VIET NAM NATIONAL UNIVERSITY – HO CHI MINH CITY
INTERNATIONAL UNIVERSITY
SCHOOL OF INDUSTRIAL ENGINEERING AND MANAGEMENT

METAHEURISTICS FOR SOLVING RE-DELIVERY ENABLED DELIVERY SYSYEM PROBLEMS: A COMPARATIVE STUDY AND PERFORMANCE ANALYSIS

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

Student: **Pham Ngoc Huy**
ID: **IELSIU19166**
Thesis Advisor: **Dr. Nguyen Hang Giang Anh**

Ho Chi Minh city, Vietnam.

February 2024

VIET NAM NATIONAL UNIVERSITY - HO CHI MINH CITY
INTERNATIONAL UNIVERSITY
SCHOOL OF INDUSTRIAL ENGINEERING AND MANAGEMENT

METAHEURISTICS FOR SOLVING RE-DELIVERY ENABLED DELIVERY SYSYEM PROBLEMS: A COMPARATIVE STUDY AND PERFORMANCE ANALYSIS

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

Student: **Pham Ngoc Huy**
ID: **IELSIU19166**
Thesis Advisor: **Dr. Nguyen Hang Giang Anh**

Ho Chi Minh city, Vietnam.
February 2024

Metaheuristics for Solving Re-Delivery Enabled Delivery System Problems: A Comparative Study and Performance Analysis

By
PHAM NGOC HUY

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

International University, Ho Chi Minh City
October 2023

Signature of Student: _____

Pham Ngoc Huy

Certified by _____

Dr. NGUYEN HANG GIANG ANH
Thesis Advisor

Approved by _____

Assoc. Prof. Dr. NGUYEN VAN HOP
Dean of IEM school

Abstract

The surge in e-commerce has brought about a rise in logistical challenges for shipping companies engaged with these digital marketplaces. This research aims to maximize profit for the company by optimizing the assignment of delivery routes, with a keen focus on weight and timing restrictions, especially considering that orders can be reassigned within a two-day window. This adjustment makes the classic Vehicle Routing Problem (VRP) more aligned with the practical realities of order reassignments. Utilizing data from an actual firm offers genuine insights into these challenges. By employing mathematical model such as Mixed Integer Linear Programming (MILP) alongside metaheuristic methods like Genetic Algorithms and Tabu Search, the study seeks to effectively manage larger operations. The research underscores how the boom in online shopping has escalated logistical expenses for supermarket chains and how the demand for swift deliveries complicates the coordination of order preparation and dispatch. It proposes a systematic method for distributing online orders, factoring in the available resources and costs, to enhance the decision-making process. The ultimate objective is to deliver superior service while lowering the overall costs associated with e-commerce fulfillment, moving from reliance on anecdotal evidence to an analytical approach.

Keywords: Vehicle Routing Problem, e-commerce, delay orders, capacity, time window, Mixed Integer Linear Programming, Genetics Algorithm, Tabu Search

Acknowledgments

We would like to begin by expressing my sincere gratitude to my advisor, Dr. Nguyen Hang Giang Anh, for her invaluable guidance and support during this challenging period. We are truly grateful for the countless hours she has devoted to providing constructive feedback, criticisms, and suggestions to enhance the quality of this work.

We extend our deepest appreciation and thanks to the esteemed Board of Deans of the School of Industrial Engineering Management, as well as all the dedicated lecturers who have selflessly shared their expertise and provided invaluable educational guidance to students pursuing a Bachelor of Engineering in Logistics and Supply Chain Management. We are grateful to the jury members for their interest in this study and for devoting their time to evaluating this thesis.

Throughout my professional journey, we have not only had the privilege of acquiring extensive knowledge in logistics operational activities but also encountered significant logistical challenges in real-world business settings, which have served as the bedrock of this thesis.

Lastly, we are indebted to my family, fellow seniors, and friends for their unwavering emotional support and generous encouragement. Without their assistance, we would not have been able to complete this study. Their contributions have played a pivotal role in the successful culmination of this thesis.

Table of Contents

List of Tables

List of Figures

CHAPTER 1. INTRODUCTION

1.1. Background

In the contemporary, rapidly evolving digital landscape, the burgeoning popularity of social media platforms and the ascendancy of online shopping have become conspicuous phenomena. Projections from the Department of E-Commerce and Economics indicate that by 2025, a substantial 55% of the populace is anticipated to engage in online shopping, with an estimated individual expenditure of 600 USD annually. This trend is poised to significantly bolster the e-commerce sector, contributing to a noteworthy commercial sales figure of 35 billion USD.

These statistics underscore the substantial scale of online consumerism and shopping patterns in Vietnam. However, this escalating demand has engendered emerging challenges in the effective management of order fulfillment, processing, and the prompt delivery of products to customers. These challenges frequently emanate from issues arising in customer-company interactions, compounded by logistical constraints that impact transportation resources. Notably, external factors often lead to delays in product delivery from manufacturers to customers, thereby inadvertently diminishing the efficiency of product delivery and the overall customer experience.

Effectively managing delayed order deliveries is crucial for businesses to enhance customer satisfaction, build trust, and foster brand loyalty. In the face of frequent order delays, redeliveries, and cancellations, addressing these challenges proactively can turn negative customer experiences into positive ones. By ensuring transparent communication, offering alternative delivery options, and compensatory gestures, companies can showcase their commitment to customer-centricity. Timely resolutions not only meet customer expectations but also demonstrate operational resilience and professionalism. Consequently, businesses that adeptly handle delayed orders not only retain customers but also gain advocates who positively endorse their brand, resulting in long-term customer loyalty and a positive reputation in the market.

Developing a mathematical model for Vehicle Routing Problems (VRPs) is academically stimulating, offering a platform to explore intricate optimization challenges and advance mathematical modeling skills. Understanding VRPs involves delving into diverse optimization techniques, including Mixed-Integer Linear Programming and metaheuristic algorithms, propelling innovations in operations research. A notable example is the study by (Li et al., 2020)on the Capacitated Vehicle Routing Problem with alternative delivery, pick-up, and time windows. Their modified hybrid approach demonstrates the practical application of mathematical models. Applying these models to address challenges such as delayed orders not only enriches academic knowledge but also holds the potential to significantly enhance the efficiency of logistics and supply chain operations, ushering these systems into a new era of effectiveness.

In a different research study, a solution was presented for the Capacitated Green Vehicle Routing Problem, which is introduced as a novel version of the traditional Vehicle Routing Problem. This specific problem involves the use of alternative fuel-powered vehicles (AFVs) for product distribution. These AFVs are designed with smaller fuel tank capacities, necessitating visits to alternative fuel stations (AFSs) for refueling while carrying out their distribution tasks (Zhang et al., 2018). The study focuses on advancing solutions through new mathematical models and formulas. Simultaneously, it explores the development of metaheuristic algorithms tailored for these models, enhancing their applicability across various operations, and increasing algorithm speed. In a specific investigation related to the vehicle routing problem, the study delves into the application of Tabu search. This application pertains to solving the time-dependent vehicle routing problem with time windows on a road network. The approach considers diverse shortest paths between any two customers, which vary according to different times of the day (Gmira et al., 2021).

1.2. Problem statement

This study explores ways to solve the complex delivery challenges faced by retailers who operate both physical stores and online platforms. These retailers sell a variety of products, including dried, fresh, and frozen foods, and they want to offer

customers different shipping choices based on their preferences. To manage these diverse orders efficiently, the retailer uses a flexible system. Customers can either choose to pick up their items from the store (click-and-collect) or have them delivered to their homes. All orders, whether from customers or stores, are processed in a central warehouse to ensure smooth operations. To stay organized, the retailer sets specific periods, referred to as "base periods," for order fulfillment. They also use a dynamic planning approach called the "rolling time horizon" to adjust order plans in real time as new customer orders come in. The study uses a weekly planning horizon as a reference but updates it daily for maximum efficiency. In essence, this research aims to find ways for retailers to handle deliveries better by providing different shipping options and using intelligent planning techniques to meet customers' needs effectively.

Certainly, our focus revolves entirely around the intricacies of the Vehicle Routing Problem (VRP), culminating in the optimal routing of vehicles to customers. Within this context, solving the VRP entails a meticulous consideration of various influential factors.

- **Capacity Problem**

Many businesses today encounter situations such as increased demand for orders, and many customer files from many different places are updated daily. With such a specific situation, currently, companies operate with a fixed fleet of vehicles but also prepare a backup plan. This backup plan is to rent a means of transportation from a third party (under contract). This causes the company to face problems of cost, limited transport volume, and time constraints. As a result, everyday vehicles must transport goods from distribution centers to customers in just one trip.

With constraints on the vehicle's maximum mass, it is impossible to transport the entire customer order in a single trip. There are still vehicles carrying orders that are less than the minimum volume that the vehicle can carry to meet the time factor for customers. For example, a vehicle with the capacity to carry 20 boxes is assigned to carry 2 boxes, which is a waste of resources and costs. Trips that are not sufficiently profitable will be discontinued as they lead to an imbalance in resource allocation. To solve this problem, this project proposes adding a minimum carrying capacity constraint

as a hard constraint. Specifically, the total number of products transported by each vehicle must be within a certain range based on the vehicle's carrying capacity.

- **Time window**

In terms of time constraints, customer orders are received and processed at a consistent daily time, typically at 5 pm, based on the operational practices of the business. To efficiently manage the logistics, transportation schedules, and vehicle coordination for the following day are planned to accommodate most incoming orders. The order pickup process starts at 6 am, concluding all deliveries by 8 pm.

It is crucial to note that orders are usually not pre-packaged upon arrival due to limited space at the warehouse loading dock. Additionally, there are food safety considerations and the need to prevent mixing with standard delivery orders, making pre-stacking unfeasible. After the specified order cutoff time, the specific order-picking process begins. Once the vehicles are loaded, the delivery process commences.

- **Delay order**

It is worth highlighting that goods may be loaded onto vehicles earlier than the stipulated time due to the travel time required to reach customers. Also, some orders might not be delivered on the first day (day 1) due to certain constraints, necessitating delivery on the second day (day 2). Following the established delivery policy, aiming for optimal customer service, all orders are delivered within a 2-day timeframe from the order placement date, excluding the order date itself.

- **Routing**

In the context of managing delivery routes, an essential emphasis is placed on meticulous routing strategies. Customers are methodically grouped into distinct regions, ensuring that a single truck serves one region within a specific time frame. The focus is on optimizing these routes, with multiple trucks potentially assigned to the same region if their routes align. This strategic approach guarantees that all orders assigned to a particular truck in a given time window are confined within the same region, facilitating streamlined and efficient deliveries. The routing process is not only about geographical mapping but also about aligning routes with specific time windows, enhancing

precision, and ensuring timely deliveries to customers' doorsteps or designated pick-up points.

- **Assigned order**

This factor holds paramount importance as it aligns with the policy of next-day delivery. Orders placed today must be delivered punctually and entirely the following day. Therefore, the arrangement of routes must meticulously adhere to these specific requirements, ensuring that deliveries are not only timely but also complete. This precision is crucial in meeting customer expectations and upholding the delivery commitment effectively.

1.3. Objectives of the Study

The fundamental objective of this research is to enhance the efficiency and optimization of goods exchange on e-commerce platforms, particularly in the context of freight transportation. The focus is on streamlining order receiving, processing, and shipping processes to improve overall efficiency, reduce operational costs, and maximize profits. Based on a thorough analysis of factors such as order destinations, delivery times, shipping costs, and available resources, the model aims to identify routes that are both efficient and cost-effective for fulfilling orders, especially those with higher costs.

Customer satisfaction remains a top priority, and the model ensures compliance with key requirements. This includes delivering orders within the specified time frame and dispatching entire orders in a single delivery, emphasizing careful preparation and optimal storage conditions. The model accommodates late orders by rescheduling them for the next day, and products are categorized as dry, fresh, or frozen, simplifying operational and recovery processes. During preparation, products are systematically grouped and transported to designated delivery points.

Moreover, the research addresses the timely processing of late orders, aiming to achieve a balance between operational efficiency and customer satisfaction. The pursuit of these goals involves rigorous academic analysis and strategic planning to foster a comprehensive and scholarly understanding of optimized shipping and order management processes in the realm of e-commerce platforms.

1.4. Scope and Limitation

1.4.1. Scope

The study aims to optimize vehicle routing within an academic research context, focusing on key factors and constraints. It considers the functionality of warehouses, which serve as both starting and ending points for vehicle routes, with each vehicle allocated a single route exclusively servicing customers. The study accommodates delays in orders, ensuring seamless delivery processes, and adheres to specific time constraints for order acceptance and delivery, operating between 6 a.m. and 8 p.m. Data processing techniques are employed to optimize vehicle utilization and adapt capacity to varying loads, minimizing cost distance based on traveled distances. Solver applications, including Mixed-Integer Linear Programming (MILP) for smaller instances and metaheuristic algorithms like Genetics Algorithm and Tabu Search for larger-scale tasks, are utilized. Comparative and sensitivity analyses evaluate solution quality and stability, providing insights for efficient route planning and decision-making in dynamic operational scenarios.

1.4.2. Limitations

In examining the proposed delivery system, several limitations have been identified, each of which poses significant constraints on its operational efficiency and flexibility:

- **Maximum Delay Restriction:**

One notable limitation of the system is the strict maximum delay restriction of one day for deliveries. This constraint implies that the delivery service lacks the flexibility to accommodate unforeseen circumstances or urgent delivery requirements. For time-sensitive deliveries or situations where customers require immediate access to goods, this constraint could severely impact customer satisfaction and operational adaptability.

- **Minimum Transport Weight Threshold:**

Goods falling below the vehicle's minimum transport weight threshold are deemed unfit for transportation. While this restriction ensures the optimal utilization of

transportation resources, it also imposes limitations on the types of products that can be transported. Smaller or lighter items, which might constitute a significant portion of certain businesses, could be excluded from the delivery service, limiting its applicability to a broader range of products and businesses.

- **Strict Adherence to Timelines:**

The system necessitates strict adherence to specified order initiation and completion times, as well as delivery commencement schedules. Any deviation from these predetermined timelines could disrupt the entire delivery process. While this adherence ensures organizational discipline, it restricts the system's ability to handle unexpected events, changes in customer demands, or variations in delivery requirements. This lack of adaptability might pose challenges in coping with dynamic market demands and customer preferences.

- **Singular Trip Limitation:**

Freight trucks are restricted to a singular trip, allowing transportation solely from the warehouse to customers. This limitation significantly impacts the volume of goods that can be transported within a given timeframe. Consequently, the system may struggle to efficiently manage high-demand periods or address fluctuations in customer orders. It also limits the optimization of delivery routes, potentially leading to inefficiencies in fuel usage and transportation costs.

- **Space Constraints and Food Conservation Requirements:**

Space limitations at the loading docks of the warehouse, coupled with food conservation requirements, add complexity to the delivery process. Loading and unloading efficiency are compromised due to limited space, potentially causing delays in shipments. Additionally, the need to adhere to food conservation standards places constraints on the types of products that can be transported, further narrowing the range of goods the system can handle effectively.

CHAPTER 2. RELATED WORKS

2.1. Overview

. The proposed solutions and recommendations for the aforementioned delivery challenges are frequently framed within the context of the vehicle routing problem. This problem is prevalent in route optimization endeavors, involving considerations of distance, time management, and the volume of goods transferred between various nodes, also referred to as distinct points.

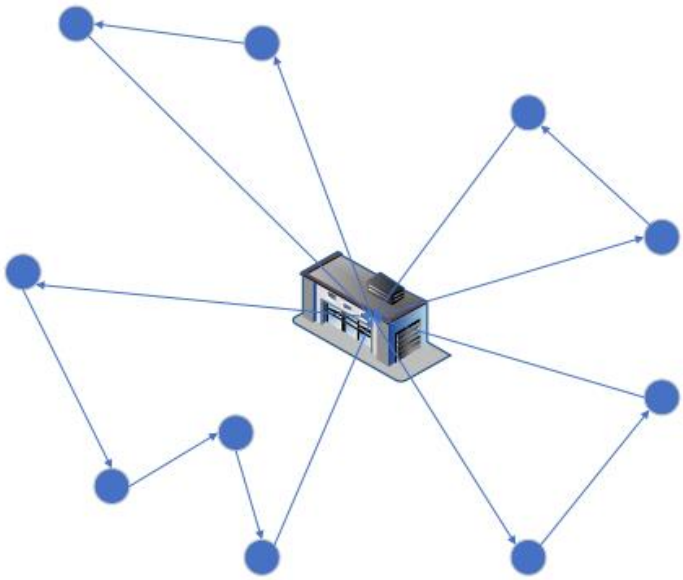

Figure 2.1: Typical simple VRP network

In practical applications, the fundamental vehicle routing problem expands with various constraints such as vehicle capacity, route length, arrival and departure times, repair durations, assortment schedules, and product deliveries. Consequently, there are multiple VRP variants tailored to specific requirements. The most well-known routing problem is the Traveling Salesman Problem (TSP), which involves a single vehicle visiting each town exactly once. While TSP serves as a foundational concept, real-world scenarios involve multiple vehicles with diverse parameters. Consequently, numerous VRP variations and specialized problems have emerged to address these complex and varied requirements.

Numerous variations of the Vehicle Routing Problem (VRP) exist; however, in this research, we specifically focus on the cases that are relevant to our specific concerns and objectives:

- The Vehicle Routing Problem with Profits (VRPP)
- Vehicle Routing Problem with Pickup and Delivery (VRPPD)
- Vehicle Routing with Time Window (VRPTW)
- Capacitated Vehicle Routing Problem (CVRP)

Upon identifying the suitable type of Vehicle Routing Problem (VRP) for the research, we will formulate an objective function incorporating the defined constraints. The primary objective is to address the core issue: determining the optimal sequence for delivering orders, specifying the delivery times, and volumes for each delivery, and determining the optimal number of vehicles required.

2.1.1. The Vehicle Routing Problem with Profits (VRPP)

The Vehicle Routing Problem with Profits (VRPP) presents a captivating twist to the traditional Vehicle Routing Problem (VRP), allowing for the strategic selection of nodes to maximize total profits within a specific time limit while ensuring vehicles commence and conclude their routes at the depot. Unlike the classical VRP, not all nodes must be visited, introducing a nuanced decision-making dimension. This flexibility demands astute resource management, considering factors like time, vehicle capacity, and profitability. VRPP's unique challenge lies in optimizing routes to balance profit maximization with adherence to strict time constraints, prompting the use of various algorithms and techniques in operations research and logistics. As businesses strive for efficient solutions, VRPP remains an intriguing problem, inspiring ongoing innovation in optimization strategies. Moreover, there is a study about VRPP-TD, which is an extension of the Vehicle Routing Problem (VRP) where each customer visit generates a fixed revenue, and there's no requirement to visit all customers. The goal is to determine the optimal number of vehicles, and their respective routes, and adhere to time constraints, all while maximizing overall profit. This profit is calculated by subtracting the total travel cost from the total revenue obtained from the visited customers (Aksen & Aras, 2006).

2.1.2. Vehicle Routing Problem with Pickup and Delivery (VRPPD)

The Pickup and Delivery Problem (PDP) revolves around optimizing routes for a fleet of vehicles assigned to fulfill transportation requests. Each vehicle is characterized by its capacity and starting and ending locations. Simultaneously, transportation requests come with load specifications, origins, and destinations. The primary challenge is to design efficient routes, ensuring all pickup and delivery points are covered. This task involves adhering to precedence constraints (demanding pickups occur before corresponding deliveries) and pairing constraints (ensuring one vehicle handles both pickup and delivery for specific loads).

2.1.3. Vehicle Routing with Time Window (VRPTW)

In the Vehicle Routing Problem with Time Windows (VRPTW), the capacity constraint remains while each customer, denoted as i, is associated with a specific time interval $[a_i, b_i]$, known as the time window, and a service time duration, s_i. These constraints limit the times when a customer can accept deliveries. Unlike the Complete Vehicle Routing Problem (CVRP), where the assumption of continuous customer availability over time is often unrealistic, VRPTW addresses real-world scenarios by incorporating specific time windows. These time windows can vary widely, from days to minutes, often aligning with the planning horizon. The introduction of time windows introduces precedence constraints on visits, making the problem asymmetric, even if the original distance and time matrices were symmetric. Solving VRPTW is also NP-hard, making finding feasible solutions challenging.

2.1.4. Capacitated Vehicle Routing Problem (CVRP)

In CVRP, vehicles are constrained by their cargo capacity while delivering goods. According to (Ekici et al., 2015),the setup includes a central depot, uniform vehicles with capacity Q, and customers with known demands. Routes for each vehicle must initiate and terminate at the central depot, and no route can surpass the vehicle's capacity. The primary objective is to minimize overall costs, mainly based on traveled distance when the number of vehicles remains constant. Additional expenses might be incurred if the number of routes fluctuates.

2.1.5. Mixed integer linear programming (MILP)

Mixed-integer linear Programming (MILP) stands as a robust mathematical optimization method employed to address problems where specific decision variables must be integers. In MILP, the objective function is linear, and constraints are expressed as linear equations or inequalities. MILP problems involve both continuous variables (taking real values) and discrete variables (limited to integers). The primary aim of MILP is to find the optimal solution by either maximizing or minimizing the objective function while ensuring compliance with all given constraints.

MILP finds widespread application across diverse fields, encompassing operations research, logistics, manufacturing, and finance. It models decision-making processes involving a mix of continuous and discrete choices. Solving MILP problems necessitates specialized algorithms capable of accommodating discrete variables' nature while optimizing the objective function within the specified constraints.

In the context of the Vehicle Routing Problem (VRP), MILP proves highly relevant and valuable. Its ability to handle discrete decisions and intricate constraints makes it particularly suitable for VRP scenarios. MILP formulations in VRP can be customized to meet specific demands, including tasks like customer-to-vehicle assignments, route optimization, and load capacity balancing.

2.2. Literature review

There is a wide array of approaches and styles in tackling the Vehicle Routing Problem (VRP). Among the numerous types of VRP, recent articles predominantly focus on addressing the Capacitated Vehicle Routing Problem (CVRP) and Vehicle Routing with Time Window (VRPTW) variations. While some articles touch upon the fundamentals of the Vehicle Routing Problem, these are relatively scarce due to the extensive research conducted in the past. Authors tend to delve into the mathematics of specific problems related to terminal transportation, picking, and unloading scenarios. The existing VRP strategies can be broadly classified into two categories: problem-specific approaches and metaheuristic algorithms. These categories have been subjects of extensive research over the years, and their literature reviews are briefly outlined below.

2.2.1. VRP and its variations

In the expansive domain of vehicle routing problems (VRPs), researchers have explored an array of innovative approaches, each adding a unique thread to the tapestry of knowledge. Their groundbreaking research challenged the conventional understanding of time-dependent VRPs (Gmira et al., 2021). While previous studies predominantly focused on the fluctuating travel times within cities, Gmira's work delved deeper. She not only emphasized the variations in travel times but also intricately analyzed the dynamic paths taken by delivery vehicles, recognizing that the routes themselves evolve throughout the day. This nuanced perspective prompted a shift in methodology, wherein the urban road network itself became the focal point of analysis. By considering travel time variations on specific road segments, Gmira provided a granular understanding of how routes transform over different times of the day, thereby offering a more holistic and context-sensitive solution to the ever-changing demands of urban logistics.

On the other hand, explored the multifaceted landscape of contemporary delivery systems, which have been significantly influenced by the rapid surge in online commerce(Wang et al., 2020). Acknowledging the need for adaptive and efficient delivery solutions, Sitek introduced innovative elements such as alternative delivery points and parcel lockers into the classical VRPs. These elements, now integral parts of modern urban environments, posed new challenges and opportunities. Sitek's approach, incorporating logical constraints, was a testament to the adaptive nature of modern e-commerce. It illuminated the pivotal role of customer expectations in shaping delivery strategies and underscored the importance of blending tradition with innovation in the dynamic realm of urban logistics.

In a parallel trajectory, Li et al., (2020) pioneered a transformative approach by addressing the logistical challenges stemming from the amalgamation of cutting-edge technology and traditional delivery methods.Li et al., (2020)’s research introduced the two-echelon vehicle routing problem with time windows and mobile satellites (2E-VRP-TM). This approach not only optimized routes for van-UAV combinations but also represented a paradigm shift in delivery scheduling. The integration of unmanned aerial vehicles (UAVs) with human-driven vans heralded a new era in the delivery landscape.

Each vehicle, equipped to serve customers at different times, epitomized the convergence of advanced technology and logistical efficiency, redefining the conventional notions of last-mile delivery.

Simultaneously, Pan et al., (2021)navigated the intricate terrain of urban transportation routing problems. Their study, a synthesis of sophisticated algorithms and real-world urban complexities, tackled a multifaceted challenge. By incorporating time-dependent travel times, multiple trips per vehicle, and loading time constraints, Pan et al., (2021) research presented a holistic perspective on urban logistics. Unlike conventional Traveling Salesman Problem (TSP) solutions, which often overlooked the intricacies of urban environments, Zhang's innovative methodology optimized routes within cityscapes. The solution, while meeting stringent time windows, vehicle capacity, and maximum trip duration constraints, represented a monumental leap in the field. It underscored the necessity of adaptive and multifaceted strategies in the dynamic and evolving landscape of modern logistics and urban delivery systems. As the world continues to urbanize and e-commerce proliferates, these diverse approaches stand testament to the innovative spirit driving advancements in urban logistics, promising a future where efficient and sustainable delivery solutions seamlessly integrate with the fabric of urban life.

2.2.2. Meta-heuristics for solving VRP and its variations.

In the realm of solving intricate vehicle routing problems (VRPs), researchers have turned to advanced meta-heuristics, harnessing the power of innovative algorithms to navigate the complexities of real-world logistics. Gmira et al., (2021)pioneering work exemplified this approach by employing a tabu search heuristic tailored to the dynamic urban environment. By considering different shortest paths between customers at various times of the day, Gmira's method showcased an adaptive and context-sensitive solution. This exemplifies the depth of exploration required in urban logistics, where the interplay of time-dependent factors demands sophisticated algorithmic approaches to optimize routes effectively.

In a parallel vein, Pan et al., (2021) took a significant leap by devising a hybrid meta-heuristic algorithm tailored specifically for the urban transportation domain.

Drawing inspiration from adaptive large neighborhood search (ALNS) for guided exploration and variable neighborhood descent (VND) for intensive exploitation, Zhenzhang's approach highlighted the synergy between exploration and exploitation in optimizing routes. The incorporation of problem-specific local search operators and removal operators further underscored the meticulous tailoring of algorithms to the unique challenges posed by urban environments. Through rigorous experimentation, this hybrid algorithm demonstrated its robustness and efficiency, even under diverse speed profiles and maximum trip duration limits.

Additionally, Tran Vu Nhat Trung's work delved into the application of Simulated Annealing, a powerful optimization technique, to address the capacited vehicle routing problem (CVRP). Trung's research showcased the adaptability of meta-heuristics to specific problem instances, emphasizing the capability of Simulated Annealing in generating optimal routing schemes. This exemplified the versatility of meta-heuristics, showcasing how tailored algorithms can be deployed to address nuanced variations within the broader spectrum of VRPs.

In synthesizing these diverse approaches, a common thread emerges — the recognition of the intricate interplay between problem specificity and algorithmic adaptability. Urban logistics, marked by dynamic variables such as varying travel times, intricate paths, and stringent time windows, demands tailored solutions. Meta-heuristics, ranging from tabu search and simulated annealing to hybrid algorithms incorporating ALNS, VND, PSO, and genetic algorithms, represent the vanguard of this endeavor. Through meticulous algorithm design and rigorous experimentation, researchers continually refine and enhance these techniques, ensuring their applicability in addressing the multifaceted challenges of modern urban transportation. This dynamic interplay between problem specificity and algorithmic adaptability stands testament to the evolving landscape of optimization in urban logistics, where innovative methodologies continue to reshape the contours of efficient and sustainable delivery systems.

In the diverse landscape of research approaches, scholars have explored various facets of logistics problems. However, in the evolving landscape of modern commerce, the challenges faced by businesses have taken on new dimensions. Issues such as

delayed deliveries, disruptions in carrier services, and customer-related complications in e-commerce transactions have become prevalent. Addressing these contemporary challenges, our research aims to bridge this gap comprehensively. We focus on understanding and incorporating these real-time complexities into our study. Our objective is not only to identify these problems but also to develop effective solutions.

To tackle these issues and enhance business profitability, we delve into the application of advanced meta-heuristic algorithms. Specifically, we employ one of the algorithms mentioned earlier to solve these problems on a larger scale. By conducting detailed comparisons and evaluating mathematical models, we strive to pinpoint the most suitable and efficient solution. Our research endeavors to provide practical and innovative strategies for businesses to navigate the intricate landscape of modern logistics, ensuring timely deliveries and customer satisfaction in the face of contemporary challenges.

2.3. Key References/Candidate Solution Methods

Upon evaluating the multifaceted aspects of the routing dilemma, bounded by delivery timelines and vehicle capacity constraints, we identified a pertinent research article that significantly aligns with our investigational focus. Hybrid genetic–tabu search algorithm to optimize the route for capacitated vehicle routing problem with time window (Akbar & Aurachmana, 2020) will serve as a primary reference for our study, providing a foundation from which we will extend our theoretical framework. The selected article extensively examines the implications of delivery timings and cargo volume limitations, adopting a lenient approach towards temporal constraints through the imposition of penalty costs. It establishes a stringent limitation solely on the cargo volume permissible for transportation, without addressing the nuances of redelivery or the detailed scheduling from order pickup to delivery.

In response to these identified gaps, our research aims to refine the model by transitioning from a flexible to a strict temporal constraint, aiming to devise a precise schedule for delivery personnel. Furthermore, we will explore the implications of order redelivery, attributing potential causes to either the delivery service or the customers themselves. To accommodate the complexity and broaden the scope of the problem, we

propose integrating hybrid metaheuristic strategies, moving beyond the confines of exact solutions suitable for smaller-scale issues. This adaptation seeks to enhance the operational efficacy and applicability of the model in managing larger scale logistics challenges.

CHAPTER 3. METHODOLOGY

3.1. Approaches Comparison and Selection

3.1.1. Mathematical model formulation

Create a mathematical model for the Capacitated Vehicle Routing Problem (CVRP) with time windows utilizing the Mixed-Integer Linear Programming (MILP) approach. Ensure the model adheres to constraints related to vehicle capacity, specific time windows for events, optimal routing, efficient delivery processes, and handling delayed orders. Make the model specific to the unique challenges of the VRP, encompassing not just capacity and time constraints but also focusing on the intricacies of routing, delivery sequencing, and managing unforeseen delays in order fulfillment.

Employ the solver tool to rigorously assess the model's accuracy by inputting specific datasets and executing the calculations. Scrutinize the results meticulously to ensure they align precisely with the predetermined constraints. If any discrepancies or violations are detected, initiate a comprehensive review and refinement of the model. In case the results adhere flawlessly to the established constraints, advance to the subsequent stages of the analysis.

3.1.2. Meta-heuristic selection

Table 3.1: Approaches comparison

	Advantages	**Disadvantages**
Tabu Search	• Efficiently explores complex solution spaces. • Allows escaping local optimal using memory structures.	• Require fine – tuning of parameters. • These might be suboptimal solutions in certain cases. • Sensitivity to parameter settings

	• Adaptable and tailored to specific problem constraints	
Genetic Algorithm (GA)	• Handles large solution spaces effectively. • Provides global optimization through population diversity. • Suitable for VRP variations with diverse constraints	• Slow convergence due to genetic operations. • Initial population setup is crucial for finding optimal solutions.
Particle Swarm Optimization (PSO)	• Fast convergence, especially in large solution spaces • Simplicity in implementation and fewer parameters to tune. • Suitable for dynamic VRP scenarios	• Prone to premature convergence. • Sensitive to parameter settings. • Limited ability to escape local optimal.
Simulated Annealing (SA)	• Robust in escaping local optimal, allowing exploration of diverse solutions. • Can handle VRP variants with complex constraints.	• Requires careful adjustment of the cooling schedule. • Slow convergence, especially in large solution spaces. • Might get stuck in local minimal.

	• Provides global optimization under proper cooling schedules.	

Selection of the Meta-heuristic:

In addressing variations of the Vehicle Routing Problem (VRP), the selection of an appropriate optimization technique holds significant importance. The VRP is a highly intricate combinatorial optimization problem, necessitating the exploration of optimal or near-optimal solutions within an extensive solution space. Among the frequently employed optimization methods in this domain are Tabu Search, Genetic Algorithm (GA), Particle Swarm Optimization (PSO), and Simulated Annealing.

While all four techniques exhibit the potential to converge towards either local or, under favorable conditions, global optima, a closer examination reveals subtle distinctions that render them more suitable for specific VRP variations.

It is noteworthy that most of these algorithms encompass two solutions that emphasize local search and two that prioritize global search. For the preliminary survey, We have chosen to delve into one of the local search methods—Tabu Search. This choice is motivated by the method's simplicity in application compared to Simulated Annealing. Additionally, Tabu Search is deemed more suitable for this problem due to its ability to efficiently separate the solution string into individual components during implementation. In this survey, Tabu Search's exploitation and exploration will be executed straightforwardly and expeditiously.

Following the completion of the local search exploration, the next step involves a hybrid approach, combining Genetic Algorithm and Tabu Search. This hybridization aims to transition from local to global search. As previously mentioned, Genetic Algorithm's drawback lies in its slow convergence due to genetic modification. In this context, Tabu Search proves to be a valuable support, facilitating rapid convergence of solutions when a child is generated, consistently surpassing its predecessor in development.

3.2. Proposed System Design/ Proposed Solution Approach

3.2.1. Proposed system design.

Building upon the fundamental understanding established in the earlier sections, this system design has been formulated through an in-depth analysis of relevant literature and essential references. The design concept for optimizing the vehicle routing problem is crafted as follows:

Figure 3.1. System design of the thesis

3.2.1.1. Mathematical Model Formulation

In optimizing operational efficiency, a company U's unique challenges are pivotal. By translating real-world obstacles into essential constraints, these challenges serve as catalysts for innovative breakthroughs within the mathematical model. Integrating these constraints with the original models ensures a harmonious blend of practicality and theoretical depth.

Strategically, real-world complexities are streamlined into linear programming paradigms for computational efficiency without compromising accuracy. This pragmatic approach empowers precise decision-making. Moreover, infusing the model with theoretical insights fortifies its adaptability, transforming it into a dynamic tool for future challenges.

This holistic strategy not only resolves current obstacles but also anticipates and prepares for future uncertainties, ensuring the company's resilience. By embracing real-world intricacies and strategic innovation, this approach becomes a powerful framework for optimizing MILP problems, enhancing the company's overall operational landscape.

3.2.1.2. Model Implementation and Solver

Upon the formulation of the mathematical model, it is imperative to implement it within Optimization Studio. This integration allows the model to be operationalized, transforming theoretical constructs into practical applications. The logical coherence of the model's outcomes is ensured by meticulously adjusting the data within the data source, thereby aligning the mathematical abstraction with real-world scenarios.

3.2.1.3. Comparison with the current system

In the realm of operational systems, a methodical iterative process stands as a fundamental principle. Rigorously comparing newly developed systems with their existing counterparts constitutes a pivotal practice. If the latest iteration falls short of its predecessor, a meticulous analysis of constraints and resource allocation ensues, guided by precise analytical insights. Adjustments are made, followed by comprehensive reevaluation. This iterative cycle, focusing on problem identification, precise solution implementation, and systematic reassessment, embodies a proactive paradigm. It ensures continual optimization and alignment with evolving operational demands. This approach, marrying theoretical depth with empirical adaptability, exemplifies a prudent strategy for perpetual system refinement.

3.2.1.4. Develop solution representation.

When initial expectations from the solver are hindered by the constraints within its programming and the limitations of the studio, a strategic approach is necessitated.

This involves crafting a solution representation tailored for meta-heuristics, thereby enhancing the scalability of the dataset. This adaptation becomes pivotal in addressing larger and more intricate problems. The solution representation crafted must be adept at accommodating all constraints, unique requirements, and assumptions inherent to the model. This not only serves as the foundational solution but also shapes the eventual outcome representation for the model, ensuring a comprehensive and adaptable approach to complex problem-solving scenarios.

3.2.1.5. Meta-Heuristics Selection

Having selected the meta-heuristics mentioned earlier, it becomes imperative to delve into their mechanisms comprehensively. Understanding the intricate workings of these heuristics is foundational. This knowledge serves as a precursor to applying the solution representation crafted previously. By integrating this representation into the meta-heuristic algorithms and launching the process, the system becomes adept at navigating complex problem landscapes. This integration not only harnesses the power of the chosen meta-heuristics but also ensures the optimal utilization of the meticulously designed solution representation. This combined approach stands poised to tackle intricate challenges, demonstrating the synergy between theoretical understanding and practical application in problem-solving contexts.

3.2.1.6. Validation with the result of Mathematical Model

In the initial run, it is prudent to begin with the smallest dataset and compare the outcomes with those obtained from the solver. This meticulous approach ensures precision and feasibility. Any disparities observed between the meta-heuristic solution

and the solver outcome demand thorough scrutiny. Delving back into the mechanism of the chosen meta-heuristics or reviewing the coding process becomes imperative. Every step must be meticulously revisited to identify potential errors until the solution aligns perfectly with the solver's results. This iterative process guarantees the accuracy of the model and underscores the commitment to achieving results that are both exact and reliable.

3.2.1.7. Comparison

Upon successful validation of each meta-heuristic through meticulous comparison and scrutiny, the model progresses to more extensive applications. Here, it is crucial to execute the model on a larger scale and meticulously record time-lapse data and solution iterations. This systematic approach serves to discern the most efficient meta-heuristic tailored for the VRP problem at hand. Through a comprehensive analysis of the recorded data, the optimal meta-heuristic one demonstrating the best fit and utmost efficiency can be identified. Subsequently, the selected meta-heuristic is implemented seamlessly, ensuring the continued pursuit of operational excellence in addressing the VRP challenge. This methodical process exemplifies a commitment to precision and efficiency in algorithm selection and implementation.

3.2.1.8. Sensitive Analysis

Once the initial results are obtained and the comparison is made, the data can be adjusted for sensitivity analysis. Sensitivity analysis involves assessing the impact of changes in input parameters or constraints on the optimized schedule and associated costs. This analysis helps identify the robustness and flexibility of the scheduling optimization model.

3.2.1.9. Conclusion and Discussion

Vehicle Routing Problems (VRP) are complex logistical challenges that require efficient solutions for optimal route planning. VRPs involve determining the most cost-effective routes for a fleet of vehicles to serve a set of customers while satisfying various

constraints. To tackle this intricate problem, employing state-of-the-art meta-heuristics can significantly enhance the efficiency and effectiveness of the solutions.

The hybrid meta-heuristic approach offers a powerful solution framework for VRPs. This synergy between exploration and exploitation enables the algorithm to navigate complex solution spaces effectively, delivering high-quality, near-optimal routes for fleet management. Embracing these cutting-edge meta-heuristics is essential for businesses seeking to optimize their vehicle routing operations, ensuring cost savings, efficient resource utilization, and enhanced customer satisfaction.

3.2.2. Technical Application

3.2.2.1. Programming tool

AMPL stands as a prominent tool in the realm of optimization for its adaptability and ease of use. It excels in managing complex issues and multiple objectives simultaneously, making it an ideal choice for a variety of optimization tasks. AMPL's ability to efficiently tackle diverse goals, constraints, and decision variables across different scenarios, such as cost minimization, efficiency enhancement, or specific performance targets, is particularly noteworthy.

One of AMPL's key advantages is its programming style, which is reminiscent of widely-used languages like C++ and C#. This similarity reduces the learning curve and allows for quick adaptation to the tool. The intuitive design and familiar programming concepts of AMPL enable users to quickly grasp its functionality, facilitating a smooth exploration of its extensive features.

AMPL proves to be exceptionally useful in projects that demand rapid decision-making due to its user-friendly approach, which streamlines the navigation through complex optimization problems and speeds up the solution-finding process. This makes AMPL an indispensable asset for professionals faced with complicated optimization challenges, ensuring both effectiveness and efficiency in their work.

In our research, AMPL plays a crucial role in developing and demonstrating mathematical models, offering a platform for formulating precise solutions using various solvers. However, it's important to note that AMPL has its limitations, particularly with the handling of a large number of variables in some instances.

Nevertheless, it proves useful for testing models on a smaller scale to validate their practicality and coherence.

Additionally, this paper employs the CPLEX IBM software for solving the Mathematical Programming (MP) model. CPLEX stands out as a comprehensive tool that supports the full optimization modeling process, encompassing everything from creation and testing to deployment and ongoing management. It offers a rich modeling language for defining optimization elements such as data, variables, goals, and constraints, along with a command language for model exploration and result analysis, and a scripting language for data handling, iterative optimization, and more.

3.2.2.2 MATLAB & Simulink

MATLAB, a widely used programming language and environment, is renowned for its prowess in numerical computing and data analysis. Developed by MathWorks, MATLAB is a versatile tool employed by scientists, engineers, and researchers for tasks ranging from complex mathematical computations to data visualization and simulation. Its user-friendly interface and extensive library of functions make it an essential asset across various disciplines, empowering professionals to tackle intricate problems and explore innovative solutions.

Simulink, an integral part of MATLAB, offers users the convenience of accessing MATLAB's functionalities without the need for application downloads. Users can utilize a trial web version, limited to 20 hours per initial account registration. This feature not only conserves memory but also enhances user experience, allowing them to enjoy the complete range of MATLAB features.

In this study, MATLAB & Simulink are employed as programming tools for implementing metaheuristics once a solution representation is established. Their use is crucial because these platforms offer accessible programming languages, making concepts such as loops and conditions easily comprehensible for readers. Once the metaheuristics for the model are programmed, validation will be conducted at a scale similar to that of AMPL. Following successful validation, the scale will be expanded, and the running time will be measured to generate the results.

CHAPTER 4. SOLUTION DEVELOPMENT

4.1. Model/ Prototype Solution

4.1.1. Assumptions

In this section, our primary focus revolves around the meticulous construction of a decision support model, adept at synchronizing the intricate dynamics of order picking and vehicle routing. Commencing this process involves a meticulous definition of parameters delineating the customer orders received at the initiation of the planning horizon. These parameters are intricately named, ensuring precision and clarity in both the order-picking and delivery subproblems.

Central to our endeavors is the formulation of a sophisticated, integrated Mixed Integer Linear Programming (MILP) model. This model's hallmark lies in its adaptability, accommodating two distinctive operational strategies. The first strategy caters to companies autonomously managing their operations, utilizing in-house resources. Conversely, the second strategy addresses businesses opting to outsource vehicles. These strategies, underpinned by specific assumptions, are instrumental in customizing the model to diverse operational frameworks, rendering it a versatile solution for the seamless integration of order picking and vehicle routing intricacies.

4.1.2. Mathematical model

Sets and Indices

S: Set of products.

C: Set of customers, including new orders and delayed orders

C_0: Each route starts and ends at the warehouse that is indexed with 0

C_D: Set of delayed orders from the previous day.

C^+: Artificial customer order

V: Set of vehicles.

Parameters

u_{ik}: Number of units of product k, which belongs to an order i.

dis_{ij} : Travelling distance from customer location i to customer location j.

$cost1$: Travelling cost per kilometer.

$cost2_l$: Hiring cost for vehicle l.

p_k: Price of product k.

$ptime_k$: Picking time for each unit of each product k.

$maxcap_{kl}$: Maximum capacity of each type of product k in vehicle l.

$mincap_l$: Minimum capacity of each vehicle l.

t_{ij}: Travelling time from customer location i to customer location j.

n_i: Earliest time of customer location i.

m_i: Latest time of customer location i.

M: Sufficient large number.

Decision variables

y_l : Binary variable, y_l = 1 if the vehicle l is assigned, otherwise 0

cus_{il}: Binary variable, cus_{il} = 1 if customer i is assigned to the vehicle l, otherwise 0

x_{ij}^l: Binary variable, x_{ij}^l = 1 if the vehicle l travels from node i to node j, otherwise 0

z_{ij}: Binary variable, z_{ij} = 1 if the order of customer i is processed immediately before the order of customer j, otherwise 0

re_i: Release time of the delivery order of customer i.

s_l: Starting time of vehicle l at the warehouse

arr_i: Arrival time of service at customer i.

$delay_i$: Binary variable, $delay_i$ = 1 if the order of the customer is delayed, otherwise 0

el_{il}: The variable gets a value for each customer i in each vehicle l , except for the depot. If a vehicle l drives from customer i to customer j, the value of el_{jl} must be bigger than the value of el_{il} [24]

Before introducing the formulated MILP model, we delve into our objective function, centered around profit. This objective function comprises three primary components, each elaborated upon in the subsequent discussion:

First components:

The total revenue function outlines the earnings generated from the delivery of all fulfilled orders.

$$\sum_{i \in C} \sum_{k \in S} \sum_{l \in V} p_k * u_{ik} * cus_{il}$$

Second components:

The traveling expenses are linked to all generated tours.

$$\sum_{l \in V} \sum_{i \in C_0} \sum_{j \in C_0} cost1 * dis_{ij} * x_{ij}^{l}$$

Third components:

The costs related to employing vehicles are contingent upon the number of vehicles utilized. It is important to note that the cost per kilometer traveled remains consistent, while the employment cost varies for each individual vehicle.

$$\sum_{l \in V} cost2_l * \ y_l$$

Now, our proposed MILP model is presented as follows:

Objective Function:

Maximize

$$\sum_{i \in C} \sum_{k \in S} \sum_{l \in V} p_k * u_{ik} * cus_{il} - \sum_{l \in V} \sum_{i \in C_0} \sum_{j \in C_0} cost1 * dis_{ij} * x_{ij}^{l} - \sum_{l \in V} cost2_l * \ y_l$$

Subject to

Constraint 1:

$$\sum_{j \in C^+, j \neq i} z_{ij} = 1, \forall \mathrm{i} \in C^+ \quad (1)$$

Constraint 2:

$$re_i \geq \sum_{k \in S} ptime_k * u_{ik}, \forall \mathrm{i} \in C \quad (2)$$

Constraint 3:

$$re_i \geq re_j + \sum_{k \in S} ptime_k * u_{ik} - M * (1 - z_{ji}), \forall \mathrm{i}, j \in C, i \neq j \quad (3)$$

Constraints (1) to (3) outline the order-picking subproblem. Constraints (2) and (3) specify the time a delivery order is released, determined by the sequence in which picking orders are assigned to a picker.

Constraint 4:

The order picking and vehicle routing subproblems are intertwined, wherein a tour cannot commence until all the relevant orders have been supplied by the warehouse.

$$s_l \geq re_i - M * (1 - cus_{il}), \forall i \in C, l \in V \quad (4)$$

Constraint 5:

According to the specified policy, it is mandatory to deliver the orders that were delayed from the previous day on the following day.

$$\sum_{l \in V} cus_{il} = 1, \quad \forall i \in C_D \quad (5)$$

Constraint 6:

In the vehicle routing subproblem, it guarantees that a customer is allocated to only one vehicle, provided their time window and capacity requirements are met. If these

conditions are not fulfilled, customer i's order will not be assigned to any vehicle that day. Essentially, this means the order will be delayed and delivered the following day.

$$\sum_{l \in V} cus_{il} \leq 1, \ \ \forall i \ \in C \tag{6}$$

Constraint 7:

$$cus_{jl} = \sum_{i \in C_0, i \neq j} x_{ij}^{l}, \forall j \ \in C, l \in V \tag{7}$$

Constraint 8:

$$cus_{jl} = \sum_{i \in C_0, i \neq j} x_{ji}^{l}, \forall j \ \in C, l \in V \tag{8}$$

Constraints (7) and (8) maintain the accuracy of route planning and customer-vehicle assignments. They also ensure the inclusion of the depot in a route that serves at least one customer, thus guaranteeing comprehensive coverage in the delivery network.

Constraint 9:

$$el_{jl} - \ el_{il} \geq \sum_{k \in S} (u_{ik} - maxcap_{kl} * \left(1 - x_{ij}^{l}\right), \ \ \forall i,j \ \in C, l \in V, i \ \neq j \tag{9}$$

Constraint 10:

$$el_{il} \geq \sum_{k \in S} u_{kl} * cus_{il}, \ \ \forall i \in C, l \in V \tag{10}$$

Constraint 11:

$$el_{il} \leq \sum_{k \in S} maxcap_{kl} * cus_{il}, \ \ \forall i \in C, l \in V \tag{11}$$

Constraints (9) - (11) prevent the formation of subroute issues by utilizing the Miller-Tucker-Zemlin (MTZ) subroute elimination formula (AIMMS How-To, n.d.).

Constraint 12:

$$\sum_{i \in C} x_{0i}^{l} \leq y_k, \forall l \in V \tag{12}$$

Constraint 13:

$$\sum_{i \in C} x_{i0}^{l} \leq y_k, \forall l \in V \tag{13}$$

Constraints (12) and (13) specify that each vehicle is limited to undertaking a single tour. These conditions ensure that the path for vehicle k begins and concludes at the depot.

Constraint 14:

$$arr_i \geq s_l + t_{0i} - M * (1 - cus_{il}), \forall i \in C, \forall l \in V \tag{14}$$

Constraint 15:

$$arr_j \geq arr_i + t_{ij} - M * \left(1 - x_{ij}^{l}\right), \forall i, j \in C, i \neq j, l \in V \tag{15}$$

Constraint 16:

$$arr_i - M * (1 - cus_{il}) \leq m_i, \forall i \in C, \forall l \in V \tag{16}$$

Constraint 17:

$$arr_i \geq n_i * cus_{il}, \forall i \in C, \forall l \in V \tag{17}$$

Constraints (14) to (17) determine the earliest time a service can begin at a customer's location, marked as arr_i . This time can't be earlier than when the vehicle arrives or the lower limit of the delivery time window. Furthermore, the service must finish before exceeding the upper limit set by the delivery time window.

Constraint 18:

Considering the limitation of the vehicle's capacity

$$\sum_{i \in C} u_{ik} * cus_{il} \leq maxcap_{kl}, \forall l \in V, \forall k \in S \tag{18}$$

Constraint 19:

$$\sum_{i \in C} cus_{il} \geq y_l, \forall l \in V \quad (19)$$

Constraint 20:

$$\sum_{i \in C} cus_{il} <= y_l * M, \forall l \in V \quad (20)$$

Constraints (19) and (20) decide the utilization of vehicle l. Specifically, if any orders from customer i are assigned to vehicle l, then vehicle l is considered in use (y_l = 1).

Constraint 21:

Indicates the minimum capacity requirement for a vehicle. If the total number of boxes fails to meet this minimum capacity, these specific orders will not be assigned to the vehicle.

$$\sum_{i \in C} \sum_{k \in S} u_{ik} * cus_{il} \geq mincap_l * y_l, \forall l \in V \quad (21)$$

Constraint 22:

Documenting the orders that will experience delays in the present schedule and necessitate delivery on subsequent days.

$$\sum_{l \in V} cus_{il} + delay_i = 1, \forall i \in C \quad (22)$$

4.2. Solution Development – Meta Heuristics Application

The section on meta-heuristics begins with an explanation of how solutions are presented, which consistently follows a three-part structure comprising the route,

separator, and delay order. This format is maintained across the two discussed algorithms. The discourse then advances to the Tabu Search algorithm, emphasizing its utility as a method for local optimization. To enhance the ability to perform global searches, the Genetic Algorithm is introduced. This algorithm, known for its global search proficiency, incorporates elements of Tabu Search within its mutation process to achieve a more robust solution-finding mechanism.

4.2.1. Solution presentation

We propose a solution comprising three elements: node routing, separator, and delayed order processing for the current day. Node routing involves rearranging all existing nodes. In the case of separators, the number of vehicles is influenced, as the position of each cell sequence represents the sequential number of the vehicles used. The separator is expressed in binary, with the number of bits for conversion to decimal determined by the total number of existing nodes to create a binary sequence. Unlike being expressed as an array, the separator is represented as a cell sequence. Similarly, the delayed order for today is also represented in binary form.

In the Tabu Search technique, the strategy is broadened through the straightforward tactic of swapping two neighboring digits. Particularly in binary systems, such swaps can modify the attributes linked to separators and the order of delays, with the impact varying based on the length of the sequence. As a result, vehicle selection from the entire pool is performed on a random basis, ensuring the chosen number aligns with the count of separators increased by one. This method guarantees a cohesive representation by fluidly merging alterations in the sequence with the distribution of resources, thus offering an all-encompassing solution structure. Moreover, this method seamlessly transitions into the application of the Genetic Algorithm, as the crossover mechanisms inherently preserve the necessary value characteristics.

Table 4.1: Solution presentation

Example: 11 node, 4 separator, 5 truck, and 4 delayed orders for today

Route										Separator				Delayed order for today			
3	2	1	5	4	6	9	8	10	7	[1010]	[1110]	[1001]	[1110]	[1111]	[1001]	[1110]	[1010]

Currently, the separators are 10, 14, 9, 14, but since there are only 10 total orders, the distribution must be adjusted. The number 14 should be changed to 6, and duplicates removed, leaving us with separators 4, 9, 10. Furthermore, the delayed orders for today are 15, 9, 10, 14, but only orders numbered 10 or less are considered (reflecting the total orders of the day). Therefore, 9 and 10 will be the delayed orders for the next day.

To sum up, we will the result are:

- Route 1 will be delivered by 1st truck: $3 \rightarrow 2 \rightarrow 1 \rightarrow 5$
- Route 2 will be delivered by 2nd truck: $4 \rightarrow 6 \rightarrow 9 \rightarrow 8 \rightarrow 10$
- Route 3 will be delivered by 3rd truck: 7

This outcome represents a hypothetical scenario and has not undergone verification through the requisite evaluative steps. As it stands, this result is presently impractical. The procedures for assessing considerations will be outlined and discussed in subsequent sections.

4.2.1.1. Permutation of Route Array

Identify all conceivable arrangements of the route array based on the number of orders. For example, when the number of orders is 20, the route array is presented as below.

Table 4.2: Permutation of Route Array

Route array																			
2	3	1	4	6	5	8	9	10	11	12	13	15	17	14	16	18	19	20	7

4.2.1.2. Random Delayed Order for Today

Introducing random delays for orders scheduled for the current day. The number of delayed orders for the following day will be capped at 40% of the total orders for the current day. If a randomly generated order matches a delayed order from the previous day, that specific order will be excluded from the current day's delayed orders, while the remaining orders will be retained. Conversely, if two identical delayed orders for the current day are randomly generated, one order will be kept, and the redundant order will be added to the delayed orders for the day. If random orders mirror those delayed from the previous day, no orders will be delayed for the current day, facilitating the efficient processing of a higher volume of orders, and minimizing delays.

4.2.1.3. Random Separator

At a fundamental level, the allocation of routes, guided by the separator, is determined by a randomly generated location reduced by one unit. The unique expression of the separator in binary form, with the number of bits determined by the total number of nodes, presents a scenario where the separator might exceed the total number of nodes. For instance, a case with 20 nodes where the separator uses 5 bits, resulting in a maximum decimal value of 31. This means decimals 21-31 are not utilized and need to be omitted. To address this, We opted to modify the separator's distribution by reassigning the range of decimals 21-31. In the original distribution, such as in the example (1/31 for each generated decimal), We would alter this distribution by mapping 21 to 2, 22 to 4, 23 to 6, and so forth, up to 31, and then looping back to 2. This adjustment maintains the original randomness while allowing for the reuse of previously discarded decimals, enhancing efficiency in the initialization process of finding a feasible solution.

For instance:

- Trucks: [1 2 3]
- Route: [1 9 8 7 5 4 3 2 6]
- Separator: [5 7]

Resulting in:

- Truck number and Route 1: [1| 1 9 8 7]

- Truck number and Route 2: [2| 5 4]
- Truck number and Route 3: [3| 3 2 6]

The total number of vehicles is determined by adding one to the overall count of separators. By converting and ensuring the uniqueness of these separators, the resulting string to be utilized will be derived.

For instance:

Separator binary = {[1 0 1 1] [1 1 1 0] [1 0 1 0] [1 1 0 0]}

Separator decimal = {11 14 10 12} => Separator transform = {11 4 10 12}

Separator sorted = {4 10 11 12} =>Number of trucks = 5 = [1 2 3 4 5]

4.2.2. Feasibility Check for Initialization/Solution Array

4.2.2.1. Routing Feasibility Check

Verify whether the routes include orders scheduled for delivery today. If not, implement randomization to ensure compliance with today's delivery requirements.

4.2.2.2. Capacity Feasibility Check

Check the feasibility of the total quantity of all orders, excluding delayed orders for today. Evaluate if the total capacity of all trucks is sufficient for the combined orders. If the condition does not match, it will random again the truck selected until it matches the condition.

4.2.2.3. Capacity Feasibility for Each Routing

Perform capacity checks for each routing, excluding today's delayed orders. Ensure that each truck's capacity is not exceeded by two cases:

1. Random again the position of the random separator with fixed iteration to limit the time lapse.
2. If the first case fails, it will move to the second case. In this case, it will random again the truck-selected array. After this case, the feasibility will be met.

4.2.2.4. Time Feasibility for Each Routing

To assess the feasibility of time windows, it's essential to note that the arrival time is composed of release time, pick-up time, and travel time, with two of these being

variable. Specifically, the arrival time is not a component of the objective function, so it only needs to be feasible and not necessarily a specific solution for each node. In the context of considering feasible time windows, the goal is to ensure that the time constraints for each node are satisfied within the specified range.

For example:

Sample route = $0 \rightarrow 3 \rightarrow 5 \rightarrow 1 \rightarrow 0$

Upper (1) = latest time (1) – travel time ($5 \rightarrow 1$)

Lower (1) = earliest time (1) – travel time ($5 \rightarrow 1$)

Resulting in:

New bound for node 5

Latest time (5) = min (upper (1), latest time (5))

Earliest time (5) = max (lower (1), earliest time (5))

Continue with other nodes when each reach the last node does not include 0.

Feasibility is achieved when, after an update, the earliest time is required to be less than the latest time. To simplify the attainment of feasibility, we have incorporated additional heuristics. These heuristics are applied before calculating the new earliest and latest times, and there are two approaches available. To determine which method to use, we allocate 70% for Method 1 and 30% for Method 2:

- Method 1: Shift the node with the highest latest time to the end of the route.
- Method 2: Rearrange all nodes from the smallest to the largest based on their latest time.

4.2.3. Heuristics for Initialization

This heuristic serves as an alternative approach to initialization, offering a remedy to the conventional method of attaining initializations through unrestricted randomization until stipulated conditions are satisfied. This method is deemed viable, leading to the identification of a feasible solution. The process of obtaining a feasible solution is noted to be time-intensive, particularly with an escalation in the number of nodes. This heuristic addresses this challenge by enhancing the efficiency of attaining a feasible solution. It is posited that an augmented number of nodes increases the likelihood of identifying a feasible solution, given the stringent nature of the time-

window constraint, which becomes particularly pronounced when the number of nodes is insufficient. The time window constraint is identified as the most formidable constraint in this problem, followed by capacity constraints, and, finally, order constraints.

4.2.3.1. Truck selection

The user's desired quantity of cars dictates a random selection process from the overall pool of available cars. This approach introduces variability in the solution generation, yielding diverse cost and profit outcomes. The randomization of the car selection is reiterated during each initialization, contributing to the diversification of the approach employed to achieve a feasible solution in the initialization phase.

4.2.3.2. Node and delayed order selection

In contrast to the initial randomization method in the initialization process, heuristics involve a process of decoding followed by re-encoding. The identification of suitable nodes and the establishment of routes involve filtering through specific criteria. Initially, priority is given to assigning orders slated for delivery on a given day, with the constraint that the count of orders permissible for postponement to the next day does not exceed 40% of the total orders for the current day. The set of orders scheduled for delivery on the current day is prioritized and sorted in ascending order based on their earliest delivery times. Subsequently, orders are selected for delivery on the current day, adhering to criteria prioritizing high revenue and early delivery times. Unselected orders are deferred to the following day.

The incremental addition of nodes occurs sequentially from left to right within the original set of nodes. Each added node undergoes an initial check against the earliest time window. The feasibility of the time check chain is then refined by filtering according to the latest arrival time, arranged from smallest to largest, thereby facilitating the attainment of a feasible time chain. If the conditions are met, the process advances to a capacity check; otherwise, the next node is selected. This iterative process continues until the cumulative capacity of the nodes equals the maximum capacity of the vehicle, at which point the next route is initiated.

4.2.3.3. Feasibility

The process of randomly selecting cars from the overall fleet produces a range of diverse solutions, some of which may lack feasibility. Feasibility, in this context, pertains to undelivered orders designated for postponement to the following day. Consequently, the random selection of vehicles and the pursuit of a feasible solution string cease when the orders slated for delivery cannot be accommodated within the deferred sequence for the subsequent day.

4.2.3.4. Separator

The selection of separators will be determined by tallying the nodes that mark the commencement and conclusion of each route. The count of separators will correspond to one less than the total number of vehicles employed.

4.2.3.5. Encoding for Initialization

Conduct encoding after confirming the feasibility of the string:

Preserve the decoded truck string; concerning the route, transform it from a cell into a matrix. For separators, perform encoding from base 10 to base 2, ensuring that the count of encoded strings matches the total number of trucks minus one. In instances where the encoded separators are insufficient, the approach involves either randomly duplicating the available separator strings or setting the entire string to 0. However, preference is given to repetition to enhance algorithmic diversity. Like the delay chain for the following day, the count of chains must equate to 40% of the overall number of orders delivered on the current day.

4.2.4. Tabu Search Algorithm

4.2.4.1. Pseudocode

Tabu Search (TS) is an optimization algorithm inspired by the human intelligence process (Guo, Liu, Liu, & Guo, 2017). However, its search performance relies heavily on the specific domain structure and initial solution, particularly when encountering local minima. This limitation hinders its ability to guarantee global optimization. To address this challenge, TS employs a flexible storage structure and a corresponding tabu criterion. This approach enables TS to efficiently locate local optima within a significantly reduced computational time.

Based on the Algorithm (Local improvement process) (Guo et al., 2017), the pseudo-code of Tabu Search Algorithm performs below:

1. *Set tabu list to null, set* x_0*;*
2. ***repeat***
3. $\{x_1, x_2, \dots, x_n\}$ = *Generate_neighborhood* (x_0)*:*
4. ***for*** $i = 1$ *to* n
5. ***if*** $tabu(i)$ *is* 0*;*
6. ***if*** x_i *meet aspiration criterion;*
7. *replace* x_b *by* x_i*,* $x_0 = x_i$ *and update the tabu list;*
8. ***else if*** x_i *is the best solution in* $\{x_1, x_2, \dots, x_n\}$
9. $x_0 = x_i$ *, update tabu list;*
10. ***endif***
11. ***endif***
12. ***endfor***
13. ***until*** *a termination condition is satisfied*
14. *Return* x_b *as the best individual.*

The Tabu Search algorithm starts by setting an empty tabu list and initializing the current solution x_0 (line 1). Then, it iteratively generates a set of neighboring solutions

$\{x_1, x_2, \ldots, x_n\}$ (lines 2–3) and evaluates each one (lines 4–12). If a candidate solution is not marked as taboo (value in the tabu list is 0), it is checked against the aspiration criterion (line 5-6). If it meets the criterion, it replaces the best solution and becomes the current solution, updating the tabu list in the process. Otherwise, the candidate solution is checked for optimality within the current iteration (line 7). If it is optimal, it replaces the current solution and the tabu list is updated again (line 9). The iterations of tabu search are repeated until a termination condition is met (line 13). Finally, the best solution x_b is returned in line 14.

4.2.4.2. Flow Chart of Tabu Search Algorithm

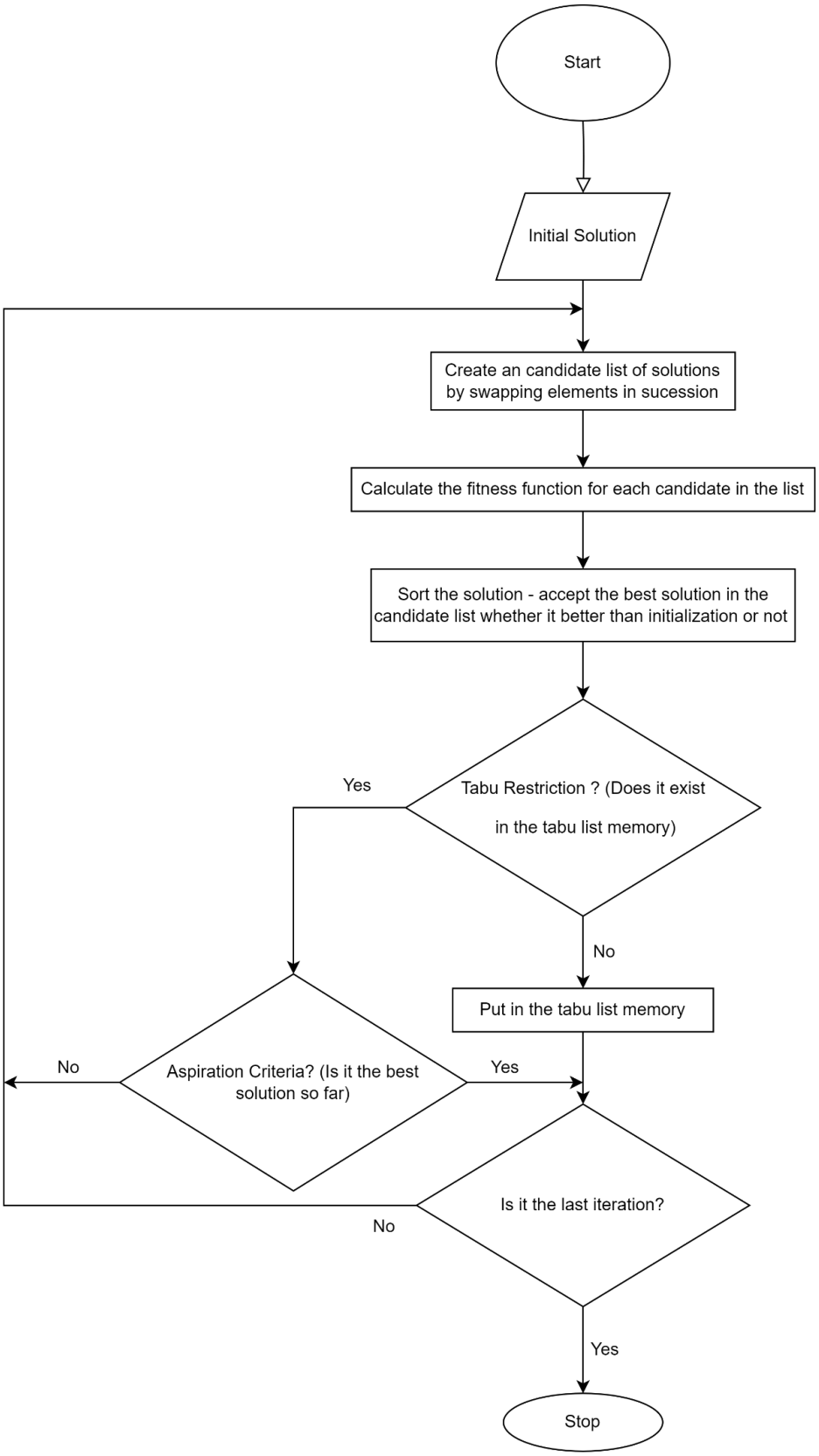

Figure 4.1: Flow chart of Tabu Search Algorithm

Step 1:	**Initialization:** Begin with an encoded string comprising three elements: routing order, separator, and delayed order for the current day. The generation of these strings must adhere to the specified rules outlined earlier. Importantly, the initial string must constitute a feasible sequence, meaning it has successfully passed feasibility checks, including capacity and arrival time validations, to be introduced into the algorithm.
Step 2:	**Calculate Fitness for Initial Array**
Step 3:	**Neighbor Generation:** • Generate a neighboring chain by executing element swaps, albeit with a unique twist due to the distinct nature of this problem. Conventional swapping is unfeasible given the presence of up to three components, each with varying sizes. In this scenario, a distinctive approach is adopted wherein a randomization process, ranging from 1 to 3, is employed to designate the components for neighbor creation. • The three components are categorized as follows: 1 for routing order, 2 for separator, and 3 for delayed order for today. This randomization mechanism introduces an element of unpredictability, ensuring that the swapping operation dynamically selects one of the three components to perform the neighbor creation.
Step 4:	**Tabu List Check:** Determine whether the reversed positions are in the tabu list. If so, exclude them from the comparison in assessing neighbor fitness. Positions not in the tabu list proceed.
Step 5:	**Feasibility Check**

	Evaluate the feasibility of each generated neighbor chain, considering capacity and time window considerations. This step is crucial to ensure that the ultimate solution comprises a sequence of improved results while maintaining feasibility, considering factors such as capacity constraints and adherence to time windows. In the event of the worst-case scenario where the strings become infeasible after the reversal operation, they will revert to the initial feasible string, initiating a repetitive cycle until a viable string is obtained to resume the iterative process.
Step 6:	**Fitness Computation for Non-Tabu and Feasible Neighbors**
Step 7:	**Comparison and Selection:** Evaluate the fitness values of neighbors against the initialization string. Two scenarios: 1. If all neighbors surpass the fitness of the initialization string, select the best among them. 2. If neighbors don't outperform the initialization string, still choose the best neighbor to prevent local traps.
Step 8:	**Save Inverse Position to Tabu List**
Step 9:	**Update Initialization String:** Set the selected string as the new initialization string for the ongoing algorithm.
Step 10:	**Termination Criteria:** Terminate under the following conditions: • Exhaustion of potential solutions in the vicinity. • Exceedance of the specified iteration limit. • Surpassing the defined threshold of consecutive iterations without enhancing the original solution. Completion of any of these conditions signifies the attainment of an optimal solution.

4.2.5. Genetics Algorithm - Tabu Search Hybrid

The genetic algorithm is an approach for addressing both constrained and unconstrained optimization problems, inspired by natural selection in biological evolution. This method involves iteratively modifying a population of individual solutions. In each iteration, individuals from the current population are chosen as parents to generate the offspring for the next generation. Across successive generations, the population undergoes an "evolutionary" process, gradually converging toward an optimal solution. Genetic algorithms (GAs) have found extensive use in contemporary metaheuristics, with numerous applications reported in solving Vehicle Routing Problems (VRP) that include time windows. Additionally, GAs have been applied to VRP variations such as those involving backhauls, multiple warehouses, and complex routing scenarios. (Baker & Ayechew, 2003)

This study aims to enhance the efficiency of population convergence in genetic algorithms by integrating Tabu search into the mutation phase. By doing so, the research accelerates the process of finding superior solutions compared to relying solely on local variable exploration. The integration of Tabu search aims to expedite convergence, ultimately improving the number of iterations required for populations to reach an optimal solution.

4.2.5.1. Pseudocode

Input:

Size α of population.

Rate β of elitism.

Rate ∂ of children creating.

Number δ of iterations.

Number ρ of outer loop mutation.

Number σ of inner loop mutation.

Output:

Solution X

// Initialization

1. Generate α feasible solutions based on heuristics with 50% randomly and 50% heuristics.
2. Save them into a population.
3. Calculate fitness function for whole solutions in population.
4. Sort the population ascending.
5. Save the global variables.

// Start the GA iteration

6. **for** i = 1 **to** δ **do**
7. **for** l = 1 **to** $\alpha * \partial$ **do**

// Selection

8. Randomly select two solutions X_A and X_B from population.

// Crossover

9. Generate 2 children X_C and X_D by crossover to X_A and X_B.
10. Select randomly between X_C and X_D to continue.

// Mutation

11. **for** j = 1 **to** ρ **do**
12. Random among 3 parts of the solution array to continue.
13. **for** k = 1 **to** σ **do**

```
                    // tabu search application
14.                     Set tabu list to null, set x_0;
15.                     {x_1, x_2, ..., x_n} = Generate_neighborhood (x_0):
16.                     if tabu(i) is 0:
17.                         Remove the solutions
18.                     elseif
19.                         Save into the population.
20.                     endif
21.                     Calculation the fitness function
22.                     If solution > global variables
23.                         Save the new global variables.
24.                         Update tabu list.
25.                     elseif
26.                         Save the best solution among solutions.
27.                     endif
28.                 endfor
29.                 Continue with the closest global variables.
30.             endfor
31.             Save the solutions to population Pop_child

            // Elitism
32.         for n = 1 to β * α do
33.             Sort the solution in Pop_child descending.
34.             Replacement
35.         endfor
36. endfor
37. until a termination condition is satisfied
```

The pseudo-code for the Genetic Algorithm (Zaritsky & Sipper, 2004) provides insight into the role of each component within the algorithm, with a special emphasis on the mutation stage. We intend to enhance this pseudo-code by integrating elements of the Tabu Search algorithm, creating a hybrid Genetic Algorithm - Tabu Search approach.

The following notes offer explanations for the terminology used in the pseudo-code, aiding in a clearer understanding of the process. This adaptation aims to leverage the strengths of both algorithms for improved performance.

- α (Population size): This denotes the total number of solutions (also known as individuals) in the population at any given time. The population size remains constant across generations in this algorithm.
- β (Elitism rate): This is the proportion of the population that is carried over to the next generation without any changes, based on their fitness. It's a way to ensure that the best solutions are preserved. The number of elite solutions per generation is calculated as $\beta * \alpha$.
- ∂ (Rate of children creating): This rate determines how many children are created in each generation as a proportion of the population size. The total number of children created per generation is $\partial * \alpha$.
- δ (Number of iterations): This represents the total number of generations or iterations the GA will run for, unless a termination condition is met earlier.
- ρ (Number of outer loop mutations): In the context of the mutation process, ρ is the number of times the outer mutation loop will run for each child solution. This outer loop allows for multiple segments of a solution to be considered for mutation.
- σ (Number of inner loop mutations): Within each outer loop mutation, σ represents the number of mutations or modifications attempted on the selected segment of the solution.
- Solution X: This is the output of the GA, representing the best solution found to the problem after all iterations.
- X_A and X_B. (Parent solutions): These are randomly selected solutions from the current population that are used for crossover to generate new child solutions (X_C and X_D).
- X_C and X_D (Child solutions): These are new solutions generated by performing crossover between two parent solutions (X_A and X_B). One of these children is randomly selected for further mutation and evaluation.

- Pop_child: This is a temporary population used to store the newly created and mutated child solutions before they are considered for inclusion in the main population through the elitism process.
- Tabu list: In the context of mutation with tabu search, the tabu list is a mechanism to avoid revisiting recently explored or undesirable solutions, thereby encouraging the exploration of new areas in the solution space.
- $\{x_1, x_2, \dots, x_n\}$: These represent solutions about the current solution being mutated. x_0 is the starting solution for mutation, and $\{x_1, x_2, \dots, x_n\}$ are new solutions generated through a neighborhood search around x_0.

4.2.5.2. Flow Chart of Genetics Algorithm -Tabu Search Hybrid

Drawing upon the findings of Genetics Algorithm Flow Chart (Dastanpour & Raja Mahmood, 2013), the workflow diagram of the Genetic Algorithm will be revised to incorporate the execution of the Tabu Search methodology within the mutation segment. This modification aims to enhance the algorithm's performance by integrating a sophisticated search mechanism during the mutation process, as outlined in the scholarly discourse.

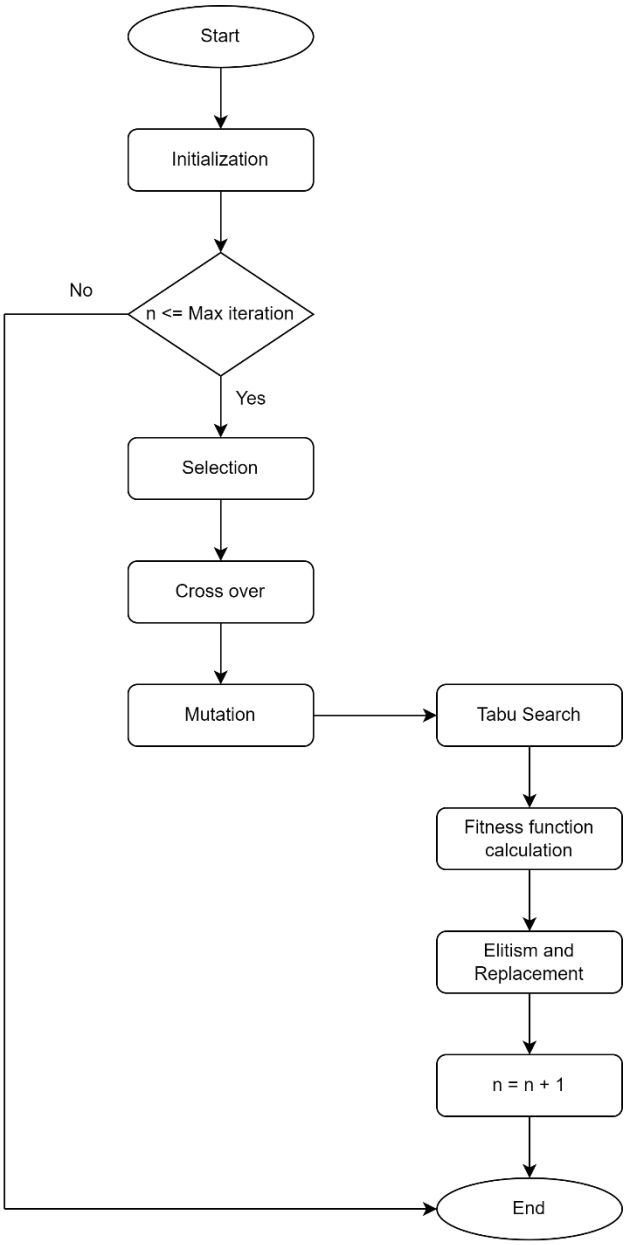

Figure 4.2: Flow Chart of Genetics Algorithm Combine Tabu Search

Step 1:	**Initialization** Begin the population initialization process by incorporating a mix of 50% heuristics-based individuals and 50% randomly generated individuals. The population must include at least one feasible chain, ensuring it adheres to constraints related to time, orders, and capacity. Note that while the randomly generated strings may not inherently satisfy these constraints, the goal is to have at least one string within the population that does. Following the population setup, proceed to compute the fitness function for the entire population. Introduce a new variable, denoted as Big M, which serves as a significant numerical value. For the strings generated randomly and deemed infeasible, subtract their respective profits from the Big M value. This approach distinguishes between feasible and infeasible sequences within the population, facilitating sorting procedures and guiding the replacement process. Essentially, the utilization of Big M allows for the identification and treatment of infeasible solutions during subsequent stages of the algorithm.
Step 2:	**Parent Selection** A parent string is chosen by selecting any two strings from the population, and if the same position is inadvertently chosen, the selection process is repeated. From these two parent chains, two children are generated. Subsequently, one of the two children is randomly chosen to proceed with the subsequent steps. In unique and infrequent situations where different positions are selected, but the two chosen strings are identical, any one of the two strings is chosen to undergo the next steps. This decision is based on their inherent properties and specific rules established in heuristics. Consequently, the generation of two identical consecutive sequences is feasible under certain conditions.
Step 3:	**Crossover:** 1. **Routing Component Crossover:**

	• A part of the routing component is selected from one of the two parent entities. The selection is random due to the integral nature of this component. • The selection process involves three steps: 1. A random size is chosen, ranging from 1 to the length of the routing string minus 2. This size determines which segment of the routing component will be retained in the child entity. The idea is to preserve potentially beneficial segments (genes) from the parent without knowing their exact location. Limiting the size to the string length minus 2 ensures that not the entire string is kept, promoting variability. 2. A starting point for this segment is then randomly chosen, ensuring the selected segment fits within the routing component's length. 3. For the remaining orders in the routing, a simple permutation is applied to rearrange these elements in the original string to produce variability. 2. **Crossover for Delay and Delimiter Components:** • Since these components are binary in nature, their crossover involves combining segments from both parent entities to form the child entity's corresponding components.
Step 4:	**Mutation:** The mutation phase involves employing an internal tabu-search, incorporating a distinct component chain. In this context, the re-tabu operation is executed many times in the outer loop (exploration) and many times in the inner loop (exploitation), fostering a balanced exploration and exploitation dynamic. This approach aims to synchronize the levels of exploration and exploitation while optimizing the runtime efficiency of the algorithm. Given the nested structure of these algorithms, an excessive

	number of tabu search iterations could impede the overall performance of the external algorithm, GA, in delivering timely results. During the mutation process, a string repair procedure is implemented once to align with specified constraints. This not only accelerates the convergence towards a superior solution but also enhances the efficiency of the overall algorithm.
Step 5:	**Fitness Function Calculation** Implement and evaluate the objective function, often referred to as the fitness function, and record its value.
Step 6:	**Elitism and Replacement** Generate six substrings from the initial population, assess the fitness function for each substring, and execute elitism by arranging them in descending order based on overall fitness. Compare and selectively substitute up to four children back into the original population. This deliberate replacement strategy ensures that, at most, four out of the six offspring integrate into the original population. This precautionary measure prevents the algorithm from prematurely converging to a provisional outcome, allowing sufficient time for comprehensive exploration and exploitation of the solution space. Consequently, the algorithm avoids settling for a locally optimal solution and strives to attain a globally optimal solution.

CHAPTER 5. RESULT ANALYSIS

5.1. Experimental Design

5.1.1. Data collection and Data Processing

- Data is acquired and processed from an anonymous entity, denoted as company U, in adherence to an agreement about information security and data integrity. This dataset is comprehensive, providing intricate details of a specific volume of orders during a typical working day. It is important to note that the volume of orders may vary, either increasing or decreasing based on dynamic data inputs. Additionally, orders delayed from the preceding day, necessitating processing on the subsequent day, are incorporated into the model's dataset, receiving distinct order designations.
- The company transitioned from a multichannel retail model to a centralized warehouse system for fulfilling online orders. This strategic shift is analyzed using data on customer interactions, warehouse operations, and logistics from the company's records. On average, the company processes 109 online orders daily, with a structured dispatch schedule and specific picking times for different product types (2 minutes for dry and fresh products, 2.5 minutes for frozen goods). A key logistic feature is the partnership with a third-party for a fleet of 12 multi-compartment trucks, catering to varying temperature needs for dry, fresh, and frozen products. This moves from owning just two vehicles to outsourcing reflects a strategic adaptation to growing demand. Costs include a $0.9 per kilometer transportation fee and a variable vehicle usage charge based on vehicle features. GPS technology is employed for precise delivery locations.
- Time values will be adjusted according to the following guidelines: Time is explicitly provided. The selection process commences at 6 a.m., while the final delivery to the customer's residence concludes at 8 p.m. Additionally, the recording of time will be temporarily halted at 5 p.m. on the preceding day:

Table 5.1: Time window change from model to real

Time window change		
Slots	Model (hours)	Real (hours)
1	2 -> 4	8 -> 10
2	4 -> 6	10 -> 12
3	6 -> 8	12 -> 14
4	8 -> 10	14 -> 16
5	10 - > 12	16 -> 18
6	12 -> 14	18 -> 20

5.1.2. Implementation

The company employs an SAP system for order management, tracking both new and delayed orders. Employees extract order data to organize daily logistics, prioritizing delayed orders based on their receipt dates. As illustrated, delayed orders (1, 2, 3) are addressed before processing new ones (4, 5, 6, 7). The SAP system efficiently updates by clearing completed orders while retaining any outstanding ones, eliminating the need for manual oversight.

Table 5.2: Implementation of data structures

Order	Product 1	Product 2	Product 3
1	4	2	1
2	1	2	2
3	1	2	3
4	1	2	1
5	3	1	2
6	1	1	1
7	1	2	3

This section aims to evaluate the model's efficiency and responsiveness through a linear programming approach implemented in AMPL. Given time constraints, the dataset,

specifically the total number of orders and vehicles, will be condensed to 10 orders for this performance assessment, while all other parameters remain constant.

5.1.3. Result of the experiments with small instances

In the following we present the results from our test instances based on data generated.

Instance 1: 10 total orders, 4 delayed orders, 6 vehicles

The objective value is $1032.27, and total cost is $204.725.

Table 5.3: Delay orders from instance 1

Order	1	2	3	4	5	6	7	8	9	10
Delay for the following day	0	0	0	0	0	0	0	0	0	0

Table 5.4: Sequence of route is processed by vehicle 1 from instance 1

	Depot (0)	1	2	3	4	5	6	7	8	9	10
Depot (0)	0	1	0	0	0	0	0	0	0	0	0
1	0	0	0	0	0	0	1	0	0	0	0
2	0	0	0	0	0	0	0	0	0	0	0
3	0	0	0	0	0	0	0	0	0	0	0
4	0	0	0	0	0	1	0	0	0	0	0
5	1	0	0	0	0	0	0	0	0	0	0
6	0	0	0	0	0	0	0	0	0	1	0
7	0	0	0	0	0	0	0	0	0	0	0
8	0	0	0	0	0	0	0	0	0	0	0
9	0	0	0	0	1	0	0	0	0	0	0
10	0	0	0	0	0	0	0	0	0	0	0

Table 5.5: Sequence of route is processed by vehicle 4 from instance 1

	Depot (0)	1	2	3	4	5	6	7	8	9	10
Depot (0)	0	0	0	0	0	0	0	0	0	0	1
1	0	0	0	0	0	0	0	0	0	0	0
2	0	0	0	0	0	0	0	0	1	0	0
3	1	0	0	0	0	0	0	0	0	0	0
4	0	0	0	0	0	0	0	0	0	0	0
5	0	0	0	0	0	0	0	0	0	0	0
6	0	0	0	0	0	0	0	0	0	0	0
7	0	0	1	0	0	0	0	0	0	0	0
8	0	0	0	1	0	0	0	0	0	0	0
9	0	0	0	0	0	0	0	0	0	0	0
10	0	0	0	0	0	0	0	1	0	0	0

Two out of the six available vehicles are strategically chosen to fulfill orders, ensuring both optimal value and punctuality. With six vehicles at its disposal, the company expertly selects the most appropriate vehicle for each order, considering the volume and the need to minimize costs. Notably, vehicles 1 and 4 are identified as the most economical options. According to the provided information, vehicle 4 is designated for delivering orders that share the same destination and time window, such as departments or apartments. This targeted approach greatly reduces travel expenses and increases operational efficiency by consolidating deliveries, thereby limiting costs to the routes between the warehouse and the specified delivery points within the same area and timeframe.

Instance 2: 10 total orders, 4 delayed orders, 1 vehicle

The objective value is $627.482, and total cost is $122. 518.

Table 5.6: Delay orders from instance 2

Order	1	2	3	4	5	6	7	8	9	10
Delay for the following day	0	0	0	0	1	1	1	0	0	1

Table 5.7: Sequence of route is processed by vehicle 1 from instance 2

	Depot (0)	1	2	3	4	5	6	7	8	9	10
Depot (0)	0	0	0	0	0	0	0	0	1	0	0
1	0	0	0	1	0	0	0	0	0	0	0
2	0	1	0	0	0	0	0	0	0	0	0
3	0	0	0	0	0	0	0	0	0	1	0
4	1	0	0	0	0	0	0	0	0	0	0
5	0	0	0	0	0	0	0	0	0	0	0
6	0	0	0	0	0	0	0	0	0	0	0
7	0	0	0	0	0	0	0	0	0	0	0
8	0	0	1	0	0	0	0	0	0	0	0
9	0	0	0	0	1	0	0	0	0	0	0
10	0	0	0	0	0	0	0	0	0	0	0

In this scenario, the company has access to just one vehicle for deliveries. Four orders, specifically numbers 5, 6, 7, and 10, have been deferred to the following day. Despite these postponements, the model in use guarantees the complete delivery of all orders that were delayed from the prior day. In practical situations, employees might

depend solely on their experience to decide which deliveries to prioritize. Yet, this system offers a more strategic approach, enabling the selection of the most advantageous orders. This might occasionally lead to breaches in minimum or maximum capacity constraints or the designated time frame. Nevertheless, the overarching goal is to maximize profits while simultaneously reducing costs.

Analysis:

The primary goal of this experiment is to highlight that, unlike the conventional Vehicle Routing Problem (VRP) where a failed delivery typically leads to the infeasibility or exclusion of the customer from the service, real-world delivery scenarios are more flexible. In practice, deliveries might not always be completed on the first attempt, often due to issues on the customer's end. For instance, consider a scenario where customer A requests a delivery to location B between 5-7 PM, but has unexpectedly left by 4 PM, resulting in a missed delivery. In such cases, the practical approach is to reschedule the delivery for the same customer the next day within the same time window. Recognizing these real-world complexities, the VRP model has been adapted to accommodate the possibility of redelivering or postponing orders to the following day. This adjustment makes the VRP more applicable and realistic by aligning it closer to actual delivery challenges and customer behaviors.

5.2. Result Illustration and Explaination

5.2.1. Mathematical model

In addressing our routing problem, we focused on a manageable scenario involving 15 nodes and a selection from 12 available vehicles, the maximum our solver could efficiently handle. The optimization led to a significant profit of $1,701.255, achieved in 49 seconds of computation time. The results, detailed in tables, offer a clear view of the routing decisions and vehicle assignments that led to this optimal outcome. This concise presentation of data demonstrates the effectiveness of our optimization approach and provides valuable insights for future route planning and fleet management strategies within similar contexts. In summary, our optimization process effectively

managed a complex routing scenario, resulting in substantial profit and demonstrating the potential of advanced routing solutions in logistical planning.

Table 5.8: Truck use for delivering

Truck Number	Truck Used (1 is used/ 0 is unused)
1	0
2	0
3	0
4	0
5	0
6	0
7	1
8	0
9	0
10	0
11	0
12	1

Table 5.9: Arrangement of orders to each truck

Order	Truck Number	
	Truck Used (1 is used/ 0 is unused)	
	5	7
1	1	0
2	0	1

3	1	0
4	1	0
5	1	0
6	1	0
7	0	1
8	0	1
9	1	0
10	0	1
11	0	1
12	0	1
13	1	0
14	0	1
15	1	0

Table 5.10: Delivery route of truck 5

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1
1	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0
2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
3	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0
4	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0
5	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

6	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0
7	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
8	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
9	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0
10	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
11	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
12	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
13	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0
14	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
15	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 5.11: Delivery route of truck 7

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0
1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
2	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0
3	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
4	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
5	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
6	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
7	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0
8	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0

9	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
10	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0
11	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0
12	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
13	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
14	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0
15	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 5.12: Related time and delay orders

	Release time of the delivery order (minutes)	Starting time of truck at warehouse (minutes)	Arrival time (minutes)	Delayed today
1	14.5	-	480	0
2	11	-	240	0
3	13.5	-	600	0
4	8.5	-	730.5	0
5	13	21.5	840	0
6	6.5	-	642.29	0
7	13.5	21.5	240	0
8	14.5	-	240	0

9	15	-	720	0
10	10.5	-	240	0
11	21.5	-	305.1	0
12	9	-	600	0
13	12.5	-	600	0
14	15	-	120	0
15	10.5	-	455.25	0

5.2.2. Meta-Heuristics

5.2.2.1. Validate Tabu Search and Genetics Algorithm – Tabu Search with Mathematical Model

To assess the effectiveness of the Genetic Algorithm – Tabu Search hybrid and compare it with the mathematical model, we configured the Tabu Search with a detailed setting that includes an inner loop count of 7 for intensive search and an outer loop count of 5 for broader exploration. Each iteration of this process involves two nested loops and utilizes a Tabu list of size 2 to manage previously explored solutions, with a particular focus on the execution time of the algorithm.

Simultaneously, the Genetic Algorithm component is set to operate with a population size of 20, generating a subset of 14 strings per iteration and undergoing a replacement process for 7 of these strings to encourage diversity and exploration in the solution space.

The effectiveness and efficiency of these settings will be benchmarked against the outcomes derived from the mathematical model. The context for this evaluation involves a scenario with 15 distinct delivery orders, from which selections are made among 12 possible vehicles. The findings from this comparative analysis are systematically presented in the subsequent table, providing a clear and direct

comparison of the results obtained from both the hybrid algorithm and the mathematical model.

Table 5.13: Acquisition time of approach solution

Number of run time	Genetics Algorithm -Tabu Search ($)	Computation Time (seconds)	$GAP = \frac{(Solver\ result - Meta\ heuristics\ result}{Solver\ result}$
1	1540.82	18.29	0.094
2	1541.46	20.27	0.094
3	1550.30	19.12	0.089
4	1572.25	19.45	0.076
5	1512.44	19.16	0.111
6	1553.36	18.52	0.087
7	1559.21	17.12	0.083
8	1572.25	20.31	0.076
9	1526.94	19.17	0.102
10	1572.25	18.25	0.076
11	1551.47	16.31	0.088
12	1554.77	19.45	0.086
13	1576.69	19.32	0.073
14	1549.40	22.40	0.089
15	1537.03	20.65	0.097
16	1553.36	22.92	0.087
17	1533.38	21.11	0.099
18	1545.17	20.66	0.092
19	1559.30	18.49	0.083
20	1572.25	19.13	0.076
21	1572.25	21.73	0.076
22	1542.85	19.25	0.093
23	1572.25	19.29	0.076
24	1533.69	20.64	0.098
25	1550.30	19.91	0.089
26	1572.25	20.12	0.076
27	1551.47	20.25	0.088
28	1517.44	20.46	0.108
29	1548.92	18.79	0.090
30	1546.25	18.04	0.091
31	1559.69	19.94	0.083
32	1554.77	19.95	0.086

33	1559.30	19.34	0.083
34	1563.51	19.51	0.081
35	1554.77	18.19	0.086
36	1549.40	22.40	0.089
37	1537.03	20.65	0.097
38	1553.36	22.92	0.087
39	1533.38	21.11	0.099
40	1545.17	20.66	0.092
41	1559.30	18.49	0.083
42	1572.25	19.13	0.076
43	1572.25	21.73	0.076
44	1542.85	19.25	0.093
45	1572.25	19.29	0.076
46	1540.82	18.29	0.094
47	1541.46	20.27	0.094
AVERAGE	1552.69	19.80	0.087

After conducting 45 different trials, the Genetic Algorithm – Tabu Search hybrid method yielded an average profit of $1,552.69, with an average completion time of 19.80 seconds. This performance shows an 8% gap compared to the results obtained from the mathematical model. Notably, the hybrid approach required less than half the time taken by the traditional solver (49 seconds) to reach a solution, achieving approximately 91% of the solver's efficiency.

This demonstrates a significant trade-off where the hybrid method offers considerable savings in time while still delivering near-optimal results. Importantly, while the solver is constrained to its current maximum scale and cannot handle larger problems, the Metaheuristic approach, exemplified by the Genetic Algorithm – Tabu Search method, shows promise for tackling larger-scale problems. The potential for scaling up this hybrid method will be explored in the following section.

5.2.2.2. Solvability of Meta Heuristics

To effectively demonstrate the enhancement and capability to process larger datasets, we will apply both Tabu Search and the Genetic Algorithm-Tabu Search (GA-Tabu) on varying scales of datasets, specifically for 20, 50, 70, and 100 orders. The fleet comprises 12 vehicles, a mix of company-owned and third-party vehicles, with the constraint that the number of orders carried over from the previous day does not exceed

40% of today's order volume. The datasets are sourced directly from Company U, ensuring a consistent basis for comparison. This setup is designed to highlight the superior performance and solutions offered by GA-Tabu compared to the traditional Tabu Search method, showcasing its improved efficiency in handling complex logistics and scheduling challenges.

TABU SEARCH:

Instance 1: 20 total orders, 8 delayed orders, 12 vehicles

The objective value is $2109.75900000000.

Time lapse: 279.840664 seconds.

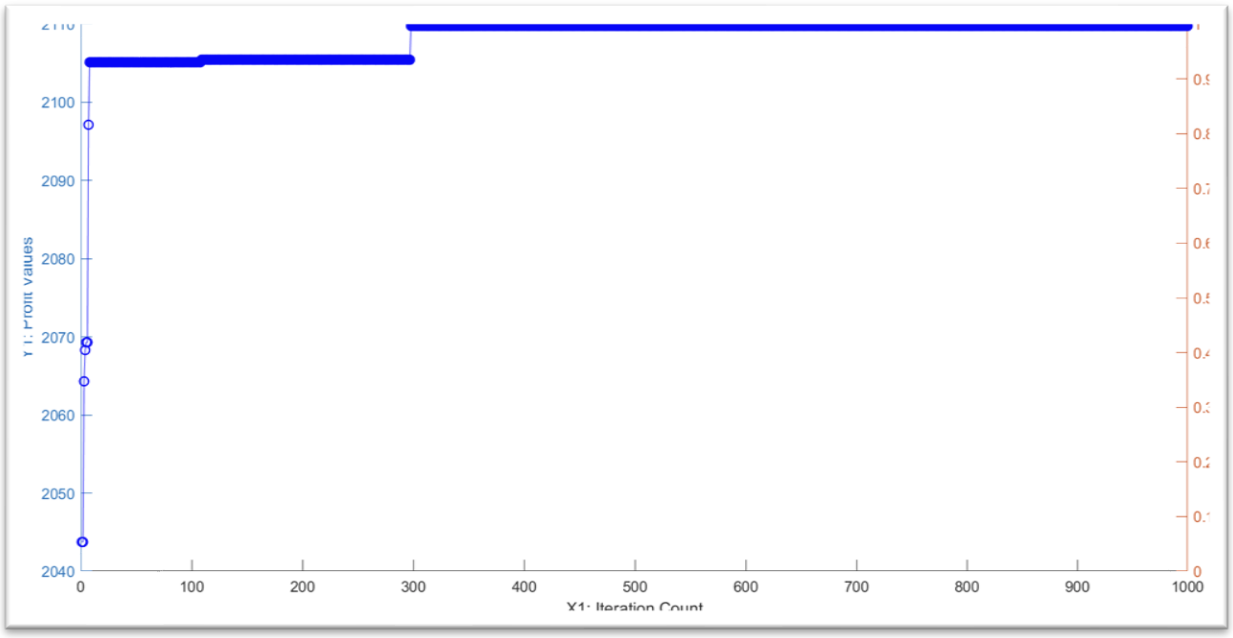

Figure 5.1: Solution record of Instance 1 by Tabu Search

Routing:

- Sequence of route is processed by vehicle 3 from instance 1 $(0 \rightarrow 14 \rightarrow 19 \rightarrow 2 \rightarrow 7 \rightarrow 10 \rightarrow 1 \rightarrow 0)$
- Sequence of route is processed by vehicle 6 from instance 1 $(0 \rightarrow 8 \rightarrow 20 \rightarrow 11 \rightarrow 17 \rightarrow 15 \rightarrow 0)$
- Sequence of route is processed by vehicle 7 from instance 1 $(0 \rightarrow 3 \rightarrow 18 \rightarrow 9 \rightarrow 6 \rightarrow 12 \rightarrow 0)$
- Sequence of route is processed by vehicle 4 from instance 1 $(0 \rightarrow 16 \rightarrow 13 \rightarrow 4 \rightarrow 5 \rightarrow 0)$

Delay order: None.

Instance 2: 50 total orders, 20 delayed orders, 12 vehicles

The objective value is $ 4599.32714500000.

Time lapse: 784.033145 seconds.

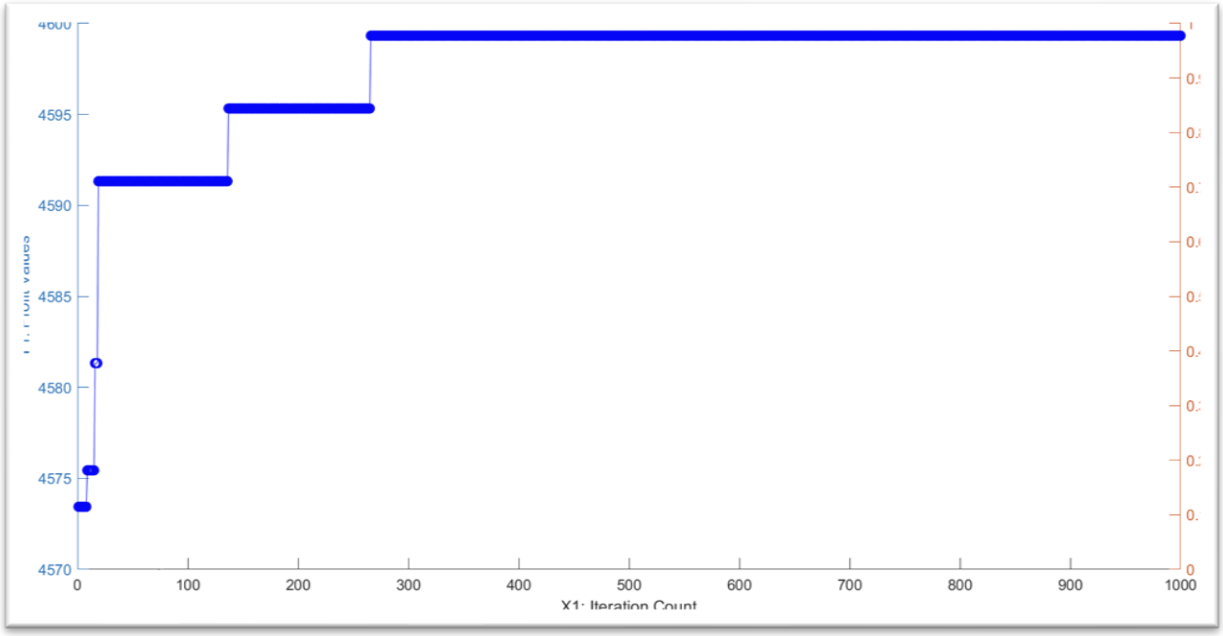

Figure 5.2: Solution record of Instance 2 by Tabu Search

Routing:

- Sequence of route is processed by vehicle 10 from instance 2 $(0 \rightarrow 14 \rightarrow 23 \rightarrow 19 \rightarrow 2 \rightarrow 7 \rightarrow 0)$
- Sequence of route is processed by vehicle 9 from instance 2 $(0 \rightarrow 8 \rightarrow 20 \rightarrow 26 \rightarrow 40 \rightarrow 33 \rightarrow 10 \rightarrow 11 \rightarrow 17 \rightarrow 15 \rightarrow 31 \rightarrow 0)$
- Sequence of route is processed by vehicle 8 from instance 2 $(0 \rightarrow 22 \rightarrow 24 \rightarrow 27 \rightarrow 29 \rightarrow 36 \rightarrow 48 \rightarrow 28 \rightarrow 34 \rightarrow 3 \rightarrow 0)$
- Sequence of route is processed by vehicle 3 from instance 2 $(0 \rightarrow 35 \rightarrow 37 \rightarrow 46 \rightarrow 39 \rightarrow 32 \rightarrow 18 \rightarrow 0)$

- Sequence of route is processed by vehicle 2 from instance 2 $(0 \rightarrow 25 \rightarrow 6 \rightarrow 9 \rightarrow 21 \rightarrow 13 \rightarrow 38 \rightarrow 39 \rightarrow 43 \rightarrow 0)$
- Sequence of route is processed by vehicle 7 from instance 2 $(0 \rightarrow 12 \rightarrow 16 \rightarrow 41 \rightarrow 47 \rightarrow 50 \rightarrow 4 \rightarrow 5 \rightarrow 0)$

Delay orders: [30, 45, 44, 42]

Instance 3: 70 total orders, 28 delayed orders, 12 vehicles

The objective value is $ 6084.400.

Time lapse: 641.467903 seconds.

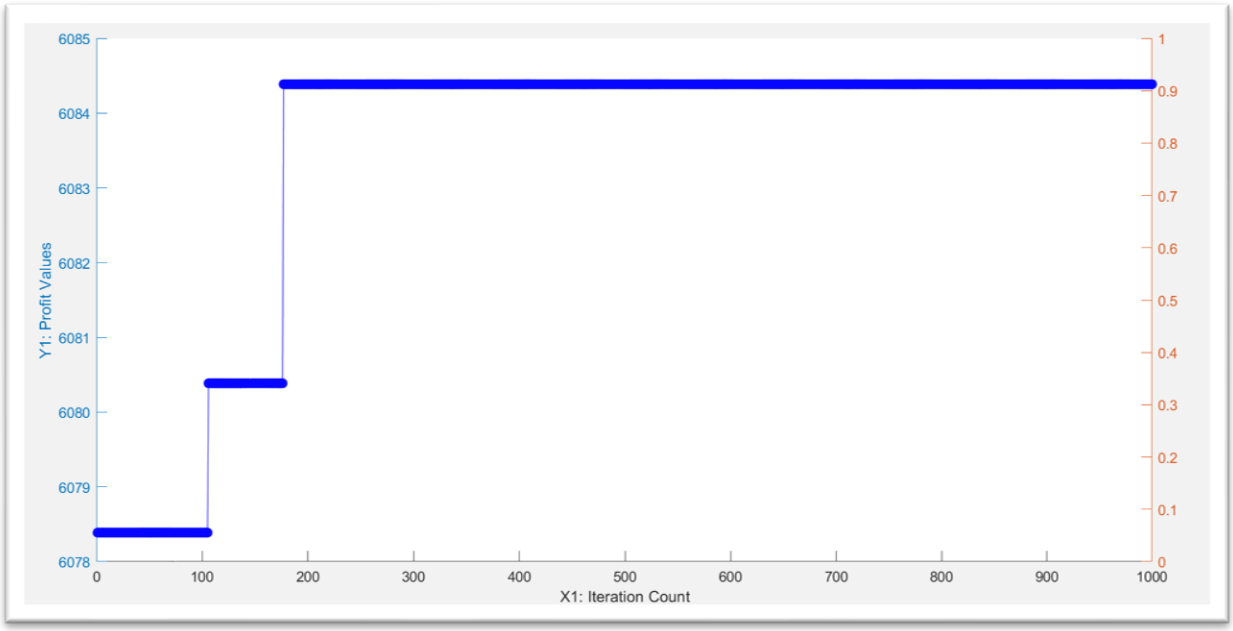

Figure 5.3: Solution record of Instance 3 by Tabu Search

Routing:

- Sequence of route is processed by vehicle 10 from instance 3 $(0 \rightarrow 8 \rightarrow 14 \rightarrow 2 \rightarrow 7 \rightarrow 10 \rightarrow 1 \rightarrow 15 \rightarrow 0)$
- Sequence of route is processed by vehicle 6 from instance 3 $(0 \rightarrow 19 \rightarrow 20 \rightarrow 26 \rightarrow 40 \rightarrow 61 \rightarrow 66 \rightarrow 68 \rightarrow 11 \rightarrow 17 \rightarrow 33 \rightarrow 35 \rightarrow 0)$
- Sequence of route is processed by vehicle 7 from instance 3 $(0 \rightarrow 3 \rightarrow 18 \rightarrow 6 \rightarrow 9 \rightarrow 12 \rightarrow 4 \rightarrow 0)$
- Sequence of route is processed by vehicle 8 from instance 3 $(0 \rightarrow 23 \rightarrow 22 \rightarrow 27 \rightarrow 25 \rightarrow 13 \rightarrow 16 \rightarrow 5 \rightarrow 0)$

- Sequence of route is processed by vehicle 1 from instance 3 $(0 \rightarrow 33 \rightarrow 52 \rightarrow 24 \rightarrow 29 \rightarrow 36 \rightarrow 48 \rightarrow 28 \rightarrow 0)$
- Sequence of route is processed by vehicle 3 from instance 3 $(0 \rightarrow 55 \rightarrow 53 \rightarrow 54 \rightarrow 59 \rightarrow 34 \rightarrow 37 \rightarrow 0)$
- Sequence of route is processed by vehicle 2 from instance 3 $(0 \rightarrow 69 \rightarrow 46 \rightarrow 49 \rightarrow 63 \rightarrow 64 \rightarrow 32 \rightarrow 58 \rightarrow 21 \rightarrow 38 \rightarrow 0)$
- Sequence of route is processed by vehicle 11 from instance 3 $(0 \rightarrow 39 \rightarrow 41 \rightarrow 43 \rightarrow 44 \rightarrow 45 \rightarrow 30 \rightarrow 51 \rightarrow 0)$
- Sequence of route is processed by vehicle 4 from instance 3 $(0 \rightarrow 47 \rightarrow 50 \rightarrow 57 \rightarrow 60 \rightarrow 42 \rightarrow 56 \rightarrow 0)$

Delay orders: [62, 65, 67, 70]

GENETICS ALGORITHM – TABU SEARCH

Instance 1: 20 total orders, 8 delayed orders, 12 vehicles

The objective value is $2128,7000.

Time lapse: 452.073255 seconds

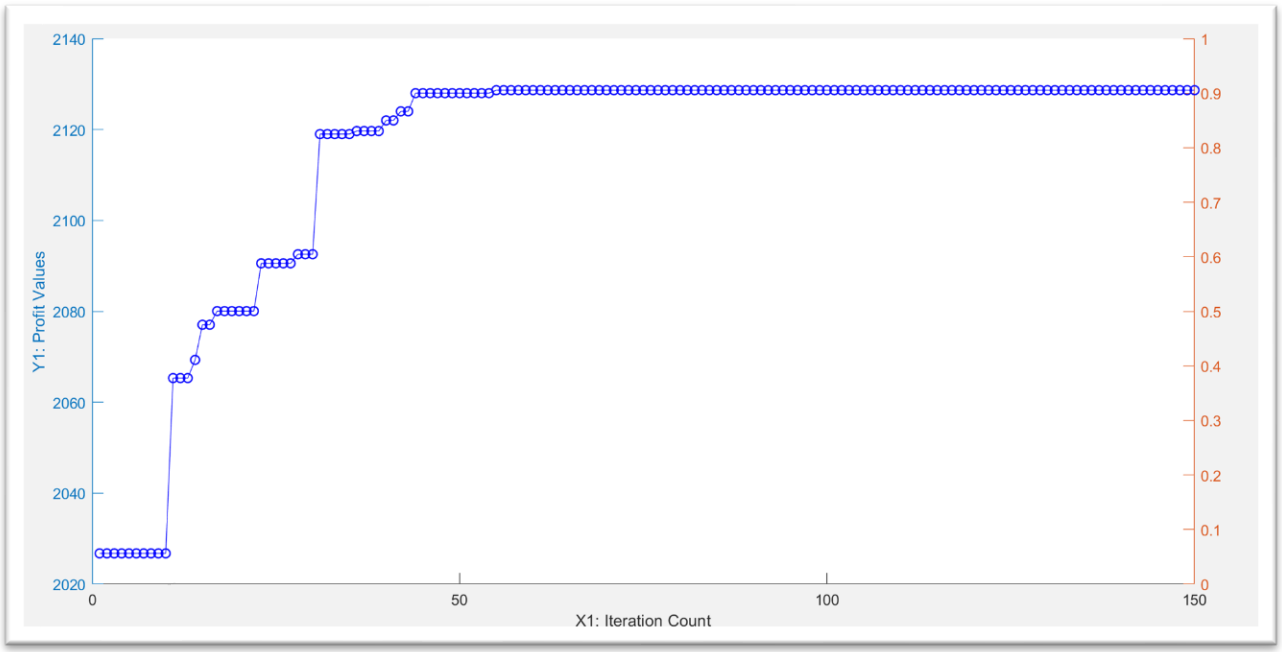

Figure 5.4: Solution record of Instance 1 by Genetics Algorithm-Tabu Search

Routing:

- Sequence of route is processed by vehicle 7 from instance 1 $(0 \rightarrow 14 \rightarrow 19 \rightarrow 8 \rightarrow 7 \rightarrow 20 \rightarrow 2 \rightarrow 10 \rightarrow 0)$
- Sequence of route is processed by vehicle 12 from instance 1 $(0 \rightarrow 11 \rightarrow 17 \rightarrow 1 \rightarrow 0)$
- Sequence of route is processed by vehicle 1 from instance 1 $(0 \rightarrow 15 \rightarrow 18 \rightarrow 3 \rightarrow 12 \rightarrow 16 \rightarrow 0)$
- Sequence of route is processed by vehicle 9 from instance 1 $(0 \rightarrow 13 \rightarrow 6 \rightarrow 9 \rightarrow 4 \rightarrow 5 \rightarrow 0)$

Delay orders: None.

Instance 2: 50 total orders, 20 delayed orders, 12 vehicles

The objective value is $4814.100.

Time lapse: 1084.380179 seconds

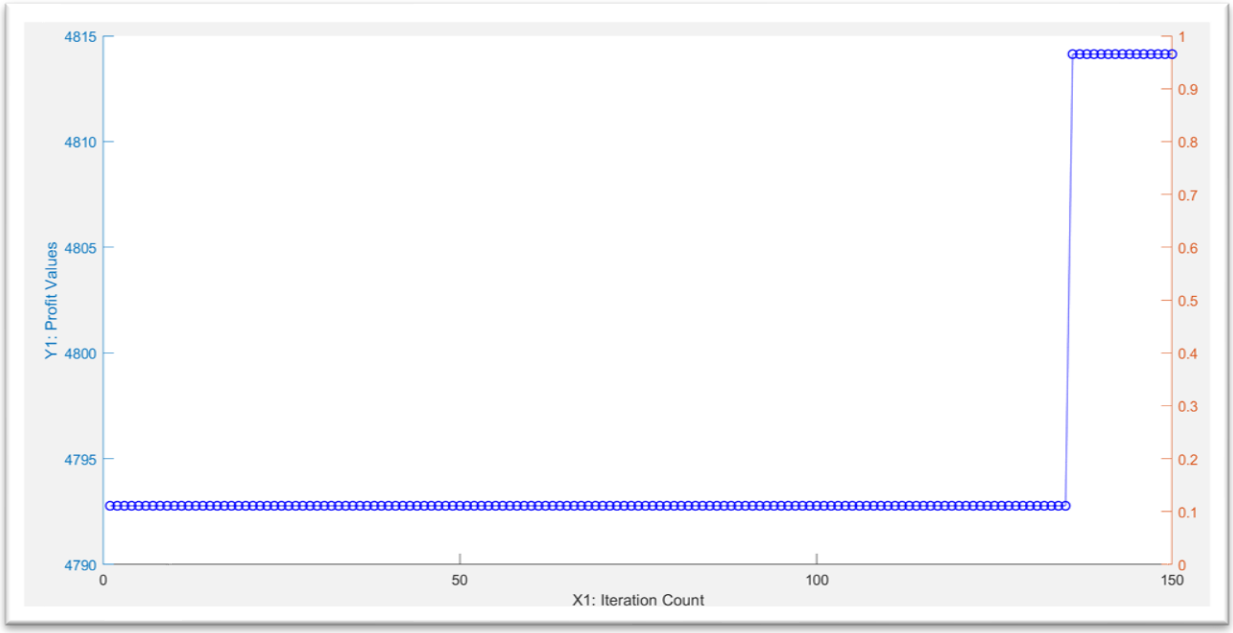

Figure 5.5: Solution record of Instance 2 by Genetics Algorithm-Tabu Search

- Sequence of route is processed by vehicle 12 from instance 2 $(0 \rightarrow 14 \rightarrow 23 \rightarrow 19 \rightarrow 2 \rightarrow 7 \rightarrow 1 \rightarrow 0)$
- Sequence of route is processed by vehicle 2 from instance 2 $(0 \rightarrow 8 \rightarrow 20 \rightarrow 26 \rightarrow 33 \rightarrow 40 \rightarrow 11 \rightarrow 10 \rightarrow 31 \rightarrow 0)$
- Sequence of route is processed by vehicle 3 from instance 2 $(0 \rightarrow 17 \rightarrow 22 \rightarrow 24 \rightarrow 27 \rightarrow 29 \rightarrow 36 \rightarrow 48 \rightarrow 15 \rightarrow 28 \rightarrow 35 \rightarrow 0)$
- Sequence of route is processed by vehicle 9 from instance 2 $(0 \rightarrow 34 \rightarrow 37 \rightarrow 46 \rightarrow 39 \rightarrow 3 \rightarrow 18 \rightarrow 32 \rightarrow 0)$
- Sequence of route is processed by vehicle 11 from instance 2 $(0 \rightarrow 25 \rightarrow 6 \rightarrow 9 \rightarrow 21 \rightarrow 13 \rightarrow 38 \rightarrow 39 \rightarrow 0)$
- Sequence of route is processed by vehicle 6 from instance 2 $(0 \rightarrow 12 \rightarrow 45 \rightarrow 44 \rightarrow 47 \rightarrow 4 \rightarrow 30 \rightarrow 5 \rightarrow 0)$
- Sequence of route is processed by vehicle 4 from instance 2 $(0 \rightarrow 43 \rightarrow 50 \rightarrow 41 \rightarrow 16 \rightarrow 42 \rightarrow 0)$

Delay orders: None.

Instance 3: 70 total orders, 20 delayed orders, 12 vehicles

The objective value is $6371.000.

Time lapse: 1550.83 seconds

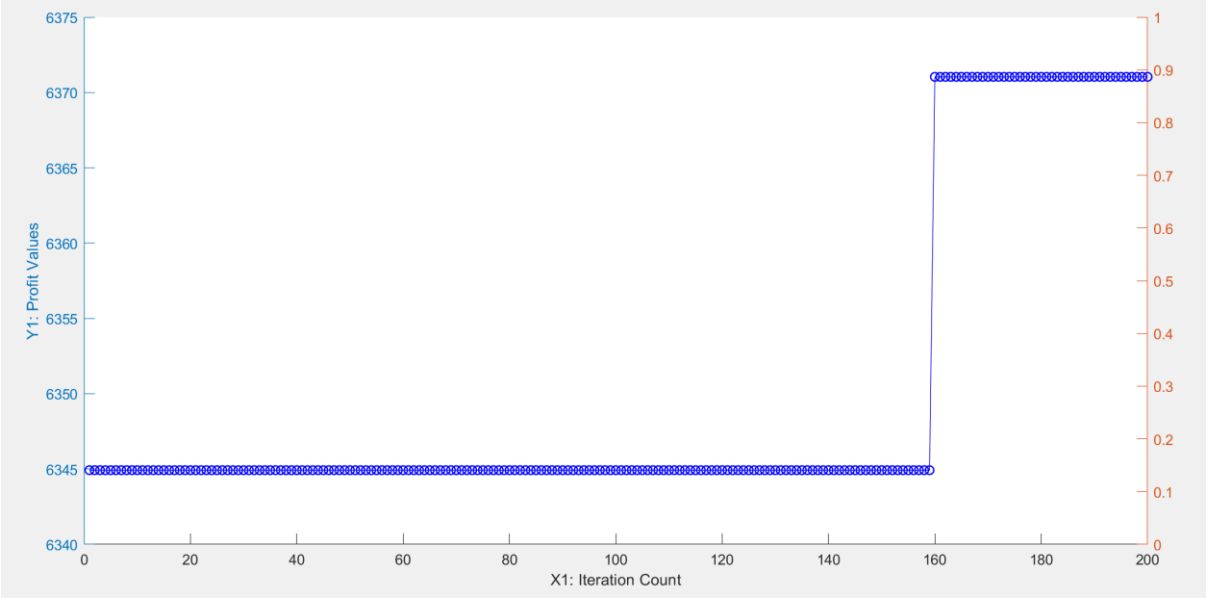

Figure 5.6: Solution record of Instance 3 by Genetics Alogirthm-Tabu Search

- Sequence of route is processed by vehicle 3 from instance 3 $(0 \rightarrow 8 \rightarrow 14 \rightarrow 2 \rightarrow 7 \rightarrow 10 \rightarrow 22 \rightarrow 24 \rightarrow 27 \rightarrow 36 \rightarrow 1 \rightarrow 15 \rightarrow 0)$
- Sequence of route is processed by vehicle 2 from instance 3 $(0 \rightarrow 19 \rightarrow 20 \rightarrow 26 \rightarrow 40 \rightarrow 61 \rightarrow 11 \rightarrow 17 \rightarrow 0)$
- Sequence of route is processed by vehicle 5 from instance 3 $(0 \rightarrow 3 \rightarrow 18 \rightarrow 6 \rightarrow 9 \rightarrow 12 \rightarrow 4 \rightarrow 5 \rightarrow 0)$
- Sequence of route is processed by vehicle 12 from instance 3 $(0 \rightarrow 23 \rightarrow 29 \rightarrow 48 \rightarrow 53 \rightarrow 54 \rightarrow 25 \rightarrow 13 \rightarrow 16 \rightarrow 0)$
- Sequence of route is processed by vehicle 10 from instance 3 $(0 \rightarrow 32 \rightarrow 52 \rightarrow 66 \rightarrow 68 \rightarrow 59 \rightarrow 31 \rightarrow 35 \rightarrow 0)$
- Sequence of route is processed by vehicle 9 from instance 3 $(0 \rightarrow 55 \rightarrow 69 \rightarrow 28 \rightarrow 34 \rightarrow 37 \rightarrow 46 \rightarrow 49 \rightarrow 67 \rightarrow 32 \rightarrow 58 \rightarrow 43 \rightarrow 0)$
- Sequence of route is processed by vehicle 1 from instance 3 $(0 \rightarrow 21 \rightarrow 38 \rightarrow 57 \rightarrow 62 \rightarrow 41 \rightarrow 39 \rightarrow 47 \rightarrow 0)$

- Sequence of route is processed by vehicle 11 from instance 3 $(0 \rightarrow 45 \rightarrow 50 \rightarrow 30 \rightarrow 51 \rightarrow 65 \rightarrow 0)$
- Sequence of route is processed by vehicle 6 from instance 3 $(0 \rightarrow 44 \rightarrow 60 \rightarrow 42 \rightarrow 56 \rightarrow 70 \rightarrow 0)$

Delay orders: [63, 64]

5.3. Result Analysis

5.3.1. Sensitive Analysis

In the sensitivity analysis section of this study, as opposed to the experimental design phase, there is a distinct inverse relationship between the number of delayed orders and profitability. Considering this observation, we have further explored the potential of enhancing profits for the entire logistics and delivery operation by expanding vehicle capacity and adjusting vehicle operational costs. Specifically, we will modify the cost parameters and the maximum allowable size for good types 2 and 3 for each vehicle to examine the resultant variations.

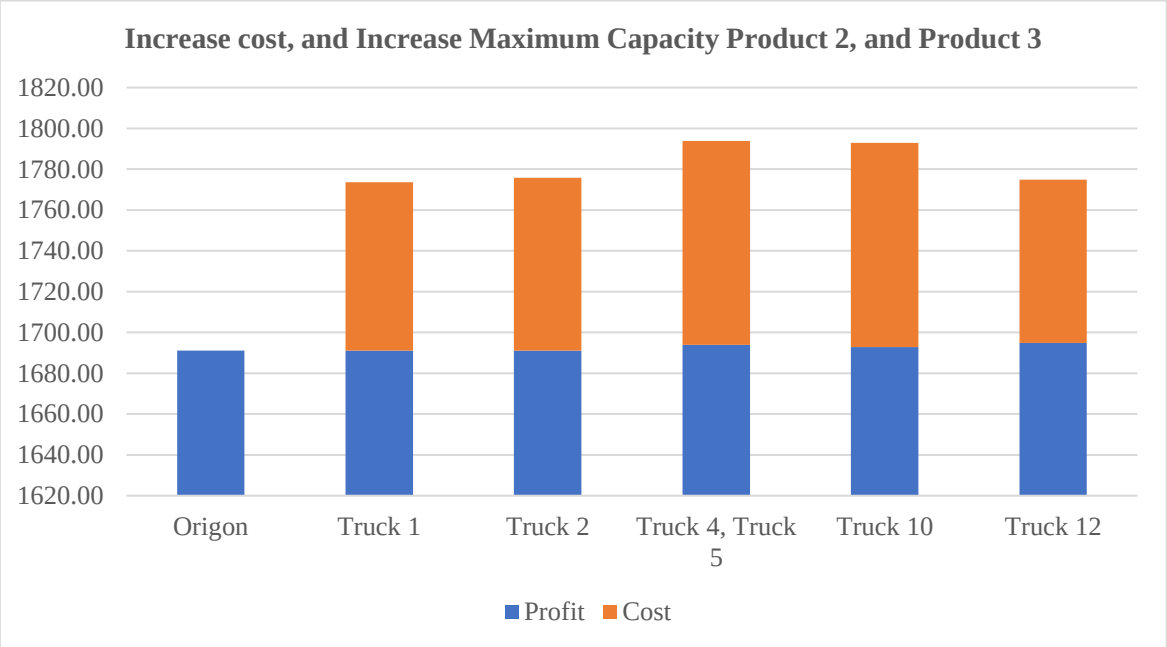

Figure 5.7: Result Analysis

The outcomes consistently show a rise in profits following the increments in both costs and vehicle capacity. This phenomenon can be attributed to the fact that enlarging the vehicle capacity enables the delivery of more orders. Consequently, this increase in delivery volume compensates for the heightened operational expenses, thereby boosting profits rather than diminishing them.

5.3.2. Environmental Impact Analysis

- **Transportation Optimization:** Optimizing transportation routes directly impacts the environment by reducing fuel consumption and, consequently, lowering greenhouse gas emissions. Specifically, efficient routing decreases the distance traveled by delivery vehicles, leading to a significant reduction in carbon footprint. This optimization can also mitigate traffic congestion, contributing to less air pollution and noise in urban areas.

5.3.3. Social Impact Analysis

- **Improved Accessibility and Equity:** Optimized transportation can enhance the accessibility of goods and services, particularly in remote or underserved regions. By streamlining delivery routes, companies can offer more reliable and timely services, bridging the gap between urban and rural areas and promoting social equity.
- **Workforce Dynamics:** The adoption of advanced routing algorithms may require a skilled workforce to manage and interpret the complex logistics software. This could lead to job displacement in traditional roles but also create opportunities in tech-driven logistics and supply chain management, necessitating workforce retraining and education.

5.3.4. Economic Impact Analysis

- **Cost Efficiency:** By minimizing unnecessary mileage and optimizing delivery schedules, companies can achieve significant savings in fuel and maintenance costs. This economic efficiency not only benefits the companies by increasing their profitability but can also lead to lower delivery costs for consumers.
- **Competitiveness and Market Expansion:** Efficient logistics can enhance a company's competitive edge by enabling faster deliveries and higher reliability, which are critical factors for customer satisfaction in the e-commerce sector. This can open new markets and expand the customer base.

5.3.5. Recommended Final Design Solution

- **Integrated Optimization System:** A comprehensive solution should incorporate advanced algorithms like Genetic Algorithms and Tabu Search to continuously analyze and optimize delivery routes. This system should be dynamic, capable of adapting to real-time changes in traffic conditions, weather, and order modifications.
- **Sustainability Focus:** The system should prioritize routes and practices that minimize environmental impact, such as favoring electric or low-emission vehicles and optimizing load capacity to reduce the number of trips.
- **User-Centric Design:** The solution should include features that enhance transparency and communication with customers, such as real-time tracking and delivery updates, to improve the overall service experience.
- **Workforce Development Program:** Implement a program to train existing employees on new technologies and systems, ensuring a smooth transition to more advanced logistics operations and maintaining job security.

CHAPTER 6. CONCLUSIONS

6.1. Results Discussion and Implication

This proposal sets out to tackle the complex challenges of fulfilling online orders from multiple distribution centers, a task that becomes significantly more complicated due to the larger scale of orders, resources, and delivery areas involved. Unlike previous studies that focused on either the picking of orders or their transportation separately, this study aims to address both issues together. We are looking at improving the entire process, from ensuring product quality during preparation to the final delivery, by considering costs, scheduling product preparation times, and optimizing delivery truck routes.

Additionally, this research dives into the complexities of shipping logistics for e-commerce, particularly focusing on efficient routing strategies. It introduces a novel idea: allowing the possibility of delaying deliveries by up to a day to increase efficiency and reduce costs. The study pays special attention to specific cases like deliveries to apartment complexes versus single homes, offering comprehensive solutions to modern logistical challenges.

Unlike earlier research that might have focused on just one aspect of logistics, like time windows or load capacities, and often compromised on strict constraints for the sake of simplicity, this study takes a no-compromise approach. It adheres to all constraints strictly, reflecting the true complexity of real-world logistics. By incorporating the flexibility of delaying orders, the research not only becomes more aligned with real-life scenarios but also enriches the logistics field by providing a detailed, practical framework for tackling today's shipping and e-commerce challenges.

6.2. Recommendations for Future Search

In the current research, the considerations for the Vehicle Routing Problem (VRP) encompass a range of constraints, including time windows and capacity limitations, to align the outcomes more closely with real-world scenarios. Additionally, the model introduces a novel element—the potential for order deferral to the subsequent

day, with a maximum allowance of one day. Future research directions may include the integration of environmental considerations, such as carbon dioxide emissions, and the prioritization of specific order characteristics to enhance customer satisfaction. For instance, the urgency and freshness of certain products could be factored into the routing decisions, elevating their priority over other orders. This approach could evolve the problem into a Multi-Objective Decision Making (MODM) framework. Moreover, there is potential for the development of a graphical user interface (GUI) application, making the model accessible to end-users beyond the confines of a developer-centric platform, thereby broadening its applicability and utility.

References

[1] Islam, M. A., Gajpal, Y., & ElMekkawy, T. Y. (2021). Hybrid particle swarm optimization algorithm for solving the clustered vehicle routing problem. *Applied Soft Computing*, *110*. https://doi.org/10.1016/j.asoc.2021.107655

[2] Hannan, M. A., Akhtar, M., Begum, R. A., Basri, H., Hussain, A., & Scavino, E. (2018). Capacitated vehicle-routing problem model for scheduled solid waste collection and route optimization using PSO algorithm. *Waste Management*, *71*. https://doi.org/10.1016/j.wasman.2017.10.019

[3] Pan, B., Zhang, Z., & Lim, A. (2021). A hybrid algorithm for time-dependent vehicle routing problem with time windows. *Computers and Operations Research*, *128*. https://doi.org/10.1016/j.cor.2020.105193

[4] Li, H., Wang, H., Chen, J., & Bai, M. (2020). Two-echelon vehicle routing problem with time windows and mobile satellites. *Transportation Research Part B: Methodological*, *138*. https://doi.org/10.1016/j.trb.2020.05.010

[5] Sitek, P., Wikarek, J., Rutczyńska-Wdowiak, K., Bocewicz, G., & Banaszak, Z. (2021). Optimization of capacitated vehicle routing problem with alternative delivery, pick-up and time windows: A modified hybrid approach. *Neurocomputing*, *423*. https://doi.org/10.1016/j.neucom.2020.02.126

[6] Kallehauge, B. (2008). Formulations and exact algorithms for the vehicle routing problem with time windows. *Computers and Operations Research*, *35*(7). https://doi.org/10.1016/j.cor.2006.11.006

[7] Bräysy, O., & Gendreau, M. (2005). Vehicle routing problem with time windows, Part II: Metaheuristics. *Transportation Science*, *39*(1). https://doi.org/10.1287/trsc.1030.0057

[8] Baldacci, R., Bartolini, E., & Laporte, G. (2010). Some applications of the generalized vehicle routing problem. *Journal of the Operational Research Society*, *61*(7). https://doi.org/10.1057/jors.2009.51

[9] Ekici, A., Özener, O. Ö., & Kuyzu, G. (2015). Cyclic delivery schedules for an inventory routing problem. *Transportation Science*, *49*(4). https://doi.org/10.1287/trsc.2014.0538

[10] Toth, P., & Vigo, D. (2002). The Vehile Routing Problem. *SIAM Society for Industrial and Applied Mathematics.*

[11] Yeun, L. C., Ismail, W. a N. R., Omar, K., & Zirour, M. (2008). Vehicle Routing Problem : Models and Solutions. *Journal of Quality Measurement and Analysis*, *4*(1).

[12] Wang, Y., Wang, L., Chen, G., Cai, Z., Zhou, Y., & Xing, L. (2020). An Improved Ant Colony Optimization algorithm to the Periodic Vehicle Routing Problem with Time Window and Service Choice. *Swarm and Evolutionary Computation*, *55*. https://doi.org/10.1016/j.swevo.2020.100675

[13] Gmira, M., Gendreau, M., Lodi, A., & Potvin, J. Y. (2021). Tabu search for the time-dependent vehicle routing problem with time windows on a road network. *European Journal of Operational Research*, *288*(1). https://doi.org/10.1016/j.ejor.2020.05.041

[14] Chen, C., Demir, E., & Huang, Y. (2021). An adaptive large neighborhood search heuristic for the vehicle routing problem with time windows and delivery robots. *European Journal of Operational Research*, *294*(3). https://doi.org/10.1016/j.ejor.2021.02.027

[15] Aksen, D., & Aras, N. (2006). Customer Selection and Profit Maximization in Vehicle Routing Problems. In *Operations Research Proceedings 2005*. https://doi.org/10.1007/3-540-32539-5_6

[16] Zhang, S., Gajpal, Y., & Appadoo, S. S. (2018). A meta-heuristic for capacitated green vehicle routing problem. *Annals of Operations Research*, *269*(1–2). https://doi.org/10.1007/s10479-017-2567-3

[17] Guo, F., Liu, Q., Liu, D., & Guo, Z. (2017). On production and green transportation coordination in a sustainable global supply chain. *Sustainability (Switzerland)*, *9*(11). https://doi.org/10.3390/su9112071

[18] Dai, H. P., Chen, D. D., & Zheng, Z. S. (2018). Effects of random values for particle swarm optimization algorithm. *Algorithms*, *11*(2). https://doi.org/10.3390/A11020023

[19] Hosseinabadi, A. A. R., Alavipour, F., Shamshirbnd, S., & Balas, V. E. (2017). A novel meta-heuristic combinatory method for solving capacitated

vehicle location-routing problem with hard time windows. Advances in Intelligent Systems and Computing, 454. https://doi.org/10.1007/978-3-319-38789-5_77

[20] Glover, F. (1989). Tabu Search—Part i. ORSA Journal on Computing, 1(3), 190–206. https://doi.org/10.1287/ijoc.1.3.190

[21] Guo, F., Liu, Q., Liu, D., & Guo, Z. (2017). On production and green transportation coordination in a sustainable global supply chain. Sustainability, 9(11), 2071. https://doi.org/10.3390/su9112071

[22] Kirkpatrick S, Gelatti CD, Vecchi MP (1983) Optimization by simulated annealing. In: Science is currently published by American association for the advancement of science, vol 220, no. 4598, pp 671–680

[23] Baker, B. M., & Ayechew, M. A. (2003). A genetic algorithm for the vehicle routing problem. Computers and Operations Research, 30(5). https://doi.org/10.1016/S0305-0548(02)00051-5

[24] "How to: Miller-Tucker-Zemlin formulation." AIMMS How-To. Accessed [27/01/2024]. https://how-to.aimms.com/Articles/332/332-Miller-Tucker-Zemlin-formulation.html.

[25] Zaritsky, A., & Sipper, M. (2004). The preservation of favored building blocks in the struggle for fitness: The puzzle algorithm. IEEE Transactions on Evolutionary Computation, 8(5). https://doi.org/10.1109/TEVC.2004.831260

[26] Dastanpour, Amin & Raja Mahmood, Raja. (2013). Feature Selection Based on Genetic Algorithm and Support Vector Machine for Intrusion Detection System. 10.13140/2.1.4289.4721.

[27] Akbar, M. D., & Aurachmana, R. (2020). Hybrid genetic–tabu search algorithm to optimize the route for capacitated vehicle routing problem with time window. International Journal of Industrial Optimization, 1(1). https://doi.org/10.12928/ijio.v1i1.1421

Appendices

Appendix A: Data Collection

Here is the link to access the dataset for the article: https://docs.google.com/spreadsheets/d/1yTgsy2MwZrweOfE_A38Hj5S4MxU4-gUh/edit#gid=1783074809.

Table 0.1: Number of product for each order

Number of product for each order			
order	**product 1**	**product 2**	**product 3**
1	4	2	1
2	1	2	2
3	1	2	3
4	1	2	1
5	3	1	2
6	1	1	1
7	1	2	3
8	4	2	1
9	5	0	2
10	2	2	1
11	4	3	3
12	1	1	2
13	3	2	1
14	3	2	2
15	4	0	1
16	5	1	1
17	5	1	1
18	3	1	1

Table 0.2: Time window of order

Time window			
order	**delivery time**	**low time**	**up time**
1	3	360	480
2	2	240	360
3	4	480	600
4	6	720	840
5	6	720	840
6	5	600	720
7	2	240	360
8	1	120	240

9	5	600	720
10	2	240	360
11	2	240	360
12	5	600	720
13	5	600	720
14	1	120	240
15	3	360	480
16	5	600	720
17	2	240	360
18	4	480	600

Table 0.3: Vehicle Information

	Vehicle					
	Product/Vehicle	**1**	**2**	**3**	**Cost**	Min cap
Oursourced vehicle	1	20	16	17	55	15
	2	25	18	19	64	15
	3	30	16	18	73	25
	4	20	16	18	55	15
	5	20	18	20	63	15
	6	25	15	16	68	15
	7	15	16	17	54	15
	8	20	17	19	63	20
	9	25	16	17	60	15
	10	20	18	20	59	15
	11	35	19	21	80	20
	12	25	18	20	60	15
Company Vehicle	1	20	13	13	61	10
	2	25	15	15	65	12

Table 0.4 Distance matrix from customer *i* to customer *j*:

	0	1	2	3	4	5	6	7	8	9	10
0	0.00	10.30	19.92	19.92	24.19	19.65	24.52	19.92	19.92	24.00	19.92
1	10.30	0.00	10.05	10.05	15.13	14.14	18.03	10.05	10.05	15.81	10.05
2	19.92	10.05	0.00	0.00	5.83	11.00	12.08	0.00	0.00	7.81	0.00
3	19.92	10.05	0.00	0.00	5.83	11.00	12.08	0.00	0.00	7.81	0.00
4	24.19	15.13	5.83	5.83	0.00	9.43	8.00	5.83	5.83	3.00	5.83
5	19.65	14.14	11.00	11.00	9.43	0.00	5.00	11.00	11.00	7.07	11.00

6	24.52	18.03	12.08	12.08	8.00	5.00	0.00	12.08	12.08	5.00	12.08
7	19.92	10.05	0.00	0.00	5.83	11.00	12.08	0.00	0.00	7.81	0.00
8	19.92	10.05	0.00	0.00	5.83	11.00	12.08	0.00	0.00	7.81	0.00
9	24.00	15.81	7.81	7.81	3.00	7.07	5.00	7.81	7.81	0.00	7.81
10	19.92	10.05	0.00	0.00	5.83	11.00	12.08	0.00	0.00	7.81	0.00

Table 0.5: Time travel from customer *i* to customer *j*

	0	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**
0	0.00	36.03	69.74	69.74	84.65	68.76	85.80	69.74	69.74	84.00	69.74
1	36.03	0.00	35.17	35.17	52.96	49.50	63.10	35.17	35.17	55.34	35.17
2	69.74	35.17	0.00	0.00	20.41	38.50	42.29	0.00	0.00	27.34	0.00
3	69.74	35.17	0.00	0.00	20.41	38.50	42.29	0.00	0.00	27.34	0.00
4	84.65	52.96	20.41	20.41	0.00	33.02	28.00	20.41	20.41	10.50	20.41
5	68.76	49.50	38.50	38.50	33.02	0.00	17.50	38.50	38.50	24.75	38.50
6	85.80	63.10	42.29	42.29	28.00	17.50	0.00	42.29	42.29	17.50	42.29
7	69.74	35.17	0.00	0.00	20.41	38.50	42.29	0.00	0.00	27.34	0.00
8	69.74	35.17	0.00	0.00	20.41	38.50	42.29	0.00	0.00	27.34	0.00
9	84.00	55.34	27.34	27.34	10.50	24.75	17.50	27.34	27.34	0.00	27.34
10	69.74	35.17	0.00	0.00	20.41	38.50	42.29	0.00	0.00	27.34	0.00

Appendix B: Mathematical Model

Model File

```
/*********************************************
 * OPL 12.8.0.0 Model
 * Author: admin
 * Creation Date: Aug 9, 2023 at 8:39:45 PM
```

```
 *********************************************/
 execute timeTermination {
   cplex.tilim = 60 *60;   // set time model stop (second)
   }
int C = 20;
{int} rangeS = {1,2,3};
{int} rangeC = {1, 2, 3, 4, 5, 6, 7, 8, 9, 10,11,12,13,14,15};
{int} rangeC0 = {0,1,2,3,4,5,6,7,8,9,10,11,12,13,14,15};// union {i | i in rangeC};
{int} rangeCD = {1,2,3,4};
{int} rangeCplus =  {i | i in     rangeC} union {C + 1} ;
{int} rangeV = {1,2,3,4,5,6,7,8,9,10,11,12};

//Parameter
int u[rangeC][rangeS] = [[4,2,1],
                                                [1,2,2],
                                                [1,2,3],
                                                [1,2,1],
                                                [3,1,2],
                                                [1,1,1],
                                                [1,2,3],
                                                [4,2,1],
                                                [5,0,2],
                                                [2,2,1],
                                                [4,3,3],
                                                [1,1,2],
                                                [3,2,1],
                                                [3,2,2],
                                                [4,0,1]
                                                /*[5,1,1],
                                                [5,1,1],
                                                [3,1,1],
                                                [4,1,1],
                                                [3,0,1]*/
                                                ];
```

```
float dis[rangeC0][rangeC0] =[
[0.00, 10.30, 19.92, 19.92, 24.19, 19.65, 24.52, 19.92, 19.92, 24.00, 19.92, 6.40,
    7.07, 19.92, 6.00, 4.00],
[10.30, 0.00, 10.05, 10.05, 15.13, 14.14, 18.03, 10.05, 10.05, 15.81, 10.05, 11.18,
    15.62, 10.05, 15.81, 7.07],
[19.92, 10.05, 0.00, 0.00, 5.83, 11.00, 12.08, 0.00, 0.00, 7.81, 0.00, 18.60,
    23.85, 0.00, 25.71, 16.16],
[19.92, 10.05, 0.00, 0.00, 5.83, 11.00, 12.08, 0.00, 0.00, 7.81, 0.00, 18.60,
    23.85, 0.00, 25.71, 16.16],
[24.19, 5.13, 5.83, 5.83, 0.00, 9.43, 8.00, 5.83, 5.83, 3.00, 5.83, 21.54,
    26.93, 5.83, 30.15, 20.22],
[19.65, 14.14, 11.00, 11.00, 9.43, 0.00, 5.00, 11.00, 11.00, 7.07, 11.00, 15.00,
    20.10, 11.00, 25.50, 15.81],
[24.52, 18.03, 12.08, 12.08, 8.00, 5.00, 0.00, 12.08, 12.08, 5.00, 12.08, 20.00,
    25.08, 12.08, 30.41, 20.62],
[19.92, 10.05, 0.00, 0.00, 5.83, 11.00, 12.08, 0.00, 0.00, 7.81, 0.00, 18.60,
    23.85, 0.00, 25.71, 16.16],
[19.92, 10.05, 0.00, 0.00, 5.83, 11.00, 12.08, 0.00, 0.00, 7.81, 0.00, 18.60,
    23.85, 0.00, 25.71, 16.16],
[24.00, 15.81, 7.81, 7.81, 3.00, 7.07, 5.00, 7.81, 7.81, 0.00, 7.81, 20.62,
    25.96, 7.81, 30.00, 20.00],
[19.92, 10.05, 0.00, 0.00, 5.83, 11.00, 12.08, 0.00, 0.00, 7.81, 0.00, 18.60,
    23.85, 0.00, 25.71, 16.16],
[6.40, 11.18, 18.60, 18.60, 21.54, 15.00, 20.00, 18.60, 18.60, 20.62, 18.60, 0.00,
    5.39, 18.60, 11.18, 5.00],
[7.07, 15.62, 23.85, 23.85, 26.93, 20.10, 25.08, 23.85, 23.85, 25.96, 23.85, 5.39,
    0.00, 23.85, 8.60, 8.60],
[19.92, 10.05, 0.00, 0.00, 5.83, 11.00, 12.08, 0.00, 0.00, 7.81, 0.00, 18.60,
    23.85, 0.00, 25.71, 16.16],
[6.00, 15.81, 25.71, 25.71, 30.15, 25.50, 30.41, 25.71, 25.71, 30.00, 25.71, 11.18,
    8.60, 25.71, 0.00, 10.00],
[4.00, 7.07, 16.16, 16.16, 20.22, 15.81, 20.62, 16.16, 16.16, 20.00, 16.16, 5.00,
    8.60, 16.16, 10.00, 0.00]
];
```

```
float cost1 = 0.9;
float cost2[rangeV] = [82.5,84.8,100,85,63,68,54,63,60,100,80,80];
int p[rangeS] = [20,22,25];
float ptime[rangeS] =  [2,2,2.5];
int maxcap[rangeS][rangeV] = [[20,25,30,20,25,15,20,25,25,20,35,25],
                              [20,20,20,10,14,12,20,14,10,25,15,20],
                              [20,20,20,10,14,12,20,14,10,25,15,20]];
int mincap[rangeV] = [15,20,25,15,15,15,15,20,15,15,20,15];
float t[rangeC0][rangeC0] = [
[0.00,  36.03,  69.74,  69.74,  84.65,  68.76,  85.80,  69.74,  69.74,  84.00,  69.74,  22.41,
        24.75,  69.74,  21.00,  14.00],
[36.03, 0.00,   35.17,  35.17,  52.96,  49.50,  63.10,  35.17,  35.17,  55.34,  35.17,  39.13,
        54.67,  35.17,  55.34,  24.75],
[69.74, 35.17,  0.00,   0.00,   20.41,  38.50,  42.29,  0.00,   0.00,   27.34,  0.00,   65.10,
        83.49,  0.00,   89.98,  56.54],
[69.74, 35.17,  0.00,   0.00,   20.41,  38.50,  42.29,  0.00,   0.00,   27.34,  0.00,   65.10,
        83.49,  0.00,   89.98,  56.54],
[84.65, 52.96,  20.41,  20.41,  0.00,   33.02,  28.00,  20.41,  20.41,  10.50,  20.41,  75.39,
        94.24,  20.41,  105.52,         70.78],
[68.76, 49.50,  38.50,  38.50,  33.02,  0.00,   17.50,  38.50,  38.50,  24.75,  38.50,  52.50,
        70.35,  38.50,  89.23,  55.34],
[85.80, 63.10,  42.29,  42.29,  28.00,  17.50,  0.00,   42.29,  42.29,  17.50,  42.29,  70.00,
        87.78,  42.29,  106.45,         72.15],
[69.74, 35.17,  0.00,   0.00,   20.41,  38.50,  42.29,  0.00,   0.00,   27.34,  0.00,   65.10,
        83.49,  0.00,   89.98,  56.54],
[69.74, 35.17,  0.00,   0.00,   20.41,  38.50,  42.29,  0.00,   0.00,   27.34,  0.00,   65.10,
        83.49,  0.00,   89.98,  56.54],
[84.00, 55.34,  27.34,  27.34,  10.50,  24.75,  17.50,  27.34,  27.34,  0.00,   27.34,  72.15,
        90.87,  27.34,  105.00,         70.00],
[69.74, 35.17,  0.00,   0.00,   20.41,  38.50,  42.29,  0.00,   0.00,   27.34,  0.00,   65.10,
        83.49,  0.00,   89.98,  56.54],
[22.41, 39.13,  65.10,  65.10,  75.39,  52.50,  70.00,  65.10,  65.10,  72.15,  65.10,  0.00,
        18.85,  65.10,  39.13,  17.50],
```

```
[24.75, 54.67, 83.49, 83.49, 94.24, 70.35, 87.78, 83.49, 83.49, 90.87, 83.49, 18.85,
        0.00, 83.49, 30.11, 30.11],
[69.74, 35.17, 0.00, 0.00, 20.41, 38.50, 42.29, 0.00, 0.00, 27.34, 0.00, 65.10,
        83.49, 0.00, 89.98, 56.54],
[21.00, 55.34, 89.98, 89.98, 105.52, 89.23, 106.45, 89.98, 89.98, 105.00,
        89.98, 39.13, 30.11, 89.98, 0.00, 35.00],
[14.00, 24.75, 56.54, 56.54, 70.78, 55.34, 72.15, 56.54, 56.54, 70.00, 56.54, 17.50,
        30.11, 56.54, 35.00, 0.00]
];
float n[rangeC] = [360,240,480,720,720,600,240,120,600,240,240,600,600,120,360];
float m[rangeC] = [480,360,600,840,840,720,360,240,720,360,360,720,720,240,480];
int M = 100000;

//decision variable
dvar boolean y[rangeV];
dvar boolean cus[rangeC][rangeV];
dvar boolean x[rangeC0][rangeC0][rangeV];
dvar boolean z[rangeC][rangeCplus];
dvar float+ re[rangeC];
dvar float+ s[rangeV];
dvar float+ arr[rangeC];
dvar boolean delay[rangeC];
dvar int+ el[rangeC][rangeV];
//objective function
maximize sum(i in rangeC, m in rangeS, k in rangeV)(p[m]*u[i][m]*cus[i][k])
                -sum(k in rangeV, i in rangeC0, j in rangeC0)(cost1*dis[i][j]*x[i][j][k])
                -sum(k in rangeV)(cost2[k]*y[k]);

//constraint
subject to {
forall(i in rangeC){
sum(j in rangeCplus: i != j) z[i][j] <= 1;
}
```

```
forall(i in rangeC){
re[i] >= sum(m in rangeS)(ptime[m]*u[i][m]);
}

forall(i in rangeC, j in rangeC: i != j){
re[i] >= re[j] + sum(m in rangeS)(ptime[m]*u[i][m]) - M*(1-z[j][i]);
}

forall(i in rangeC,k in rangeV){
s[k] >= re[i] - M*(1-cus[i][k]);
}



forall(i in rangeC, j in rangeC, k in rangeV: i != j) {
        el[j][k] - el[i][k] >= sum(m in rangeS)(u[i][m] - maxcap[m][k]*(1 - x[i][j][k]));
}

forall(i in rangeC, k in rangeV){
        el[i][k] >= sum(m in rangeS)u[i][m]*cus[i][k];
        el[i][k] <= sum(m in rangeS)maxcap[m][k]*cus[i][k];
}

forall(i in rangeCD){
sum(k in rangeV)cus[i][k]  == 1;
}

forall(i in rangeC){
sum(k in rangeV)cus[i][k] <= 1;
}

forall(j in rangeC, k in rangeV){
cus[j][k] == sum(i in rangeC0: i != j)x[i][j][k];
cus[j][k] == sum(i in rangeC0: i != j)x[j][i][k];
;}
```

```
forall(k in rangeV){
sum(i in rangeC)x[0][i][k] <= y[k];
sum(i in rangeC)x[i][0][k] <= y[k];

}


forall(i in rangeC, k in rangeV){
arr[i] >= s[k] + t[0][i] - M*(1-cus[i][k]);
}

forall(i in rangeC, j in rangeC, k in rangeV: i != j){
arr[j] >= arr[i] + t[i][j] - M*(1 - x[i][j][k]);
}

forall(i in rangeC, k in rangeV) {
arr[i] - M*(1 - cus[i][k]) <= m[i];
}

forall(i in rangeC, k in rangeV) {
arr[i] >= n[i]*cus[i][k];
}

forall(k in rangeV, m in rangeS) {
sum(i in rangeC)(u[i][m] * cus[i][k]) <= maxcap[m][k];
}

forall(k in rangeV){
sum(i in rangeC)cus[i][k] >= y[k];
sum(i in rangeC)cus[i][k] <= y[k]*M;
}

forall(k in rangeV){
```

```
sum(i in rangeC, m in rangeS)(u[i][m]*cus[i][k]) >= mincap[k]*y[k];
}

forall(i in rangeC){
sum(k in rangeV)(cus[i][k]) + delay[i] == 1;
}

}
```

Appendix C: Meta Heuristics

Tabu Search

```
tic
run('datasource.m')
run('input_parameter.m')
%run('InitializationforTabu.m')
run('Heuristics')
% truck_encode = {[0 0 0 0]   [1 1 0 0]    [1 0 1 1]    [1 0 1 1]    [1 0 1 1]    [1 1 1 0]};
% routing_order = [8  11  2  4  12  3  5  6  1  9  7  10];
% delayed_yesterday = [2 3 4];
% delayed_today_encode = {[0 1 0 1]    [0 0 0 1]    [1 0 1 0]    [0 1 1 1]    [0 1 0 0]};

figure;
yyaxis left;
ax1 = gca;
lineObj1 = animatedline('Color', 'b', 'Marker', 'o', 'LineStyle', '-');
xlabel('X1: Iteration Count');
ylabel('Y1: Profit Values');
iter = 1;

% initial fitnessfunction
```

```
[revenue_init, employee_cost_init,distance_travel_init,travel_cost_init,profit_init] =
fitnessfunction(routing_mat,Num_product,price_product,Order_route_quantity,truck_
decode,employ_cost,route_used,distance,lengths,travelling_cost,sep_decode);

% tabu_applying
array_1 = [routing_order, cell2mat(truck_encode), cell2mat(delayed_today_encode)];

%record solution
global_fitness = profit_init;
init_route = route_used;
init_truck = truck_decode;
global_route = route_used;
global_truck = truck_decode;
global_routing = array_1;

% initial define
tabulist = zeros(tabusize,tabusize);
solutionrecord = zeros(Number_Iteration,1);
solutionrecord_inner = zeros(Number_Iteration,Number_inner_loop);
solutionmatrix_inner = zeros(Number_Iteration,Number_inner_loop);


% start outer loop
while iter <= Number_Iteration
    array_sample = array_1;
    feasible_matrix = zeros(Number_inner_loop,1);
    disp(['Iteration-outer: ' num2str(iter)]);
    rand_part = round(random('Uniform',1,3+1)-0.5);
    disp("rand_part:")
    disp(rand_part)
    % start inner loop
```

```
   for inner = 1:Number_inner_loop
       array_inner = array_sample;
       % define l?i cái routing_order,truck_encode, delayed_today_encode
       routing_order = array_inner(1,1:Num_route);
       truck_encode = mat2cell(array_inner(1,Num_route + 1: Num_route +
numBits_sep*Num_truck),1);
       delyed_today_encode = mat2cell(array_inner(1, Num_route +
numBits_sep*Num_truck + 1:end),1);
%        disp(['Iteration-inner: ' num2str(inner)]);
       [tabu_record, step_record, array_record] = swaprandom(array_inner, rand_part,
routing_order, truck_encode, delayed_today_encode, tabulist, tabusize);

       % remove tabu and full zeros rows
       is_tabu_location = find(tabu_record == 1);
       array_record(is_tabu_location,:) = [];
       step_record(is_tabu_location,:) = [];
       rows_to_remove_step = all(step_record == 0,2);
       rows_to_remove_array = all(array_record == 0,2);
       array_record = array_record(~rows_to_remove_array, :);

       % handle the case that if the array_record is empty, this case
       % happen when the previous rand case  is equal to the following
       % rand case when the inner loop is cover all the tabu - step
       % for example, inner loop 2 , and the rand-case = 2, and the
       % tabulist of the previous is [ 6,7 ; 7,8 ] and the following loop
       % is the same with previous , it will gain conflict,
       % this may be gain local trap when it keep repeat

%        if isempty(array_record)
%            %array_record = array_inner;
%            % or increase outer loop one more time
```

```
%            array_1 = array_inner;
%            break
%        end

        step_record = step_record(~rows_to_remove_step, :);
        profit_matrix = zeros(size(array_record,1),1);
        % transform and calculate fitness function
        % init for array_modified
        array_modified = zeros(size(array_record,1),length(array_inner));
        % init for matrix route_used_decode
        array_route_decode = cell(size(array_record,1),1);
        % processing the neighbor
        for i = 1:size(array_record,1)
            % other init
            routing_order_1 = array_record(i,1:length(routing_order));
            truck_encode_1 = array_record(i,length(routing_order)+1:
length(routing_order)+length(cell2mat(truck_encode)));
            delayed_encode_today_1 = array_record(i,length(routing_order) +
length(cell2mat(truck_encode)) + 1: length(routing_order) +
length(cell2mat(truck_encode)) + length(cell2mat(delayed_today_encode)));
            %turn into cell for truck_encode and delayed_encode and processing
                % Num_cell = size(routing_order(i),2)/numBits_sep;
                % Cellsize = repmat(numBits_sep, 1, Num_cell);
            truck_sep_used = mat2cell(truck_encode_1, size(truck_encode_1,1),
repmat(numBits_sep,1,size(truck_encode_1,2)/numBits_sep));
            [sep_decode,truck_decode,sep_decode_org] =
truckselect(Num_truck,truck_sep_used,Num_route);
            % the same with the truck_encode
            delayed_used =
mat2cell(delayed_encode_today_1,size(delayed_encode_today_1,1),
repmat(numBits_sep,1,size(delayed_encode_today_1,2)/numBits_sep));
```

```
        if isempty(delayed_used)
            delayed_decode_feasible = 0;
        else
            [delayed_decode,delayed_decode_feasible] =
delayed(Num_route,delayed_yesterday, delayed_used,size_delayed);
        end
        % routediv
        [route_divide,route_used,truck_decode] =
div(sep_decode,routing_order_1,delayed_decode_feasible,truck_decode);
        % check element in route_used
        feasible_route = feasibleroute(delayed_yesterday, route_used);
        if feasible_route == 1
            feasible_full = 0;
        % Capacity Check
        else
          [capacity_check,total_capacity_truck,total_capacity_routing,route_check] =
checking(route_used,Capacity_truck_max,
Order_route_quantity,truck_decode,Num_product);
%           disp("Capcity_check_truck: 1/infeasible - 0/feasible")
%           disp(capacity_check)
            if capacity_check == 1
                feasible_full = 0;
            else
                [route_capacity,lengths,div_route_check,total_quantity] =
capacitycheck(truck_decode,route_used, Order_route_quantity, Capacity_truck_min,
Capacity_truck_max,Num_product);
  %             disp("Capacity_check: 1/infeasible - 0/feasible")
  %             disp(div_route_check)
              if div_route_check == 1
                  feasible_full = 0;
              else
```

```
                % Time Check
                [upper,lower,earliest,latest,time_check_feasible,route_used] =
arrivaltime(lengths,route_used,earliest_time,latest_time,travel_time,distance);
   %                disp("Time_check: 1/infeasible - 0/feasible")
   %                disp(time_check_feasible)
                if time_check_feasible == 1
                   feasible_full = 0;
                else
                   feasible_full = 1;
                end
             end
          end
       end
       routing_mat = cell2mat(route_used);
       % collect array after modified in time_check
       array_modified(i,:) =
[routing_mat,delayed_decode_feasible,cell2mat(truck_sep_used),cell2mat(delayed_us
ed)];
       % collect route_used matrix to define global_route
       array_route_decode{i} = route_used;
       % feasible check
%          if feasible_route == 0 && capacity_check == 0 && div_route_check == 0
&& time_check_feasible == 0
%             feasible_full = 1;
%          else
%             feasible_full = 0;
%          end
       % processing after feasible check - if feasible => calculate profit/infeasible =>
profit = 0
       if feasible_full == 1
```

```
            [revenue, employee_cost,distance_travel,travel_cost,profit] =
fitnessfunction(routing_mat,Num_product,price_product,Order_route_quantity,truck_
decode,employ_cost,route_used,distance,lengths,travelling_cost,sep_decode);
            profit_matrix(i) = profit;
        else
            profit_matrix(i) = 0;
        end
    end
    % condition for all the swap matrix is infeasible solution
    if all(profit_matrix == 0)
        r1 = round(random("Uniform",1,size(array_modified,1)+1)-0.5);
        array_sample = array_record(r1,:);
%           disp('No feasible found')
        % disp(r1)
        % disp(array_1)
    % if there is still 1 or more feasible solution
    else
        [location_new,t1,t2,x_best,x_result,x_route] =
localoptimization(profit_matrix,step_record,array_modified,array_route_decode);
        % after calculate the fitness function => if > globalfitness/update into the
global
        if x_result >= global_fitness
            global_fitness = x_result;
            global_route = x_route;
            global_routing = x_best;
            global_truck = truck_decode;
        end
        %update tabulist
        [tabulist] = updatetabulist(tabusize,tabulist,t1,t2);
        disp(tabulist)
        %update new result and truck and route array
```

```
            solutionrecord_inner(iter,inner) = x_result;
            array_sample = x_best;
            disp(x_best)
            disp(x_result)
        end
    end
%     escape = round(random("Uniform",1,2+1)-0.5);
%     if escape == 1
        array_1 = global_routing;
%     else
%         array_sample = x_best;
%     end
    solutionroute{iter} = route_used;
    solutionrecord(iter) = global_fitness;
    %% plot
    addpoints(lineObj1, iter, global_fitness);
    drawnow;
    iter = iter + 1;
end
toc
```

Genetics Algorithm – Tabu Search Hybrid

```
tic
run("initializationforGA.m");

%% sort fitness population in descend
[sort_pop_fitness_matrix,location_sort] = sort(pop_fitness_matrix,"ascend");
sort_population = pop_cal(location_sort,:);
sort_pop_truck = pop_truck(location_sort);
sort_pop_route = pop_route(location_sort);
```

```
%% selection
figure;
yyaxis left;
ax1 = gca;
lineObj1 = animatedline('Color', 'b', 'Marker', 'o', 'LineStyle', '-');
xlabel('X1: Iteration Count');
ylabel('Y1: Profit Values');
tabulist = zeros(tabusize,tabusize);

iter_GA = 1;
while iter_GA <= Number_of_GA_Iteration
    disp(['Iteration-GA: ' num2str(iter_GA)]);
    %disp(sort_pop_route);
    cell_in = population;
    %disp(cell_in)
    %% init
    chil = zeros(child_set,size(cell2mat(population),2));
    truck = cell(child_set,1);
    solu = zeros(child_set,1);
    rou = cell(child_set,1);
    for c = 1:child_set
        %disp(['Iteration-chil-set: ' num2str(c)]);
        parent = selection(pmax,cell_in);
        %% split route - separator - delayed matrix
        [route_cell,delayed_cell,separator_cell] =
split_component(parent,Num_route,Num_truck,numBits_sep);
        %% cross-over
        numParents = size(parent, 1);
        rand_parent = round(rand * (numParents - 1)) + 1;
        % cross over routing
        child_route = generateChildRoute(route_cell, parent,rand_parent);
```

```
        % cross over separator & cross over delayed_order
        [child_sep, child_del] = generateChildDelayAndSep(separator_cell, delayed_cell,
rand_parent);
        array_child = [child_route,child_sep,child_del];
        %% muitation for full array - tabu applying - and calculate the fitnessfunction
        [init_profit,route_used_init,sep_decode_init,truck_decode_init] =
calculateFitness(child_sep, child_del, child_route, Num_route,Num_truck,
numBits_sep, Capacity_truck_max, Capacity_truck_min,Order_route_quantity,
Num_product, price_product, employ_cost, distance, travelling_cost, earliest_time,
latest_time, travel_time, BigM, size_delayed, delayed_yesterday);

        %record solution
        global_fitness = init_profit;
        init_route = route_used_init;
        init_truck = truck_decode_init;
        global_route = route_used_init;
        global_truck = truck_decode_init;
        global_routing = array_child;

        % initial define
        solutionrecord = zeros(Number_outer_loop_muitation,1);
        solutionrecord_inner =
zeros(Number_outer_loop_muitation,Number_inner_loop);
        solutionmatrix_inner =
zeros(Number_outer_loop_muitation,Number_inner_loop);
        % iter_outer = 1;
        % start outer loop
        for iter_outer = 1: Number_outer_loop_muitation
            array_sample = array_child;
            %disp(['Iteration-outer-muitation: ' num2str(iter_outer)]);
            rand_part = round(random('Uniform',1,3+1)-0.5);
```

```
        %rand_part = round(random('Uniform',2,3+1)-0.5);
        % start inner loop
        for inner = 1:Number_inner_loop
            array_inner = array_sample;
            routing_order = array_inner(1,1:Num_route);
            truck_encode = mat2cell(array_inner(1,Num_route + 1: Num_route +
numBits_sep*Num_truck),1);
            delayed_today_encode = mat2cell(array_inner(1, Num_route +
numBits_sep*Num_truck + 1:end),1);
            % init for swap
            %disp(['Iteration-inner-muitation: ' num2str(inner)]);
            [tabu_record, step_record, array_record] = swaprandom(array_inner,
rand_part, routing_order, truck_encode, delayed_today_encode, tabulist, tabusize);
            % remove tabu and full zeros rows
            is_tabu_location = find(tabu_record == 1);
            array_record(is_tabu_location,:) = [];
            step_record(is_tabu_location,:) = [];
            rows_to_remove_step = all(step_record == 0,2);
            rows_to_remove_array = all(array_record == 0,2);
            array_record = array_record(~rows_to_remove_array, :);
            if isempty(array_record)
                %array_record = array_inner;
                % or increase outer loop one more time
                array_child = array_inner;
                break
            end
            step_record = step_record(~rows_to_remove_step, :);
            profit_matrix = zeros(size(array_record,1),1);
            % transform and calculate fitness function
            % init for array_modified
            array_modified = zeros(size(array_record,1),length(array_inner));
```

```
        % init for matrix route_used_decode
        array_route_decode = cell(size(array_record,1),1);
        % processing the neighbor

        for i = 1:size(array_record,1)
            % other init
            routing_order_1 = array_record(i,1:Num_route);
            truck_encode_1 = array_record(i,Num_route + 1: Num_route +
numBits_sep*Num_truck);
            delayed_encode_today_1 = array_record(i,Num_route +
numBits_sep*Num_truck + 1:end);
            %turn into cell for truck_encode and delayed_encode and processing
                % Num_cell = size(routing_order(i),2)/numBits_sep;
                % Cellsize = repmat(numBits_sep, 1, Num_cell);
            truck_sep_used = mat2cell(truck_encode_1, size(truck_encode_1,1),
repmat(numBits_sep,1,size(truck_encode_1,2)/numBits_sep));
            [sep_decode,truck_decode,sep_decode_org] =
truckselect(Num_truck,truck_sep_used,Num_route);
            % the same with the truck_encode
            delayed_used =
mat2cell(delayed_encode_today_1,size(delayed_encode_today_1,1),
repmat(numBits_sep,1,size(delayed_encode_today_1,2)/numBits_sep));
            if isempty(delayed_used)
               delayed_decode_feasible = 0;
            else
               [delayed_decode,delayed_decode_feasible] =
delayed(Num_route,delayed_yesterday, delayed_used,size_delayed);
            end
            % routediv
            [route_divide,route_used,truck_decode] =
div(sep_decode,routing_order_1,delayed_decode_feasible,truck_decode);
```

```
            % check element in route_used
            feasible_route = feasibleroute(delayed_yesterday, route_used);

            % Capacity Check
            [capacity_check,total_capacity_truck,total_capacity_routing,route_check]
= checking(route_used,Capacity_truck_max,
Order_route_quantity,truck_decode,Num_product);
            [route_capacity,lengths,div_route_check,total_quantity] =
capacitycheck(truck_decode,route_used, Order_route_quantity, Capacity_truck_min,
Capacity_truck_max,Num_product);

            % Time Check
            [upper,lower,earliest,latest,time_check_feasible,route_used] =
arrivaltime(lengths,route_used,earliest_time,latest_time,travel_time, distance);

            routing_mat = cell2mat(route_used);
            % collect array after modified in time_check
            % collect route_used matrix to define global_route
            array_route_decode{i} = route_used;
            array_modified(i,:) =
[routing_mat,delayed_decode_feasible,cell2mat(truck_sep_used),cell2mat(delayed_us
ed)];
            [revenue, employee_cost,distance_travel,travel_cost,profit] =
fitnessfunction(routing_mat,Num_product,price_product,Order_route_quantity,truck_
decode,employ_cost,route_used,distance,lengths,travelling_cost,sep_decode);
            % feasible check
            if feasible_route == 0 && capacity_check == 0 && div_route_check == 0
&& time_check_feasible == 0
               profit_matrix(i) = profit;
            else
               profit_matrix(i) = profit - BigM;
```

```
            end
        end
        [location_new,t1,t2,x_best,x_result,x_route] =
localoptimization(profit_matrix,step_record,array_modified,array_route_decode);
        % after calculate the fitness function => if > globalfitness/update into the
global
        if x_result >= global_fitness
            global_fitness = x_result;
            global_route = x_route;
            global_routing = x_best;
            global_truck = truck_decode;
        end
        %update tabulist
        [tabulist] = updatetabulist(tabusize,tabulist,t1,t2);
        %update new result and truck and route array
        solutionrecord_inner(iter_outer,inner) = x_result;
        array_sample = x_best;
    end
    array_child = global_routing;
    % solutionroute{iter_outer} = route_used;
    solutionrecord(iter_outer) = global_fitness;
end
% record the solution
chil(c,:) = global_routing;
truck{c} = global_truck;
solu(c,1) = global_fitness;
rou{c} = global_route;
end
%% collect and sorting
[sort_solu,elitism_loca] = sort(solu,"descend");
sort_chil = chil(elitism_loca,:);
```

```
    sort_truck = truck(elitism_loca);
    sort_route = rou(elitism_loca);
    %% elitism and replacement
    [sort_pop_fitness_matrix,sort_pop_route,sort_population,sort_pop_truck] =
elitism(sort_pop_fitness_matrix,sort_pop_route,sort_pop_truck,sort_population,sort_s
olu,sort_chil,sort_truck,sort_route,child_set);
    %update the pop_cal
    pop_cal = sort_population;
    fix = cell(pmax,1);
    for i = 1:size(pop_cal,1)
        fix{i} = pop_cal(i,:);
    end
    disp(sort_pop_fitness_matrix)
    %disp(sort_pop_truck);
    % attach to the new population var for the next loop using
    population = fix;
    best_solution = sort_pop_fitness_matrix(pmax,1);
    %% plot
    addpoints(lineObj1, iter_GA, best_solution);
    drawnow;
    iter_GA = iter_GA + 1;
end
toc
```

Function

InitializationforGA.m

```
%% initialization
tic
run("datasource.m");
run("input_parameter.m");
p = 1;
```

```
population = cell(pmax,1);
pop_fitness_matrix = zeros(pmax,1);
pop_truck = cell(pmax,1);
pop_route = cell(pmax,1);

while p <= pmax
   if p <= percentage_of_infeasible
      run("Heuristics.m")
      [revenue_init, employee_cost_init,distance_travel_init,travel_cost_init,profit_init]
=
fitnessfunction(routing_mat,Num_product,price_product,Order_route_quantity,truck_
decode,employ_cost,route_used,distance,lengths,travelling_cost,sep_decode);
      population{p} =
[routing_order,cell2mat(truck_encode),cell2mat(delayed_today_encode)];
      pop_fitness_matrix(p) = profit_init;
      pop_truck{p} = truck_decode;
      pop_route{p} = route_used;
   else
      population{p} = [randperm(Num_route),
randi([0,1],1,numBits_sep*(Num_truck)), randi([0,1],1,numBits_sep*size_delayed)];
   end
   p = p + 1;
end
%% calculate the fitness function of initialization for which random
% check feasibility
pop_cal = cell2mat(population);
%calculation the fitness function
pop_modified = zeros(size(pop_cal,1),size(pop_cal,2));
route_pop_decode = cell(size(pop_cal,1),1);
for i = percentage_of_infeasible +1:pmax
   %other init
```

```
    routing_pop = pop_cal(i,1:Num_route);
    sep_pop = pop_cal(i,Num_route + 1: Num_route + numBits_sep*Num_truck);
    del_pop = pop_cal(i,Num_route + numBits_sep*Num_truck + 1:end);
    %turn into cell for truck_encode and delayed_encode and processing
%         Num_cell = size(routing_order(i),2)/numBits_sep;
%         Cellsize = repmat(numBits_sep, 1, Num_cell);
    sep_pop_used = mat2cell(sep_pop, size(sep_pop,1),
repmat(numBits_sep,1,size(sep_pop,2)/numBits_sep));
    [sep_decode_pop,truck_decode_pop,sep_decode_org_pop] =
truckselect(Num_truck,sep_pop_used,Num_route);
    %the same with the truck_encode
    del_pop_used = mat2cell(del_pop,size(del_pop,1),
repmat(numBits_sep,1,size(del_pop,2)/numBits_sep));
    [delayed_decode_pop,delayed_decode_feasible_pop] =
delayed(Num_route,delayed_yesterday, del_pop_used,size_delayed);
    %routediv
    [route_divide_pop,route_used_pop,truck_decode_pop] =
div(sep_decode_pop,routing_pop,delayed_decode_feasible_pop,truck_decode_pop);
    %record truck
    pop_truck{i} = truck_decode_pop;
    %check element in route_used
    feasible_route = feasibleroute(delayed_yesterday, route_used);
    %Capacity Check
    [capacity_check_pop,~,~,~] = checking(route_used_pop,Capacity_truck_max,
Order_route_quantity,truck_decode_pop,Num_product);
    [~,lengths,div_route_check_pop,~] =
capacitycheck(truck_decode_pop,route_used_pop, Order_route_quantity,
Capacity_truck_min, Capacity_truck_max,Num_product);
    %Time Check
```

```
    [~,~,~,~,time_check_feasible_pop,~] =
time_check_without_repair(lengths,route_used_pop,earliest_time,latest_time,travel_ti
me);
    pop_route{i} = route_used_pop;
    routing_mat_pop = cell2mat(route_used_pop);
    if capacity_check_pop == 0 && div_route_check_pop == 0 &&
time_check_feasible_pop == 0 && feasible_route == 0
        feasible_full_pop = 1;
    else
        feasible_full_pop = 0;
    end
    %processing after feasible check - if feasible => calculate profit
    %profit/infeasible => profit = profit - BigM
    [revenue_pop, employee_cost_pop,distance_travel_pop,travel_cost_pop,profit_pop]
=
fitnessfunction(routing_mat_pop,Num_product,price_product,Order_route_quantity,tr
uck_decode_pop,employ_cost,route_used_pop,distance,lengths,travelling_cost,sep_de
code_pop);
    if feasible_full_pop == 1
        pop_fitness_matrix(i) = profit_pop;
    else
        pop_fitness_matrix(i) = profit_pop - BigM;
    end
end
toc
```

Heuristics.m

```
run('datasource.m');
run('input_parameter.m')
feasible_route = 1;
while feasible_route == 1
```

```
%      truck_number = 1:length(employ_cost);
%      truck_to_use = randperm(numel(truck_number), Num_truck + 1);
%      truck_decode = truck_number(truck_to_use);
%    truck_decode = [7 12];
    truck_decode = randperm(numel(1:Num_truck+1));
    Num_truck_real = length(truck_decode);
    % for truck
    % total_capacity_truck = zeros(1, Num_truck_real);
    % employ_cost = employ_cost(1:Num_truck_real);
    % [max_values, truck_decode] = mink(employ_cost, Num_truck_real);
    % for route_used
    revenue_order = sum(price_product .* Order_route_quantity(1:Num_route, :), 2);

    [sort_revenue, order_position] = maxk(revenue_order, Num_route);
    % find the order that can delayed
    order_can_delayed = setdiff(order_position,delayed_yesterday);
    % sort the order that need to deliver today according to earliest_time
    if ~isempty(delayed_yesterday)
        [sort_earliest_1, order_rush] = sort(earliest_time(delayed_yesterday),"ascend");
        delayed_yesterday = delayed_yesterday(1,order_rush);
    end
    % sort the order that don't need to deliver today according to earliest_time
    [sort_earliest_2,order_not_rush] = sort(earliest_time(order_can_delayed),"ascend");
    order_can_delayed = transpose(order_can_delayed(order_not_rush,:));
    % perform into an complete array
    order = [delayed_yesterday,order_can_delayed];

    capacity_sort = sum(Order_route_quantity, 2);

    %% Processing assign array
    route_used = cell(1, Num_truck_real); % Initialize the route_used sample
```

```
    % Sort nodes in descending order of revenue
    sortedPositions = order;

    for i = 1:Num_truck_real
        for j = 1:Num_route
            if isempty(sortedPositions)
                break;
            end
            if j == 1
                route_used{i}(j) = sortedPositions(1);
                sortedPositions(1) = [];
            else
                for k = 1:length(sortedPositions)
                    temp_route = [route_used{i}, sortedPositions(k)];
                    [time_check_feasible,earliest,latest] = checkTimeFeasibility(temp_route,
earliest_time, latest_time, travel_time,distance);
                    % disp(time_check_feasible)
                    if time_check_feasible == 1
                        [value, route_order_sort] = sort(latest_time(temp_route),"ascend");
                        temp_route = temp_route(route_order_sort);
                        [time_check_feasible,earliest,latest] =
checkTimeFeasibility(temp_route, earliest_time, latest_time, travel_time,distance);
                    end
                    % disp(time_check_feasible)
                    if all(Capacity_truck_max(truck_decode(i),:) >=
sum(Order_route_quantity(temp_route,:),1))
                        capacity_feasible = 0;
                    else
                        capacity_feasible = 1;
                    end
                    % disp(capacity_feasible)
```

```
                if capacity_feasible == 0 && time_check_feasible == 0
                    route_used{i} = temp_route;
                    sortedPositions(k) = [];
                    break;
                else
                    temp_route = route_used{i};
                end
            end
        end
    end
    if Capacity_truck_min(truck_decode(i)) >
sum(sum(Order_route_quantity(route_used{i},:),2))
        sortedPositions = [sortedPositions,route_used{i}];
        truck_decode(i) = 0;
        route_used{i} = [];
    end
end
node_used = cell2mat(route_used);
delayed_today_order = setdiff(order,node_used);

route_used = route_used(~cellfun('isempty', route_used));
truck_decode = truck_decode(truck_decode ~= 0);
if all(~ismember(delayed_yesterday,delayed_today_order)) &&
length(delayed_today_order) <= size_delayed
    feasible_route = 0;
else
    feasible_route = 1;
end
disp('feasibility - all the delayed yesterday is delivering:')
disp(feasible_route)
```

```
    disp('Route for each truck:')
    disp(route_used)
    disp('Truck used:')
    disp(truck_decode)
end
%% encoding the array back to the right form
routing_order = [node_used,delayed_today_order];
sep_decode = zeros(1,length(truck_decode)-1);
for i = 1:length(truck_decode) - 1
    if i == 1
        sep_decode(i) = (numel(route_used{i}) + 1);
    else
        sep_decode(i) = sep_decode(i-1) + numel(route_used{i});
    end
end
empty_separator_encode = Num_truck - length(sep_decode);
truck_encode_2 = cell(1,empty_separator_encode);
truck_encode_1 = base10to2(sep_decode,numBits_sep);
for i = 1:empty_separator_encode
    rand_apply = round(random("Uniform",1,empty_separator_encode + 1)-0.5);
    if any(rand_apply == 1:length(truck_encode_1))
        truck_encode_2{i} = truck_encode_1{rand_apply};
    else
        truck_encode_2{i} = zeros(1,numBits_sep);
    end
end
truck_encode = [truck_encode_1,truck_encode_2];
delayed_today_encode_1 = base10to2(delayed_today_order,numBits_sep);
%encode l?i cho delayed v?i s? l??ng b?ng 0.4*Num_route
number_delayed_order_left = size_delayed - length(delayed_today_encode_1);
delayed_today_encode_2 = cell(1,number_delayed_order_left);
```

```
for j = 1:number_delayed_order_left
   rand_position = round(random("Uniform",1,number_delayed_order_left+1)-0.5);
   if any(rand_position == 1:length(delayed_today_encode_1))
      delayed_today_encode_2{j} = delayed_today_encode_1{rand_position};
   else
      delayed_today_encode_2{j} = zeros(1,numBits_sep);
   end
end
delayed_today_encode = [delayed_today_encode_1,delayed_today_encode_2];
routing_mat = cell2mat(route_used);
lengths = zeros(1,length(route_used));
for i = 1:length(route_used)
   lengths(i) = numel(route_used{i});
end
```

fitnessfunction.m

```
function [revenue, employee_cost,distance_travel,travel_cost,profit] =
fitnessfunction(routing_mat,Num_product,price_product,Order_route_quantity,truck_
decode,employ_cost,route_used,distance,lengths,travelling_cost,sep_decode)
sum_price_product = zeros(1,Num_product);
sum_employ_cost = zeros(1,length(truck_decode));
distance_travel = cell(size(route_used));
% revenue calculated
for i = 1:length(routing_mat)
   for m = 1:Num_product
      sum_price_product(i,m) =
sum(price_product(m)*Order_route_quantity(routing_mat(i),m));
   end
end
revenue = sum(sum(sum_price_product));
% travel cost calculated
```

```
for i = 1:length(route_used)
    for j = 1:numel(route_used{i})+1
        if j == 1
            distance_travel{i}(j) = distance(1, route_used{i}(j) + 1);
        elseif j == numel(route_used{i}) + 1
            distance_travel{i}(j) = distance(route_used{i}(j-1) + 1, 1);
        else
            distance_travel{i}(j) = distance(route_used{i}(j-1) + 1, route_used{i}(j) + 1);
        end
    end
end
distance_travel = cell2mat(distance_travel);
travel_cost = travelling_cost*sum(distance_travel);
% employee cost calculated
for k = 1:(length(truck_decode))
    sum_employ_cost(k) = sum(employ_cost(truck_decode(k)));
end
employee_cost = sum(sum_employ_cost);
% profit calculated
profit = revenue - travel_cost - employee_cost;
end
```

delayed.m

```
%% Delayed - for today /decoding process
function [delayed_decode,delayed_decode_feasible] =
delayed(Num_route,delayed_yesterday, delayed_today_encode,size_delayed)
    encode = zeros(1, numel(delayed_today_encode));
    for i = 1:length(delayed_today_encode)
        binaryArray = delayed_today_encode{i}; % Keep the original order
        Numbits = numel(binaryArray);
        decimalNumber = 0;
```

```
        for j = 1:Numbits
            decimalNumber = decimalNumber + binaryArray(end-j+1) * 2^(j-1);
        end
        encode(i) = decimalNumber;
    end
    delayed_decode = unique(encode);
    common_elements = delayed_decode(ismember(delayed_decode,
delayed_yesterday));
    delayed_decode_feasible = delayed_decode(~ismember(delayed_decode,
[common_elements, 0]));
    delayed_decode_feasible = delayed_decode_feasible(delayed_decode_feasible <=
Num_route);
end
```

swaprandom.m

```
function [tabu_record, step_record, array_record] = swaprandom(array_1, rand_case,
routing_order, truck_encode, delayed_today_encode, tabulist, tabusize)
    % Initialize records
    numElements = length(array_1);
    tabu_record = zeros(1, numElements);
    recordstep = cell(1, numElements);
    array_record = zeros(numElements, numElements);
    step_record = zeros(numElements, 2);

    % Determine the start and end index for swapping based on rand_case
    if rand_case == 1
        startIndex = 1;
        endIndex = length(routing_order) - 1;
    elseif rand_case == 2
        startIndex = length(routing_order) + 1;
        endIndex = length(routing_order) + length(cell2mat(truck_encode)) - 1;
```

```
    elseif rand_case == 3
        startIndex = length(routing_order) + length(cell2mat(truck_encode)) + 1;
        endIndex = numElements - 1;
    end

    % Perform swaps and record
    for i = startIndex:endIndex
        [is_tabu, swappedArray] = performSwapAndCheckTabu(array_1, i, tabulist,
tabusize);
        tabu_record(i) = is_tabu;
        array_record(i,:) = swappedArray;
        step_record(i,:) = [i, i+1];
    end
end

function [is_tabu, swappedArray] = performSwapAndCheckTabu(array, index,
tabulist, tabusize)
    swappedArray = array;
    swappedArray([index, index+1]) = array([index+1, index]);
    is_tabu = checkTabuCondition(index, tabulist, tabusize);
end

function is_tabu = checkTabuCondition(index, tabulist, tabusize)
    is_tabu = any((tabulist(1:tabusize,1) == index & tabulist(1:tabusize,2) == index+1) |
(tabulist(1:tabusize,2) == index & tabulist(1:tabusize,1) == index+1));
end
```

truckselect.m

```
%% decoding truck and sep array
function [sep_decode,truck_decode,sep_decode_org] =
truckselect(Num_truck,truck_encode,Num_route)
```

```
sep_decode_org = zeros(1, Num_truck);
%truck_array = [];
for i = 1:length(truck_encode)
    binaryArray = truck_encode{i}; % Keep the original order
    Numbits = numel(binaryArray);
    decimalNumber = 0;
    for j = 1:Numbits
        decimalNumber = decimalNumber + binaryArray(end-j+1) * 2^(j-1);
    end
    sep_decode_org(i) = decimalNumber;
    % modified distribution probability function/ reuse sep that > Num
    % route
    if sep_decode_org(i) >= Num_route
        sep_decode_org(i) = 2 * (sep_decode_org(i) - Num_route);
        if sep_decode_org(i) >= Num_route
        sep_decode_org(i) = sep_decode_org(i) - Num_route;
        end
    end
end
sep_decode_org = unique(sep_decode_org);
% Handle the case when sep_decode_org is empty
if all(sep_decode_org == 0)
    sep_decode = [];
    truck_decode = randi(Num_truck + 1);  % Set truck_decode to 1 for an empty
sep_decode
else
    % Sort sep_decode_org and remove zeros
    sep_decode = sort(sep_decode_org);
    sep_decode(sep_decode == 0) = [];
    % Generate truck numbers
    num_trucks_needed = length(sep_decode) + 1;
```

```
    % Randomly select trucks to use
    truck_decode = randperm(Num_truck+1, num_trucks_needed);
    % old technique
    % truck_decode = 1:(length(sep_decode)+1);
    % truck_decode(~location_zeros) = [];
end
end
```

feasibleroute.m

```
function feasible_route = feasibleroute(delayed_yesterday, route_used)
    if all(ismember(delayed_yesterday, cell2mat(route_used)))
        feasible_route = 0;
    else
        feasible_route = 1;
    end
end
```

capacitycheck.m

```
%% capacity check - for min and max
function [route_capacity,lengths,div_route_check,total_quantity] =
capacitycheck(truck_decode,route_used, Order_route_quantity, Capacity_truck_min,
Capacity_truck_max,Num_product)
    lengths = zeros(1,length(route_used));
    total_quantity = zeros(1,length(route_used));
    for i = 1:length(route_used)
        lengths(i) = numel(route_used{i});
    end
    route_capacity = zeros(length(route_used),Num_product);
    for j = 1:length(route_used)
        % for k = 1:Num_product
        %     route_capacity(i, k) = sum(Order_route_quantity(route_used{i}, k));
```

```
        % end
        % total_quantity(i) = sum(route_capacity(i, :),2);  % Calculate total quantity for
each route
        % for k = 1:Num_product
            if all(sum(Order_route_quantity(route_used{j},:),1) <=
Capacity_truck_max(truck_decode(j),:)) &&
sum(sum(Order_route_quantity(route_used{j},:),2)) >=
Capacity_truck_min(truck_decode(j))
                div_route_check = 0;
            else
                div_route_check = 1;
                break;
            end
                % if div_route_check == 1
                %    break;
                % end
        % end
        %

    end
end
```

sorted_case1.m

```
%% case 1 - inversion route
function [route_used,latest] = sorted_case1(latest,num_routes,route_used)
location = zeros(1,num_routes);
num_routes = length(route_used);
for i = 1:num_routes
    [~, maxIndex] = max(latest{i}(1,:));
    location(i) = maxIndex;
    [latest{i}(maxIndex), latest{i}(end)] = deal(latest{i}(end), latest{i}(maxIndex));
```

```
    [route_used{i}(maxIndex), route_used{i}(end)] = deal(route_used{i}(end),
route_used{i}(maxIndex));
end
end
```

sorted_case2.m

```
%% case 2 - inversion route
function [route_used,latest] = sorted_case2(latest,num_routes,route_used)
C = cell(num_routes, 2);
num_routes = length(route_used);

for i = 1:num_routes
    [sorted_latest, index_latest] = sort(latest{i});
    C{i, 1} = sorted_latest;
    C{i, 2} = route_used{i}(index_latest);
end

route_used = C(:, 2)';
latest = C(:, 1)';
end
```

arrivaltime.m

```
%% calculate arrival time
function [upper,lower,earliest,latest,time_check_feasible,route_used] =
arrivaltime(lengths,route_used,earliest_time,latest_time,travel_time,distance)
earliest_data = cell(size(route_used));
latest_data = cell(size(route_used));
num_routes = length(route_used);
time_check_feasible = 0;
% take out earliest_time and latest_time
for i = 1:num_routes
```

```
    for j = 1:lengths(i)
        latest_data{i}(j) = latest_time(route_used{i}(j));
    end
end
% choosing type of sorting
rand_case = round(random('Uniform',1,10+1) -0.5);
if rand_case <= 7
    [route_used,latest_data] = sorted_case1(latest_data,num_routes,route_used);
else
    [route_used,latest_data] = sorted_case2(latest_data,num_routes,route_used);
end

for i = 1:num_routes
    for j = 1:lengths(i)
        earliest_data{i}(j) = earliest_time(route_used{i}(j));
    end
end
upper = cell(size(route_used));
lower = cell(size(route_used));
latest = cell(size(route_used));
earliest = cell(size(route_used));
% Calculate upper and lower
    for u = 1: num_routes
        upper{u} = zeros(1, lengths(u));
        lower{u} = zeros(1, lengths(u));
        earliest{u}(lengths(u)) = earliest_data{u}(lengths(u));
        latest{u}(lengths(u)) = latest_data{u}(lengths(u));
        for j = lengths(u):-1:2
            if distance(route_used{u}(j)+1, route_used{u}(j-1)+1) == 0 &&
earliest_data{u}(j) == earliest_data{u}(j-1)
                earliest{u}(j-1) = earliest{u}(j);
```

```
            latest{u}(j-1) = latest{u}(j);
        else

            lower{u}(j) = earliest{u}(j) - travel_time(route_used{u}(j) + 1,
route_used{u}(j -1 ) + 1);
            upper{u}(j) = latest{u}(j)  - travel_time(route_used{u}(j) + 1,
route_used{u}(j -1 ) + 1);
            if upper{u}(j) < 0
                time_check_feasible = 1;
                break
            end
            earliest{u}(j-1) = max(lower{u}(j), earliest_data{u}(j-1));
            latest{u}(j-1) = min(upper{u}(j), latest_data{u}(j-1));
        end
        if earliest{u}(j-1) <= latest{u}(j-1)
            time_check_feasible = 0;
        else
            time_check_feasible = 1;
            break
        end
    end

    if time_check_feasible == 1 %sum(sum_of_ones) <= 0
        time_check_feasible = 1;
        break
    end
  end
end
```

InitializationforTabu.m

```
%% Main loop
```

```
run('datasource.m')
run('input_parameter.m')
% kh?i t?o initialization array
%% quá tŕnh encoding
% t?o chu?i routing
tic
    time_check_feasible = 1;
    div_route_check = 1;
    capacity_check = 1;
    feasible_route = 1;
    roll = 0;
    routing_order = randperm(Num_route);
    %sep_decode_org = 0;
    % t?o chu?i delayed today
    delayed_today_encode = delayedtodayencode(Num_route,size_delayed);
    % decode cho delayed_order
    [delayed_decode,delayed_decode_feasible] =
delayed(Num_route,delayed_yesterday, delayed_today_encode,size_delayed);
    while capacity_check == 1 || div_route_check == 1 || feasible_route == 1 ||
time_check_feasible == 1
        % t?o chu?i separator và truck number
          truck_encode = truckencode(Num_truck,Num_route);
        % decode cho sep và truck
          [sep_decode,truck_decode,sep_decode_org] =
truckselect(Num_truck,truck_encode,Num_route);
        % t?o chu?i delayed today
        % decode cho delayed_order
        % div routing theo separator và b? các ??n delayed today
          [route_divide,route_used,truck_decode] =
div(sep_decode,routing_order,delayed_decode_feasible,truck_decode);
        % route check when delayed yesterday is in the route yet ?
```

```
        feasible_route = feasibleroute(delayed_yesterday, route_used);
        routing_mat = cell2mat(route_used);
        disp(route_used)
      % Capacity Check
        [capacity_check,total_capacity_truck,total_capacity_routing,route_check] =
checking(route_used,Capacity_truck_max,
Order_route_quantity,truck_decode,Num_product);
        [route_capacity,lengths,div_route_check,total_quantity] =
capacitycheck(truck_decode,route_used, Order_route_quantity, Capacity_truck_min,
Capacity_truck_max,Num_product);
      % arrival time bound
        %[upper,lower,earliest,latest,time_check_feasible,route_used] =
arrivaltime(lengths,route_used,earliest_time,latest_time,travel_time);
        [arrival_time,time_check_feasible] =
time_solu2(route_used,earliest_time,latest_time,travel_time,Order_route_quantity,pic
k_time);
        if feasible_route == 0 && capacity_check == 0 &&  div_route_check == 0
%&& time_check_feasible == 0
            disp('Feasible solution found: Approve')
            break;
        else
            disp('Feasible solution found: Check...')
            disp(route_used);
            disp(sep_decode);
            disp(delayed_decode_feasible);
            roll = roll + 1;
            if roll == 10 % tr??ng h?p initialization không c?n m?t qquas nhi?u th?i gian
?? ́m do có thth? chu?i ? trên infeasible
                roll = 0;
                routing_order = randperm(Num_route);
                % t?o chu?i delayed today
```

```
            delayed_today_encode = delayedtodayencode(Num_route,size_delayed);
            % decode cho delayed_order
            [delayed_decode,delayed_decode_feasible] =
delayed(Num_route,delayed_yesterday, delayed_today_encode,size_delayed);
        end

    end
  end
toc
```

calculateFitness.m

```
function [pop_fitness_matrix,route_used_pop,sep_decode_pop,truck_decode_pop] =
calculateFitness(child_sep, child_del, child_route, Num_route,Num_truck,
numBits_sep, Capacity_truck_max, Capacity_truck_min,Order_route_quantity,
Num_product, price_product, employ_cost, distance, travelling_cost, earliest_time,
latest_time, travel_time, BigM, size_delayed, delayed_yesterday)
    % Decoding and processing
    sep_pop_used = mat2cell(child_sep, size(child_sep, 1), repmat(numBits_sep, 1,
size(child_sep, 2) / numBits_sep));
    [sep_decode_pop, truck_decode_pop, ~] = truckselect(Num_truck,
sep_pop_used, Num_route);

    del_pop_used = mat2cell(child_del, size(child_del, 1), repmat(numBits_sep, 1,
size(child_del, 2) / numBits_sep));

    [~, delayed_decode_feasible_pop] = delayed(Num_route, delayed_yesterday,
del_pop_used, size_delayed);
    [~, route_used_pop, truck_decode_pop] = div(sep_decode_pop, child_route,
delayed_decode_feasible_pop, truck_decode_pop);
    % Capacity Check
```

```
    [capacity_check_pop, ~, ~, ~] = checking(route_used_pop, Capacity_truck_max,
Order_route_quantity, truck_decode_pop, Num_product);
    [~, lengths, div_route_check_pop, ~] = capacitycheck(truck_decode_pop,
route_used_pop, Order_route_quantity, Capacity_truck_min, Capacity_truck_max,
Num_product);
    [~, ~, ~, ~, time_check_feasible_pop, route_used_pop] =
time_check_without_repair(lengths, route_used_pop, earliest_time, latest_time,
travel_time);

    % Convert cell array to matrix for further processing
    routing_mat_pop = cell2mat(route_used_pop);
    % Feasibility Check
    if capacity_check_pop == 0 && div_route_check_pop == 0 &&
time_check_feasible_pop == 0
        feasible_full_pop = 1;
    else
        feasible_full_pop = 0;
    end
    % Calculate Profit
    [~, ~, ~, ~, profit_pop] = fitnessfunction(routing_mat_pop, Num_product,
price_product, Order_route_quantity, truck_decode_pop, employ_cost,
route_used_pop, distance, lengths, travelling_cost, sep_decode_pop);
    % Update Fitness Matrix
    if feasible_full_pop == 1
        pop_fitness_matrix = profit_pop;
    else
        pop_fitness_matrix = profit_pop - BigM;
    end
end
```

selection.m

```
%% random the position of the parent in the population
function parent = selection(pmax,population)
check = 0;
while check == 0
   r1 = round(random("Uniform",round(3*pmax/4),pmax+1)-0.5);
   r2 = round(random("Uniform",1,pmax+1)-0.5);
   if r1 == r2
      check = 0;
   else
      check = 1;
   end
end
%% parent take out
parent = cell(2,1);
parent{1} = population{r1};
parent{2} = population{r2};
end
```

div.m

```
% %% routing divide by separator
function [route_divide,route_used,truck_decode] =
div(sep_decode,routing_order,delayed_decode_feasible,truck_decode)
   route_divide = cell(1,length(sep_decode)+1);
   if ~isempty(sep_decode)
      for i = 1:length(sep_decode)
         if i == 1
            route_divide{i} = routing_order(1:sep_decode(i)-1);
         else
            route_divide{i} = routing_order(sep_decode(i-1):sep_decode(i)-1);
         end
         route_divide{end} = routing_order(sep_decode(end):end);
```

```
        end
    else
        route_divide = {routing_order};
    end
    route_used = cell(size(route_divide));
    % remove the delayed order today out of the routing
    for i = 1:numel(route_divide)
        currentCell = route_divide{i};
        if any(ismember(currentCell, delayed_decode_feasible))
            route_used{i} = currentCell(~ismember(currentCell,
delayed_decode_feasible));
        else
            route_used{i} = currentCell;
        end
    end
    for i = numel(route_used):-1:1
        if all(route_used{i} == 0)
            route_used(i) = [];
            truck_decode(i) = [];
        end
    end
end
```

delayedtodayencode.m

```
%% delayed today creating -encode
function delayed_today_encode = delayedtodayencode(Num_route,size_delayed)
    delayed_today_encode = cell(1,size_delayed);
    numBits_route = ceil(log2(Num_route));
    for i = 1:size_delayed
        delayed_today_encode{i} = randi([0, 1], 1, numBits_route);
    end
```

```
end
```

checking.m

```
%% Capacity feasibility check for total quantity of all orders which exclude delayed order for today with total capacity of all trucks.
function [capacity_check,total_capacity_truck,total_capacity_routing,route_check] = checking(route_used,Capacity_truck_max,
Order_route_quantity,truck_decode,Num_product)
    total_capacity_truck = 0;
    total_capacity_routing = 0;
    route_check = cell2mat(route_used);
    for j = 1:length(route_used) % use route-used because there is some truck is not being used due to drop the delayed order today out of the rout
        for k = 1:Num_product
            total_capacity_truck = total_capacity_truck + Capacity_truck_max(truck_decode(j),k); %valid
        end
    end
    for i = 1:length(route_check)
        for k = 1:Num_product
            total_capacity_routing = total_capacity_routing + Order_route_quantity(route_check(i),k); %valid
        end
    end
    if total_capacity_truck >= total_capacity_routing
      capacity_check = 0;
    else
      capacity_check = 1;
    end
end
```

checktimeFeasibility.m

```
%% time check feasible for heuristics

function [time_check_feasible,earliest,latest] = checkTimeFeasibility(route,
earliest_time, latest_time, travel_time,distance)
    % Initialize variables
    num_locations = numel(route);
    earliest = zeros(1, num_locations);
    latest = zeros(1, num_locations);

    % Assign earliest and latest times for each location in the route
    for j = 1:num_locations
        earliest(j) = earliest_time(route(j));
        latest(j) = latest_time(route(j));
    end

    % Check time feasibility
    time_check_feasible = 0;
    for j = num_locations:-1:2
        if distance(route(j)+1, route(j-1)+1) == 0 && earliest(j) == earliest(j-1)
            earliest(j-1) = earliest(j);
            latest(j-1) = latest(j);
        else
            travelTimeToNext = travel_time(route(j-1)+1, route(j)+1);
            earliestArrival = earliest(j) - travelTimeToNext;
            latestArrival = latest(j) - travelTimeToNext;
            % Update earliest and latest times
            earliest(j-1) = max(earliest(j-1), earliestArrival);
            latest(j-1) = min(latest(j-1), latestArrival);
        end
          if earliest(j-1) > latest(j-1)
```

```
            time_check_feasible = 1;
            break;
          end
    end
end
```

truckencode.m

```
%% Truck selected and seperator encoding
function truck_encode = truckencode(Num_truck,Num_route)
    numBits_sep = ceil(log2(Num_route));
    truck_encode = cell(1, Num_truck);
    for i = 1:Num_truck
        truck_encode{i} = randi([0, 1], 1, numBits_sep);
    end
end
```

time_check_without_repair.m

```
%% calculate arrival time
function [upper,lower,earliest,latest,time_check_feasible,route_used] =
time_check_without_repair(lengths,route_used,earliest_time,latest_time,travel_time)
earliest_data = cell(size(route_used));
latest_data = cell(size(route_used));
num_routes = length(route_used);
time_check_feasible = 0;
% take out earliest_time and latest_time
for i = 1:num_routes
    for j = 1:lengths(i)
        latest_data{i}(j) = latest_time(route_used{i}(j));
        earliest_data{i}(j) = earliest_time(route_used{i}(j));
    end
end
```

```
upper = cell(size(route_used));
lower = cell(size(route_used));
latest = cell(size(route_used));
earliest = cell(size(route_used));
% Calculate upper and lower
for u = 1: num_routes
   upper{u} = zeros(1, lengths(u));
   lower{u} = zeros(1, lengths(u));
   earliest{u}(lengths(u)) = earliest_data{u}(lengths(u));
   latest{u}(lengths(u)) = latest_data{u}(lengths(u));
   for j = lengths(u):-1:2
         lower{u}(j) = earliest{u}(j) - travel_time(route_used{u}(j) + 1,
route_used{u}(j -1 ) + 1);
         upper{u}(j) = latest{u}(j)  - travel_time(route_used{u}(j) + 1, route_used{u}(j
-1 ) + 1);
         if upper{u}(j) < 0
            time_check_feasible = 1;
            break
         end
         earliest{u}(j-1) = max(lower{u}(j), earliest_data{u}(j-1));
         latest{u}(j-1) = min(upper{u}(j), latest_data{u}(j-1));
         if earliest{u}(j-1) <= latest{u}(j-1)
            time_check_feasible = 0;
         else
            time_check_feasible = 1;
            break
         end
   end
   if time_check_feasible == 1
      time_check_feasible = 1;
      break
```

```
    end
end
end
```

splitcomponent.m

```
function [route_cell,delayed_cell,separator_cell] =
split_component(population,Num_route,Num_truck,numBits_sep)
    route_cell = cell(2,1);
    delayed_cell = cell(2,1);
    separator_cell = cell(2,1);
    for i = 1:2
        route_cell{i} = population{i}(1:Num_route);
        separator_cell{i} =
population{i}(Num_route+1:Num_route+numBits_sep*Num_truck);
        delayed_cell{i} = population{i}(Num_route+numBits_sep*Num_truck+1:end);
    end
end
```

updatetabulist.m

```
function [tabulist] = updatetabulist(tabusize,tabulist,t1,t2)
for n = tabusize : -1 : 2
    tabulist(n,1) = tabulist((n-1),1);
    tabulist(n,2) = tabulist((n-1),2);
end
tabulist(1,1) = t1;
tabulist(1,2) = t2;
end
```

localoptimization.m

```
function [location,t1,t2,x_best, x_result, x_route] =
localoptimization(profit_matrix,step_record,array_record,array_route_decode)
   location = find(profit_matrix == max(profit_matrix));
   %disp(location);
   rand_solu = round(random('Uniform',1,length(location)+1)-0.5);
   location_new = location(rand_solu);
   %disp(location_new)
   t1 = step_record(location_new,1);
   t2 = step_record(location_new,2);
   x_best = array_record(location_new,:);
   x_result = profit_matrix(location_new,:);
   x_route = array_route_decode{location_new};
end
```

elitism.m

```
function [sort_pop_fitness_matrix,sort_pop_route,sort_population,sort_pop_truck] =
elitism(sort_pop_fitness_matrix,sort_pop_route,sort_pop_truck,sort_population,sort_s
olu,sort_chil,sort_truck,sort_route,replacement)
for etil = 1:replacement
    if(sort_pop_fitness_matrix(1) < sort_solu(etil)) %keep if the child is better the
worst of the population
       sort_pop_fitness_matrix(1) = sort_solu(etil);
       sort_population(1,:) = sort_chil(etil,:);
       sort_pop_truck{1} = sort_truck{etil};
       sort_pop_route{1} = sort_route{etil};
       %remove when the child is worse than the worst of the population
    end
    [sort_pop_fitness_matrix,loc_sort] = sort(sort_pop_fitness_matrix,"ascend");
    sort_population = sort_population(loc_sort,:);
    sort_pop_truck = sort_pop_truck(loc_sort);
    sort_pop_route = sort_pop_route(loc_sort);
```

```
end
end
```

GenerateChildDelayandSep.m

```
function [child_sep, child_del] = generateChildDelayAndSep(separator_cell,
delayed_cell, rand_parent)
    % Calculate the middle positions of the arrays in 'separator_cell' and 'delayed_cell'
    pos_sep = round(size(cell2mat(separator_cell), 2) / 2);
    pos_delayed = round(size(cell2mat(delayed_cell), 2) / 2);

    % Generate children based on 'rand_parent'
    if rand_parent == 1
        % For 'separator_cell', take the first half from the first cell and the second half
from the second cell
        child_sep = [separator_cell{1}(1:pos_sep), separator_cell{2}(pos_sep+1:end)];

        % For 'delayed_cell', take the first half from the first cell and the second half from
the second cell
        child_del = [delayed_cell{1}(1:pos_delayed),
delayed_cell{2}(pos_delayed+1:end)];
    else
        % For 'separator_cell', take the first half from the second cell and the second half
from the first cell
        child_sep = [separator_cell{2}(1:pos_sep), separator_cell{1}(pos_sep+1:end)];

        % For 'delayed_cell', take the first half from the second cell and the second half
from the first cell
        child_del = [delayed_cell{2}(1:pos_delayed),
delayed_cell{1}(pos_delayed+1:end)];
    end
```

```
end
```

GenerateChildRoute.m

```
function child_route = generateChildRoute(route_cell, parent,rand_parent)
    % Pre-calculate sizes to avoid repeated computation
    numRoutes = size(cell2mat(route_cell), 2);

    % Generate random size using more efficient methods
    rand_size_par = round(rand * (numRoutes - 2)) + 1;
    % Generate random starting position
    rand_pos_par = round(random("Uniform",0,numRoutes - rand_size_par)-0.5);
    % Determine if the parents are identical
    identicalParents = isequal(parent{1}, parent{2});

    % Initialize 'child_route' to improve efficiency
    child_route = zeros(1, numRoutes); % Adjust according to the actual data structure

    % Generate 'child_route' based on the conditions
    if identicalParents
        route_remain = route_cell{1,1}(rand_pos_par + 1:(rand_pos_par +
rand_size_par));
    else
        route_remain = route_cell{rand_parent,1}(rand_pos_par + 1:(rand_pos_par +
rand_size_par));
    end

    % Permutation of the remaining route
    %route_permutation = route_remain(randperm(length(route_remain)));
    route_permutation(1:rand_pos_par) = route_cell{rand_parent,1}(1:rand_pos_par);
```

```
    route_permutation(rand_pos_par + 1: numRoutes - rand_size_par) =
route_cell{rand_parent,1}(rand_pos_par + 1 + rand_size_par: end);

    order_pos = randperm(length(route_permutation));
    route_permutation = route_permutation(order_pos);
    % Constructing the child route
    % case 1:
    %child_route(1:rand_pos_par) = route_cell{rand_parent}(1:rand_size_par);
    %child_route(rand_size_par + 1:end) = route_permutation;

    child_route(1:rand_pos_par) = route_permutation(1:rand_pos_par);
    child_route(rand_pos_par + 1: rand_pos_par + rand_size_par) = route_remain;
    child_route(rand_pos_par + 1 + rand_size_par: end) =
route_permutation(rand_pos_par + 1: end);
end
```