DELIVERY OPTIMIZATION VIA HETEROGENEOUS FLEET

BY

PEESIT TANGTUNG

AN INDEPENDENT STUDY SUBMITTED IN PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR THE DEGREE OF MASTER OF ENGINEERING (LOGISTICS AND SUPPLY CHAIN SYSTEMS ENGINEERING)
SIRINDHORN INTERNATIONAL INSTITUTE OF TECHNOLOGY
THAMMASAT UNIVERSITY
ACADEMIC YEAR 2025

THAMMASAT UNIVERSITY

SIRINDHORN INTERNATIONAL INSTITUTE OF TECHNOLOGY

INDEPENDENT STUDY

BY

PEESIT TANGTUNG

ENTITLED

DELIVERY OPTIMIZATION VIA HETEROGENEOUS FLEET

was approved as partial fulfillment of the requirements for
the degree of Master of Engineering
(Logistics and Supply Chain Systems Engineering)

on November 23, 2025

Member and Advisor	[illegible]
	(Associate Professor Sun Olapiriyakul, Ph.D.)
Member	[illegible]
	(Associate Professor Jirachai Buddhakulsomsiri, Ph.D.)
Director	[illegible]
	(Associate Professor Kriengsak Panuwatwanich, Ph.D.)

Independent Study Title	DELIVERY OPTIMIZATION VIA HETEROGENEOUS FLEET
Author	Peesit Tangtung
Degree	Master of Engineering (Logistics and Supply Chain Systems Engineering)
Faculty/University	Sirindhorn International Institute of Technology/ Thammasat University
Advisor	Associate Professor Sun Olapiriyakul, Ph.D.
Academic Years	2025

ABSTRACT

This research optimizes tire distribution operations through a heterogeneous fleet vehicle routing problem with time window (HFVRPTW) solved with Google OR-Tools. The model minimizes total transportation costs, including fixed and distance-based variable costs, while meeting customer demands from a central depot. A heterogeneous fleet with varying capacities and cost structures is considered, enabling efficient vehicle allocation and allowing multiple trips per vehicle. By addressing key constraints such as vehicle capacity, demand, and route distances, the model determines optimal routes and vehicle assignments. Results show that a heterogeneous fleet provides greater flexibility and significant cost savings compared to a uniform fleet, offering practical insights for logistics optimization and efficiency improvement in distribution networks.

Keywords: Vehicle routing problem (vrp), Fleet routing problem, Distribution routing problem, Delivery routing problem, Route optimization problem, Distribution system optimization, Time window, Heterogeneous fleet

ACKNOWLEDGEMENTS

First and foremost, I would like to express my deepest and most sincere gratitude to my advisor, Associate Professor Sun Olapiriyakul, Ph.D., for his invaluable guidance, and dedicated support throughout the entire process of this thesis. His expertise, insightful feedback, and patience have not only shaped the direction of my research but have also greatly enriched my academic journey. Without his kind mentorship, this thesis would not have been possible. I am truly indebted for the countless hours he has devoted to guiding me, the constructive criticisms that challenged me to think critically, and the inspiration he has given me to strive for academic excellence.

I am also profoundly grateful to the faculty members and staff of the Sirindhorn International Institute of Technology (SIIT), Thammasat University, for providing me with a world-class academic environment in which to learn, grow, and pursue my Master's degree in Logistics and Supply Chain Systems Engineering (LSCSE). Their knowledge, professionalism, and support have been essential in expanding my perspective and deepening my understanding in this field.

I would like to extend heartfelt appreciation to my friends and classmates, who have shared this academic journey with me. The discussions, collaborations, and encouragement we exchanged not only helped me academically but also made this journey enjoyable and fulfilling. Their friendship and support have created cherished memories that I will carry with me long after this program.

Most importantly, I am profoundly grateful to my beloved family for their unconditional love, understanding, and endless encouragement. Their faith in me has been a constant source of strength and motivation. To my parents, who have always believed in my potential and supported me in every possible way, this achievement is as much theirs as it is mine.

Finally, to everyone who has contributed, directly or indirectly, to the completion of this thesis, I extend my sincere thanks and appreciation. This work stands as a testament not only to my efforts but also to the collective support, inspiration, and kindness I have been fortunate to receive along the way.

Peesit Tangtung

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF SYMBOLS/ABBREVIATIONS

Symbols/Abbreviations	**Terms**
ALNS	Adaptive Large Neighborhood Search
CP	Constraint Programming
CVRP	Capacitated Vehicle Routing Problem
GLS	Guided Local Search
HFVRP	Heterogeneous Fleet Vehicle Routing Problem
HFVRPTW	Heterogeneous Fleet Vehicle Routing Problem with Time Windows
KPIs	Key Performance Indicators
LNS	Large Neighborhood Search
LSCSE	Logistics and Supply Chain Systems Engineering
MDVRP	Multi-Depot Vehicle Routing Problem
MILP	Mixed-Integer Linear Programming
MVRP	Multi-Vehicle Routing Problem
OT	Overtime
SA	Simulated Annealing
TS	Tabu Search
VNS	Variable Neighborhood Search
VRP	Vehicle Routing Problem

CHAPTER 1
INTRODUCTION

In the evolving domain of logistics and supply chain management, transportation planning has emerged as a strategic function that directly influences cost efficiency, service reliability, and operational agility. As distribution networks grow more complex and customer expectations for on-time delivery intensify, organizations must adopt intelligent planning tools to manage operations with precision. This study explores a real-world transportation challenge at "DC NP TIRE," a Bangkok-based tire distributor, where the selection of appropriate vehicle types from a heterogeneous fleet plays a pivotal role in optimizing performance.

This chapter begins by establishing the background and context of the specific problem facing DC NP TIRE, followed by a discussion of the multifaceted challenges inherent in its transportation planning. It then introduces the company's heterogeneous fleet as a strategic lever for cost control. Finally, the chapter outlines the objectives and methodological framework of the study, highlighting the application of Google OR-Tools to solve a complex Vehicle Routing Problem (VRP) and uncover a non-intuitive "Sweet Spot" (Win-Win) solution.

1.1 Background and Problem Context

Transportation planning is a cornerstone of the operation at DC NP TIRE. It determines not only how goods are moved to its 15 key customers but also how effectively resources are utilized across this network. In this tire distribution context, where delivery demands vary and service requirements differ by location, relying on manual planning or a "one-size-fits-all" strategy leads to significant inefficiencies such as underutilized vehicle capacity, excessive fuel consumption, and missed delivery windows. These inefficiencies are amplified by the need to manage driver overtime and avoid costly penalties.

1.1.1 Challenges in Transportation Planning

The complexity of delivering goods to 15 destinations with varying demand levels, all while adhering to specific ideal time windows, presents a formidable optimization problem. Without a strategic framework, the system suffers from mismatched vehicle assignments, redundant trips, and elevated operational costs. The core challenge lies in balancing two conflicting objectives:

- Cost Minimization: Reducing fixed costs (number of vehicles used), variable costs (fuel and distance), and overtime pay.
- Service Quality: Maximizing customer satisfaction by ensuring on-time delivery, especially to VIP customers where late arrivals incur real financial penalties.

1.1.2 The Role of a Heterogeneous Fleet

Deploying a heterogeneous fleet—comprising 5 "Cage Trucks" (70 capacity) and 8 "Small Lorries" (120 capacity), each with different fixed and variable operating costs—offers a pragmatic solution to these challenges. By aligning the correct vehicle selection with specific route characteristics and customer demands, the organization can achieve higher resource utilization and lower total costs. This study leverages this fleet diversity as a key tactical advantage in the optimization model.

1.2 Objectives and Methodological Framework

The primary objective of this study is to develop a delivery optimization model tailored to the specific context of DC NP TIRE. The model evaluates customer demand and determines the optimal vehicle type, sequence, and route for each delivery.

A key objective is to challenge the conventional wisdom that higher service must result in higher costs. This study seeks to prove that a "globally optimal" solution exists that is both lower in cost and higher in service than the baseline "cost-focused" strategy. The model emphasizes integrating real-world constraints, including vehicle capacities, ideal time windows, and the asymmetric cost of service failures.

1.2.1 Application of Google OR-Tools

To address the complexity of this Heterogeneous Fleet Vehicle Routing Problem with Time Windows (HFVRPTW), the study employs Google OR-Tools. This library is well-suited for this application, supporting multi-vehicle routing, capacity constraints, and complex cost structures.

The model is used to conduct a comparative scenario analysis (Baseline vs. No-OT models) and a decisive sensitivity analysis on service-level penalties. The outputs optimal routes, total costs, and satisfaction scores serve as actionable insights to prove the existence of the "Sweet Spot" and provide a clear, data-driven recommendation for management.

CHAPTER 2
REVIEW OF LITERATURE

2.1 Importance of Vehicle Routing Optimization in Heterogeneous Fleet Distribution Systems

In today's increasingly complex and competitive logistics landscape, optimizing vehicle routing is essential for improving delivery efficiency, reducing operational costs, and enhancing customer satisfaction. The Vehicle Routing Problem (VRP) and its variants such as the Capacitated VRP (CVRP), Multi-Vehicle Routing Problem (MVRP), and Fleet Routing Problem play a central role in distribution system optimization, especially when dealing with heterogeneous fleets. These fleets, composed of vehicles with varying capacities, costs, and operational constraints, introduce additional layers of complexity that demand advanced algorithmic solutions.

Traditional VRP models often assume homogeneous fleets and static demand, which limits their applicability in real-world scenarios. To address this, modern approaches incorporate fleet heterogeneity, time windows, and multiple depots. **Liu, Jabali, and Dekker (2023)** proposed a green VRP model that integrates multi-depot, multi-tour, and split deliveries for heterogeneous fleets, demonstrating significant improvements in both cost and environmental performance. Their model reflects the growing need for flexible routing systems that accommodate diverse vehicle characteristics and sustainability goals. **Nalepa and Blocho (2021)** conducted a comprehensive review of VRP models, highlighting the evolution of routing strategies in response to real-world constraints such as traffic congestion, delivery time windows, and fleet diversity. Their classification framework provides a foundation for selecting appropriate algorithms based on operational context, reinforcing the importance of tailored routing solutions. **Eksioglu, Vural, and Reisman (2020)** emphasized the role of advanced heuristics and metaheuristics in solving large-scale VRPs, particularly those involving heterogeneous fleets. Their study demonstrated that hybrid approaches combining genetic algorithms, tabu search, and local search can effectively balance route efficiency with computational feasibility. **Bouanane, Amrani and Benadada (2022)** further explored the impact of fleet diversity on routing performance, showing that heterogeneous fleet models outperform homogeneous ones in terms of cost savings and service reliability. Their findings underscore the practical benefits of incorporating vehicle-specific parameters into routing algorithms.

As logistics systems evolve, the integration of intelligent routing algorithms with scalable optimization tools offers promising avenues for enhancing distribution performance. By capturing the nuances of vehicle diversity and operational constraints, heterogeneous fleet models enable more realistic and efficient routing strategies, paving the way for smarter transport networks.

2.2 Leveraging Heterogeneous Fleet Models for Realistic Routing Scenarios

In real-world logistics operations, fleets are rarely uniform. Vehicles differ in capacity, cost, fuel efficiency, and operational constraints, making heterogeneous fleet modeling a critical advancement in vehicle routing research. The Heterogeneous Fleet Vehicle Routing Problem (HFVRP) extends classical VRP formulations by incorporating these variations, enabling more accurate and cost-effective route planning. Traditional VRP models often assume identical vehicles, which oversimplifies the complexities of modern distribution systems. **Subramanian and Ochi (2012)**. addressed this gap by proposing a hybrid algorithm for HFVRP with time windows, combining genetic algorithms and local search to improve solution quality. Their model demonstrated superior performance in minimizing total travel distance while respecting vehicle-specific constraints, highlighting the importance of hybrid heuristics in heterogeneous environments. **Afshar-Nadjafi (2016)** further explored HFVRP by integrating multiple depots and delivery time windows. Their optimization framework used adaptive large neighborhood search (ALNS) to dynamically adjust routes based on vehicle characteristics and depot availability. This approach significantly reduced operational costs and improved delivery reliability, especially in large-scale urban networks. **Avci and Topaloglu (2016)** introduced a hybrid metaheuristic for HFVRP with split deliveries, allowing partial fulfillment of customer demand across multiple vehicles. This flexibility is particularly useful in scenarios with fluctuating demand and limited vehicle capacity. Their results showed that split delivery strategies, when combined with fleet diversity, can enhance service levels and reduce the number of required trips. **Kaewman and Akararungruangkul (2018)** contributed to the practical implementation of HFVRP by developing heuristic algorithms tailored to fleets with excessive demand and longest time constraints. Their framework emphasized the importance of balancing delivery efficiency with service time limitations, offering valuable insights for logistics providers operating under tight schedules and varied vehicle capabilities.

As logistics systems become more dynamic and customer expectations rise, heterogeneous fleet models provide the necessary flexibility to meet diverse operational demands. By capturing the nuances of vehicle diversity, these models enable more realistic and efficient routing solutions, paving the way for smarter distribution strategies.

2.3 Addressing Multi-Vehicle and Multi-Depot Routing Challenges

As distribution networks scale and diversify, the complexity of coordinating multiple vehicles across multiple depots becomes a central challenge in logistics optimization. The Multi-Vehicle Routing Problem (MVRP) and Multi-Depot Vehicle Routing Problem (MDVRP) extend classical VRP models by introducing additional layers of operational coordination, such as depot assignment, vehicle scheduling, and route synchronization. **Despaux and Basterrech (2016)** tackled the MDVRP with time windows and heterogeneous fleets, proposing a hybrid algorithm that integrates tabu search and adaptive memory programming. Their model demonstrated improved delivery efficiency and reduced total cost, especially in scenarios with tight time constraints and diverse vehicle capabilities. This highlights the importance of depot-aware routing strategies in large-scale logistics systems. **Stodola (2018)** explored metaheuristics for MDVRP, emphasizing the role of variable neighborhood search (VNS) and genetic algorithms in solving complex routing problems. Their study showed that combining multiple heuristics can significantly enhance solution quality and computational speed, particularly in high-dimensional routing environments. Wang, **Zhe , Sun and Wang (2022)** introduced a coordinated routing framework for multi-vehicle systems with depot constraints. Their model incorporated depot-specific service levels and vehicle availability, enabling dynamic reassignment of routes based on real-time demand fluctuations. This approach proved effective in minimizing delivery delays and balancing workload across depots. **Bektas and Laporte (2021)** provided a comprehensive review of MVRP models and solution methods, categorizing them based on fleet composition, depot structure, and routing objectives. Their work serves as a foundational reference for researchers and practitioners seeking to design scalable and flexible routing systems. By addressing the intricacies of multi-vehicle and multi-depot coordination, these models enable logistics providers to optimize resource allocation, reduce operational bottlenecks, and improve overall service reliability.

2.4 Integrating OR-Tools and AI for Scalable Routing Solutions

The rise of open-source optimization libraries and artificial intelligence has transformed the way vehicle routing problems are approached. Google's OR-Tools, in particular, has become a go-to platform for solving complex routing scenarios, offering flexibility, scalability, and integration with real-time mapping APIs. **Cuvelier,Didier, Furnon, Gay, Mohajeri and Perron** conducted a comparative study between OR-Tools and SCIP, demonstrating that OR-Tools not only achieved faster execution times but also required fewer lines of code to solve CVRP with time windows. This makes it highly suitable for real-time delivery routing applications, especially in dynamic urban environments developed a suite of customized VRP models using OR-Tools and Python, incorporating constraints such as load/unload demands, depot scheduling, and time windows. The integration with Google's Distance Matrix API allowed for accurate travel time estimation, showcasing the practical utility of OR-Tools in transport network optimization. **Tahir (2024)** explored AI-enhanced routing using deep learning and OR-Tools, proposing a hybrid framework that combines predictive modeling with constraint-based optimization. Their approach improved route efficiency and reduced computational overhead, particularly in scenarios with fluctuating demand and traffic conditions.

Google Developers (n.d.) provide extensive documentation on OR-Tools, including methods like AddDimensionWithVehicleCapacity, which allow for vehicle-specific capacity modeling. This flexibility is crucial for heterogeneous fleet routing and real-time logistics planning.

By integrating AI and scalable optimization tools, logistics systems can achieve higher levels of responsiveness, adaptability, and cost-efficiency, paving the way for smarter and more sustainable delivery networks.

2.5 Integrating Sensitivity Analysis for Robust Vehicle Routing and Distribution Optimization

In complex logistics systems, small changes in input parameters such as demand, travel time, fuel cost, or vehicle capacity can significantly impact routing decisions and overall performance. Sensitivity analysis plays a crucial role in evaluating the robustness of vehicle routing models by identifying which variables most influence outcomes and how resilient solutions are to uncertainty. **Sabet and Farooq (2022)**conducted a sensitivity analysis on heterogeneous fleet routing models, examining how variations in fuel cost and vehicle capacity affect total delivery cost and route feasibility. Their findings revealed that even minor fluctuations in fuel prices could shift optimal vehicle assignments, underscoring the need for adaptable routing strategies in volatile environments Yu, Zhang, Yu, Sun, & Huang (2020) applied sensitivity analysis to a multi-depot vehicle routing problem with time windows, focusing on demand variability and service time constraints. Their study demonstrated that route efficiency and customer satisfaction were highly sensitive to changes in delivery time windows, suggesting that flexible scheduling mechanisms are essential for maintaining performance under uncertainty.

Table 2.1 Literature on VRP Models and Solution Methods

Authors	Routing Model	Objective Function	Fleet Type	Constraints Considered	Depot Structure	Time Window	Solution Method
Liu, Jabali & Dekker (2023)	Green VRP with split deliveries	Minimize cost & emissions	Heterogeneous	Capacity, emissions, multi-tour	Multi-depot	Yes	Heuristic + MILP
Nalepa & Blocho (2021)	VRP classification review	Model taxonomy	Mixed	Traffic, fleet diversity, delivery constraints	Various	Mixed	Literature synthesis
Eksioglu, Vural & Reisman (2020)	VRP solution methods	Minimize cost	Mixed	Capacity, routing complexity	Various	Mixed	Heuristic & metaheuristic
Bouanane, K., Amrani & Benadada (2022)	VRP model review	Cost & service reliability	Mixed	Fleet diversity, demand patterns	Various	Mixed	Comparative analysis

Kaewman & Akararungruangkul (2018)	HFVRP with time constraints	Minimize cost	Heterogeneous	Excess demand, longest time	Single depot	Yes	Heuristic algorithm
Subramanian, Penna, Uchoa & Ochi (2012)	HFVRP with time windows	Minimize distance	Heterogeneous	Capacity, time windows	Single depot	Yes	Hybrid Genetic Algorithm
Afshar-Nadjafi (2016).	HFVRP with multiple depots	Minimize total cost	Heterogeneous	Depot assignment, time windows	Multi-depot	Yes	Adaptive Large Neighborhood
Avci & Topaloglu, (2016)	HFVRP with split deliveries	Minimize cost	Heterogeneous	Split delivery, route length	Single depot	No	Hybrid Metaheuristic
Despaux & Basterrech (2016)	MDVRP with heterogeneous fleet	Minimize total cost	Heterogeneous	Capacity, time windows	Multi-depot	Yes	Tabu Search + Adaptive Memory
Stodola (2022)	MDVRP	Minimize route cost	Homogeneous	Depot coordination, route synchronization	Multi-depot	Yes	Variable Neighborhood Search
Wang, Zhe , Sun & Wang (2022)	Coordinated multi-vehicle routing	Minimize delay & cost	Mixed	Depot constraints, vehicle availability	Multi-depot	Yes	Dynamic Reassignment
Cuvelier, Didier, Furnon, Gay, Mohajeri & Perron	CVRP with time windows	Minimize cost	Mixed	Capacity, time windows	Single depot	Yes	OR-Tools vs SCIP
Tahir (2024)	AI-enhanced VRP	Minimize route cost	Mixed	Traffic, demand prediction	Single depot	Yes	Deep Learning + OR-Tools
Euchi & Yassine (2023)	Sustainable VRP	Minimize fuel & distance	Mixed	Fuel usage, route length	Single depot	No	Hybrid metaheuristics
Yu, Zhang,Yu, Sun, & Huang (2020)	Green VRP with LNS	Minimize cost & emissions	Mixed	Neighborhood selection, emissions	Single depot	No	Large Neighborhood Search
This paper	CVRP, HFVRP, VRPTW, Sensitivity Analysis (After Proposal)	Minimize total distance, vehicle usage, cost	Heterogeneous	Vehicle capacity, demand, time windows (after proposal), fuel cost, demand variability	Single depot	Yes	OR-Tools (Python), constraint modeling, LNS, scenario testing

Table 2.1 summarizes key studies on VRP models and solution methods. It compares routing types, objectives, fleet configurations, constraints, and solution techniques, providing context for the methodological choices in this study.

2.6 Research Gap and Contributions of the Study

While vehicle routing problems (VRP) have been widely studied, many existing models still rely on simplified assumptions such as homogeneous fleets, static demand, and limited operational constraints. These limitations reduce their applicability in real-world logistics systems, which often involve diverse vehicle types, fluctuating delivery conditions, and the need for flexible, scalable solutions.

Moreover, although Google OR-Tools is a powerful and accessible optimization platform, few academic studies provide detailed, customizable implementations that reflect practical routing challenges especially for heterogeneous fleets. Sensitivity analysis and time window constraints, which are crucial for evaluating robustness and responsiveness, are often underrepresented in current literature.

This study addresses these gaps by developing a customizable VRP framework using Google OR-Tools that supports heterogeneous fleet configurations and capacity constraints. The model is designed to reflect operational realities and can be adapted to various logistics scenarios.

In addition, the study provides a modular codebase that can be extended to include time window constraints and sensitivity analysis in future phases. This flexibility ensures that the model remains relevant as delivery conditions evolve and complexity increases.

To support both academic and practical use, the research also includes a structured comparative literature review and implementation summary. These components serve as a reference for future studies and logistics teams seeking to apply optimization tools in real-world settings.

Finally, the study lays the foundation for scenario testing and robustness evaluation, which will be conducted in the next stage of the research. By bridging the gap between theoretical models and practical routing systems, this work contributes a scalable and adaptable approach to distribution system optimization, with clear pathways for future enhancement.

CHAPTER 3
METHODOLOGY

3.1 Research Design

This study employs a quantitative, computational modeling methodology to investigate the impact of operational policies on logistics network performance. The research design is anchored in a three-phase comparative scenario analysis, with each phase structured to isolate specific variables and build upon the insights of the last.

The entire experimental framework is formulated as a **Heterogeneous Fleet Vehicle Routing Problem with Time Windows (HFVRPTW)**. This complex optimization model accounts for the use of different vehicle types (heterogeneous fleet) and the critical constraint of delivery deadlines (time windows). The model is computationally solved using the Google OR-Tools library, a powerful open-source suite for combinatorial optimization.

The core of the methodology unfolds across the following three experimental scenarios.

3.1.1 Scenario 1: Baseline Performance Quantification

This initial phase serves as the experimental control for the entire study. It is designed to create a high-fidelity benchmark of the company's current logistics network.

- **Objective:** To establish a comprehensive, quantitative benchmark of the existing operational *status quo*.
- **Method:** The HFVRPTW model is configured to mirror the company's current operational reality, most notably by strictly enforcing its "overtime-constrained" policy. All existing fleet sizes, driver schedules, customer locations, and demand data are held constant.
- **Expected Insight:** The output from this scenario provides a clear baseline dataset. It quantifies key performance indicators (KPIs) such as total operational cost, on-time delivery percentages, total mileage, and asset (vehicle/driver) utilization under the current rule set. This baseline is the

essential point of comparison against which all subsequent scenarios are measured.

3.1.2 Scenario 2: Unconstrained Theoretical Optimum

The second phase moves from a descriptive model to a diagnostic one. It seeks to understand the "true" capabilities and inherent trade-offs of the network by computationally removing its most significant operational constraint.

- **Objective:** To isolate and quantify the specific impact of the overtime policy on network performance and to identify the system's "true" unconstrained trade-offs.
- **Method:** This scenario utilizes the same core data as Scenario 1 but with one critical modification: the overtime constraint is completely removed. This allows the optimization solver to operate in a "blue-sky" or "ideal-state" environment, prioritizing the most efficient routing solutions regardless of arbitrary work-hour limits.
- **Expected Insight:** By comparing the results of Scenario 2 to Scenario 1, this study can precisely quantify the *cost of the constraint*—that is, how much cost, time, and inefficiency are directly attributable to the overtime policy alone. This phase reveals the latent optimization potential within the network and provides a theoretical "best-case" ceiling for performance.

3.1.3 Scenario 3: Prescriptive Optimization & Robustness Testing

The final and most complex phase transitions the research from descriptive analysis to prescriptive recommendation. It is a two-part experiment designed to first find a new "globally optimal" policy and then validate its practicality against real-world pressures.

- **Objective:** To identify a new, globally optimal "Sweet Spot" by tuning service-level penalties and subsequently to test the robustness of this new policy against the complexities of real-world operational exceptions.
- **Method (Part A: Sensitivity Analysis):** This sub-phase conducts a comprehensive sensitivity analysis by systematically iterating the "service penalty" parameter within the model. This parameter assigns a high virtual cost

for failing to meet a time window, allowing the model to find the most effective balance between the *cost of service*(e.g., using more trucks) and the *cost of failure* (e.g., missing a delivery). The configuration that yields the best system-wide cost-service balance is identified as the "globally optimal" *Sweet Spot*.

- **Method (Part B: Robustness Test):** The newly identified optimal policy from Part A is then subjected to a final "stress test." In this sub-phase, the model incorporates a realistic "VIP Customer Policy"—an exception-handling rule that forces the network to prioritize certain high-value clients, even if it is not a "globally" efficient decision.
- **Expected Insight:** This scenario provides the study's primary recommendation. It moves beyond a simple theoretical optimum (Scenario 2) to propose a *practically robust* policy. By testing the *Sweet Spot* policy against the VIP customer constraint, the research can confidently determine if the new policy is resilient enough to handle the pragmatic, non-optimal demands of the real world, ensuring the final recommendation is both data-driven and operationally viable.

3.2 Mathematical Model Formulation

The mathematical foundation of this study is built upon the Heterogeneous Fleet Vehicle Routing Problem with Time Windows (HFVRPTW). This model is specifically adapted to include Soft Time Window Penalties, which are the core mechanism for the sensitivity analysis in this research.

3.2.1 Sets and Indices

- $N = \{0,1,\dots,n\}$: Set of nodes, where node 0 is the depot
- C = Set of customer node, C = $N\ \{0\}$
- $V = \{0,1,\dots,m-1\}$: Set of vehicles
- $T = \{0,1\}$: Set of vehicle types (0 = small, 1 = large)
- $t(k)$ = Function mapping vehicle $k \in V$ to its type t ∈ T
- $C_{VIP} \subset C$: Set of VIP customer nodes (e.g., {1, 2}
- $C_{REG} = C \setminus C_{VIP}$: Set of Regular customer nodes.

3.2.2 Parameters

- d_{ij}: Distance (km) between node i and node j
- tt_{ij}: Travel time (minutes) from node i to node j (derived from d_{ij} and average speed).
- st_i: Service time (minutes) required at node i (where st_0 = 0).
- q_i: Demand (units) at customer node i (where q_0 = 0).
- Q_t: Capacity (units) of type t.
- c_t: Variable cost per kilometer (THB/km) for vehicle t.
- f_t: Fixed cost (THB) for using vehicle t
- $[e_i, l_i]$: The ideal time window for node i, where e_i is the earliest arrival time and l_i is the latest arrival time.
- P_{OT} : Penalty cost per minute for route duration exceeding 480 minutes.
- P_{EARLY} : Penalty cost per minute for arriving at a customer before e_i
- P_{LATE} : Penalty cost per minute for arriving at a customer after l_i
- P_{VIP} : Penalty cost for arriving at a C_{VIP} after l_i

3.2.3 Decision Variables

- x_{ijk}: A binary variable, = 1 if vehicle k travels directly from node k to node j , and 0 otherwise.
- y_k: A binary variable, = 1 if vehicle k is used, and 0 otherwise.
- s_{ik}: A continuous variable representing the arrival time of vehicle k at node i.
- b_{ik}: A continuous variable representing the departure time of vehicle k from node i.
- Dev_{ik}^E: Amount of time (minutes) vehicle k arrives early at node i
- Dev_{ik}^L: Amount of time (minutes) vehicle k arrives late at node i
- OT_k: Amount of time (minutes) vehicle k's route duration exceeds 480 minutes.

3.2.4 Objective Function

The objective is to minimize the Total System Cost, which is a composite function of operational costs and service penalties, depending on the scenario.

$$\text{Minimize } Z = Z_{operational} + Z_{Service\ penalty} + Z_{OT\ penalty} + Z_{VIP\ Penalties}$$

Where:

1. Operational Cost

$$Z_{Operational} = \sum_{k\in V} f_{t(k)} \cdot y_k + \sum_{k\in V} \sum_{i,j\in N} c_{t(k)} \cdot d_{ij} \cdot x_{ijk}$$

The operational cost component represents the tangible financial expenditure required for fleet execution, aggregating both the fixed costs associated with activating specific vehicle types and the variable costs incurred per unit of distance traveled. This baseline cost ensures the model accounts for the fundamental economic resources needed to service the logistical network, independent of service quality metrics.

2. Service Penalty

$$Z_{ServicePenalty} = \sum_{k\in V} \sum_{i\in C} (P_{Early} \cdot DEV_{ik}^{E} + P_{LATE} \cdot DEV_{ik}^{L}$$

The service penalty quantifies the degradation in general customer satisfaction by penalizing temporal deviations outside the designated soft time windows. This function accounts for both premature arrivals, which may necessitate waiting or cause inventory inconveniences, and delayed arrivals, which directly negatively impact the standard service level agreements.

3. Overtime Penalty

$$Z_{OTPenalty} = \sum_{k\in V} (P_{OT} \cdot OT_{k})$$

The overtime penalty captures the additional costs incurred when the total duration of a vehicle's route exceeds the standard allowable working hours. This component serves to regulate resource utilization by discouraging excessive route lengths, thereby ensuring adherence to driver labor constraints and minimizing extra workforce expenditures.

4. VIP Penalty

$$Z_{VIP\ Penalties} = \sum_{i \in C_{vip}} \sum_{k \in V} (\, P_{VIP} \cdot DEV_{ik}^{L})$$

The VIP penalty introduces a strictly weighted cost associated with service delays for high-priority clients, distinct from the standard service penalty. By assigning a significantly higher penalty rate to lateness for these specific customers, the model is mathematically coerced to prioritize punctuality for key accounts to mitigate potential reputational damage or severe contractual violations.

(Note: In Scenario 1, $Z_{ServicePenalty}$was set to 0 or ∞. In Scenarios 2 & 3, $Z_{OTPenalty}$ was set to 0, and the "Knobs" $P_{Early}$and P_{Late} were adjusted. The *real* financial penalty for C_{VIP} is handled as a business rule in the experimental design, Section 3.4.3, not in this solver's objective function.)

3.2.5 Constraints

Routing Constraints:

Each customer is visited exactly once:

$$\sum_{k \in V} \sum_{i \in N} x_{ijk} = 1 \quad \forall_j \in C \tag{3.1}$$

Flow Conservation For each vehicle, the number of arrivals equals the number of departures:

$$\sum_{i \in N} x_{ijk} - \sum_{i \in N} x_{jik} = 0 \quad \forall_j \in N\,, \forall k \ \in V \tag{3.2}$$

Each used vehicle stars and ends at the depot:

$$\sum_{j \in C} x_{0k} = y_k \text{ and } \sum_{i \in C} x_{i0k} = y_k \ \forall k \ \in V \quad (3.3)$$

The fundamental routing restrictions, including the mandate that every customer node is visited exactly once (Equation 3.1) and the flow conservation principle ensuring equal arrivals and departures at each node (Equation 3.2), are enforced structurally within the Google OR-Tools framework. Specifically, the initialization of the pywrapcp.RoutingIndexManager and pywrapcp.RoutingModel classes constructs the underlying graph topology. By defining the number of nodes and vehicles, along with the specific depot index, the solver implicitly applies these constraints to generate valid, continuous closed-loop tours for each active vehicle starting and ending at the designated depot as defined in Equation 3.3.

Capacity Constraints:

To ensure feasibility regarding fleet limitations, the capacity constraints are modeled using the AddDimensionWithVehicleCapacity method within the OR-Tools routing model. This function integrates a registered demand callback, which retrieves the specific load requirement for each location, and cross-references the cumulative load against the defined capacity vector for the heterogeneous fleet. The solver tracks the accumulated demand variable along the route and strictly invalidates any solution where the total load exceeds the maximum capacity of the specific vehicle type assigned to that route.

Time Window and Flow Constraints:

Departure time from node i:

$$b_{ik}1 = \max(e_i, s_{ik}) + st_i \ \forall_i \in C \ , \forall k \in V \quad (3.4)$$

Arrival ime at node j (if j follows i) :

$$s_{jk} \geq (b_{ik} + \ tt_{ij}) - M(1 - \ x_{ijk}) \ \forall i,j \ \in N \ , \forall k \ \in V \quad (3.5)$$

(where M is a large constant)

The temporal propagation across the network, as mathematically defined in Equations 3.4 and 3.5, is implemented through the creation of a dedicated Time Dimension using the routing.AddDimension method. This mechanism relies on a registered transit callback function that computes the traversal cost of an arc by summing the estimated travel time between nodes and the requisite service duration at the origin node. This dimension accumulates the time variable cumulatively along the route path, effectively modeling the continuity of time and ensuring that the arrival time at a subsequent node logically follows the departure time from its predecessor.

Time window definition (for soft penalties):

$$Dev_{ik}^{E} \geq e_i - s_{ik}$$
$$Dev_{ik}^{E} \geq 0 \tag{3.6}$$
$$Dev_{ik}^{L} \geq s_{ik} - l_i$$
$$Dev_{ik}^{L} \geq 0 \tag{3.7}$$

The implementation of soft time window constraints, which permit temporal deviations subject to penalty costs (Equations 3.6 and 3.7), is achieved through the SetCumulVarSoftLowerBound and SetCumulVarSoftUpperBoundmethods applied to the cumulative time variable of each node. These functions automatically compute the linear deviation between the actual arrival time and the pre-defined preferred time window boundaries. If the arrival time occurs earlier than the lower bound or later than the upper bound, the solver adds a cost to the objective function proportional to the magnitude of the violation multiplied by the specified early or late penalty coefficients.

3.3 Model Assumptions

To ensure computational tractability and conceptual clarity, the mathematical model is constructed under a set of simplifying assumptions that reflect a controlled logistics environment. These assumptions serve as foundational constraints for the initial phase of model development, allowing for focused analysis of routing efficiency and fleet utilization.

It is assumed that all customer demand (q_i) is fully known and remains constant throughout the defined planning horizon. This eliminates the need for real-time demand forecasting and enables deterministic route planning. Each customer is served exactly once by a single vehicle, thereby excluding scenarios involving multiple visits or demand splitting. All vehicles are dispatched from a single central depot and are required to return to the same location upon completion of their assigned routes, reinforcing a closed-loop delivery structure.

Travel distances between nodes are considered symmetric and deterministic, implying that the cost and distance from point A to point B are identical to those from point B to point A. This assumption disregards complex real-world factors such as traffic congestion, road conditions, and time-dependent travel variability, which will be addressed in future iterations of the model.

Crucially, Time Window Constraints (VRPTW) are central to the current formulation, defining the core challenge. This model utilizes Soft Penalties as the primary mechanic, allowing the solver to deviate from the ideal window $[e_i , l_i]$ by incurring a weighted cost (P_{EARLY}, P_{LATE}). This design is the foundation for the sensitivity analysis used to locate the optimal solution.

By establishing these assumptions, the study creates a structured baseline for evaluating heterogeneous fleet performance under controlled conditions. This approach facilitates rigorous testing of routing algorithms and provides a scalable foundation for future enhancements that reflect the operational intricacies of real-world distribution systems.

3.4 Tools and Technologies

To implement and evaluate the heterogeneous fleet vehicle routing model, this study employs a suite of computational tools and technologies that support algorithmic development, constraint modeling, and solution visualization. The integration of these tools enables a robust experimental framework capable of handling complex logistics scenarios and delivering actionable insights.

3.4.1 Python Programming Language

Python serves as the foundational programming language for model construction and execution. Its versatility and extensive ecosystem of scientific libraries make it particularly well-suited for operations research and optimization tasks. Python facilitates the seamless integration of data structures, algorithmic logic, and visualization modules, allowing for end-to-end development of the routing model. The language’s readability and modular design also support iterative prototyping, enabling rapid refinement of model components as new constraints and features are introduced.

3.4.2 Google OR-Tools

Google OR-Tools functions as the core optimization engine within the modeling framework. Designed specifically for combinatorial optimization problems, OR-Tools offers powerful capabilities for solving the Vehicle Routing Problem (VRP) under heterogeneous fleet conditions. The library supports constraint programming, routing index management, and advanced metaheuristic techniques such as Large Neighborhood Search (LNS), which are critical for navigating the vast solution space of multi-vehicle routing scenarios. OR-Tools allows for precise modeling of vehicle-specific attributes, including capacity limits, fixed and variable costs, and crucially, time window constraints and soft penalties, as defined in the mathematical model. Its efficient solver architecture ensures scalability and responsiveness, even when applied to real-world datasets with complex delivery requirements.

Together, Python and Google OR-Tools provide a synergistic platform for developing, executing, and analyzing the HFVRPTW model. This combination enables the study to move beyond theoretical formulation and into practical simulation, offering a replicable and adaptable framework for future logistics optimization research.

3.4.3 Routing Model Components

The core implementation of the vehicle routing model leverages key components from Google OR-Tools, a specialized optimization library designed for solving combinatorial problems. Central to the model architecture is the **RoutingIndexManager**, which serves as the interface for mapping logical node indices to physical locations and managing vehicle assignments. This abstraction simplifies the handling of depot and customer nodes, especially in scenarios involving multiple vehicles and complex routing constraints.

The **RoutingModel** acts as the computational engine that defines the problem structure and executes the optimization process. It encapsulates the routing logic, cost functions, and constraint definitions, enabling the solver to generate feasible and cost-effective delivery routes. Critical constraints are implemented using OR-Tools' **dimension features**, which allow for the accumulation and tracking of quantities across routes. These dimensions were utilized as follows:

1. **Capacity Dimension:** Enforces vehicle load limits and demand constraints $(Q_{t(k)})$.
2. **Time Dimension:** This is the core experimental dimension. It tracks cumulative time (travel time + service time) and is essential for:
    - **Time Window Constraints (VRPTW):** Ensuring routes respect the customer's preferred arrival windows $[e_i , l_i]$.
    - **Soft Penalties:** Allowing the tracking of service deviation and applying the penalty rates (P_{EARLY}, P_{LATE}) central to the **Sweet Spot sensitivity analysis**.
    - **Overtime Penalties:** Applying soft penalties for exceeding the 8-hour workday limit (Scenario 1).

By modularizing the routing logic and constraint handling using these advanced dimension features, the model achieves both scalability and adaptability. This design ensures that the model can accurately test strategic operational policies, such as balancing service cost against utilization rates, which is essential for a robust and high-fidelity simulation.

3.4.4 Metaheuristic Optimization

To enhance solution quality and computational efficiency, the model incorporates advanced metaheuristic optimization strategies, with a particular emphasis on Large Neighborhood Search (LNS). LNS is a highly robust technique designed specifically for solving complex combinatorial problems like the VRP. Unlike conventional local search methods that only make small, incremental adjustments, LNS operates by strategically removing a large part of the current solution (the Destroy phase) and then intelligently rebuilding that section (the Repair phase). This process allows the solver to take "big jumps" across the solution space, effectively avoiding being trapped in local optima which is critical for identifying the true global optimum.

In the context of heterogeneous fleet routing (HFVRPTW), LNS proves especially effective because its design facilitates global restructuring of routes, adapting to the complex interplay between diverse vehicle capacities, fixed costs, and soft penalties. This ensures that when the service penalty "knob" is applied (Scenario 3), the solver is capable of finding the counter-intuitive "Sweet Spot" solution—where costs are lower and service is higher which a simpler algorithm (that gets stuck in local optima) would have missed. The integration of LNS within the OR-Tools framework positions the model as a practical, high-efficiency decision-support tool.

Table 3.1 Table comparing different metaheuristics

Metaheuristic	Strengths	Weaknesses	Typical Performance in VRP
Guided Local Search (GLS)	Escapes local minima by penalizing frequent edges	May take longer to diversify	Very good for medium–large VRP
Tabu Search (TS)	Strong diversification, avoids cycling	Can be slower to converge	Good for finding robust solutions
Simulated Annealing (SA)	Simple to implement, probabilistic escape	Sensitive to cooling schedule	Works but less efficient than GLS/TS
Large Neighborhood Search (LNS)	Removes large parts of solution, rebuilds for big jumps	Rebuild step can be computationally heavy	Excellent for large-scale VRP, often best quality

Table 3.1 presents a comparative overview of three prominent metaheuristics applied in Vehicle Routing Problems: Guided Local Search (GLS), Tabu Search combined with Simulated Annealing (TS-SA), and Large Neighborhood Search (LNS). Each method offers distinct strengths and trade-offs. GLS effectively escapes local minima but may require longer convergence times. TS-SA is simple to implement and robust, though sensitive to parameter tuning. LNS stands out for its ability to restructure large portions of the solution space, making it particularly suitable for large-scale VRP instances. Given its performance advantages and compatibility with Google OR-Tools, LNS is selected as the primary search strategy in this study.

3.4.5 Visualization Tools

To enhance interpretability and support stakeholder communication, this study utilizes Google My Maps as a visualization platform for the optimized vehicle routes generated by the routing model. Geographic data including customer locations, depot coordinates, and route sequences is first structured in Microsoft Excel. These datasets are then imported into My Maps to produce interactive maps that reflect the spatial distribution of delivery points and the movement of vehicles across the network.

This visualization approach serves multiple purposes. It provides a clear and intuitive representation of route efficiency, vehicle coverage, and geographic clustering, which are critical for evaluating the practical feasibility of the model’s output. Moreover, map-based visuals offer an accessible medium for conveying complex routing results to non-technical stakeholders, such as logistics managers or business decision-makers. By bridging the gap between algorithmic output and operational insight, Google My Maps supports both analytical validation and strategic planning, reinforcing the model’s applicability in real-world logistics environments.

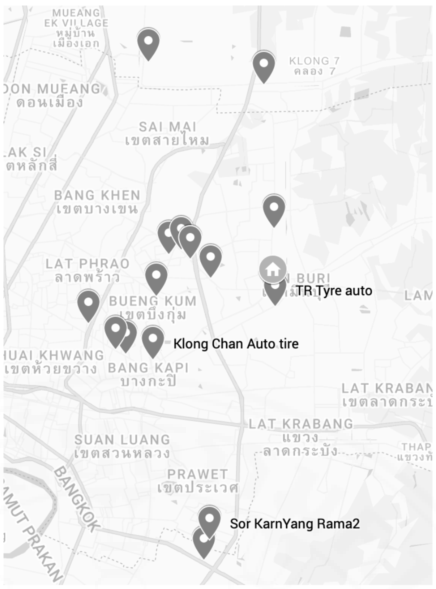

Figure 3.1 Visualization Map from Google my map

Figure 3.1 displays a geographic visualization of selected tire service centers across Bangkok, including locations where customer demand originates. This spatial representation supports the routing model by illustrating the distribution of delivery points and informing distance matrix construction. It also provides a practical reference for evaluating route feasibility and vehicle assignment based on location clustering.

3.5 Data Preparation

The data preparation phase serves as the essential translation layer between the physical supply chain and the computational model. The input dataset for the heterogeneous fleet vehicle routing model is systematically prepared, involving the organizing of operational parameters including raw customer demand profiles, geographic coordinates, and the derived inter-location distance matrices. This phase is crucial for transforming raw business requirements into normalized computational parameters.

Prior to integration into the optimization engine, meticulous efforts are focused on defining fleet heterogeneity within the data. This involves classifying vehicle parameters to ensure that the distinct cost structures (e.g., the 17 THB/km vs. 22 THB/km variable cost) are accurately mapped to the corresponding load capacity.

Furthermore, the ideal time windows are precisely structured, as these soft constraints define the operational performance boundaries for the entire network.

A rigorous data validation process is conducted to ensure completeness, accuracy, and internal consistency. This validation goes beyond simple error checking; it confirms the computational feasibility of the initial constraints and verifies that the time window data is suitable for the subsequent application of the asymmetric service penalties. By establishing a clean and reliable data foundation, the study ensures that the optimization process operates on robust inputs, thereby enabling the credible identification of the non-intuitive "Sweet Spot" solution in the analysis phase

Table 3.2 Node to Customer Mapping and Time window for Routing Model

Node	Customer	Time window	Customer status
1	NV yangyont	08:00 - 09:15	VIP
2	TR Tyre auto	08:30 - 08:45	VIP
3	V Auto Tire	09:00 - 09:15	Regular
4	PWM Power Max	09:15 - 09:30	Regular
5	Lamlukka Max Shop	09:45 - 10:15	Regular
6	NumKarnYang Klong 4	10:30 - 10:40	Regular
7	Klong Chan Auto tire	11:00 - 11:35	Regular
8	G Max Wheels	11:30 - 11:55	Regular
9	Nawamin auto tyre	13:00 - 13:15	Regular
10	Sermmit Tire	13:45 - 14:15	Regular
11	Sportmag100	14:20 - 14:40	Regular
12	Max Design	14:45 - 15:20	Regular
13	LPAutomag	15:30 - 15:40	Regular
14	Sor KarnYang Rama2	16:00 - 16:30	Regular
15	71HHH AUTO.TIRE	16:40 - 17:00	Regular
Depot	DC NP TIRE	08:00 - 17:00	

Table 3.2 provides the essential node-to-customer mapping and its corresponding Time Window constraints for the routing model. Each customer is assigned a unique node index to facilitate consistent referencing within the optimization logic. This structured mapping ensures data fidelity and is crucial for implementing the Heterogeneous Fleet Vehicle Routing Problem with Time Windows (HFVRPTW).

Critically, the table specifies the Customer Status (VIP/Regular), allowing the model to perform asymmetric risk assessment and calculate service penalties based on the priority of the customer, which is central to the analysis in Scenario 3.

Table 3.3 Customer Distance Matrix

Location	Depot	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Depot	0	17.5	16.6	12.9	23.1	27	26	24	25	21	22	20.7	20	26.1	37	30.5
1	18.1	0	11.5	15	16.5	19.5	18.5	17.5	18.5	16	16.5	16.5	16	17.6	31.6	26.2
2	17	11.7	0	13.8	15.5	18.5	17.8	16.8	17.8	15.5	16	17	15.9	15.4	30.5	25.1
3	13.2	15.5	14.1	0	22.1	25.1	24.3	22.1	23.3	19.7	20.7	18.6	18.1	22.8	33.5	27.3
4	24	17	15.9	22.1	0	13.8	12.7	13.3	14.4	15.4	16.5	18.1	18.6	15.9	26.2	23
5	28.2	20.3	19.1	25.1	13.8	0	11.6	12.8	13.8	17	17.6	18.6	19.7	18.1	25.1	21.9
6	27.3	19.2	18.4	24.3	12.7	11.6	0	11.1	12.2	15.9	17	17.6	18.6	17	24	20.8
7	25	18.1	17.4	22.1	13.3	12.8	11.1	0	11.7	14.3	15.4	16.5	17	15.9	21.9	19.2
8	26.1	19.2	18.4	23.3	14.4	13.8	12.2	11.7	0	15.5	16	17.1	17.6	16.5	23	19.7
9	21.8	16.5	15.9	19.7	15.4	17	15.9	14.3	15.5	0	13.2	14.3	14.8	14.3	20.8	17.6
10	22.8	17	16.5	20.7	16.5	17.6	17	15.4	16	13.2	0	12.7	13.2	13.8	21.9	18.6
11	20.7	16.5	17	18.6	18.1	18.6	17.6	16.5	17.1	14.3	12.7	0	12.7	14.3	20.8	18.1
12	20	16	15.9	18.1	18.6	19.7	18.6	17	17.6	14.8	13.2	12.7	0	14.8	21.9	17.6
13	26.1	17.6	15.4	22.8	15.9	18.1	17	15.9	16.5	14.3	13.8	14.3	14.8	0	24	20.8
14	37	31.6	30.5	33.5	26.2	25.1	24	21.9	23	20.8	21.9	20.8	21.9	24	0	19.5
15	30.5	26.2	25.1	27.3	23	21.9	20.8	19.2	19.7	17.6	18.6	18.1	17.6	20.8	19.5	0

Table 3.3 presents the customer distance matrix used in the routing model. It quantifies the pairwise distances between the depot and each customer location, forming the basis for route cost calculations and feasibility checks. These values are essential for constructing the cost matrix within the optimization engine and directly influence route selection, vehicle assignment, and total travel cost estimation.

Table 3.4 Customer Nodes and Weekly Demand

Node	Customer Name	Weekly Demand
DEPOT	DC NP TIRE	0
1	NV yangyont	70
2	TR Tyre auto	120
3	V Auto Tire	80
4	PWM Power Max	30
5	Lamlukka Max Shop	70
6	NumKarnYang Yong	110
7	Klong Chan auto tire	105
8	G Max Wheels	95
9	Nawamin auto tyre	110
10	Sermmit Tire	25
11	Sportmag100	10
12	G Max Design	85
13	LPAutomag	25
14	Sor KarnYang Rama2	65
15	7HIHH AUTO.TIRE	10

Table 3.4 outlines the weekly demand associated with each customer node in the routing model. This data serves as a critical input for vehicle capacity planning and route feasibility analysis. By quantifying demand at the node level, the model ensures that vehicle assignments align with load constraints and that delivery routes are optimized to meet service requirements efficiently.

Table 3.5 Vehicle Specifications by Type

Vehicle Type	Capacity (Units)	Cost per Kilometer (THB)	Available Unit	Fixed Cost (THB)
Cage Truck	70	10.5	5	600
Small Lorry	120	19	8	1,000

Table 3.5 summarizes the specifications of the two vehicle types used in the routing model: Cage Truck and Small Lorry.These parameters are critical for modeling fleet heterogeneity and directly influence route feasibility, cost calculations, and vehicle assignment decisions within the optimization framework.

3.6 Implementation Process

The implementation of the heterogeneous fleet vehicle routing model is carried out using Python in conjunction with Google OR-Tools, a specialized optimization library designed for solving complex routing problems. The process begins with the definition of fundamental model components (sets, parameters, and decision variables) which form the mathematical backbone of the model and are structured to reflect the operational realities of tire distribution logistics.

Once the foundational elements are defined, the RoutingModel acts as the computational engine. Critical constraints are implemented using OR-Tools' dimension features, which allow for the enforcement of path history and cumulative constraints. While the Capacity Dimension enforces volumetric limits, the Time Dimension is the core mechanism for policy testing and multiobjective evaluation. This dimension tracks cumulative time (travel time + service time), which is essential for implementing the soft time window constraints and the soft overtime penalty across all strategic scenarios. To enhance solution quality and guarantee convergence to the global optimum, the model incorporates Large Neighborhood Search (LNS) metaheuristic optimization. LNS is selected because it effectively balances Exploration (discovering new areas of the solution space) and Exploitation (refining known good solutions). This strategic balance ensures that the solver is not trapped in the local optima observed in Scenario 2a, but successfully identifies the globally superior Sweet Spot solution. The final stage then synthesizes the solver's output with post-processing logic to yield the final real cost (Travel + Fixed + VIP Penalty) and the crucial service metrics, completing the robust experimental framework.

CHAPTER 4
RESULT

This chapter presents the empirical findings derived from the computational optimization model. The analysis is structured as a sequential investigation across three distinct strategic policy environments (Scenarios). The objective is to move beyond a simple presentation of costs to provide a deep interpretation of the underlying operational dynamics.

This chapter will:

1. Deconstruct the legacy policy (Scenario 1) to quantify its structural inefficiencies.
2. Establish the true minimal expenditure baseline (Scenario 2) by removing artificial constraints, thereby identifying a critical "Constraint Trap."
3. Present the core discovery (Scenario 3) by conducting a sensitivity analysis on service penalties, culminating in the identification of a non-linear, globally optimal "Sweet Spot" policy that challenges the conventional cost-service trade-off.

The analysis is performed by evaluating the interaction between Total Overall Cost (THB), Average Customer Satisfaction (%), and Fleet Utilization (Fixed Cost).

4.1 Scenario 1 Analysis: The Legacy Policy Structure

Objective: To quantify the operational performance and cost ceiling of the current operational paradigm, which is defined by a rigid, 8-hour workday constraint (simulated via a soft OT penalty).

Table 4.1 Legacy Policy Trade-off (OT-Constrained Model)

Strategy	Policy Intent	Total Overall Cost (THB)	Avg. Satisfaction	Vehicles Used	Total OT Cost (THB)
1a	Legacy Cost Baseline	20,200.40	69.80%	10	0
1b	Legacy Service Focus	21,047.95	98.50%	**11**	260.85

Table 4.1 presents the results of the baseline analysis (Scenario 1), comparing the financial and service outcomes under the rigid operational assumption of the soft OT penalty. The data establishes the high structural cost imposed by the legacy constraint.

4.1.1 Interpretation of Findings Scenario 1

The analysis of Scenario 1 reveals that the legacy policy framework is structurally inefficient and expensive. While the solver could find a 10-vehicle solution for the pure cost model (1a), the resulting service level 69.8% was unacceptably low, marked by significant, unpenalized service failures.

More critically, when forced to achieve high service (1b), the model's cost increased by 847.55 THB. This cost was not just from variable overtime; it was structurally driven by the necessity to deploy an eleventh vehicle (a Cage Truck, increasing fixed cost by 600 THB. This proves that the 8-hour constraint prevented the optimal utilization of the 10-vehicle fleet, forcing an expensive and inefficient expansion of the asset base. This 21,047.95 THB cost represents the high operational ceiling imposed by the legacy policy.

4.2 Scenario 2: Establishing the True Cost Baseline (The Constraint Trap)

Objective: To isolate the system's inherent cost floor by removing the OT penalty (P_OT= 0), allowing the solver to optimize purely for asset utilization (Fixed Cost) and variable cost (Travel Cost).

Table 4.2 Analysis of Legacy Constraint Impact on Operational Cost

Policy Intent	Total Overall Cost (THB)	Avg. Satisfaction	Total Fixed Cost (THB)	Net Change in Cost
Scenario 1A	20,200.40	69.80%	8,800.00	Benchmark
Scenario 2A	17,410.00	82.40%	7,600.00	-2,790.4 THB (-13.8%)

Table 4.2 presents the comparative analysis demonstrating the financial and service impact of removing the obsolete OT penalty (moving from Scenario 1A to

Scenario 2A). This comparison establishes the minimal expenditure baseline for the entire network.

4.2.1 Interpretation of Findings: The Constraint Trap

The elimination of the OT penalty (moving from S1a to S2a) yielded a profound and counter-intuitive result: 13.8% reduction in total cost (2,790.40 THB savings) while service simultaneously increased by 12.6%.

This discovery confirms the existence of a Constraint Trap. The legacy OT rule *forced* the solver into an inferior route structure to avoid the penalty, resulting in a higher operational cost. The source of this saving is revealed in the Total Fixed Cost. The S1a solution utilized a 10-vehicle mix (3 Cage Trucks, 7 Small Lorries) for a fixed cost of 8,800 THB. Freed from the time constraint, the S2a solver found a superior 10-vehicle mix (6 Cage Trucks, 4 Small Lorries) with a fixed cost of only 7,600 THB.

This demonstrates that the legacy constraint was a structural cost driver that prevented optimal fleet mix allocation. The 17,410.00 THB cost is therefore established as the verified minimal expenditure baseline for all subsequent policy calibration.

4.3 Scenario 3: Optimal Policy Discovery & Sensitivity Analysis

Objective

This final scenario details the core research contribution: the calibration of the P_Late service penalty (the "Knob") to find the globally optimal solution that truly balances cost and service.

4.3.1 The Marginal Cost of Perfection (Policy Extremes)

First, the analysis quantifies the cost of the two policy extremes in the unconstrained model: a pure cost focus (Knob =0) versus a pure service focus (Knob =1000).

Table 4.3 Marginal Cost Analysis of Service Level Extremes

Policy Comparison	Total Overall Cost (THB)	Avg. Satisfaction	Net Cost for 17.5% Sat Gain
Cost Floor (S2a, Knob=0)	17,410.00	82.40%	Baseline

Perfection Ceiling (S3, Knob=1000)	17,418.50	99.90%	add 8.50 THB

Table 4.3 presents the comparative analysis between the minimal expenditure baseline (Cost Floor) and the highest service achieved (Perfection Ceiling). This comparison quantifies the marginal financial cost required to achieve maximum customer satisfaction in the unconstrained model.

Interpretation: The cost difference between the 82.4% satisfaction baseline and the 99.9% ceiling is only 8.50 THB. This negligible marginal cost proves that the resources required for a 17.5% service gain were already available within the optimally utilized 10-vehicle fleet. The improvement was entirely a function of superior route sequencing precision, confirming that high service is "functionally free" when the system is properly optimized.

4.3.2 The Global Optimum ("Sweet Spot" Policy)

This section analyzes the result of the intermediate Knob = 50 test, which challenges the assumption that either extreme (0 or 1000) is the optimal choice.

Table 4.4 Sensitivity Analysis and Global Optimum Discovery

penalty_early (Fixed)	penalty_late (Knob)	Avg. Satisfaction	Total Overall Cost	Notes
100	0.00	82.40%	17,410.00 THB	Baseline (High lateness)
100	50.00	97.80%	17,321.60 THB	Sweet Spot (Win-Win)
100	**100**	**99.90%**	**17,418.50 THB**	**Diminishing Returns**
100	1,000.00	99.90%	17,418.50 THB	Stabilized (No change)

Table 4.4 presents the results of the sensitivity analysis performed by adjusting the penalty_late "Knob" against a fixed early penalty (penalty_early = 100$). This analysis was conducted to empirically validate the non-linear relationship between service policy and total operational cost.

4.3.2.1 Analysis and Conclusion

The sensitivity analysis conclusively identifies the optimum strategic policy at P_Late = 50 . This discovery proves the existence of a globally superior solution that defies the conventional linear trade-off.

- The Win-Win Relationship: The P_Late = 50 penalty served as an Intelligent Steering Mechanism for the LNS solver. It forced the algorithm to abandon the inferior Local Optimum (Knob = 0) and discover a new solution path that resulted in a cost decrease of 88.40 THB while simultaneously achieving a 15.4% increase in service.
- Definitive Policy Recommendation: The Sweet Spot Policy is the definitive, optimal strategic recommendation for DC NP TIRE. The solution is also robust, having achieved 0.00 THB in VIP Penalties (Scenario 3), confirming its safety and efficiency. This proves that a modest focus on service is not a cost, but a catalyst for achieving true global efficiency.

CHAPTER 5
DISCUSSION AND CONCLUSION

5.1 Discussion of Key Findings

The results presented in Chapter 4 provide empirical validation of a non-linear and counter-intuitive phenomenon in logistics optimization. The analysis moves beyond a simple quantification of costs to interpret the strategic implications of optimizing policy constraints. The findings are discussed in three distinct thematic discoveries.

5.1.1 The Global Optimum and the Failure of Pure Cost Minimization

The most profound finding of this research is the empirical discovery of the Sweet Spot Policy (P_Late = 50), which generated a 15.4% service gain while simultaneously achieving a cost reduction of 88.40 THB (Table 4.3).

This "Win-Win" outcome refutes the conventional linear cost-service trade-off assumption. It proves that the pure cost-focused baseline (Knob = 0) was, in fact, a Local Optimum. The P_Late = 0 policy, by having an over-simplified objective function, trapped the solver in an inefficient solution space, finding only the cheapest route *given its limited view*.

The moderate P_Late =50 penalty acted as an Intelligent Steering Mechanism. It *enriched* the objective function, providing the LNS metaheuristic with just enough new information to justify exploring a different, more "rugged" part of the solution space. In doing so, it was forced to discard the local minimum and converge upon the true Global Optimum a solution path that was inherently more efficient in both sequencing and asset utilization, thus yielding both lower costs and higher service. The 88.40 THB saving is the quantifiable value of this new strategic information.

5.1.2 The Economic Impact of Legacy Constraints

The comparative analysis of Scenario 1 (OT) and Scenario 2 (No-OT) revealed a 13.8% cost leakage. The 2,790.40 THB loss under the legacy policy was not merely an operational variance; it was a structural burden imposed by a flawed policy.

The root cause was not variable cost but sub optimal asset allocation. The rigid 8-hour constraint forced the solver into an inefficient Fleet Mix (3 Cage Trucks, 7 Small Lorries) with a fixed cost of 8,800.00 THB. By removing this artificial constraint (Scenario 2a), the solver was free to achieve optimal fleet utilization, selecting a superior 10-vehicle mix (6 Cage Trucks, 4 Small Lorries) with a fixed cost of only 7,600.00 THB. This 1,200.00 THB saving in fixed costs alone provides crucial evidence that obsolete operational rules can impose severe, hidden financial penalties on an otherwise efficient logistics operation.

5.1.3 The De-coupling of Service Quality from Resource Scarcity

The analysis of the service ceiling (moving from 82.4% Sat. to 99.9% Sat.) confirms that the marginal cost for achieving near-perfect service in the unconstrained model is negligible (+8.50 THB).

This minimal expenditure is a critical insight: it proves that the service failures in the baseline (82.4% Sat) were not a problem of resource scarcity (i.e., not enough trucks or drivers). The resources required for a 17.5% service gain were already available within the network's asset base. The failure was one of combinatorial sequencing. The cost of high service, therefore, was not for *more assets*, but for the *algorithmic precision* required to generate a superior sequencing plan, validating the power of optimization over manual or simplified heuristic approaches.

5.2 Conclusion

This research successfully developed an empirical framework that moved beyond a simplistic cost-minimization exercise to challenge the fundamental logistics cost-service assumption. The investigation conclusively answers the main research question: Yes, a non-linear, globally optimal "Sweet Spot" solution exists.

The final recommended policy achieved a 15.4% service gain while simultaneously reducing the total operational cost by 88.40 THB. This contribution demonstrates that service optimization, when performed using advanced computational methods, functions not as a "cost center," but as a "lever" for achieving superior system-wide efficiency and asset utilization. The research quantified the high cost of legacy constraints (2,790.40 THB) and demonstrated that high service is a function of algorithmic precision, not just resource allocation.

5.3 Strategic Recommendations

The findings dictate a clear paradigm shift in operational policy for DC NP TIRE Management:

- Adopt the Global Optimum Policy: The company must formally adopt the Sweet Spot Policy (P_Late = 50) and the No-OT Planning Model. This strategy provides the best possible trade-off: lowest total cost 17,321.60 THB and highest safe service level 97.8%.
- Shift from Micro-Management to Policy Management: Management should stop managing the 8-hour rule (a micro-level constraint) and start managing the objective function (a macro-level policy). The funds saved by adopting the Sweet Spot policy should be used to cover any necessary driver overtime, as the total expenditure will still be 3,629.45 THB cheaper than the legacy policy (S1b).
- Leverage Algorithmic Robustness: The P_Late = 50$ policy is confirmed to be robust. The 0.00 THB VIP Penalty (Scenario 3) proves that no special, expensive constraints are needed to protect high-value customers; the "Sweet Spot" policy protects them automatically, simplifying the entire operational framework.

5.4 Limitations and Future Research

While the model is highly robust, its deterministic nature imposes certain academic limitations, which in turn suggest clear avenues for future research.

- Limitation: Static Data: The model assumes static demand (q_i) and fixed service times (st_i). This does not account for real-world demand variability or on-site delays.
    - Future Research: Future work should explore Stochastic VRP (S-VRP) to model demand variability and service time uncertainty, creating plans that are not just optimal, but also resilient to disruption.
- Limitation: Static Travel Times: The most critical limitation is the model's reliance on static, average travel times, which does not reflect the congestion patterns of the Bangkok Metropolitan Region.
    - Future Research: The model should be evolved into a Dynamic VRP (D-VRP) by integrating with real-time traffic APIs (e.g., Google Distance Matrix API). This would transform the model from a static strategic planner into a tactical, adaptive routing tool.

REFERENCES

Afshar-Nadjafi, B., & Afshar-Nadjafi, A. (2016). Multi-depot time dependent vehicle routing problem with heterogeneous fleet and time windows. *International Journal of Operational Research, 26(*1), 88-103.

Avci, M., & Topaloglu, S. (2016). A hybrid metaheuristic algorithm for heterogeneous vehicle routing problem with simultaneous pickup and delivery. *Expert Systems with Applications, 53*, 160-171.

Bouanane, K., Amrani, M. E., & Benadada, Y. (2022). The vehicle routing problem with simultaneous delivery and pickup: a taxonomic survey. *International Journal of Logistics Systems and Management*, *41*(1-2), 77-119.

Cuvelier, T., Didier, F., Furnon, V., Gay, S., Mohajeri, S., & Perron, L. (2023, February). Or-tools' vehicle routing solver: A generic constraint-programming solver with heuristic search for routing problems. In *24e congrès annuel de la société française de recherche opérationnelle et d'aide à la décision.*

Despaux, F., & Basterrech, S. (2016). Multi-trip vehicle routing problem with time window and heterogeneous fleet. *International Journal of Computer Information Systems and Industrial Management Applications, 8,* 9-9.

Euchi, J., & Yassine, A. (2023). A hybrid metaheuristic algorithm to solve the electric vehicle routing problem with battery recharging stations for sustainable environmental and energy optimization. *Energy Systems*, 14(1), 243-267.

Garlotta, D. (2001). A literature review of poly (lactic acid). *Journal of Polymers and the Environment, 9*(2), 63-84.

Kaewman, S., & Akararungruangkul, R. (2018). Heuristics algorithms for a heterogeneous fleets VRP with excessive demand for the vehicle at the pickup points, and the longest traveling time constraint: a case study in Prasitsuksa Songkloe, Ubonratchathani Thailand. *Logistics*, *2*(3), 15.

Kabadurmus, O., & Erdogan, M. S. (2023). A green vehicle routing problem with multi-depot, multi-tour, heterogeneous fleet and split deliveries: a mathematical model and heuristic approach. *Journal of Combinatorial Optimization*, *45*(3), 89.

Prokop, A., Helling, H. J., Hahn, U., Udomkaewkanjana, C., & Rehm, K. E. (2005).

Biodegradable implants for pipkin fractures. *Clinical Orthopaedics and Related Research*, *12*(32), 226-233.

Sabet, S., & Farooq, B. (2022). Green vehicle routing problem: State of the art and future directions. *IEEE Access, 10,* 101622-101642.

Stodola, P. (2018). Using metaheuristics on the multi-depot vehicle routing problem With modified optimization criterion. *Algorithms, 11*(5), 74.Wang, Y., Zhe, J.,

Subramanian, A., Penna, P. H. V., Uchoa, E., & Ochi, L. S. (2012). A hybrid algorithm for the heterogeneous fleet vehicle routing problem. *European Journal of Operational Research, 221*(2), 285-295.

Tahir, M. A. (2024). Revolutionizing International Cargo Transportation: A Data-Driven Strategy for Fleet Management Optimization and Workforce Efficiency.

Tan, S. Y., & Yeh, W. C. (2021). The vehicle routing problem: State-of-the-art classification and review. *Applied Sciences*, *11*(21), 10295.

Wang, X., Sun, Y., & Wang, H. (2022). Collaborative multidepot vehicle routing problem with dynamic customer demands and time windows. *Sustainability, 14*(11), 6709.

Yu, Z., Zhang, P., Yu, Y., Sun, W., & Huang, M. (2020). An Adaptive Large Neighborhood Search for the Larger-Scale Instances of Green Vehicle Routing Problem with Time Windows. *Complexity, 2020*(1), 8210630.

Zhang, H., Ge, H., Yang, J., & Tong, Y. (2022). Review of vehicle routing problems: Models, classification and solving algorithms. *Archives of Computational Methods in Engineering*, *29*(1), 195-221.

APPENDICES

APPENDIX A
SOURCE CODE

The following Python source code implements the Heterogeneous Fleet Vehicle Routing Problem with Time Windows and Soft Penalties (HFVRPTW-SP). The code was executed using Google Colab and the OR-Tools library. It defines all customer demands, vehicle types (heterogeneous), distance matrix, and time window constraints. The implementation uses Dimensions for capacity and time, and Soft Penalties (P_Early = 100, P_Late = 50) to model the 'Sweet Spot' policy.

A.1 Data Setup

```
from ortools.constraint_solver import pywrapcp, routing_enums_pb2
import math

weekly_demands = [0, 70, 120, 80, 30, 70, 110, 105, 95, 110, 25, 10, 85, 25, 65, 10]

distance_matrix = [
[0.0, 17.5, 16.6, 12.9, 23.1, 27.0, 26.0, 24.0, 25.0, 21.0, 22.0, 20.0, 19.3, 25.0, 35.0, 29.0],
    [18.1, 0.0, 11.5, 15.0, 16.5, 19.5, 18.5, 17.5, 18.5, 16.0, 16.5, 16.0, 15.5, 17.0, 30.0, 25.0],
    [17.0, 11.7, 0.0, 13.8, 15.5, 18.5, 17.8, 16.8, 17.8, 15.5, 16.0, 16.5, 15.5, 15.0, 16.5, 29.0],
    [13.2, 15.5, 14.1, 0.0, 21.5, 24.5, 23.5, 21.5, 22.5, 19.0, 20.0, 18.0, 17.5, 22.0, 32.0, 26.0],
    [24.0, 17.0, 15.9, 22.1, 0.0, 13.5, 12.5, 13.0, 14.0, 15.0, 16.0, 17.5, 18.0, 15.5, 25.0, 22.0],
    [28.2, 20.3, 19.1, 25.1, 13.8, 0.0, 11.5, 12.5, 13.5, 16.5, 17.0, 18.0, 19.0, 17.5, 24.0, 21.0],
    [27.3, 19.2, 18.4, 24.3, 12.7, 11.6, 0.0, 11.0, 12.0, 15.5, 16.5, 17.0, 18.0, 16.5, 23.0, 20.0],
    [25.0, 18.1, 17.4, 22.1, 13.3, 12.8, 11.1, 0.0, 11.5, 14.0, 15.0, 16.0, 16.5, 15.5, 21.0, 18.5],
    [26.1, 19.2, 18.4, 23.3, 14.4, 13.8, 12.2, 11.7, 0.0, 15.0, 15.5, 16.5, 17.0, 16.0, 22.0, 19.0],
    [21.8, 16.5, 15.9, 19.7, 15.4, 17.0, 15.9, 14.3, 15.5, 0.0, 13.0, 14.0, 14.5, 14.0, 20.0, 17.0],
    [22.8, 17.0, 16.5, 20.7, 16.5, 17.6, 17.0, 15.4, 16.0, 13.2, 0.0, 12.5, 13.0, 13.5, 21.0, 18.0],
    [20.7, 16.5, 17.0, 18.6, 18.1, 18.6, 17.6, 16.5, 17.1, 14.3, 12.7, 0.0, 12.5, 14.0, 20.0, 17.5],
    [20.0, 16.0, 15.9, 18.1, 18.6, 19.7, 18.6, 17.0, 17.6, 14.8, 13.2, 12.7, 0.0, 14.5, 21.0, 17.0],
    [26.1, 17.6, 15.4, 22.8, 15.9, 18.1, 17.0, 15.9, 16.5, 14.3, 13.8, 14.3, 14.8, 0.0, 23.0, 20.0],
    [37.0, 31.6, 30.5, 33.5, 26.2, 25.1, 24.0, 21.9, 23.0, 20.8, 21.9, 20.8, 21.9, 24.0, 0.0, 19.5],
    [30.5, 26.2, 25.1, 27.3, 23.0, 21.9, 20.8, 19.2, 19.7, 17.6, 18.6, 18.1, 17.6, 20.8, 19.5, 0.0]
  ]

time_windows_min = [
  (480, 1020), (480, 555), (510, 525), (540, 555),
  (585, 615), (630, 640), (660, 695), (690, 715),
  (555, 570), (780, 795), (825, 855), (860, 880),
  (885, 920), (930, 940), (960, 990), (1000, 1020)
]

service_times_min = [0, 40, 40, 40, 40, 40, 40, 40, 40, 40, 40, 40, 40, 40, 40, 40]

avg_speed_kmh = 30
time_matrix_min = []
for from_node_dist in distance_matrix:
  row = []
  for to_node_dist in from_node_dist:
    time_hours = to_node_dist / avg_speed_kmh
    time_minutes = math.ceil(time_hours * 60)
    row.append(time_minutes)
  time_matrix_min.append(row)
```

A.2 Vehicle & Fleet Parameters

```
vehicle_capacities = [70, 120]
vehicle_km_costs = [17, 22]
vehicle_fixed_costs_thb = [600, 1000]
demands = weekly_demands

num_vehicles_by_type = [ 5, 8
]
num_vehicles = sum(num_vehicles_by_type)
```

```
num_vehicle_types = len(vehicle_capacities)
```

A.3 Model Setup

```
depot = 0 num_locations = len(distance_matrix)
manager = pywrapcp.RoutingIndexManager(num_locations, num_vehicles, depot)
routing = pywrapcp.RoutingModel(manager)                              (3.1 - 3.3)
```

A.4 Capacity and Distance Constraints

```
def create_distance_callback(distance_matrix, vehicle_costs_per_km, vehicle_type):
def distance_callback(from_index, to_index): from_node =
manager.IndexToNode(from_index) to_node = manager.IndexToNode(to_index)
return int(distance_matrix[from_node][to_node] *
vehicle_costs_per_km[vehicle_type] * 100) return distance_callback for vehicle_id in
range(num_vehicles): vehicle_type_id = vehicle_id % num_vehicle_types
transit_callback_index = routing.RegisterTransitCallback(
create_distance_callback(distance_matrix, vehicle_km_costs, vehicle_type_id))
routing.SetArcCostEvaluatorOfVehicle(transit_callback_index, vehicle_id)
scaled_fixed_costs = [cost * 100 for cost in vehicle_fixed_costs_thb]
vehicle_index_counter = 0 for type_id, num_v in enumerate(num_vehicles_by_type):
for _ in range(num_v): routing.SetFixedCostOfVehicle(scaled_fixed_costs[type_id],
vehicle_index_counter) vehicle_index_counter += 1
```

A.5 Capacity Constraint

```
def demand_callback(from_index): from_node = manager.IndexToNode(from_index)
return demands[from_node] demand_callback_index =
routing.RegisterUnaryTransitCallback(demand_callback) capacities_list = [cap for
cap, num in zip(vehicle_capacities, num_vehicles_by_type) for _ in range(num)]
```

```
routing.AddDimensionWithVehicleCapacity( demand_callback_index, 0,
capacities_list, True, 'Capacity')                                    (3.4 – 3.5)
```

A.6 Time Constraints & Soft Penalties

```
def create_time_callback(time_matrix, service_times):
    def time_callback(from_index, to_index):
        from_node = manager.IndexToNode(from_index)
        to_node = manager.IndexToNode(to_index)
        travel_time = time_matrix[from_node][to_node]
        service_time = service_times[from_node]
        return int(travel_time + service_time)
    return time_callback

time_callback_index = routing.RegisterTransitCallback(
    create_time_callback(time_matrix_min, service_times_min)
)

time_dimension_name = 'Time'
horizon_minutes = 24 * 60
routing.AddDimension(
    time_callback_index, horizon_minutes, horizon_minutes, False,
time_dimension_name
)
time_dimension = routing.GetDimensionOrDie(time_dimension_name)

# --- Sweet Spot Policy Settings (P_Late = 50) ---
penalty_early = 100
penalty_late = 50
# ------------------------------------------------

depot_open_time = time_windows_min[depot][0]
```

```
depot_close_time = time_windows_min[depot][1]

for i in range(num_locations):
    index = manager.NodeToIndex(i)
    window_start = time_windows_min[i][0]
    window_end = time_windows_min[i][1]

    if i == depot:
        time_dimension.CumulVar(index).SetRange(window_start, window_end)
    else:
        time_dimension.SetCumulVarSoftLowerBound(index, window_start,
penalty_early)                                                        (3.6)
        time_dimension.SetCumulVarSoftUpperBound(index, window_end,
penalty_late)                                                         (3.7)

for vehicle_id in range(num_vehicles):
    start_index = routing.Start(vehicle_id)
    time_dimension.CumulVar(start_index).SetRange(depot_open_time,
depot_close_time)
```

A.7 Search Parameters & Execution

```
search_params = pywrapcp.DefaultRoutingSearchParameters()
search_params.first_solution_strategy = (
    routing_enums_pb2.FirstSolutionStrategy.AUTOMATIC)
search_params.time_limit.seconds = 120

solution = routing.SolveWithParameters(search_params)
```

A.8 Solution Printing & Post-Analysis

```
def format_time(minutes):
    hours = int(minutes // 60)
    mins = int(minutes % 60)
    return f"{hours:02d}:{mins:02d}"

def print_solution(manager, routing, solution, distance_matrix, demands,
                   vehicle_capacities, vehicle_km_costs,
                   vehicle_fixed_costs_thb,
                   time_dimension, time_windows, service_times,
                   penalty_early_val, penalty_late_val,
                   vip_nodes, vip_late_penalty_thb):

    vehicle_type_names = ["cage truck", "small lorry"]
    node_names = {
        0: "DC NP TIRE", 1: "NV yangyont", 2: "TR Tyre auto", 3: "V Auto Tire",
        4: "PWM Power Max", 5: "Lamlukka Max Shop", 6: "NumKarnYang Klong 4",
        7: "Klong Chan Auto tire", 8: "G Max Wheels", 9: "Nawamin auto tyre",
        10: "Sermmit Tire", 11: "Sportmag100", 12: "Max Design", 13: "LPAutomag",
        14: "Sor KarnYang Rama2", 15: "71HHH AUTO.TIRE"
    }

    if solution:
        print(f"Objective (Travel + Fixed + Solver Penalties):
{solution.ObjectiveValue() / 100:.2f} THB")
        total_distance = 0
        total_travel_cost = 0

        total_solver_early_penalty = 0
        total_solver_late_penalty = 0
        total_real_vip_penalty = 0
        vip_violation_details = []

        used_vehicles_with_load = []
        node_arrival_times = {}

        for vehicle_id in range(manager.GetNumberOfVehicles()):
            index = routing.Start(vehicle_id)
            route_load = 0
            temp_index = index
            while not routing.IsEnd(temp_index):
                node_index = manager.IndexToNode(temp_index)
                route_load += demands[node_index]
                temp_index = solution.Value(routing.NextVar(temp_index))
            if solution.Value(routing.NextVar(index)) != index and route_load > 0:
                used_vehicles_with_load.append(vehicle_id)
        print(f"\n--- VIP Customer Policy ---")
        vip_names = [node_names[i] for i in vip_nodes if i in node_names]
        print(f"VIP Customers (Must not be late): {', '.join(vip_names)}")
        print(f"VIP Late Penalty (Real Cost): {vip_late_penalty_thb:.2f} THB per
incident")

        print("\n--- Routes for Used Vehicles with Load > 0 ---")
        for count, vehicle_id in enumerate(used_vehicles_with_load):
            index = routing.Start(vehicle_id)
            route_distance = 0
            route_load = 0
            vehicle_type_id = vehicle_id % len(vehicle_capacities)
            vehicle_type_name = vehicle_type_names[vehicle_type_id]

            plan_output = f"Route for Vehicle {count + 1} (Virtual ID: {vehicle_id + 1},
Type: {vehicle_type_name}, Cap: {vehicle_capacities[vehicle_type_id]}, Cost/km:
{vehicle_km_costs[vehicle_type_id]}):\n"

            time_var = time_dimension.CumulVar(index)
            arrival_time_min = solution.Value(time_var)

            node_index = manager.IndexToNode(index)
            plan_output += f" -> {node_names[node_index]} (D:
{demands[node_index]})"
            plan_output += f" [Start Time: {format_time(arrival_time_min)}]\n"

            previous_index = index
            index = solution.Value(routing.NextVar(index))

            while not routing.IsEnd(index):
                time_var = time_dimension.CumulVar(index)
                arrival_time_min = solution.Value(time_var)

                node_index = manager.IndexToNode(index)
                node_arrival_times[node_index] = arrival_time_min
                route_load += demands[node_index]

                tw_start = time_windows[node_index][0]
                tw_end = time_windows[node_index][1]

                early_diff = max(0, tw_start - arrival_time_min)
                late_diff = max(0, arrival_time_min - tw_end)

                status_str = ""
                if node_index in vip_nodes and late_diff > 0:
                    total_real_vip_penalty += vip_late_penalty_thb
                    vip_violation_details.append(f"{node_names[node_index]} (Late
{late_diff} min)")
                    status_str = f" (!!! VIP LATE {late_diff} min !!!)"

                elif late_diff > 0:
                    total_solver_late_penalty += late_diff * penalty_late_val
                    status_str = f" (LATE {late_diff} min)"
                elif early_diff > 0:
                    total_solver_early_penalty += early_diff * penalty_early_val
                    status_str = f" (Early {early_diff} min)"
                else:
                    status_str = f" (On-Time)"

                plan_output += f" -> {node_names[node_index]} (D:
{demands[node_index]})"
                plan_output += f"\n    [Window: {format_time(tw_start)}-
{format_time(tw_end)}]"
                plan_output += f" [Arrive: {format_time(arrival_time_min)}]"
                plan_output += status_str
                plan_output += f"\n"

                route_distance +=
distance_matrix[manager.IndexToNode(previous_index)][manager.IndexToNode(ind
ex)]
                previous_index = index
                index = solution.Value(routing.NextVar(index))

            time_var = time_dimension.CumulVar(index)
            arrival_time_min = solution.Value(time_var)
            node_index = manager.IndexToNode(index)
            plan_output += f" -> {node_names[node_index]}"
            plan_output += f" [Arrive back: {format_time(arrival_time_min)}]\n"

            route_distance +=
distance_matrix[manager.IndexToNode(previous_index)][manager.IndexToNode(ind
ex)]

            route_travel_cost = route_distance * vehicle_km_costs[vehicle_type_id]
            plan_output += f"Total load on route: {route_load} | Distance:
{route_distance:.2f} km | Travel Cost: {route_travel_cost:.2f} THB\n"
            print(plan_output)

            total_distance += route_distance
            total_travel_cost += route_travel_cost

        print(f"\n--- Summary ---")
        print(f"Total number of vehicles used with load > 0:
{len(used_vehicles_with_load)}")
        type1_count = sum(1 for v_id in used_vehicles_with_load if v_id %
len(vehicle_capacities) == 0)
        type2_count = sum(1 for v_id in used_vehicles_with_load if v_id %
len(vehicle_capacities) == 1)
        print(f"  - {vehicle_type_names[0]} (Cap {vehicle_capacities[0]}, Cost
{vehicle_fixed_costs_thb[0]}): {type1_count}")
        print(f"  - {vehicle_type_names[1]} (Cap {vehicle_capacities[1]}, Cost
{vehicle_fixed_costs_thb[1]}): {type2_count}")

        print(f"Total distance of all routes with load > 0: {total_distance:.2f} km")
        print(f"Total load served (across all routes): {sum(demands[1:])}")

        print(f"\n--- Customer Satisfaction Summary ---")
        print(f"{'Node':<20} | {'Ideal Time':<13} | {'Actual Arrival':<14} |
{'Deviation':<20} | {'Satisfaction':>12}")
        print("-" * 87)

        SAT_PENALTY_LATE_PER_MIN = 1
        SAT_PENALTY_EARLY_PER_MIN = 0.1

        all_satisfaction_scores = []
        total_demand_nodes = range(1, manager.GetNumberOfNodes())

        served_nodes = set(node_arrival_times.keys())
        for node_index in vip_nodes:
            if node_index != depot and node_index not in served_nodes:
                total_real_vip_penalty += vip_late_penalty_thb
                vip_violation_details.append(f"{node_names[node_index]} (Not Served)")

        for node_index in total_demand_nodes:
            node_name = node_names[node_index]
            tw_start = time_windows[node_index][0]
            tw_end = time_windows[node_index][1]
            ideal_time_str = f"{format_time(tw_start)}-{format_time(tw_end)}"

            if node_index in node_arrival_times:
                arrival_time_min = node_arrival_times[node_index]
                arrival_str = format_time(arrival_time_min)

                early_diff = max(0, tw_start - arrival_time_min)
                late_diff = max(0, arrival_time_min - tw_end)

                satisfaction = 100.0
                dev_str = ""
```

```
            if node_index in vip_nodes and late_diff > 0:
                dev_str = f"!!! VIP LATE {late_diff} min !!!"
                sat_penalty = late_diff * SAT_PENALTY_LATE_PER_MIN * 10
                satisfaction = max(0, 100 - sat_penalty)

            elif early_diff > 0:
                if early_diff >= 60: dev_str = f"Early {early_diff / 60:.1f} hr"
                else: dev_str = f"Early {early_diff} min"
                sat_penalty = early_diff * SAT_PENALTY_EARLY_PER_MIN
                satisfaction = max(0, 100 - sat_penalty)

            elif late_diff > 0:
                if late_diff >= 60: dev_str = f"Late {late_diff / 60:.1f} hr"
                else: dev_str = f"Late {late_diff} min"
                sat_penalty = late_diff * SAT_PENALTY_LATE_PER_MIN
                satisfaction = max(0, 100 - sat_penalty)

            else:
                dev_str = "On Time"
                satisfaction = 100.0

            all_satisfaction_scores.append(satisfaction)
            sat_str = f"{satisfaction:.0f}%"

        else:
            arrival_str = "---"
            dev_str = "Not Served"
            sat_str = "0%"
            all_satisfaction_scores.append(0.0)

        print(f"{node_name:<20} | {ideal_time_str:<13} | {arrival_str:<14} |
{dev_str:<20} | {sat_str:>12}")

    if all_satisfaction_scores:
        avg_sat = sum(all_satisfaction_scores) / len(all_satisfaction_scores)
        print("-" * 87)
        print(f"{'Average Customer Satisfaction:':<69} {avg_sat:>11.1f}%")
    else:
        print("No customer nodes found to average.")

    print(f"\n--- Cost Breakdown ---")
    print(f"Total travel cost: {total_travel_cost:.2f} THB")

    total_fixed_cost = 0
    for v_id in used_vehicles_with_load:
          vehicle_type_id = v_id % len(vehicle_capacities)
          total_fixed_cost += vehicle_fixed_costs_thb[vehicle_type_id]

      print(f"Total fixed vehicle cost: {total_fixed_cost:.2f} THB")

      print(f"Total VIP Customer Penalty (Real Cost): {total_real_vip_penalty:.2f}
THB")
      if vip_violation_details:
          print(f"  (Violations: {', '.join(sorted(list(set(vip_violation_details))))})")

      print(f"Total Early Penalty (Solver Guide): {total_solver_early_penalty /
100:.2f} THB")
      print(f"Total Late Penalty (Non-VIP, Solver Guide): {total_solver_late_penalty /
100:.2f} THB")

      total_overall_cost = total_travel_cost + total_fixed_cost + total_real_vip_penalty
      print(f"Total overall cost (Travel + Fixed + VIP Penalty):
{total_overall_cost:.2f} THB")

      print(f"(Solver Objective Value (Travel + Fixed + *Solver* Penalties):
{solution.ObjectiveValue() / 100:.2f} THB)")



   else:
      print("No solution found.")
      print("\nPossible reasons for no solution:")
      print("- Total demand exceeds total vehicle capacity.")
      print("- Time windows are too tight / infeasible (e.g., travel time > window).")
      print("- Solver could not find a solution within search limits/time.")

      try:
          total_demands_sum = sum(demands[1:])
          num_vehicles_potential = manager.GetNumberOfVehicles() if manager else
num_vehicles
          total_potential_capacity_sum = sum([vehicle_capacities[i %
len(vehicle_capacities)] for i in range(num_vehicles_potential)])
          print(f"\nTotal demand to serve (excluding depot): {total_demands_sum}")
          print(f"Total potential capacity across all virtual vehicles:
{total_potential_capacity_sum}")
          if total_demands_sum > total_potential_capacity_sum:
              print("Diagnosis: Total demand is greater than total potential capacity.")
          else:
               print("Diagnosis: Total demand is less than or equal to total potential
capacity. Infeasibility is likely due to TIME CONSTRAINTS.")
      except Exception as e:
           print(f"Diagnosis: Could not perform detailed capacity check. Error: {e}")
```

A.9 Main Execution

```
VIP_NODES = {1, 2}
VIP_LATE_PENALTY_THB = 5000.00

if solution:
   print_solution(manager, routing, solution, distance_matrix, demands,
               vehicle_capacities, vehicle_km_costs,
               vehicle_fixed_costs_thb,
               time_dimension, time_windows_min, service_times_min,
               penalty_early, penalty_late,
               VIP_NODES, VIP_LATE_PENALTY_THB)
else:
   print_solution(manager, routing, None, distance_matrix, demands,
               vehicle_capacities, vehicle_km_costs,
               vehicle_fixed_costs_thb,
               None, time_windows_min, service_times_min,
               penalty_early, penalty_late,
               VIP_NODES, VIP_LATE_PENALTY_THB)
```

APPENDIX B
SOLVER OUTPUT

```
Objective (Travel + Fixed + Solver Penalties): 18938.09 THB

--- VIP Customer Policy ---
VIP Customers (Must not be late): NV yangyont, TR Tyre auto
VIP Late Penalty (Real Cost): 5000.00 THB per incident

--- Routes for Used Vehicles with Load > 0 ---
Route for Vehicle 1 (Virtual ID: 1, Type: cage truck, Cap: 70, Cost/km: 17):
 -> DC NP TIRE (D: 0) [Start Time: 08:00]
 -> Lamlukka Max Shop (D: 70)
   [Window: 10:30-10:40] [Arrive: 10:30] (On-Time)
 -> DC NP TIRE [Arrive back: 12:07]
Total load on route: 70 | Distance: 55.20 km | Travel Cost: 938.40 THB

Route for Vehicle 2 (Virtual ID: 5, Type: cage truck, Cap: 70, Cost/km: 17):
 -> DC NP TIRE (D: 0) [Start Time: 08:00]
 -> NV yangyont (D: 70)
   [Window: 08:00-09:15] [Arrive: 09:15] (On-Time)
 -> DC NP TIRE [Arrive back: 10:32]
Total load on route: 70 | Distance: 35.60 km | Travel Cost: 605.20 THB

Route for Vehicle 3 (Virtual ID: 6, Type: small lorry, Cap: 120, Cost/km: 22):
 -> DC NP TIRE (D: 0) [Start Time: 08:00]
 -> Nawamin auto tyre (D: 110)
   [Window: 13:00-13:15] [Arrive: 13:00] (On-Time)
 -> DC NP TIRE [Arrive back: 14:24]
Total load on route: 110 | Distance: 42.80 km | Travel Cost: 941.60 THB

Route for Vehicle 4 (Virtual ID: 7, Type: cage truck, Cap: 70, Cost/km: 17):
 -> DC NP TIRE (D: 0) [Start Time: 08:00]
 -> G Max Wheels (D: 95)
   [Window: 09:15-09:30] [Arrive: 09:15] (On-Time)
 -> Sermmit Tire (D: 25)
   [Window: 13:45-14:15] [Arrive: 13:45] (On-Time)
 -> DC NP TIRE [Arrive back: 15:11]
Total load on route: 120 | Distance: 63.30 km | Travel Cost: 1076.10 THB

Route for Vehicle 5 (Virtual ID: 8, Type: small lorry, Cap: 120, Cost/km: 22):
 -> DC NP TIRE (D: 0) [Start Time: 08:00]
 -> Klong Chan Auto tire (D: 105)
```

```
   [Window: 11:30-11:55] [Arrive: 11:30] (On-Time)
 -> DC NP TIRE [Arrive back: 13:00]
Total load on route: 105 | Distance: 49.00 km | Travel Cost: 1078.00 THB

Route for Vehicle 6 (Virtual ID: 9, Type: cage truck, Cap: 70, Cost/km: 17):
 -> DC NP TIRE (D: 0) [Start Time: 08:00]
 -> NumKarnYang Klong 4 (D: 110)
   [Window: 11:00-11:35] [Arrive: 11:00] (On-Time)
 -> Sportmag100 (D: 10)
   [Window: 14:20-14:40] [Arrive: 14:20] (On-Time)
 -> DC NP TIRE [Arrive back: 15:42]
Total load on route: 120 | Distance: 63.70 km | Travel Cost: 1082.90 THB

Route for Vehicle 7 (Virtual ID: 10, Type: small lorry, Cap: 120, Cost/km: 22):
 -> DC NP TIRE (D: 0) [Start Time: 08:00]
 -> V Auto Tire (D: 80)
   [Window: 09:00-09:15] [Arrive: 09:00] (On-Time)
 -> DC NP TIRE [Arrive back: 10:07]
Total load on route: 80 | Distance: 26.10 km | Travel Cost: 574.20 THB

Route for Vehicle 8 (Virtual ID: 11, Type: cage truck, Cap: 70, Cost/km: 17):
 -> DC NP TIRE (D: 0) [Start Time: 08:00]
 -> Max Design (D: 85)
   [Window: 14:45-15:20] [Arrive: 14:45] (On-Time)
 -> LPAutomag (D: 25)
   [Window: 15:30-15:40] [Arrive: 15:54] (LATE 14 min)
 -> DC NP TIRE [Arrive back: 17:27]
Total load on route: 110 | Distance: 59.90 km | Travel Cost: 1018.30 THB

Route for Vehicle 9 (Virtual ID: 12, Type: small lorry, Cap: 120, Cost/km: 22):
 -> DC NP TIRE (D: 0) [Start Time: 08:00]
 -> TR Tyre auto (D: 120)
   [Window: 08:30-08:45] [Arrive: 08:34] (On-Time)
 -> DC NP TIRE [Arrive back: 09:48]
Total load on route: 120 | Distance: 33.60 km | Travel Cost: 739.20 THB

Route for Vehicle 10 (Virtual ID: 13, Type: cage truck, Cap: 70, Cost/km: 17):
 -> DC NP TIRE (D: 0) [Start Time: 08:00]
 -> PWM Power Max (D: 30)
   [Window: 09:45-10:15] [Arrive: 10:15] (On-Time)
 -> Sor KarnYang Rama2 (D: 65)
   [Window: 16:00-16:30] [Arrive: 16:00] (On-Time)
 -> 71HHH AUTO.TIRE (D: 10)
   [Window: 16:40-17:00] [Arrive: 17:19] (LATE 19 min)
 -> DC NP TIRE [Arrive back: 19:00]
Total load on route: 105 | Distance: 98.10 km | Travel Cost: 1667.70 THB
```

```
--- Summary ---
Total number of vehicles used with load > 0: 10
  - cage truck (Cap 70, Cost 600): 6
  - small lorry (Cap 120, Cost 1000): 4
Total distance of all routes with load > 0: 527.30 km
Total load served (across all routes): 1010

--- Customer Satisfaction Summary ---
Node                 | Ideal Time    | Actual Arrival | Deviation          | Satisfaction
------------------------------------------------------------------------------------
NV yangyont          | 08:00-09:15   | 09:15          | On Time            |    100%
TR Tyre auto         | 08:30-08:45   | 08:34          | On Time            |    100%
V Auto Tire          | 09:00-09:15   | 09:00          | On Time            |    100%
PWM Power Max        | 09:45-10:15   | 10:15          | On Time            |    100%
Lamlukka Max Shop    | 10:30-10:40   | 10:30          | On Time            |    100%
NumKarnYang Klong 4  | 11:00-11:35   | 11:00          | On Time            |    100%
Klong Chan Auto tire | 11:30-11:55   | 11:30          | On Time            |    100%
G Max Wheels         | 09:15-09:30   | 09:15          | On Time            |    100%
Nawamin auto tyre    | 13:00-13:15   | 13:00          | On Time            |    100%
Sermmit Tire         | 13:45-14:15   | 13:45          | On Time            |    100%
Sportmag100          | 14:20-14:40   | 14:20          | On Time            |    100%
Max Design           | 14:45-15:20   | 14:45          | On Time            |    100%
LPAutomag            | 15:30-15:40   | 15:54          | Late 14 min        |     86%
Sor KarnYang Rama2   | 16:00-16:30   | 16:00          | On Time            |    100%
71HHH AUTO.TIRE      | 16:40-17:00   | 17:19          | Late 19 min        |     81%
------------------------------------------------------------------------------------
Average Customer Satisfaction:                                     97.8%

--- Cost Breakdown ---
Total travel cost: 9721.60 THB
Total fixed vehicle cost: 7600.00 THB
Total VIP Customer Penalty (Real Cost): 0.00 THB
Total Early Penalty (Solver Guide): 0.00 THB
Total Late Penalty (Non-VIP, Solver Guide): 16.50 THB
Total overall cost (Travel + Fixed + VIP Penalty): 17321.60 THB
(Solver Objective Value (Travel + Fixed + *Solver* Penalties): 18938.09 THB)
```

APPENDIX C
SOLVER SETTINGS AND CONFIGURATION

The heterogeneous fleet VRP model was implemented in Python using Google Colab and solved with the OR-Tools routing library. Solver settings and platform details are summarized below.

Table C.1 Solver configuration and platform details

Component	Configuration Details
Platform	Google Colab (Python 3 environment)
Solver	OR-Tools Constraint Solver (Routing Library)
Engine	Constraint Programming (CP) with LNS/GLS Metaheuristics
Model Type	Heterogeneous Fleet VRP with Time Windows (HFVRPTW) Constraints: Capacity, Time Windows, Soft Penalties
Time Limit	120 seconds per run (stopping condition for search)
Metaheuristic	Large Neighborhood Search (LNS) / Guided Local Search (GLS)
Search Strategy	AUTOMATIC (Combines first solution heuristics + local search)
Fixed Costs	Applied per vehicle type (Cage Truck = 600 THB, Small Lorry = 1,000 THB)
Output	Route assignments, vehicle usage, total cost (Fixed + Travel + VIP), satisfaction
Validation	Feasibility checked via Capacity and Time Window dimensions

BIOGRAPHY

Name	Peesit Tangtung
Education	2024: Bachelor of Science (Engineering Management)
	Sirindhorn International Institute of Technology
	Thammasat University