for effectively selecting depots and routing vehicles.

After applying the Cuckoo Search Algorithm, the following depots were selected to remain open: ['Depot 1', 'Depot 2', 'Depot 3', 'Depot 4', 'Depot 5', 'Depot 6', 'Depot 7', 'Depot 8', 'Depot 9', 'Depot 10', 'Depot 11', 'Depot 14'] (Fig. 33.1). The total distance covered by all vehicles, including the fixed costs for the selected depots, was 20171.56689544507 units. In this instance, the Cuckoo Search Algorithm (CSA) did not surpass the solution yielded by the Nearest Neighbor Algorithm (NNA). **[31]**

```
Solution Total Distance: 20171.56689544507
Active Depots:
['Depot 1', 'Depot 2', 'Depot 3', 'Depot 4', 'Depot 5', 'Depot 6', 'Depot 7', 'Depot 8', 'Depot 9', 'Depot 10', 'Depot 11', 'Depot 14']
Depot ID: 1
    Vehicle ID: 0
        Vehicle 0 goes through [18, 94]
Total Vehicle distance: 9.219544457292887
Depot ID: 2
    Vehicle ID: 0
        Vehicle 0 goes through [3, 50, 84, 78, 60, 96, 29, 7, 69]
Total Vehicle distance: 587.2816825059797
Depot ID: 3
    Vehicle ID: 0
        Vehicle 0 goes through [92, 48, 21, 85, 34, 63, 116, 117, 23, 12, 100, 103, 111, 102]
Total Vehicle distance: 3764.35261583347
Depot ID: 4
    Vehicle ID: 0
        Vehicle 0 goes through [90, 99, 37, 41, 40, 73, 72, 68, 80, 25, 74, 61, 89, 82, 42, 17, 36, 11, 88, 26, 77, 64, 14]
Total Vehicle distance: 2212.267329225103
Depot ID: 5
    Vehicle ID: 0
        Vehicle 0 goes through [91, 39, 22, 81]
Total Vehicle distance: 62.66742837328087
Depot ID: 6
    Vehicle ID: 0
        Vehicle 0 goes through [2]
Total Vehicle distance: 0.0
Depot ID: 7
    Vehicle ID: 0
        Vehicle 0 goes through [1]
Total Vehicle distance: 0.0
Depot ID: 8
    Vehicle ID: 0
        Vehicle 0 goes through [53, 76, 13, 54, 9, 30, 75, 43, 95]
Total Vehicle distance: 736.2575860934551
```

```
Depot ID: 9
    Vehicle ID: 0
        Vehicle 0 goes through [46, 62, 98, 70, 58, 97, 31, 57, 51]
Total Vehicle distance: 747.5034545911868
Depot ID: 10
    Vehicle ID: 0
        Vehicle 0 goes through [47, 35, 59, 4, 66, 15, 55, 10, 79, 20, 27, 71, 105, 107, 113, 108, 106, 112, 114, 115, 101, 109, 104, 110]
Total Vehicle distance: 4376.619986892323
Depot ID: 11
    Vehicle ID: 0
        Vehicle 0 goes through [52, 44]
Total Vehicle distance: 29.5296461204668
Depot ID: 14
    Vehicle ID: 0
        Vehicle 0 goes through [93, 33, 67, 6, 32, 16, 19, 83, 38, 5, 28, 45, 8, 65, 49, 87, 86, 24, 56]
Total Vehicle distance: 715.0676213525103
```

Fig. 33: Solution for Or76-117x14

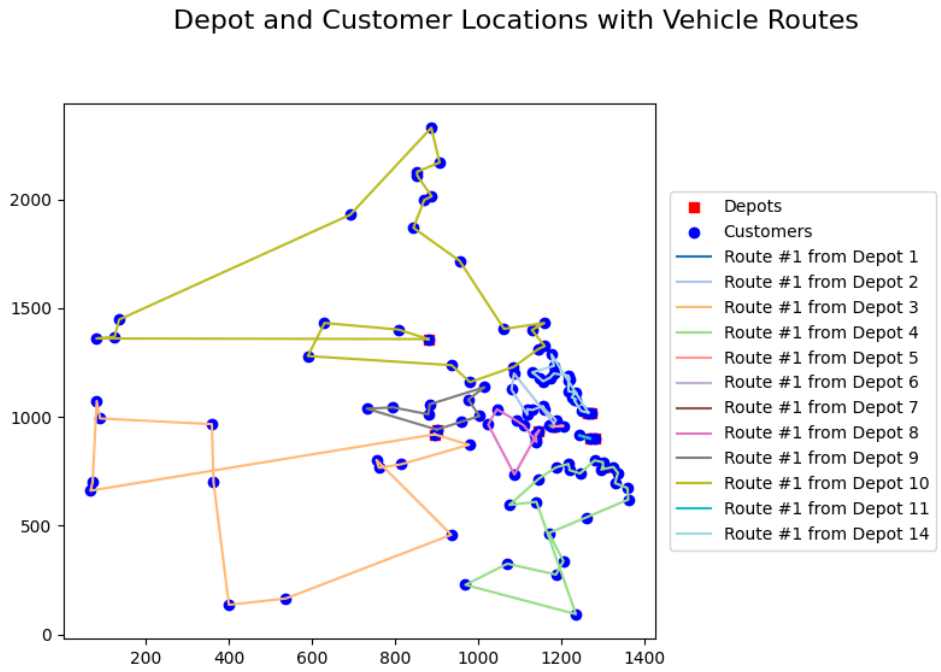

Fig. 33.1: Graph for Or76-117x14

Performance Evaluation Against Upper Bounds

Instance	Upper Bound	Algorithm Solution	Difference	Percentage Difference
Christofides69-50x5	582.7	631.584873	48.88487304	8%
Christofides69-75x10	886.3	946.376114	60.07611403	7%
Christofides69-100x10	889.4	1102.391498	212.991498	24%
Daskin95-88x8	384.9	971.4671735	586.5671735	152%
Daskin95-150x10	46642.7	74269.90351	27627.20351	59%
Gaskell67-21x5	435.9	438.825341	2.925340954	1%
Gaskell67-22x5	591.5	619.4042713	27.90427134	5%
Gaskell67-29x5	512.1	560.9411666	48.84116658	10%
Gaskell67-32x5	571.7	573.0818602	1.381860165	0%
Gaskell67-32x5	511.4	558.5182684	47.11826838	9%
Gaskell67-36x5	470.7	535.9326293	65.23262932	14%
Min92-27x5	3062	3171.995541	109.9955411	4%
Min92-134x8	6238	6865.367882	627.3678822	10%
Perl83-12x2	204	296.7403193	92.74031934	45%
Perl83-55x15	1136.2	3477.514534	2341.314534	206%
Perl83-85x7	1656.9	2320.992702	664.0927021	40%
Perl83-318x4	580680.2	681618.7467	100938.5467	17%
Perl83-318x4	747619	763214.8838	15595.88381	2%
Or76-117x14	12474.2	20171.5669	7697.366895	62%

6.3 Sensitivity Analysis

The effectiveness of metaheuristic algorithms like the Cuckoo Search Algorithm (CSA) relies heavily on choosing the right parameter values. To better understand how changes in these parameters affect the algorithm's performance—including its ability to converge, the quality of solutions it produces, and its overall robustness—sensitivity analysis plays a crucial role. This section focuses on the influence of three key parameters on the CSA's success in addressing the Location Routing Problem (LRP): the step size scaling factor (α), the probability of abandoning a solution (pa), and the number of initial solutions created using the Nearest Neighbor Algorithm (NNA).

This analysis aims to pinpoint parameter configurations that balance exploring new possibilities and exploiting known good solutions, ensuring a diverse set of solutions and efficiently delivering high-quality results. Each parameter will be discussed in detail, outlining its importance in the algorithm and the reasoning behind the chosen values.

The parameter α in the Cuckoo Search Algorithm is essential for regulating the step size during Lévy flights, which significantly impacts the algorithm's ability to balance exploration and exploitation. A smaller value of α narrows the search to local optimization, allowing for more precise adjustments within a limited area, while larger values facilitate broader exploration of the solution space.

This study examined three distinct values of α: 0.01, 0.05, and 0.1. Experimental findings revealed that an intermediate value of α = 0.05 achieved an optimal balance, providing adequate exploration while still enabling convergence on high-quality solutions. This parameter configuration successfully maintained a robust search across the solution landscape, mitigating excessive randomness and reducing the risk of premature convergence to suboptimal solutions.

Ultimately, the choice of α = 0.05 for the final implementation consistently yielded reliable outcomes across diverse datasets. This selection effectively harnessed the algorithm's global and local search strengths, ensuring optimal decision-making for depot selection and routing in the context of the Location Routing Problem.

The parameter pa in the Cuckoo Search Algorithm is vital for determining how often less favorable solutions are discarded. This introduces diversity into the solution pool, which is crucial for avoiding premature convergence and encouraging exploration of various areas within the solution space. A higher pa value results in more frequent abandonment of solutions, enhancing exploration, while a lower value allows solutions to remain for longer, emphasizing refinement and exploitation.

In most cases examined in this study, pa was set to 0.25, providing a balanced approach that enabled the algorithm to navigate the solution space effectively while retaining promising options. However, in certain instances where solutions exhibited excessive randomness or struggled to converge, the parameter was adjusted to 0.10. This reduction in the abandonment rate led to greater stability and refinement in the solution process.

This selective adjustment of pa highlights its importance in adapting the algorithm's behavior to the complexity and characteristics of different datasets. By carefully tuning this parameter, the Cuckoo Search Algorithm maintained a robust search process, resulting in reliable depot configurations and routing decisions in the Location Routing Problem.

The number of initial solutions produced by the Nearest Neighbor Algorithm (NNA) is crucial for the overall performance of the Cuckoo Search Algorithm (CSA). These initial solutions act as a starting point for the CSA's iterative improvement process, affecting both the variety of solutions it can generate and its ability to navigate the solution space effectively.

Typically, 20 initial solutions were created using NNA, striking a good balance between diversity and computational efficiency. However, there were cases where generating 20 initial solutions wasn’t practical due to specific constraints of the problems or didn't yield high-quality outcomes. In those situations, a smaller set of initial solutions was used, customized to fit the unique features of the dataset.

This adaptability in the number of initial solutions highlights the necessity of tailoring the algorithm’s parameters to the specific problem at hand. By achieving the right balance between solution variety and efficiency, the CSA effectively addressed the Location Routing Problem across all datasets while maintaining manageable computational demands.

CHAPTER 7: CONCLUSION AND EPILOGUE

This thesis addressed the intricate and computationally demanding Location Routing Problem (LRP), an essential concern in logistics optimization, by creating and implementing a hybrid optimization framework. The approach combined the Nearest Neighbor Algorithm (NNA) to generate efficient initial solutions with the Cuckoo Search Algorithm (CSA) to refine these solutions. The main objective was to reduce the total distance traveled by vehicles while optimizing the selection of depots and routing strategies.

The research began with a deep dive into the core concepts of logistics and supply chain management, examining various routing challenges, with a specific focus on LRP. A mathematical model of the Capacitated Location Problem (CLRP) was created to provide a clear definition of the issue. The study also addressed the difficulties in solving LRPs, which arise from their NP-hard nature and the interconnectedness of location and routing decisions.

To tackle these challenges, the proposed solution utilized the CSA, which is highly capable of balancing global exploration and local refinement. This enhanced the initial solutions generated by NNA. The algorithm's performance was tested on 18 different datasets, and the analyses highlighted its capability to deliver high-quality solutions while adhering to real-world constraints. A sensitivity analysis emphasized the crucial role of parameter tuning, especially for achieving an optimal balance between exploration and exploitation.

Key contributions of this research include the creation of a novel and scalable framework for resolving large-scale LRPs by merging NNA with CSA, effectively addressing both depot selection and vehicle routing. Comprehensive testing across 18 diverse datasets provided valuable insights into how the algorithm behaved under various parameter settings and dataset characteristics. Additionally, the implementation of the algorithm in Python, using libraries like NumPy and Matplotlib, ensures that the results can be replicated and sets the stage for further research and development.

Despite its successes, this thesis points to several future research directions. Comparing the performance of CSA with other metaheuristic strategies, such as Genetic Algorithms, Particle Swarm Optimization, or Simulated Annealing, could offer a broader perspective on its effectiveness. Extending the framework to accommodate dynamic customer demands or unpredictable variables like uncertain delivery times could significantly boost its practical usability. Furthermore, exploring hybrid optimization methodologies that combine CSA with other techniques might enhance both computational efficiency and solution quality. Testing the algorithm on larger and more intricate logistics datasets would also assess its robustness and generalizability.

Overall, this thesis illustrates the effectiveness of metaheuristics, especially the Cuckoo Search Algorithm, in tackling real-world logistics challenges. By merging advanced optimization techniques with practical problem-solving strategies, the research contributes to the wider field of logistics optimization. This study's findings lay the groundwork for creating more efficient, scalable, and sustainable supply chain solutions, offering valuable tools for academia and industry alike.

REFERENCES

[1] Islam, D. M. Z., Meier, J. F., Aditjandra, P. T., Zunder, T. H., & Pace, G. (2013). Logistics and supply chain management. Research in transportation economics, 41(1), 3-16.

[2] Salhi, S., & Rand, G. (1989). The effect of ignoring routes when locating depots. European Journal of Operational Research, 39, 150-156

[3] M.O. Ball et al., Eds., Handbooks in OR & MS, VoL 7, Chapter 4

[4] Applegate, D. L., Bixby, R. E., Chvatal, V., & Cook, W. J. (2011). The traveling salesman problem: A computational study. Princeton University Press.

[5] Archetti, C., Bianchessi, N., & Speranza, M. G. (2014). Branch-and-cut algorithms for the split delivery vehicle routing problem. European Journal of Operational Research, 238(3), 685–698. http://dx.doi.org/10.1016/j.ejor.2014.04.026.

[6] Dantzig, G. , Fulkerson, R. , Johnson, S. , 1954. Solution of a large-scale traveling-sales- man problem. J. Oper. Res. Soc. Am. 2 (4), 393–410.

[7] Dantzig, G. B., & Ramser, J. H. (1959). The truck dispatching problem. Management Science, 6(1), 80–92. http://dx.doi.org/10.1287/mnsc.6.1.80

[8] Clarke, G., & Wright, J. W. (1964). Scheduling of vehicles from a central depot to a number of delivery points. Operations Research, 12(4), 568–582. http://dx.doi.org/10.1287/opre.12.4.568

[9] Eksioglu, B., Vural, A. V., & Reisman, A. (2009). The vehicle routing problem: A taxonomic review. Computers & Industrial Engineering, 57(4), 1472–1483. http://dx.doi.org/10.1016/j.cie.2009.05.009.

[10] Amaya, A., Langevin, A., & Trépanier, M. (2007). The capacitated arc routing problem with refill points. Operations Research Letters, 35(1), 45–53.

[11] Jacobsen, S., & Madsen, O. (1980). A comparative study of heuristics for a two-level routing-location problem. European Journal of Operational Research, 6, 378–387.

[12] Albareda-Sambola, M., Fernández, E., & Laporte, G. (2007). Heuristic and lower bound for a stochastic location-routing problem. European Journal of Operational Research, 179(3), 940–955.

[13] Li F, Golden B, Wasil E. The open vehicle routing problem: Algorithms, large-scale test problems, and computational results. Comput Oper Res 2007;34 (10):2918–30.

[14] Liu, S., & Lin, C. (2005). A heuristic method for the combined location routing and inventory problem. International Journal of Advanced Manufacturing Technology, 26(4), 372–381.

[15] Klibi, W., Lasalle, F., Martel, A., & Ichoua, S. (2010). The stochastic multiperiod location transportation problem. Transportation Science, 44(2), 221–237

[16] Karaoglan, I., Altiparmak, F., Kara, I., & Dengiz, B. (2012). The location-routing problem with simultaneous pickup and delivery: Formulations and a heuristic approach. Omega, 40(4), 465–477.

[17] Lin, C. , Choy, K.L. , Ho, G.T. , Chung, S. , Lam, H. , 2014. Survey of green vehicle routing problem: past and future trends. Expert Syst. Appl. 41 (4), 1118–1138.

[18] Derigs, U., Pullmann, M., & Vogel, U. (2013). Truck and trailer routing – Problems, heuristics and computational experience. Computers and Operations Research, 40(2), 536–546. ISSN 0305-054.

[19] Lin, C., & Kwok, R. (2006). Multi-objective metaheuristics for a location-routing problem with multiple use of vehicles on real data and simulated data. European Journal of Operational Research, 175(3), 1833–1849.

[20] Guerrero, W., Prodhon, C., Velasco, N., & Amaya, C. (2013). Hybrid heuristic for the inventory location-routing problem with deterministic demand. International Journal of Production Economics, 146(1), 359–370.

[21] T. M. Cover and P. E. Hart, "Nearest Neighbor Pattern Classification", IEEE Trans. Inform. Theory, Vol. IT-13, pp 21-27, Jan 1967.

[22] Derbel, H., Jarboui, B., Hanafi, S., & Chabchoub, H. (2012). Genetic algorithm with iterated local search for solving a location-routing problem. Expert Systems with Applications, 39(3), 2865–2871.

[23] Lin, S.-W., Yu, V. F., & Chou, S.-Y. (2009). Solving the truck and trailer routing problem based on a simulated annealing heuristic. Computers and Operations Research, 36(5), 1683–1692.

[24] Ting, C.-J., & Chen, C.-H. (2013). A multiple ant colony optimization algorithm for the capacitated location routing problem. International Journal of Production Economics, 141(1), 34–44.

[25] Yang, X.-S.; Deb, S. Cuckoo Search via Lévy flights. In Proceedings of the 2009 World Congress on Nature & Biologically Inspired Computing (NaBIC), Coimbatore, India, 9–11 December 2009; pp. 210–214.

[26] Christofides, Nicos, Eilon, Samuel, 1969. An algorithm for the vehicle-dispatching problem. Operational Research Quarterly 20 (3) 309-318.

[27] Daskin, Mark S., 1995. Network and discrete location: models, algorithms and applications. John Wiley & Sons, Inc., New York.

[28] Gaskell, T. J., 1967. Bases for vehicle fleet scheduling. Operational Research Quarterly 18 (3), 281-295

[29] Min, Hokey, Current, John, Schilling, David, 1992. The multiple depot vehicle routing problem with backhauling. Journal of Business logistics, 13 (1) 259-288.

[30] Pearl, Jossef, 1983. A unified warehouse location-routing analysis. Ph. D. Dissertetion. Northwestern University, Evanston, Illinois, USA.

[31] Or, Ilhan, 1976. Traveling salesman–type combinatorial problems and their relation to the logistics of regional blood banking. Ph. D. Northwestern University, Evanston, Illinois.