12-2023

Data-driven optimization approaches for dynamic urban logistics operational problems

Jingfeng YANG
Singapore Management University, jfyang.2018@phdcs.smu.edu.sg

DATA-DRIVEN OPTIMIZATION APPROACHES FOR DYNAMIC URBAN LOGISTICS OPERATIONAL PROBLEMS

JINGFENG YANG

SINGAPORE MANAGEMENT UNIVERSITY
2023

Data-Driven Optimization Approaches for Dynamic Urban Logistics Operational Problems

by

Jingfeng Yang

Submitted to School of Computing and Information Systems in partial fulfillment of the requirements for the Degree of Doctor of Philosophy in Computer Science

Dissertation Committee:

LAU Hoong Chuin (Supervisor / Chair)
Professor of Computer Science
Singapore Management University

CHENG Shih-Fen
Associate Professor of Computer Science
Singapore Management University

WANG Hai
Associate Professor of Computer Science
Singapore Management University

ZHANG Yingqian
Associate Professor of Industrial Engineering & Innovation Sciences
Eindhoven University of Technology

Singapore Management University
2023

I hereby declare that this PhD dissertation is my original work

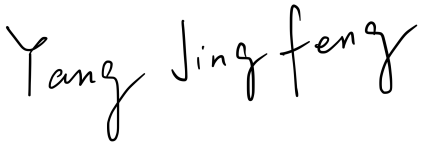
and it has been written by me in its entirety.

I have duly acknowledged all the sources of information

which have been used in this dissertation.

This PhD dissertation has also not been submitted for any degree

in any university previously.



Jingfeng Yang

20 September 2023

Data-Driven Optimization Approaches for Dynamic Urban Logistics Operational Problems

Jingfeng Yang

Abstract

Given the rapid pace of urbanization, there is a pressing need to optimize urban logistics delivery operations for enhanced capacity and efficiency. Over recent decades, a multitude of optimization approaches have been put forth to address urban logistics challenges, encompassing routing and scheduling within both static and dynamic contexts. In light of the rising computational capabilities and the widespread adoption of machine learning in recent times, there is a growing body of research aimed at elucidating the seamless integration of data and machine learning within conventional urban logistics optimization models. Additionally, the ubiquitous utilization of smartphones and internet innovations presents novel research challenges in the realm of urban logistics, notably in the domains of last-mile delivery collaboration and on-demand food delivery services.

The necessity of addressing these emerging challenges is what motivates my doctoral research, with a focus on the investigation of data-driven optimization methodologies. This thesis will encompass a comprehensive discussion of my research conducted in three key domains: (1) collaborative urban delivery with alliances; (2) dynamic service area sizing optimization for on-demand food delivery services; and (3) optimization of dynamic matching time intervals for on-demand food delivery services. The specific details are outlined as follows:

1. We study the pickup and delivery problem within a collaborative framework, focusing on multiple small Logistics Service Providers (LSPs) operating in an urban setting. These LSPs establish trusted alliances, enabling the shared execution of delivery tasks. Specifically, we address a prevalent challenge in urban logistics: the daily operational tasks of LSPs involve collecting goods from one location and delivering them to another, with each request featuring a delivery time window. We

formulate this problem as a Mixed-Integer Programming (MIP) model. To manage the substantial daily delivery volume effectively, we introduce a two-stage approach. First, we determine the allocation of LSP requests among alliances, followed by vehicle routing optimization within each alliance. In the first stage, we propose machine learning models to learn the values of delivery costs from past delivery data, which serve as a surrogate for deciding how requests are assigned. In the second stage, we introduce a tabu search heuristic. Experimental results on a standard dataset show that our proposed learning-based optimization framework is efficient and effective in outperforming the direct use of tabu search in most instances. Using our approach, we demonstrate that substantial savings in costs and, hence, improvement in sustainability can be achieved when these LSPs form alliances and requests are optimally assigned to these alliances.

2. We investigate the combined demand and supply management for on-demand food delivery services by adjusting the radius of their customer service and driver dispatch areas. For each restaurant, the platform needs to decide the (1) customer service area, i.e., the radius of the area within which the customers can see the restaurant's information and order food from it; and (2) driver dispatch area, i.e., the radius of the area within which the drivers can see the restaurant's information and deliver orders from it. Leveraging a real dataset from a food delivery platform, we propose a data-driven optimization framework that combines machine learning methods for order delivery time estimation and an MIP model for the optimization of the two areas at the same time. The objective is to maximize the total number of orders served with minimal impact on the average order delivery time. Extensive experiments using real-world data demonstrate that the proposed framework outperforms several benchmarks in current practice.

3. We focus on the optimization of order dispatching time for on-demand food delivery services by dynamically optimizing the time intervals for dispatching orders on such on-demand food delivery platforms. This study is motivated by a practical

challenge encountered by a food delivery platform, wherein customer orders need to be allocated to couriers responsible for collecting food from designated centers and delivering it to customers within specified time windows. This setting poses a dynamic pickup and delivery problem where prompt delivery without delays is the critical objective. Specifically, we address this challenge by formulating the problem as a Markov decision process (MDP) and proposing a two-stage framework that integrates a multi-agent reinforcement learning (RL) approach for order dispatching and a heuristic method for courier routing. The multi-agent reinforcement algorithm determines the optimal timing for each order's entry into the matching pool, while the routing method incorporates orders into the couriers' delivery routes for pickups and deliveries. Extensive experiments were conducted, evaluating our approach using real-world data and a well-designed simulator. The results demonstrate the superior performance of our proposed framework compared to the currently practiced strategy.

In summary, this thesis addresses new research problems arising from on-demand delivery, drawing upon new methods in AI. Experiments conducted with real-world urban delivery data demonstrate that our proposed data-driven optimization approaches can significantly enhance operational efficiency and reduce delivery costs. This thesis also opens various opportunities for future research, as discussed in the concluding chapter. Specifically, those emerging approaches that leverage machine learning and deep learning to develop optimization methods for vehicle routing problems from end to end for real-world urban delivery scenarios.

Table of Contents

List of Figures

List of Tables

Chapter 1

Introduction

Urban logistics plays a pivotal role in maintaining the city's functionality by ensuring timely delivery of goods to both individuals and businesses. The rapid urbanization and the surge in e-commerce have led to heightened research interest in addressing operational challenges within urban logistics. These challenges include, but are not limited to, vehicle routing in last-mile delivery, resource allocation at urban freight consolidation centers, and order dispatch mechanisms for on-demand food delivery services. Researchers hailing from diverse fields, including Operations Research (OR), Machine Learning (ML), and economics, have collectively directed their focus toward these issues. To tackle these challenges, scholars within the OR community have devised an array of optimization techniques, all aimed at enhancing the operational efficiency of urban logistics systems. Simultaneously, the availability of extensive data collections presents promising prospects for ML researchers to formulate data-driven decisions through effective data mining techniques for prediction tasks. Presently, there is a concerted effort to foster synergy between the fields of OR and ML, with a particular focus on advancing the integration of data and machine learning techniques into conventional models for urban logistics optimization. This thesis draws inspiration from the data-driven optimization frameworks, such as predict-then-optimize framework introduced in [33], the integrated predictive-and-prescriptive modeling framework in [10] and end-to-end optimization via deep Reinforcement Learning (RL) [112]. Our primary objective is to explore data-driven decision-

making optimization strategies tailored to address the challenges of urban logistics within dynamic and complex urban environments.

Thus, motivated by the growing enthusiasm for employing the aforementioned frameworks to address the burgeoning challenges within urban delivery operations, this thesis endeavors to devise novel methodologies that assimilate pertinent techniques, ultimately aiming to enhance operation efficiency for the delivery systems within the data-driven framework context.

1.1 Motivations Problems and Contributions

Within the realm of modern urban logistics, this thesis finds its specific motivation in addressing two real-world applications: collaborative urban delivery and on-demand food delivery.

Collaborative urban delivery, involving multiple entities sharing resources and infrastructure, has gained immense importance as cities grapple with traffic congestion, pollution, and the increasing demand for sustainable deliveries. The quest for optimal collaborative vehicle routing, and resource allocation within this complex framework calls for intelligent, data-driven strategies capable of harnessing the power of real-world data sources to unlock cost savings, minimize environmental impact, and enhance overall service quality.

On the other hand, the surge in on-demand food delivery services has reshaped consumer expectations, placing a premium on timely, accurate, and reliable deliveries. Meeting these expectations necessitates dynamic decision-making that can adapt to different supply and demand scenarios. The utilization of data-driven optimization methods can have a significant impact on facilitating the intricate coordination between restaurants, delivery drivers, and customers, ultimately resulting in the efficient administration of the entire process and ensuring a smooth dining experience.

This thesis embarks on a journey to explore, develop, and apply data-driven approaches to address the multifaceted challenges within collaborative urban delivery and

on-demand food delivery.

1.1.1 Collaboration in Urban Delivery

Collaboration within the logistics industry has been a recurring theme in urban logistics research, typically realized through two distinct approaches: vertical and horizontal collaboration. Vertical collaboration often entails the participation of various stakeholders, such as suppliers, logistics service providers, and customers, within the context of last-mile delivery services. An exemplary form of vertical collaboration is the urban consolidation center, facilitating the separation and consolidation of goods by suppliers and carriers for subsequent delivery. In contrast, horizontal collaboration exclusively encompasses logistics service providers (LSPs) operating at the same supply chain tier. As an illustration, there exist numerous small-scale LSPs offering pickup and delivery services with constrained vehicle fleets within a highly competitive environment. LSPs execute their daily operations, involving the pickup of goods from one location and their subsequent delivery to another, all while adhering to specified delivery time windows. In the absence of cooperation, each LSP independently devises route plans for their respective requests. With the overarching goal of enhancing delivery efficiency and cost-effectiveness, we explore a collaborative approach involving the existence of multiple alliances in the market, enabling LSPs within the same alliance to pool requests and collectively determine routing decisions. Nevertheless, as the number of participating LSPs and requests within an alliance expands, the complexity of solving the joint routing problem escalates. Within this thesis, we formulate the collaborative delivery challenge as a Multi-Alliance Multi-Depot Pickup and Delivery Problem with Time Windows (MAD-PDPTW) problem and introduce a novel learning-based optimization framework to address it.

1.1.2 On-Demand Food Delivery Service

On-demand food delivery platforms like Grab Food, Uber Eats, and Meituan have experienced rapid growth in recent years, particularly during the COVID-19 pandemic. These

platforms enable customers to order their preferred meals from restaurants using mobile applications. According to the data from Meituan, the total daily orders placed by customers are over 30 million. The food delivery service presents significant challenges, primarily stemming from the complexity of orchestrating the supply of delivery resources to align with fluctuating and unpredictable customer demand within exceptionally stringent timeframes. For instance, the majority of restaurants are mandated to complete food deliveries within a one-hour. Extensive literature review reveals that much of the research in food delivery operations concentrates on the development of optimization algorithms aimed at addressing order assignment and driver routing challenges. The objective is to efficiently allocate prepared orders to drivers and formulate optimal routes to ensure punctual deliveries. Simultaneously, achieving a balance between supply and demand in the context of food delivery has garnered considerable attention. n this thesis, our primary focus centers on addressing two intricate operational challenges within the on-demand food delivery service domain: (1) the problem of optimizing area sizing for effective supply and demand management; and (2) the challenge of determining the optimal dispatching time intervals for order assignment and driver routing.

Area Sizing Optimization

Typically, supply and demand management techniques rely on dynamic pricing strategies, which entail modifying the delivery charges for specific restaurants. However, the efficacy of this approach may be limited in practice due to the relatively low prices associated with deliveries. Recently, alternative strategies have emerged to address the supply and demand equilibrium in food delivery services by directly modifying a restaurant's customer service area. For instance, Meituan introduced an innovative mechanism to define the delivery scope, represented as a spatial polygon serving as the customer service area, for restaurants in [29]. In this thesis, we expand upon this approach by exploring integrated demand and supply management strategies for on-demand food delivery services. Our research encompasses modifications not only to the customer service area on the demand side but also to the driver dispatch area on the supply side. Chapter 5 introduces a

data-driven optimization framework that integrates machine learning techniques for estimating order delivery times and a Mixed-Integer Programming (MIP) model for simultaneous optimization of both the demand-side customer service area and the supply-side driver dispatch area.

Dynamic Matching Time Interval Optimization

Conventional order dispatching strategies implemented in practical settings (e.g., [151]) typically involve fixed time intervals (e.g., every 10 minutes), within which the platform accumulates all incoming orders and consolidates them into a matching pool. Subsequently, the orders are collectively assigned to couriers for simultaneous delivery. However, recent investigations in the realm of ridesourcing (e.g., [57, 140]) have revealed that by extending the dispatching time interval, it is possible to optimize the allocation of couriers to passengers, leading to reduced wait times for both parties involved. Drawing inspiration from the efficacy of extended dispatching time intervals in ridesourcing, our research concentrates on the dynamic optimization of order dispatching time intervals for on-demand food delivery services. Specifically, our focus lies in determining the opportune moment for orders to enter the matching pool to initiate the dispatching process, given their arrival on the platform.

1.1.3 Contributions

This thesis presents two principal contributions. Firstly, we delineate and formalize the novel and intricate operational challenges entailed in collaborative urban delivery and on-demand food delivery services. Secondly, we introduce innovative data-driven optimization approaches that intelligently integrate established frameworks to effectively address these emerging problem variants, which, owing to their complexity and the dynamic nature of urban environments, often pose challenges beyond the capabilities of traditional optimization methods, such as heuristic approaches.

1.2 Structure of the Thesis

This thesis is organized as follows:

- Chapter 2 delves into the foundational concepts and prior research related to the general and dynamic aspects of pickup and delivery challenges. It also encompasses a detailed examination of various data-driven optimization strategies that have been investigated to tackle complex operational issues in urban logistics. These frameworks are pivotal for enhancing efficiency and responding to the urban delivery systems.

- In Chapter 3, we offer a comprehensive exposition of background information and relevant research pertaining to both collaborative urban delivery and on-demand food delivery service challenges. Additionally, we delve into optimization methodologies devised to address these operational intricacies. We furnish an in-depth elucidation of these two problem domains, accompanied by an extensive examination of pertinent research in these areas.

- Chapter 4 delves into the urban delivery challenge within alliance-based logistics, cast as a collaborative pickup and delivery problem. We introduce an innovative learning and optimization framework designed to resolve this complex issue. We substantiate the importance of the introduced learning and optimization framework, showcasing its ability to achieve reduced delivery costs while demanding less computational time. Additionally, we extract valuable managerial insights relevant to Logistics Service Providers (LSPs). Furthermore, this research has garnered recognition, being accepted for presentation at the *International Conference on Computational Logistics 2021* and the *30th International Joint Conference on Artificial Intelligence (IJCAI-21) workshop on Data Science meets Optimization*.

- Chapter 5 centers on the area size optimization challenge within on-demand food delivery services. Here, the platform dynamically orchestrates the equilibrium between supply and demand by flexibly adapting the radius of both its customer ser-

vice and driver dispatch areas. We introduce a data-driven optimization framework that amalgamates machine learning techniques for order delivery time estimation with mathematical programming to concurrently optimize both the customer service and driver dispatch areas. Furthermore, we subject our approach to rigorous evaluation utilizing a real dataset obtained from a food delivery platform, affirming the superior performance of the proposed framework when compared to several existing benchmarks. The paper version is under review in journal *Transportation Research Part C: Emerging Technologies*.

- In Chapter 6, we focus on the dynamically optimizing the time intervals of dispatching orders for on-demand food delivery platforms. Specifically, we address this challenge by formulating the problem as a Markov decision process (MDP) and proposing a two-stage framework that integrates a multi-agent reinforcement learning (RL) approach for order dispatching and a heuristic method for courier routing. The multi-agent reinforcement algorithm determines the optimal timing for each order's entry into the matching pool, while the routing method incorporates orders into the couriers' delivery routes for pickups and deliveries. Extensive experiments were conducted, evaluating our approach using real-world data and a well-designed simulator. The results demonstrate the superior performance of our proposed framework compared to the currently practiced strategy. The paper version is under review in journal *Transportation Research Part E: Logistics and Transportation Review*.

- In the final chapter, Chapter 7, we bring this thesis to a close by presenting our concluding thoughts on its primary contributions and delving into prospective avenues for future research that can arise from the endeavors undertaken in this thesis.

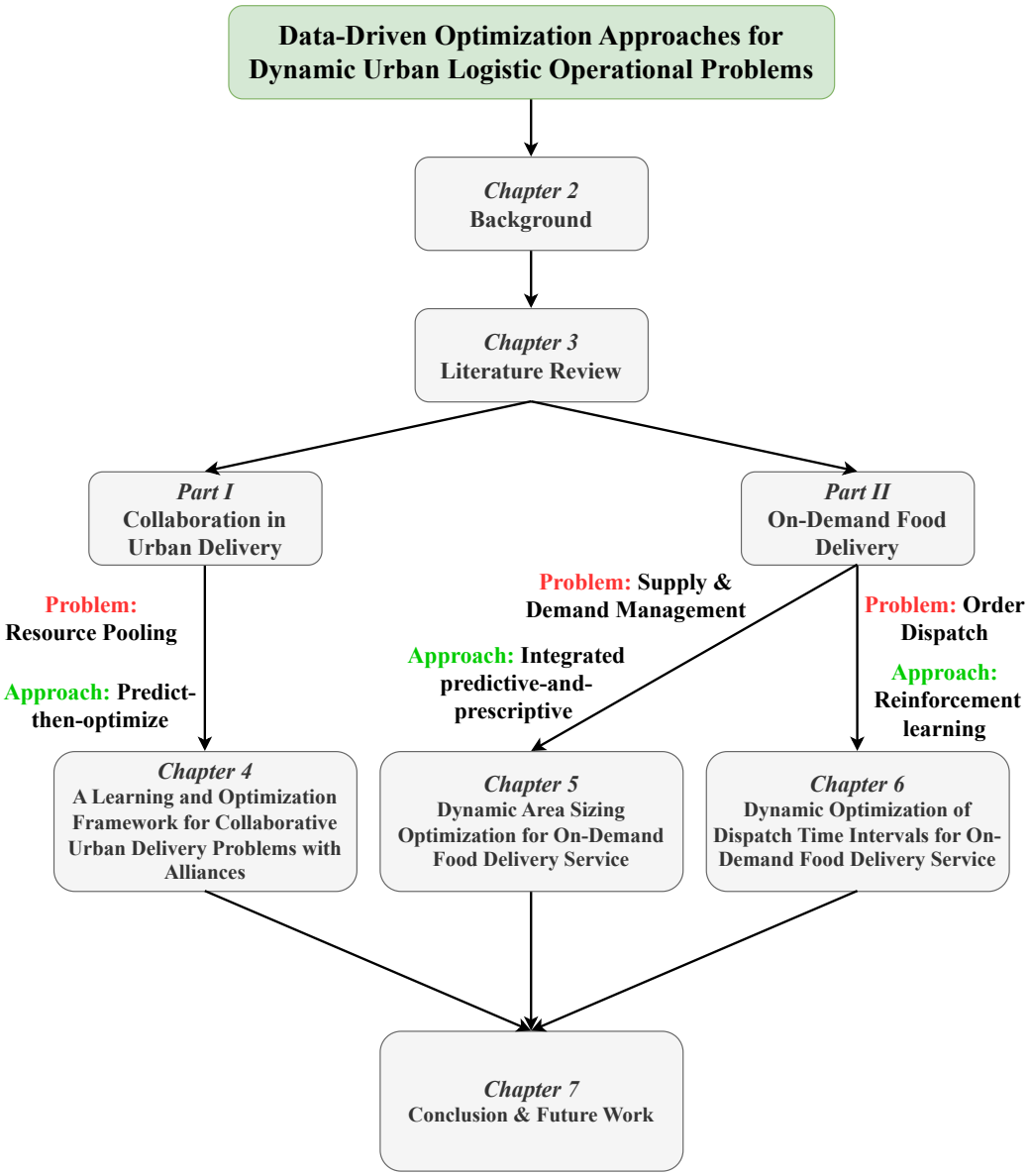


Figure 1.1: The organization of the dissertation.

Chapter 2

Background

In this chapter, we commence by establishing essential terminology and notation, a foundational step to enhance the readability and comprehension of this thesis. Subsequently, we delve into the exposition of background information and related research within the domain of urban logistic operational problems, along with the utilization of data-driven approaches to address such challenges. Finally, we offer comprehensive insights into the motivating problems and their associated scholarly contributions.

2.1 Terminologies

To enhance clarity and facilitate a deeper understanding of the content within this thesis, we provide comprehensive definitions and discussions of key terminologies in the following paragraphs.

Data-driven Optimization. In this thesis, we define data-driven optimization as the branch of optimization methods that primarily uses data as a source of information and knowledge to formulate and solve optimization problems. Data-driven optimization approaches are pivotal in designing models, solving problems, and making decisions to improve operational efficiency and effectiveness. In data-driven optimization, machine learning and statistical methods are often integrated to discern patterns and insights from data, refining the optimization models and enhancing predictive accuracy. The imple-

mentation of these approaches is crucial for dealing with uncertainties and variability in complex systems, providing a robust framework for informed decision-making and enabling organizations to adapt to changing environments and achieve operational excellence. In subsequent subsections, we shall furnish a detail introduction to the data-driven optimization frameworks employed within this dissertation.

Dynamic. In the context of urban delivery services, the term "dynamic" signifies the constant evolution and adaptability required to address the multifaceted challenges inherent to city logistics and operations. From an OR perspective, dynamic approaches are essential to contend with the uncertainties and real-time variations prevalent in urban environments, such as fluctuating demand, traffic conditions, and route alterations. These dynamic elements necessitate the development of advanced, data-driven optimization models and algorithms that can rapidly adjust to changing inputs and constraints, providing real-time, optimal solutions for delivery routing and scheduling. Take the DPDP as an example, the general structure of such dynamic problem is illustrate in Figure 2.1. In a dynamic setting, some of the input data (e.g., orders locations, time windows) are revealed or updated during the period of time in which operations take place (drivers are delivery orders follow by the predetermined delivery plans). Contrary to a static PDP where the planning horizon is predetermined, the planning horizon of a dynamic PDP may potentially be unbounded. Consequently, resolving a dynamic problem does not yield a static output. Instead, it necessitates the development of a optimization strategy that, utilizing the information unveiled over time, delineates the requisite actions, which is represented by the step 3 in Figure 2.1 to be executed as time progresses. Step 3 distinctly signifies the central research focus of this dissertation, posing critical inquiries regarding: *the methodologies for employing real-world data resources to apprehend and forecast the intricate dynamics inherent in urban logistic delivery services, and the strategies for implementing pioneering data-driven optimization techniques, such as reinforcement learning, with an aim to augment the overall operational efficiency of specified urban logistic systems.*

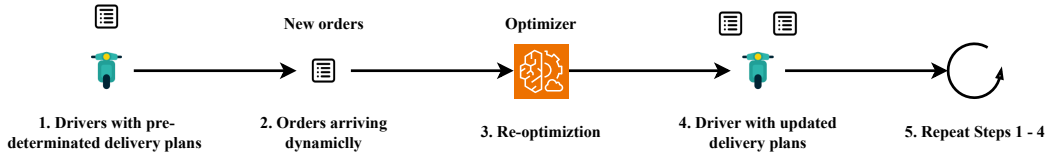


Figure 2.1: An illustration of the general structure of a dynamic pickup and delivery problem in urban delivery service.

2.2 Pickup and Delivery Problem

The Pickup and Delivery Problem (PDP) is a classic problem in the field of operations research and urban logistics optimization. It involves determining the most efficient way to transport goods or items between a set of pickup locations and their corresponding delivery locations, typically using a fleet of vehicles. The PDP can be defined as a variant or extension of Vehicle Routing Problem (VRP). The VRP made its initial introduction in the work by [23]. For an in-depth exploration of VRP, its various problem variants, and solution methodologies, we direct interested readers to the comprehensive overview provided in [118]. The general PDP was first introduced in [108]. In the PDP, each pickup location is paired with a corresponding delivery destination. The primary objective is to determine optimal routes for a fleet of vehicles, facilitating the collection of items at their designated pickup points and their subsequent delivery to corresponding destinations. This optimization process aims to minimize specific objective functions, including total distance traveled, total time, and total cost. Crucial attributes and factors to contemplate in the context of the PDP encompass:

- *Vehicle Capacities*: Each vehicle possesses a finite carrying capacity, and it is imperative that the aggregate demand for items to be picked up remains within this stipulated capacity.

- *Time Windows*: Temporal limitations can be linked to the timing of pickups and deliveries. For instance, a pickup point may have restricted accessibility within specific timeframes.

- *Multiple Vehicles*: Typically, multiple vehicles are available for utilization, and the

challenge lies in determining the appropriate vehicle to allocate for each corresponding pickup and delivery pair.

- *Objective Function*: The goal is to minimize a selected measure of cost, potentially comprising the total distance traversed, the overall time consumed, or an amalgamation of assorted expenses, including vehicle operational costs and driver remunerations.

The PDP boasts myriad applications within the realm of urban logistics, encompassing E-commerce deliveries, on-demand food delivery services, and waste collection, to name a few. A common refinement of the traditional PDP, in the aforementioned application scenarios, is the dynamic PDP (DPDP), which accommodates the dynamic nature inherent in real-world logistics operations. In such practical scenarios, new pickup and delivery requests may emerge progressively, complicating the problem due to its dynamic essence. Unlike the traditional PDP, which necessitates predefined route planning, the DPDP demands instantaneous decision-making to integrate emerging requests into existing routes or to formulate new routes promptly. Additionally, uncertainties often associated with the arrival times, locations, and attributes of new requests augment the challenges in decision-making.

Attaining an optimal solution to the DPDP represents a significant computational challenge, prompting researchers to develop an array of algorithms and heuristics intended to uncover near-optimal solutions efficiently. These solutions contribute to enhanced efficiency, reduced transportation costs, and improved customer service in delivery operations. For a more comprehensive discussion on the DPDP, general issues, and solution strategies, we direct readers to [8, 15].

2.3 Data-driven Optimization Frameworks

In OR, data-driven optimization frameworks refer to structured approaches that utilize empirical data to inform, validate, and refine decision-making processes and optimiza-

tion models. These frameworks leverage vast amounts of real-time and historical data to uncover patterns, relationships, and insights, enabling the development of more accurate, robust, and effective optimization models for solving complex operational problems. This thesis draws inspiration from the existing data-driven optimization frameworks, such as predict-then-optimize framework [109, 85], the integrated predictive-and-prescriptive modeling framework [10] and end-to-end optimization via deep Reinforcement Learning (RL) [112]. Our primary objective is to explore data-driven decision-making optimization strategies tailored to address the challenges of urban logistics within dynamic and complex urban environments. We provide a concise introduction to these data-driven frameworks in following subsections.

2.3.1 Predict-then-optimize

The "predict-then-optimize" framework is a decision-making approach commonly used in the field of operations research and optimization. It combines predictive modeling with optimization techniques to make better decisions in situations where there is uncertainty or variability in the underlying data. Typically, as described in [33], this framework comprises two distinct phases: the prediction step and the optimization step. During the prediction step, the framework estimates the unknown parameters related to an optimization problem, while the subsequent optimization step employs these predictions to solve the underlying optimization problem. As an illustrative example, consider the last-mile delivery problem within urban logistics, characterized by the recurrent occurrence of solving a vehicle routing problem with multiple daily resolutions. Initially, a well-trained prediction (e.g., Machine Learning) model furnishes an estimate of travel time. This estimate relies on current traffic conditions, weather, time, and other pertinent factors. Subsequently, the routing algorithm employs the predicted travel time as input to determine the most optimal delivery routes for drivers. We now describe the "Predict-then-optimize" framework in practice in a mathematical optimization way. Consider an optimization problem of interest with a linear objective function $\mathbf{c}^T\mathbf{x}$, where the deci-

sion variables $\mathbf{x} \in \mathbb{R}^n$ are precisely defined and part of the cost parameters $\mathbf{c} \in \mathbb{R}^n$ in the objective function or part of the input parameters $\mathbf{A} \in \mathbb{R}^{m \times n}$ and $\mathbf{b} \in \mathbb{R}^m$ are unavailable.

$$\min \quad \mathbf{c}^T \mathbf{x} \tag{2.1}$$

$$s.t. \quad \mathbf{A}\mathbf{x} = \mathbf{b} \tag{2.2}$$

$$\mathbf{x} \in \mathbb{R}^n \tag{2.3}$$

Typically, if the undetermined parameters $(\mathbf{c}, \mathbf{A}, \mathbf{b})$ can be initially discerned from historical data through the application of trained ML models, then resolving this optimization problem can be substantially facilitated. We describe the "predict-then-optimize" framework in Figure 2.2.

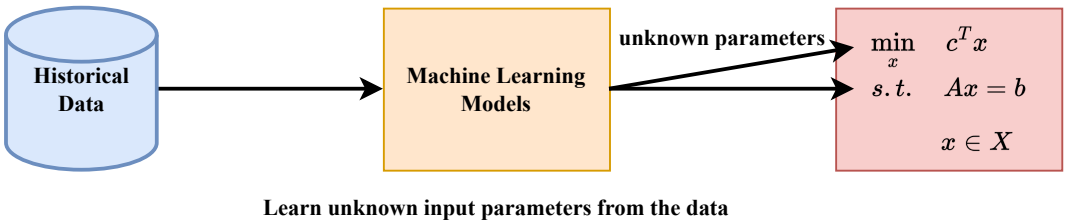


Figure 2.2: An illustration of the "predict-then-optimize" framework.

Recently, [33] proposed a novel data-driven framework termed "Smart Predict-then-Optimize" (SPO) aimed at aligning the objectives of prediction and optimization [33]. Differing from the standard "predict-then-optimize" framework, the SPO approach specifically addresses the decision error resulting from inaccurate predictions by introducing a loss function that measures this discrepancy. To address computational challenges in model training, they have developed a convex surrogate loss function. Complementing this, El et al. [32] offer a range of generalization bounds for the SPO loss function. Extending the application of this framework, Mandi et al. [81] explored its utility in solving discrete optimization problems more realistically. The SPO framework has been further adapted to address real-world transportation problems, including ship maintenance planning [116], efficient ship inspection [136], and traffic signal control [139].

2.3.2 Integrated predictive-and-prescriptive

The integrated predictive and prescriptive framework is a modeling paradigm that seamlessly integrates two crucial components: a prediction model and an optimization model within a unified framework. This framework enables researchers and practitioners to incorporate machine learning models into an optimization context. The primary innovation lies in offering modeling constructs that enable the explicit incorporation of widely employed predictive models, along with their associated characteristics, as both constraints and variables within the optimization model. The integration of ML techniques and mathematical optimization has gained considerable scholarly interest, as evidenced by notable works like [83, 2]. Several software packages [10, 16] and the platform introduced in [92] have been developed to facilitate a more profound integration between predictive and prescriptive analytics. The integrated predictive-and-prescriptive modeling framework introduced by [10] aims to solve an optimization problem as follows:

$$\max_{\boldsymbol{x}} \quad \sum_{j=1}^{n_1} c_j x_j + \sum_{k=1}^{n_2} d_k y_k \tag{2.4}$$

$$\textbf{s.t.} \quad \sum_{j=1}^{n_1} a_j^i x_j \leq b_i, \quad \forall i \in \{1, \cdots, m\} \tag{2.5}$$

$$y_k = g_k(\alpha_1^k, \cdots, \alpha_{p_k}^k; \theta_k), \quad \forall k \in \{1, \cdots, n_2\} \tag{2.6}$$

$$\alpha_l^k = e_l^k \cdot x, \quad \forall l \in \{q_k + 1, \cdots, p_k\}, k \in \{1, \cdots, n_2\} \tag{2.7}$$

$$x_j \in X_j, \quad j \in \{1, \cdots, n1\}. \tag{2.8}$$

The framework consider two set of decision variables: (1) regular variables $x = (x_1, x_2, \cdots, x_{n_1})$, and (2) predicted variables $y = (y_1, y_2, \cdots, y_{n_2})$. Each variable x_j belongs to a finite or continuous set X_j and are constrained via linear inequalities. The predicted variables y_k are set by predictive (e.g., machine learning) models, each predicted variable y_k is associated with a pre-trained predictive model g_k (e.g., linear regression, decision tree or neural network) with trained parameters θ_k, input features

$\alpha^k = (\alpha_1^k, \cdots, \alpha_{p_k}^k)$, where p_k denotes the number of features for model g_k. Note that the features are divided into two parts, the first q_k ($1 \leq q_k \leq p_k$) features are fixed and given, and the remaining $p_k - q_k$ features are regular variables determined by equation (2.7). e_l^k is a n_1-length binary unit-vector with a 1 in the position of the corresponding regular variable and 0 elsewhere. Readers can refer to [10] for details. We illustrate the "Integrated predictive-and-prescriptive" framework in Figure 2.3.

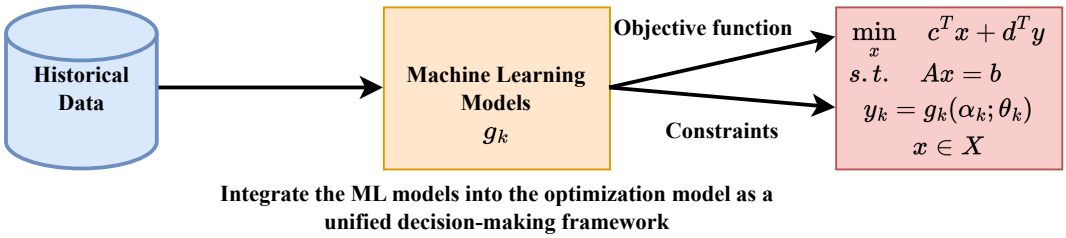


Figure 2.3: An illustration of the "integrated predictive and prescriptive" framework.

2.3.3 Reinforcement Learning

In recent years, the application of Reinforcement Learning (RL) has emerged as a groundbreaking approach to tackle complex end-to-end optimization problems across various domains. The adaptability and capacity of RL to learn optimal strategies through interaction with dynamic environments have made it particularly effective in data-driven optimization scenarios for urban environments, such as traffic signal control [50], rail-hailing marketplace [105], logistics and supply chain management [137]. RL is usually used to solve sequential decision-making problems, which is the Markov decision process (MDP). RL and MDP are pivotal in the realm of decision-making under uncertainty. RL, symbolized by the interaction between an agent and an environment, revolves around maximizing cumulative rewards $E(G_t|S_t = s)$. The agent's learning mechanism, described through sequences of states s, actions a, and rewards r, is effectively modeled using MDP. An MDP can be defined by a tuple (S, A, P, R). In this context, the agent observes the current state $s \in S$, takes an action $a \in A$, transitions to a new state $s' \in S$ based on the probabilistic $p(s'|s, a)$, and receives a immediate reward $r(s, a)$. The ob-

jective is to determine a optimal policy π^*, which is a mapping from states to actions (s, a), that maximizes the expected cumulative reward. RL is a robust method for addressing complex decision-making problems across various domains. We illustrate the MDP framework in Figure 2.4 and summarize the common notation used in Table 2.1.

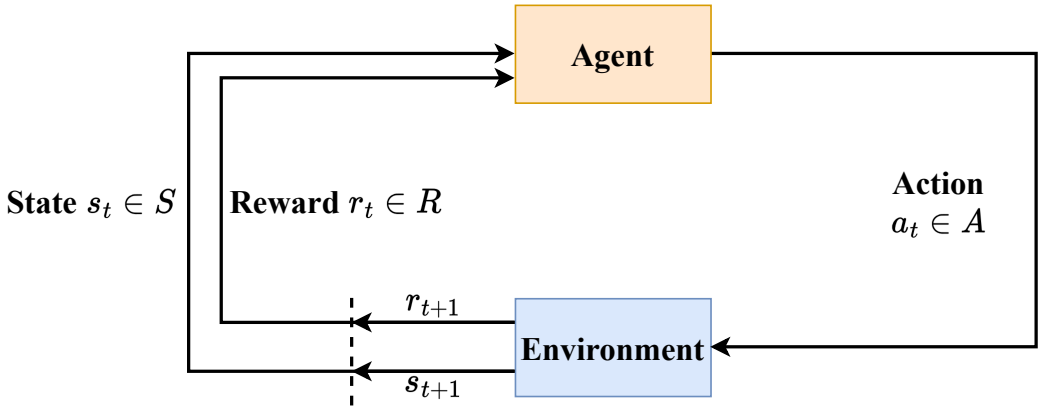


Figure 2.4: An illustration of the MDP framework, adapted from [112].

In general, there are two main approaches to train our agent to find this optimal policy π^*, value-based and policy-based approaches. In the value-based paradigm, algorithms like Q-learning endeavor to compute a value function $V(s)$ or action-value function $Q(s, a)$. These functions capture the expected cumulative reward starting from state s or upon taking action a in state s, respectively. An optimal policy $\pi^*(s)$ is subsequently derived from these values. Conversely, policy-based methods, exemplified by techniques like policy gradient, operate by directly optimizing the policy function $\pi(s)$ without the intermediary of a value function.

2.4 Problems Settings

In this section, we discuss the comprehensive descriptions of the motivating problems and addressing potential data-driven optimization approaches to such problems.

Table 2.1: Notation for MDP and RL.

Notation	Description
S	Set of states
A	Set of actions
R	Set of rewards
P	Probability transition model
t	Discrete time step in the planning time horizon
π	Policy (decision making rule)
π^*	Optimal policy
$\pi(s)$	Action taken in state s under a deterministic policy π
S_t	State at time t
A_t	Action at time t
R_t	Reward at time t
$v_\pi(s)$	Value of state $s \in S$ under policy π
$v_*(s)$	Value of state $s \in S$ under policy π_*
$q_\pi(s)$	Value of taking action a at state $s \in S$ under policy π
$q_*(s)$	Value of taking action a at state $s \in S$ under policy π_*
G_t	Reward following time t
$p(s'\|s,a)$	Probability of transition to state s' from state s
$r(s,a)$	Expected immediate reward from state s after take action a
$V(s)$	Estimate of state-value function
$Q(s,a)$	Estimate of action-value function

2.4.1 MAD-PDPTW

We consider a new variant of the collaborative urban delivery problem which is Multi-Alliance Multi-Depot Pickup and Delivery Problem with Time Windows (MAD-PDPTW). In the market, numerous small logistics service providers (LSPs) specialize in providing urban pickup and delivery services. In pursuit of additional delivery cost reduction, these LSPs engage in collaborative efforts to establish various alliances. Within an alliance, LSPs operate with a shared distribution center, commonly referred to as a depot, and jointly utilize vehicles for their operations. Orders from LSPs belonging to the same alliance are consolidated and delivered collectively. It is important to highlight that an individual LSP may opt to participate in multiple alliances to realize more substantial cost reductions. Starting from the perspective of a third-party platform that facilitates these alliances, the primary objective is the minimization of the overall distribution cost, which is equivalent to the total delivery distance. To provide a more comprehensive un-

derstanding of this matter, we illustrate this new in Figure 2.5. We propose a two-stage learning and optimization approach which follows the predict-then-optimize framework to solve this problem.

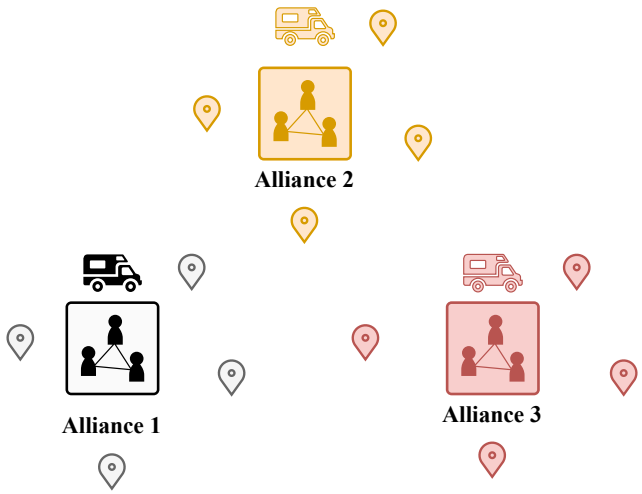


Figure 2.5: An simple illustration of the MAD-PDPTW.

2.4.2 Restaurant Area Sizing Optimization

We introduce a new approach for balancing the supply and demand for on-demand food delivery service. For each restaurant in a specified operating time horizon, considering from the platform perspective, we need decide the (1) customer service area (CSA), the radius of the surrounding area within which customers can see the restaurant's information and order food from it; and (2) driver dispatch area (DDA), the radius of the surrounding area within which drivers can see the restaurant's information and deliver orders from it. We endeavor to mitigate the issue of supply-demand imbalance in the context of food delivery for restaurants by dynamically adjusting both CSA and DDA simultaneously. Our primary objective is to maximize the served order quantity, concurrently ensuring that the average delivery time adheres to predetermined specifications. We named this problem as Restaurant Area Sizing Optimization (RASO) problem, and give an illustration in Figure 2.6. We proposed a data-driven optimization approach with the integrated predictive-and-prescriptive modeling framework to solve this problem.

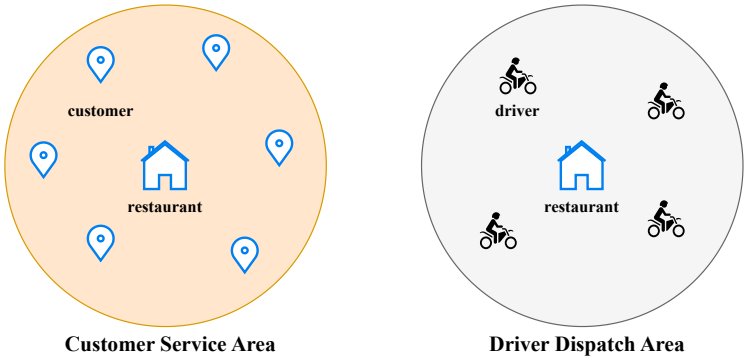


Figure 2.6: A simple illustration of the restaurant's CSA and DDA.

2.4.3 Dispatch Time Interval Optimization

We address the order dispatch challenge in on-demand food delivery services and introduce a novel dispatching strategy through the dynamic optimization of order dispatching time intervals. In this context, orders, accompanied by pickup and delivery locations and stipulated delivery times (e.g., 45 minutes) set by the platform, arrive dynamically within a planning horizon. We define the "matching pool" as a dynamic reservoir comprising available orders placed by customers awaiting dispatch to drivers for pickup and delivery. Approaching the problem from the platform's perspective, we aim to determine: (1) the optimal timing for releasing orders into the matching pool, and (2) the most efficient method to allocate orders in the matching pool to available drivers. Our primary goal is to curtail the total delivery distance for drivers while maintaining a high service level for processed orders and adhering to the order delivery times. To address this challenge, we advocate a two-stage optimization framework utilizing reinforcement learning techniques.

Chapter 3

Literature Review

In this chapter, we commence by providing an introduction to the background information and the existing body of research that addresses operational challenges within collaborative urban delivery and on-demand food delivery services. Subsequently, we conduct a comprehensive review of the extant literature to pinpoint research gaps.

3.1 Collaborative Urban Delivery Problems

This section offers a concise overview of prior studies that center on logistics collaboration and distance approximation within the context of vehicle routing problems (VRP). Logistics collaboration has been a prominent subject within urban logistics studies, typically categorized into two forms: vertical and horizontal collaboration [107]. In this thesis, our primary focus lies on horizontal collaboration, which entails the cooperation of logistics service providers (LSPs) operating at the same supply chain tier. A comprehensive examination of the prospects and challenges associated with horizontal collaborative logistics services was undertaken by [21]. They conducted a survey involving 1,537 logistics service providers (LSPs) in Belgium, and the results indicated that the majority of LSPs hold the belief that collaboration can lead to enhanced profitability and service quality.

Horizontal Collaboration: Numerous studies on horizontal collaboration within logistics systems have been conducted in recent decades. Interested readers can find more

comprehensive information in [38] and [123]. These studies can be broadly categorized into two main themes: (1) the development of optimization models and mechanisms for collaborative network planning and design, aimed at assisting Logistics Service Providers (LSPs) in profit maximization and cost reduction; and (2) the introduction of cooperative and non-cooperative game theory approaches for equitable cost/gain sharing, facilitating the establishment and sustainability of improved collaborations. Our study will primarily concentrate on optimization models pertinent to collaborative multi-LSP delivery problems, and the literature review will be conducted accordingly. [9] introduced a decentralized control and auction-based exchange mechanism aimed at maximizing overall profits through collaborative efforts among individual carriers. [60] pursued a similar line of research, focusing on centralized control with iterative auctions designed to minimize the distance traveled with empty vehicles. [22] investigated a pickup and delivery vehicle routing problem with time windows (PDVRPTW) with the objective of minimizing the overall delivery cost. [63] examined the pickup and delivery problem with requests exchange, aiming to maximize total profits. [95] addressed a multi-depot vehicle routing problem with the objective of minimizing the total traveled distance, employing a local search method. In contrast to request exchange or vehicle sharing, [37] introduced a novel vehicle routing problem wherein customers can be served by multiple carriers. The objective is to minimize overall operational costs through such collaborative arrangements. For a more extensive exploration of various vehicle routing problems within a collaborative setting, interested readers can consult the survey conducted by [42].

Approximations of Routes: The Vehicle Routing Problem (VRP) has received considerable attention over the past decade, resulting in the development of numerous exact and heuristic algorithms aimed at achieving optimal solutions or reducing computational time. In contrast to optimization algorithms, which aim to obtain optimal or high-quality solutions, Continuum Approximation (CA) models have emerged as tools to approximate route distances without tackling the intricacies of solving the complex routing problem. These CA models offer rapid and reasonably accurate estimates of route distances, making them valuable in various applications, including terminal design prob-

lems [93], supply chain distribution network design [68], and collaboration mechanism design [41]. However, when confronted with large-scale complex problems, most CA approaches exhibit limited accuracy. Recently, a few studies [69, 91] have turned to machine learning approaches to directly estimate the total travel distance of routes. In this thesis, we introduce a machine learning approach designed to estimate the delivery cost for the Pickup and Delivery Problem with Time Windows (PDPTW). Leveraging the learned cost estimates, we can seamlessly integrate them into the request assignment procedure, facilitating the allocation of orders to the appropriate alliances.

As previously discussed, the majority of studies have concentrated on optimizing collaborative planning and operational issues from the standpoint of an entire coalition, where all Logistics Service Providers (LSPs) are part of a single coalition. [146] explored the problem of decision-making in less-than-truckload collaboration for e-commerce logistics networks, with the objective of maximizing the total profit of the entire alliance. To the best of our knowledge, [48] were the pioneers in addressing the coalition configuration problem, allowing companies to participate in multiple coalitions (in this thesis, we prefer to use the term "alliance") within collaborative transport. They developed an optimization model to facilitate the identification of the optimal coalition configuration, thus revealing research gaps through the examination of existing literature. In our thesis, alliances are introduced as inputs in our model, enabling LSPs within one alliance to collaborate on requests and engage in centralized planning for urban delivery services. More specifically, our focus centers on the optimization of collaborative urban delivery services, where certain LSPs have the flexibility to engage in multiple alliances.

3.2 On-Demand Food Delivery Service

My thesis are closely related to four streams of literature: (1) supply and demand management, (2) order dispatching and driver routing, (3) delivery time prediction, and (4) learning-based methods for on-demand platforms.

Supply and demand management. Customer demand and driver supply are two

sides of on-demand platforms, which are usually unbalanced. Many scholars paid attention to supply and demand management in food delivery [125, 67] and ride-sourcing markets [128], including service area management and dynamic pricing.

In the area of service area management, to the best of our knowledge, [143] are the first to analyze how service area size impacts the profit of a delivery platform. The authors derive a functional dependency between revenue and service area size and other parameters, such as customer arrival rate, revenue per customer, compensation per delivery and miles traveled, and customer satisfaction. [82] also investigates how the service area, which is called the "service outlet", affects a store's supply and demand. [120] study the dynamic service area sizing problem in an urban delivery and model it as a Markov decision process, and propose a value function approximation method to decide the radius of the customer service area. [4] study the customer service area problem and focus on matching the levels of supply and demand. A mixed-integer programming model is proposed to determine the optimal service radius that maximizes the number of orders served. [29] study restaurant delivery scope problem for restaurant using machine learning algorithms to rank potential delivery scopes and then combinatorial optimization to select delivery scope. In on-demand ride-sourcing market, [140] investigate the optimal matching radius with an objective of enhancing system efficiency in terms of passenger waiting time, vehicle utilization, and matching rate.

In the area of surge pricing, [117] study the dynamic pricing strategy for Online-to-Offline (O2O) on-demand food service in China from both theoretical and empirical perspectives. They demonstrate that platforms that employ dynamic pricing strategies have much more demand than platforms that use static pricing systems. [80] provide an empirical investigation of the adoption of a dynamic pricing algorithm in an environment with time-varying demand and firm capacity restrictions in restaurants. They find that dynamic pricing can reduce demand volatility, which results in an increase in the proportion of transactions during periods of low demand. [62] investigate the pricing strategies of online food delivery platforms through the lens of demand-supply interaction models. A mathematical model has been developed to ascertain the optimal service

charge and wage rate aimed at maximizing profits. [152] adjust the actual supply and demand through two-sided pricing and constructed a streamlined model to examine the optimal pricing and compensation for a platform catering to customers sensitive to delays and agents sensitive to income. [6] propose a pricing framework for an on-demand service platform, and examine how various factors affect the optimal price, wage, and commission with an objective of maximizing the platform's profit or social welfare. Numerous studies have investigated the implementation of pricing strategies to regulate supply and demand in ride-sourcing markets. [13] examine spatial pricing discrimination in a ride-sharing platform, emphasizing the influence of demand patterns on pricing, profits, and consumer surplus. [154] propose a Mean-field Markov decision process to depict the dynamics in ride-sourcing systems with mixed agents for spatial-temporal subsidies to solve the supply-demand imbalance issue. [71] investigates the impacts of the prevailing threshold-based driver incentives on ride-sourcing drivers' labor supply with extensive ride-sourcing dataset. Extensive relevant research exists on the pricing and incentive challenges within the ride-sourcing market, spanning various perspectives, such as pricing for pooling services [58, 144, 5, 56, 72]; pricing for platform's regulations [65, 124]; driver incentives and multi-homing [110, 3, 49]; and third-party platform-integration [153].

Order dispatching and driver routing. Order dispatching is important for on-demand transportation services such as ride-hailing, ride-sharing, and food delivery. For ride-hailing, the task is dispatching vehicles to serve passengers; for food delivery, the task is dispatching drivers with recommendations of delivery routes to deliver food within a promised time period. Readers who are interested in ride-hailing can refer to [145, 135, 99, 77, 76].

For relevant studies on food delivery services, the order dispatching and driver routing problem is formalized as meal delivery routing problem (MDRP) in [104], which is a variant of dynamic pickup and delivery problem (DPDP) and has been studied in recent decades [97]. In a recent study on MDRP, [142] formulate the problem assuming that the platform has perfect information about order arrivals and solve it using a combined column and row generation approach. [121] formulate the problem as a stochas-

tic dynamic pickup and delivery problem using a route-based Markov decision process (MDP). An anticipatory customer assignment policy is proposed to for order dispatching and vehicle routing. [133] aim to improve the working conditions of drivers by order dispatching algorithm design. A queuing-model-based algorithm is proposed with the objective of minimizing the waiting time of drivers while ensuring a good user experience. In addition, machine learning methods are commonly employed in food delivery order dispatching and driver routing problem. On can refer to [14, 151, 44, 17, 149].

Delivery time prediction. Delivery time prediction is a variant of the ETA (Estimated Time of Arrival) problem [131, 132, 127]. It estimates travel time between two points, which is often approximated by analytical functions or predicted using machine learning and deep learning models. Compared with the traditional ETA problem, delivery time prediction in on-demand food delivery services is more challenging since the delivery time is endogenously affected by the demand and supply in the market. Only limited works focus on order delivery time prediction in food delivery. [69] develop a data-driven framework that integrates travel time prediction and order dispatching for a single restaurant. [51] propose two methods: an offline method that predicts order arrival time based on state features by means of gradient-boosted decision trees (GBDTs), and an offline-online method that exploits an offline supervised learning approximation with a deep neural network to perform detailed online simulations in real-time. [44] discuss a time prediction module that simultaneously predicts order pickup time from the restaurants, driving time on the road, and delivery time to the customer's location. In the context of online retailing, [106] develop a data-driven framework to predict the distribution of order delivery time and set the delivery time promised to customers using tree-based machine learning models.

Learning-based methods. We observe that more recent studies recognize the value of integrating machine learning and reinforcement learning approaches in solving the operational problems in urban logistics, such as vehicle routing problems and its variants [88, 59, 55, 27, 31]; driver-passenger matching problem in ride-sourcing market [135, 114, 54, 150, 57, 100]; last-mile delivery [18, 64]; and pickup and delivery problems for

on-demand food delivery services [155, 14, 73].

[19] solve a driver dispatching problem with a multi-agent reinforcement learning to maximize the revenue of served requests. This work demonstrates the effectiveness of RL in practical dispatch problems. [30] model the last-mile package delivery problem as a MDP and develop a reinforcement learning model to learn order dispatching strategies from massive passenger data and package data of a crowd-sourcing delivery system. [66] develop a graph relational leaning approach for large-scale practical DPDP considering many realistic constraints. Solving a similar problem, [78] present a hierarchical RL solution framework that the upper-level agent dynamically partitions the DPDP into static sub-problems, then a lower-level agent solves those sub-problems by local search. [35] proposes a dynamic on-demand crowd-shipping solution based on deep reinforcement learning. By embedding heuristic strategies and constraints into a double dueling deep Q-network (DDQN), the optimization of vehicle routing is achieved to improve the efficiency and cost-effectiveness of crowd-shipping logistics. [70] develop an improved inverse reinforcement learning method for food delivery route planning with the consideration of deliverymen's preferences. To our best of knowledge, [14] is the first one who study the order dispatching and driver routing by using DDQN to guide the driver to pickup an order and deliver it to destination. Although the conducted experiments exhibit a certain level of simplicity, they solely focus on the routing aspect of a singular driver.

3.2.1 Area Sizing Optimization

In Table 3.1, we present an overview of the relevant literature that focuses on the area sizing optimization problem for same-day delivery services (SDD) and food delivery services (FD). The table delineates the specific characteristics and features explored in the Chapter 5: "Objective" specifies the objective function employed in the optimization model; "Decision" indicates the decision variables to be optimized; "Methodology" identifies the modeling method used; "Approach" identifies the primary approaches employed to solve

the model; "Instance" denotes whether the experiments conducted in the respective studies employed real-world datasets as instances for evaluation.

In chapter 5, we present the first work to jointly optimize the customer service area and driver dispatch area, which requires demand estimation and order delivery time prediction. This is a new lever for balancing supply and demand in food delivery services.

3.2.2 Dynamic Dispatching Time Interval

We summarize the most relevant literature that focuses on optimizing the dispatching (matching) time of orders and drivers (vehicles). The majority of existing research on optimizing matching time intervals primarily concentrates on the ride-sourcing sector. [57] proposed a reinforcement learning model to determine the delay time for each order, which aligns with our settings. However, their method uses fixed time intervals and can only make matches at the end of each interval. [140] propose a spatial probability model that characterizes the matching process between idle drivers and passengers awaiting service in a ride-sourcing system. [98] also explore the issue of delayed matching in ride-hailing services using reinforcement learning algorithms. They advocate for a suite of policy-based methods, specifically Actor-Critic (AC) and ACER (Actor-Critic with experience replay), to determine the optimal matching time interval. [126] have developed an innovative real-time driver-request assignment algorithm that takes into account both the waiting time in matching decisions and the pickup distance in the assignment process. However, their model operates under the assumption that drivers remain on the platform until they are assigned a new request and the customer leaves with a predefined quitting distribution.

Above studies are applicable only to ride-sourcing platforms, whereas the optimization of matching time intervals has also garnered considerable attention outside the ride-sourcing problems. [138] employ a policy-based approach to identify an optimal withholding strategy within the resource allocation problem, specifically aimed at improving on-site service delivery in urban logistics. [1] study the allocation problem in remote tele-

operation, they model the problem as online matching in bipartite graphs which allows for delayed assignment. An LP-based online algorithm algorithm is proposed to solve the problem with a competitive ratio of 0.5. [134] study the general online resource allocation problem and have shown that a slight delay can be beneficial in the context of real-time online decision-making.

In the context of on-demand food delivery services, [148] have developed real-time matching algorithms. These algorithms strategically postpone the pairing of drivers with orders to accommodate the variability in food preparation times. Their findings suggest that this intentional delay contributes to a more robust marketplace, which, in turn, enhances the availability of drivers and orders, thereby reducing the overall operational costs. Additionally, [78] utilize reinforcement learning techniques to tackle the dynamic pickup and delivery problem (DPDP), a foundational routing challenge in the sphere of on-demand delivery services. They propose a hierarchical framework with an upper agent that dynamically partitions the DPDP into a series of sub-problems to optimize vehicle routing. Similar to our model, they also take into account soft time window constraints and vehicle routing decisions. Our model specifically caters to the food delivery service, considering stricter constraints on order delivery time and driver routing decisions. The main difference lies in the fact that in our model, each order is treated as an independent agent capable of deciding when to enter the matching pool, whereas they only permit the release of all accumulated orders at once. Furthermore, by allowing order re-dispatch, we have developed a novel heuristic-based driver routing algorithm for order pickups and deliveries, while they employ a reinforcement learning-based neighborhood search algorithm.

Table 3.1: Literature on area sizing optimization on food delivery and same-day delivery.

	Paper	Objective	Decision	Methodology	Approach	Instance
SDD	[120].	Served orders	Service Radius	MDP	CA + VFA	×
	[7].	Served orders	Service Radius	IP	Rolling horizon + CA	✓
FD	[143].	Total profit	Service Radius	IP	AfQ + FOA	×
	[4].	Served orders	Service Radius	IP	Solver	×
	[29].	GMV	Candidate Service Scopes	IP	ML + Heuristics	✓
	RASO-LT.	Served orders	Service & Dispatch Radius	MIQCP	MNL + ML + Solver	✓

Note. GMV: Global Merchandise Volume; MDP: Markov decision process; IP: integer programming; CA: continuous approximation; VFA: value function approximate; AfQ: approximation for queueing; FOA: first-order algorithm MNL: multinomial logit model; ML: machine learning.

Chapter 4

A Learning and Optimization Framework for Collaborative Urban Delivery Problems with Alliances

4.1 Motivation and Background

With the rapid pace of urbanization, it has become imperative to optimize urban delivery systems for enhanced capacity and efficiency. The surging demand for deliveries not only poses challenges for major Logistics Service Providers (LSPs) like Amazon and Cainiao but also intensifies competition among small and medium-sized LSPs. Given the inherent uncertainty in daily delivery demands and locations, LSPs encounter operational challenges spanning from excess idle capacity to shortages in vehicles and manpower.

To address these challenges, one approach is to foster collaboration among logistics entities. As elucidated by Savelsbergh and Woensel in [107], collaboration or cooperation is often recognized as a strategic avenue to consolidate freight volumes, ultimately leading to more efficient and optimal resource utilization. Alliances formed by two or more companies create opportunities for shared information and resources, enabling the joint handling of delivery tasks. The subject of collaboration within urban logistics systems has garnered significant research attention in recent years.

In this chapter, we study the pickup and delivery routing problem in a collaborative setting. In particular, we consider the problem that frequently occurs in urban delivery: LSPs perform their daily operations to pickup goods from one location and deliver to another location, and each request has a delivery time window. In the absence of collaboration, each LSP independently devises route plans for their respective set of requests. However, in the collaborative routing context, we assume the existence of multiple alliances within the market. LSPs belonging to the same alliance have the capacity to share requests and collectively make routing decisions. For simplicity, we assume that LSPs within a given alliance utilize a common depot as the base for their vehicle operations. Additionally, an LSP may choose to participate in more than one alliance, potentially to cater to different types of goods. Importantly, it is worth noting that this chapter does not address the coalition structure generation problem, which concerns the partitioning of agents into mutually exclusive coalitions to maximize the overall total reward over the long term. Instead, we operate under the assumption that the structure of alliances, including the composition of LSPs within each alliance, is predefined and serves as an input parameter for our model. Our primary focus lies in addressing the operational challenge of achieving efficient deliveries within an environment where an LSP may be affiliated with multiple alliances.

From a sustainability perspective, the ideal scenario involves cooperative LSPs, wherein the planning process optimally operates within an existing alliance structure, aiming to maximize the overarching system-wide objective of minimizing the total travel cost. We formulate this complex problem as the Multi-Alliance Multi-Depot Pickup and Delivery Problem with Time Windows (MAD-PDPTW).

The primary contributions of this chapter can be succinctly summarized as follows: (1) We introduce a Mixed-Integer Programming (MIP) model, which serves as a formal representation for the MAD-PDPTW; (2) We develop an effective tabu search-based heuristic method, designed to tackle the complexities of the problem, particularly when dealing with large instances; (3) To enhance the efficiency of problem-solving, we employ a two-stage approach, following the predict-then-optimize framework introduced

in Chapter 1, decomposing the MAD-PDPTW into a sequential process. First, we utilize data to learn the delivery cost, followed by the optimization of request reassignment and vehicle routing; (4) We highlights the significance of the proposed learning and optimization framework, showcasing its ability to achieve lower delivery costs with reduced computational time. Furthermore, we provide valuable managerial insights for LSPs.

4.2 Problem Formulation

In this section, we present our collaborative urban logistics delivery problem within the framework of multiple LSPs and multiple alliances. Given that each LSP may have its specialization in handling various types of goods (e.g., groceries and electronics), which may or may not be co-loaded within the same vehicle, and each LSP may also maintain its set of trusted partners, it is plausible to have multiple alliances with participants that overlap.

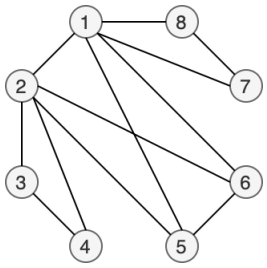


Figure 4.1: Multiple alliances with overlapping LSPs.

Figure 4.1 illustrates an example comprising 8 Logistics Service Providers (LSPs) and 3 alliances. Each node represents an LSP, and the presence of an edge between two nodes signifies the ability of those two LSPs to share requests. In this context, an alliance is defined as a complete sub-graph, where a unique edge connects every pair of distinct vertices. Specifically, in this example, we have alliances denoted as $[2, 3, 4]$, $[1, 2, 5, 6]$, and $[1, 7, 8]$. The primary objective of this study is to evaluate the potential advantages of collaborative routing among LSPs, involving the sharing of requests and coordinated planning. This entails the operation of a centralized platform responsible for determining the optimal assignment of requests within each alliance, with the crucial constraint that requests

cannot be shared across different alliances.

If this problem is considered comprehensively, it necessitates concurrent decisions regarding the distribution of LSPs' requests across various alliances and the routing of assigned requests within each alliance. Notably, for even a modest-sized problem instance, conventional meta-heuristic approaches like Tabu Search might not be computationally efficient, and as our experimental data indicates, may not yield an effective solution. Before presenting our mathematical programming model, we first introduce the notation in Table 4.1.

Table 4.1: Notation.

Notation	**Description**
G	A complete direct graph
N	Set of all LSPs
A	Set of all alliances as well as depots
K	Set of vehicles
R	Set of all requests, each request r has a pickup node and delivery node
P	Set of pickup nodes
D	Set of delivery nodes
V	Set of all nodes in graph G
K_a	Set of vehicles only belong to alliance a
d_a	Depot node for alliance a
$[e_i, l_i]$	Time windows for node i, earliest pickup time and latest delivery time
s_i	Service time at location i
q_i	Weight of goods to pickup or delivery at node i
c_{ij}	Travel cost between node i and node j
t_{ik}	Time node i served by vehicle k
w_{ik}	Weight of vehicle k after visit node i
Q	Vehicle capacity
y_{ijk}	Binary variable, 1 if the vehicle k visited node j directly after visited node i, 0 otherwise

In light of the aforementioned notation, we formalize the MAD-PDPTW as outlined below:

$$\textbf{min} \quad \sum_{i \in V} \sum_{j \in V} \sum_{k \in K} c_{ij} y_{ijk} \tag{4.1}$$

$$\textbf{s.t.} \quad \sum_{i \in V} \sum_{k \in K} y_{ijk} = 1 \quad \forall j \in P \cup D \tag{4.2}$$

$$\sum_{i \in V} y_{ijk} - \sum_{i \in V} y_{jik} = 0 \quad \forall j \in P \cup D, \forall k \in K \tag{4.3}$$

$$\sum_{i \in V} y_{id_a k} = \sum_{j \in P \cup D} y_{d_a jk} \leq 1 \quad \forall a \in A, \forall k \in K_a \tag{4.4}$$

$$\sum_{j \in P \cup D} y_{ijk} - \sum_{j \in P \cup D} y_{(i+|R|)jk} = 0 \quad \forall i \in R, \forall k \in K \tag{4.5}$$

$$t_{ik} + s_i + c_{ij} - M(1 - y_{ijk}) \leq t_{jk} \quad \forall i, j \in P \cup D \tag{4.6}$$

$$e_i \leq t_{ik} \leq l_i \quad \forall i \in P \cup D, \forall k \in K \tag{4.7}$$

$$t_{ik} \leq t_{(i+|R|)k} \quad \forall i \in P \tag{4.8}$$

$$t_{ak} = 0 \quad \forall a \in A, \forall k \in K \tag{4.9}$$

$$w_{jk} \leq w_{ik} + q_j + M(1 - y_{ijk}) \; \forall i, j \in V, \forall k \in K \tag{4.10}$$

$$w_{jk} \geq w_{ik} + q_j - M(1 - y_{ijk}) \; \forall i, j \in V, \forall k \in K \tag{4.11}$$

$$w_{ik} \leq Q \quad \forall i \in V, \forall k \in K \tag{4.12}$$

$$y_{ijk} = 0 \quad \forall i \notin V_a, \forall j \notin V_a, \forall k \in K_a \tag{4.13}$$

We categorize the constraints into four distinct groups. The first group focuses on the flow in and out between each pickup and delivery node. Constraint (4.2) ensures that each pickup or delivery node is visited exactly once. Constraint (4.3) stipulates that the same vehicle k must serve each pickup or delivery node. Constraint (4.4) imposes constraints on each depot and ensures that each vehicle k belongs to depot K_l will start and back to depot d with at most once. Constraint (4.5) guarantees the pickup node i and delivery node $i + |R|$ belonging to one request will be served within the same tour.

The second set of constraints pertains to the visiting precedence of pickup nodes, delivery nodes, and time windows. Constraint (4.6) represents the Miller-Tucker-Zemlin

(MTZ) sub-tour elimination constraint. If $y_{ijk} = 1$, then we have $t_{ik} + s_i + c_{ij} \leq t_{jk}$; otherwise, we establish a constraint with a right-hand side (RHS) set to a sufficiently large positive value. Constraint (4.7) imposes time window restrictions, ensuring that the delivery times for each request fall within the specified time window. Constraint (4.8) enforces a precedence condition that ensures each request is serviced at its pickup node before the delivery. Lastly, Constraint (4.9) signifies that the arrival time for each vehicle at the depots is equal to 0.

The third set of constraints pertains to the capacity limitations. Constraints (4.10) through (4.11) compute the vehicle's weight after visiting each node. Furthermore, we establish that $q_i = -q_{i+r}$ for $i \in R_p$. Constraint (4.12) stipulates that for each vehicle k, after serving node i, its load cannot surpass the predetermined capacity. The final constraint concerns request assignment, ensuring that vehicles affiliated with one alliance are restricted from delivering requests associated with another alliance. In essence, each alliance assumes responsibility for its own requests.

4.3 Two-Stage Learning and Optimization Framework

The preceding section presents a Mixed-Integer Programming (MIP) model aimed at determining the optimal request assignment and routing for multiple alliances. Within this MIP model, the decision variable y_{ijk} serves a dual purpose: it not only determines the delivery sequence from node i to node j, but also plays a pivotal role in making decisions for Logistics Service Providers (LSPs) participating in multiple alliances regarding request assignment (namely, choosing the alliance for sharing requests). However, it is imperative to note that the underlying problem is NP-hard, rendering it computationally intractable for larger instances. To address this challenge, we propose a comprehensive learning and optimization framework consisting of two distinct stages, encompassing request assignment to vehicle routing.

Specifically, the first stage focuses on resolving the assignment of requests to alliances for LSPs participating in multiple alliances (as detailed in Section 4.3.1). Subsequently, the second stage employs a tabu search-based heuristic algorithm to address the Pickup and Delivery Problem with Time Windows (PDPTW) for each alliance, incorporating the assigned requests (as outlined in Section 4.3.2). The entire framework is illustrated in Figure 4.2.

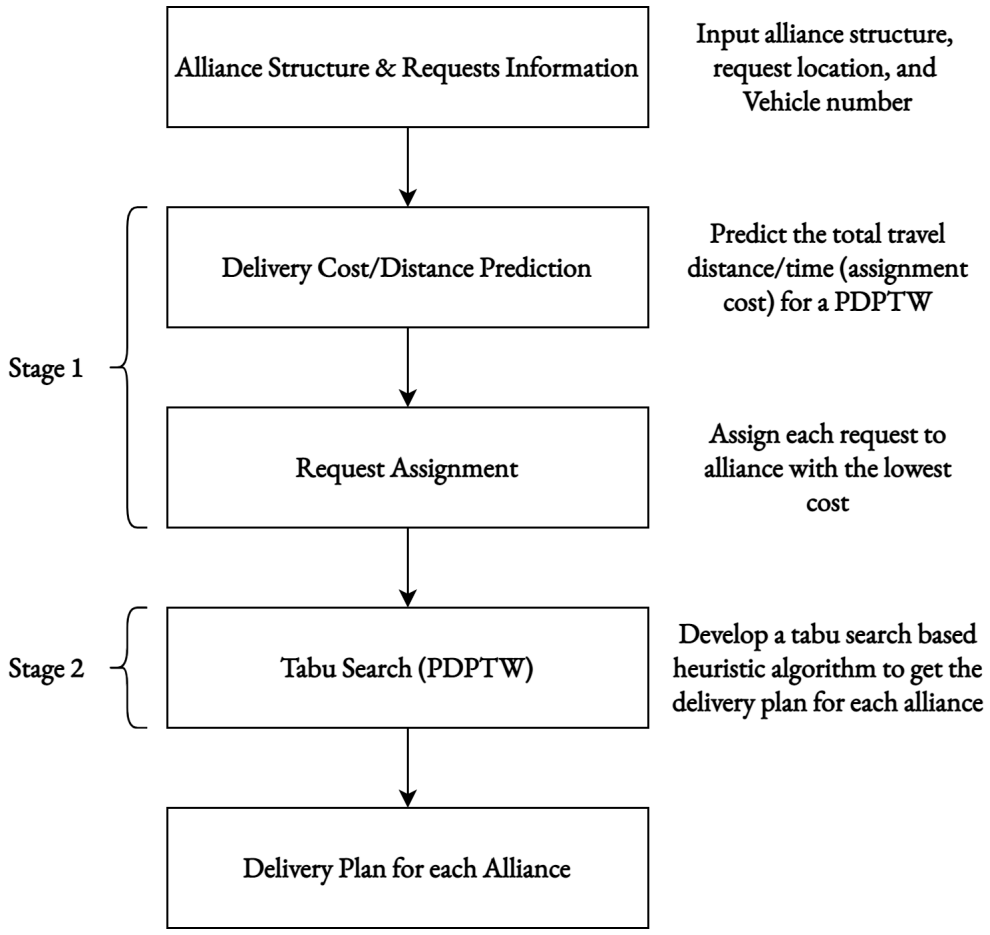


Figure 4.2: A two-stage learning and optimization framework to solve the MAD-PDPTW.

4.3.1 Delivery Cost Prediction and Request Assignment

In this subsection, we initially delve into the development of a predictive model for estimating the delivery cost associated with each alliance. Subsequently, we employ the estimated delivery cost as input parameters for the request assignment process.

Cost prediction: Previous research has introduced approximate analytical formulas for TSP and VRP across various application scenarios, as documented in the literature. However, analytical approaches tend to exhibit suboptimal performance when confronted with larger problem instances or real-world complexities, such as the Capacity Vehicle Routing Problem with Time Windows (CVRPTW). In light of these challenges, we turn to machine learning models to forecast the delivery cost in the context of the Pickup and Delivery Problem with Time Windows (PDPTW). In this study, we consider the delivery cost as the aggregate travel distance.

To initiate our cost prediction process, we first generate a set of promising features that encapsulate factors impacting the total travel distance. These predictors can be categorized broadly based on factors such as the number of locations, the geographical area covered, inter-node distances, node distribution, time windows, and route quantity. Table 4.2 provides an exhaustive listing of all the features incorporated into our learning model, comprising a total of 19 distinct attributes employed within our prediction model.

Table 4.2: Features for total travel distance prediction.

Features	**Definitions**
f_1	Number of locations need to be visited
f_2, f_3	Min/max distance between customers and depots
f_4, f_5	Min/max x distance between customers and depots
f_6, f_7	Min/max y distance between customers and depots
f_8	Average distance between customers and depots
f_9	Average x distance between customers and depots
f_{10}	Average y distance between customers and depots
f_{11}	Standard deviation of distance between customers (and depots)
f_{12}	Area of the smallest rectangle covering customer locations
f_{13}	Area of the smallest rectangle covering customer and depot locations
f_{14}	Sum of the length of time windows
f_{15}	Standard deviation of the length of time windows
f_{16}	Sum of the length of overlap time windows
f_{17}	Standard deviation of the length of overlap time windows
f_{18}	Total demand/Vehicle capacity ratio
f_{19}	Vehicle capacity/Average demand ratio

Upon extracting the pertinent features for the Pickup and Delivery Problem with Time Windows (PDPTW), the subsequent step involves obtaining actual solutions for

the given problem instances, which serve as labeled data. Given the NP-hard nature of PDPTW, generating a substantial number of exact solutions for this purpose would present formidable computational challenges. In this study, we circumvent this hurdle by approximating the best solution using our tabu search algorithm, as expounded upon in the upcoming section. Experimental results corroborate the efficacy of our algorithm, revealing an average deviation of less than 5% when compared to the best-known solutions. This lends credence to the accuracy and precision of the labeled data generated through this approach.

The subsequent phase entails selecting an appropriate machine learning model amenable to request assignment optimization. In our endeavor, we have explored a broad spectrum of machine learning regression models, encompassing linear techniques such as Ordinary Least Squares (OLS), LASSO, and Ridge Regression, as well as nonlinear models, including Decision Trees and Random Forests. In summation, our aim is to identify prediction models that strike a balance between exceptional performance and interpretability. Detailed insights into our model selection process, guided by the aforementioned criteria, are presented in the numerical experiments outlined in Section 5.

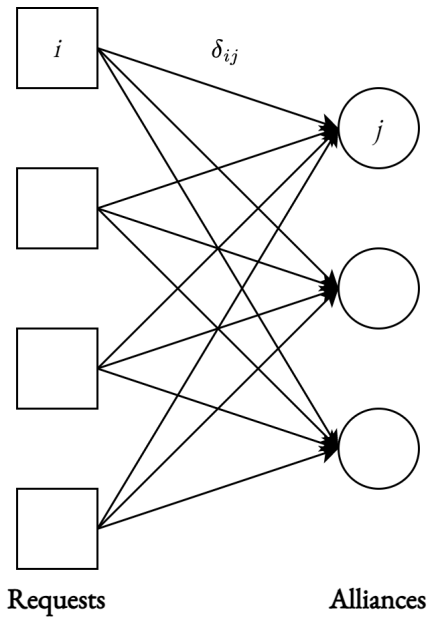


Figure 4.3: Illustration of the requests assignment.

Request assignment: It is important to note that in the context of the MAD-

PDPTW, each request must be exclusively assigned to one alliance. Since requests belong to LSPs that may participate in more than one alliances (e.g., LSP 1 in Figure 4.1), we need to assign each request to an alliance. As shown in Figure 4.3, assume there are $I = \{1, 2, ..., |I|\}$ requests to be assigned to the set of alliances $A = \{1, 2, ..., |A|\}$. Assume each request must be served and there is no limit on how many requests each alliance can have. The request assignment problem is to find a partition of the requests with the minimum total cost, which can be modelled as the set partitioning problem. Let the subset j of locations assigned to an alliance $a \in A$ be associated with an estimated delivery cost c_j, whose value can be predicted by our machine learning models. Since this predicted cost may contain errors, we handle the prediction uncertainly via adding an error term $\widetilde{e}$ which represents the prediction error. Consistent with the methodology employed in [69], we leverage the empirical distribution of $\widetilde{e}$ to generate scenarios within our Sample Average Approximation (SAA) framework.

$$\textbf{min} \quad \mathbb{E}_{\xi}\big[\sum_{j\in Z}(c_j + \widetilde{e}_j(\xi))v_j\big] \tag{4.14}$$

$$\textbf{s.t.} \quad \sum_{j\in Z}\delta_{ij}v_j = 1 \quad \forall i \in I \tag{4.15}$$

$$\sum_{j\in J} v_j = |A| \tag{4.16}$$

$$v_j \in \{0, 1\} \tag{4.17}$$

Z is the set of all possible partition of requests. Decision variable v_j equals to 1 if subset j is selected. δ_{ij} equals to 1 if request i belongs to subset j, and 0 otherwise. Here, we can convert the objective function into a mean value formulation $\frac{1}{|\xi|} \cdot \sum_{\xi}\sum_{j\in Z}(c_j + \widetilde{e}_j(\xi))v_j$. As stated in [25], the expected value (solution of the mean value problem) can provide a robust solution to original stochastic problem. Constraints (4.15) ensure that every request is assigned to an alliance and constraint (4.16) ensures the number of selected subsets equal to the number of alliances $|A|$. The problem involves an exponential

number of variables (columns) since the number of possible subsets grows exponentially in the number of requests waiting for assignment. And predict the cost c_j of all possible partition of request is also very time-consuming. Instead of enumerating all the possible partitions, we provide a simply greedy heuristic approach to solve the request assignment iteratively. We randomly rank requests sequence of unassigned requests, and assign one request to one alliance at each iteration. Here, c_{ia} denotes the cost for request i assigned to alliance a, which equals to the predict expected cost: $\widetilde{c}_j = \frac{1}{|\xi|} \cdot \sum_{\xi} \sum_{j \in Z} (c_j + \widetilde{e}_j(\xi)) v_j$. predicted by our machine learning models introduced in **cost prediction**. In this case, the problem becomes a simple facility location problem and we can simply assign each request i to the alliance a with lowest cost c_{ia}. Then the total cost equals to $\sum_{i \in I, j \in J} c_{ia}$.

4.3.2 Tabu Search Algorithm

In this subsection, we first develop an efficient tabu search algorithm to solve the PDPTW for each alliance. Furthermore, we make minor adjustments by including constraint (4.13) in the tabu search algorithm during Step 3 when performing insertion and removal operations. This ensures that candidate requests can only be inserted or removed from routes (denoted by k) that belong to the same alliances. We can use the adjusted tabu search algorithm to directly solve the MAD-PDPTW as a baseline method in our numerical experiments in Section 5.

Tabu search [46] is one of the well-known metaheuristics. It takes a potential solution and searches its neighborhood iteratively to find improved solutions. It has been successfully applied to various routing problems [113, 20]. In what follows, we introduce the full framework of our algorithm, including initial solution construction and the tabu search algorithm. The procedure to construct an initial solution s_0 is described here. We construct the initial solution s_0 where not all the constraints defined in PDPTW need to be satisfied. Given the request set R, pickup node set P, delivery node set D, and available vehicles K as inputs, for each vehicle $k \in K$, we iteratively select request c from the pickup set P and check whether it satisfies the earliest pickup time constraint $e_i \leq e_c \leq e_{i+1}$. If

yes, we add both the pickup node and delivery node of request c to vehicle k; otherwise, we assign it to a new vehicle $k+1$. When there are no requests in set P, we end up with the initial solution s_0.

Algorithm 1 Tabu search algorithm

Input: s_0, best solution $s^* = s_0$, tabu list $\mathcal{L} = \emptyset$
Output: Best solution s^*

procedure TABU
  Let current solution $s_c = s_0$
  while $i \leq I_{max}$ **do**
    Do insertion and removal operation
    Get the neighborhood solution N_s of s_c
    for $s_i \in N_s$ **do**
      Calculate fitness function $f(s_i)$
      if $s_i \notin \mathcal{L}$ **and** $f(s_i) \leq f(s_c)$ **then**
        $s_c = s_0$
      end if
    end for
    if $f(s_c) \leq f(s^*)$ **then**
      $s^* = s_c$
    end if
    if Size of $\mathcal{L} \geq L_{min}$ **then**
      Update $\mathcal{L}$
    end if
  end while
end procedure

Based on the initial solution found, the tabu search-based heuristic algorithm is described in Algorithm 1. Here, the termination condition is that the maximum number of iterations I_{max} is reached. The fitness function is described as $f(s) = C(s) + \alpha \cdot Q(s) + \beta \cdot T(s)$, where $C(s)$ is the value of the objective function (1), $Q(s)$ denotes the total weight that exceeds the vehicle capacity, and $T(s)$ represents the total units of time that violate the time windows constraint. The fitness function consists of two parts: the original objective function and the penalty cost. Parameters α and β are both positive penalty terms that make the solution s more likely to meet the capacity and time windows constraints, respectively. To achieve this, we introduce a new parameter θ with a small value (e.g., 0.1) as a step size to adjust the values of α and β. If either $Q(s)$ or $T(s)$ is not equal to 0, we multiply it by $(1+\theta)$ in the next iteration. Another important

component is the tabu list, which represents a set of solutions that have been visited in the recent past. Here, we define the maximum length of the tabu list as L_{max} and use it to memorize the insertion operations when we insert the pickup node i and delivery node $i+r$ into route k. To solve the MAD-PDPTW, we only need to add constraint (12) before performing insert and remove operations, ensuring that nodes i and $i+r$ can only be added to or removed from route $k \in K_a$ that belongs to the same alliance.

4.4 Numerical Experiments

This section presents the experimental setup for problem instance generation and delivery cost prediction. We also compare our learning and optimization framework against tabu search in solving the MAD-PDPTW. Computational experiments are conducted to validate the developed framework's performance for multiple alliances under different settings. All computational experiments are carried out on a desktop computer with an Intel Core i5 2.3 GHz processor and 16GB RAM. The tabu search algorithm is implemented in Java, while the machine learning models are coded in Python 3.7.

4.4.1 Problem Instance Generation

The dataset proposed by Li et al. [61] is a popular standard dataset in the study of PDPTW and is used to generate sampled PDPTW instances in our work. We need to construct two types of instances, which are synthesized from the PDPTW benchmark dataset. The first type is used as a training and testing dataset for delivery cost prediction, while the second type is prepared for running MAD-PDPTW. For the first type of instances, we randomly sample instances with a total number of requests ranging from 100 to 200. The labeled data for each PDPTW instance is computed using the tabu search algorithm described in Section 4.3. In total, we obtain 500 instances, of which 400 are randomly selected as the training set, and the remaining 100 serve as the test set.

To set up the multiple alliance structures, we construct a second type of instance by sampling from the original data and randomly reallocating the requests to LSPs and al-

Table 4.3: Instances generated from the PDPTW benchmark dataset.

Notation	Description
x	The x coordinate of the pickup/delivery locations
y	The y coordinate of the pickup/delivery locations
q_i	Demand of node i
e_i	Earliest pickup/delivery time of node i
l_i	Latest pickup/delivery time of node i
s_i	Service time of node i
p_i	Pickup (index to sibling) of node i
d_i	Delivery (index to sibling) of node i
L_i	LSP index of node i

liances. In comparison to the first type of instances, the second type includes an additional column that provides request ownership information. Table 4.3 gives a brief description of the second type of sampled instances. For detailed parameters and illustrations of the alliance structures for all second-type test instances, please refer to Table 4.4 and Figure 4.4.

Table 4.4: Detail parameters setting for all test instances.

No.	Structure	Alliance	LSPs	Requests	Size	Request Configuration
1	1	2	3	9	small	[3, 3, 3]
2	1	2	3	18	small	[6, 6, 6]
3	1	2	3	20	small	[7, 7, 6]
4	1	2	3	24	small	[8, 8, 8]
5	3	3	6	30	small	[5, 5, 5, 5, 5, 5]
6	1	2	3	65	medium	[30, 10, 25]
7	1	2	3	65	medium	[20, 15, 30]
8	1	2	3	65	medium	[25, 5, 35]
9	2	2	6	60	medium	[10, 10, 10, 10, 10, 10]
10	3	3	6	105	large	[30, 10, 20, 10, 20, 15]
11	3	3	6	105	large	[40, 5, 25, 10, 15, 10]
12	3	3	6	120	large	[30, 15, 30, 20, 15, 10]
13	4	4	8	135	large	[20, 10, 5, 30, 15, 10, 15, 30]
14	4	4	8	135	large	[15, 15, 5, 20, 25, 15, 20, 20]
15	4	4	8	135	large	[30, 5, 15, 25, 10, 20, 10, 20]
16	5	5	10	150	large	[20, 10, 20, 5, 15, 10, 15, 15, 20, 20]
17	5	5	10	180	large	[30, 15, 25, 20, 15, 30, 10, 10, 10, 15]
18	5	5	10	185	large	[25, 5, 25, 30, 10, 20, 15, 20, 10, 25]

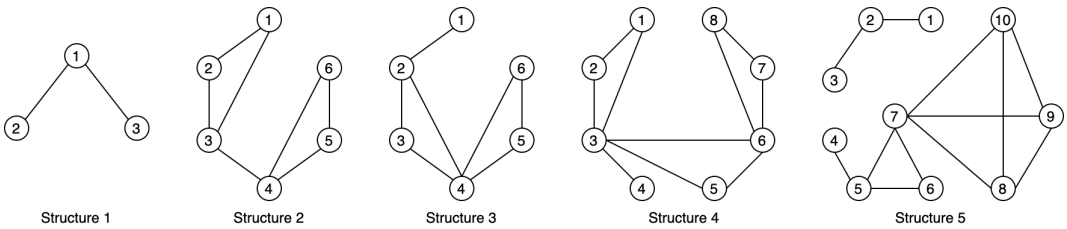


Figure 4.4: Alliances structure setting for the case study.

4.4.2 Prediction Model Selection

We have tested five different machine learning models: linear regression, LASSO regression, ridge regression, elastic net, decision trees, and random forest. To achieve the best performance, we implemented 5-fold cross-validation (5-CV) to select the best hyperparameters (e.g., coefficient value for the regularization term, maximum depth of the tree) for all models. All the training and validation procedures were carried out in Python 3.7. Table 4.5 summarizes the average cross-validation R^2 value and the mean absolute percentage value (MAPE). The MAPE is defined as: $\frac{1}{|S|}\sum_{t\in S}\frac{|l_s-\hat{l}s|}{ls}$, where l_s denotes the best solutions we obtained by tabu search, and $\hat{l}_s$ denotes the predicted delivery cost for a sample s in each fold S of the training set.

Based on the evaluation results, all the machine learning models mentioned above have demonstrated reasonably good performance in delivery cost prediction. Notably, the LASSO regression model stands out with the lowest test error and the highest R^2 score. Additionally, LASSO produces sparse coefficients, which reduces the number of features in the model and maintains good interpretability. Therefore, we have decided to adopt the LASSO regression model as the prediction model in our framework.

Table 4.5: Performance evaluation of the machine learning models.

Model	**5-CV R^2**	**5-CV MAPE**	**Test R^2**	**Test MAPE**
LR	**0.969**	**0.067**	0.904	0.140
LASSO	0.966	0.072	**0.972**	**0.066**
Ridge	0.967	0.071	0.953	0.095
Elastic Net	0.947	0.101	0.939	0.099
Decision Tree	0.937	0.089	0.961	0.085
Random Forest	0.965	0.068	0.966	0.069

4.4.3 Performance Comparison

In this subsection, we compare the results of delivery costs obtained by four different approaches: (1) self-routing by LSPs without collaboration, (2) collaborative routing with alliances solving by tabu search heuristic alone, (3) collaborative routing with alliances solving by proposed learning-based optimization framework, and (4) collaborative routing with fully collaboration, which means each LSP can cooperate with each other and exchange requests from both the computational and management perspectives. The experimental results for all instances are presented in Table 4.6 for small instances and Table 4.7 for medium and large instances.

Table 4.6: Experimental results for small size test instances.

No.	Gurobi	Tabu	Learning	Learning + Error
1	996	1026	996	996
2	1609	1609	1642	1621
3	1709	1715	1724	1715
4	1934	2035	2012	1987
5	2838	2972	2851	2851

Table 4.7: Experimental results for medium and large size test instances.

No.	$\mathcal{I}$	$\mathcal{F}$	$\mathcal{L}$	$\mathcal{A}_{min}$	$\mathcal{A}_{max}$	$\mathcal{S}_1$ (%)	$\mathcal{S}_{min}$ (%)	$\mathcal{S}_{max}$ (%)
6	5699	4165	4660	4915	5096	18.23	5.18	8.50
7	5263	3958	4706	4587	4784	10.58	-2.59	1.63
8	5067	3855	4804	4747	4887	5.19	-0.64	1.67
9	5659	3705	4550	4678	4785	18.45	2.81	5.16
10	8990	5685	7337	7805	7935	18.39	5.99	7.17
11	8499	5711	7342	7613	7763	13.61	3.79	5.65
12	12857	8955	11058	11286	11886	13.99	2.02	6.96
13	14978	9472	12856	13434	13628	14.17	4.30	5.66
14	14969	10187	12711	13205	13676	15.08	3.74	7.06
15	14416	10536	12302	13313	13700	14.66	**7.59**	10.20
16	17060	10572	14005	14986	15241	17.91	6.55	8.11
17	21029	11375	16802	17512	20297	**20.10**	4.05	**17.22**
18	19850	13645	16550	17166	17576	16.62	3.58	5.83

We have observed that our Tabu search method and learning method, both with and without error term estimation, for small-sized instances can obtain solutions with small

optimal gaps compared to Gurobi. As previously mentioned in Section 4.3.1, errors will always exist in **cost prediction**. To provide a more comprehensive evaluation of the benefits of our learning-based approach, we also investigated the influence of error cascades. In Table 4.6, the last column displays the results of the learning method that incorporates errors in request assignment. This indicates that the learning method, when considering prediction errors, yields more accurate results.

For medium and large-sized instances, Gurobi fails to provide feasible solutions within the 3600-second time limit, whereas both our tabu search and learning-based framework can find good solutions in less than 1 minute. Columns $\mathcal{I}$, $\mathcal{F}$ and $\mathcal{L}$ denote the delivery costs obtained by self-routing without collaboration, collaborative routing with fully collaboration (represents the upper bound of the cost savings via collaboration) and collaborative routing with alliances solving by our learning-based approach, respectively. Columns $\mathcal{A}_{min}$, $\mathcal{A}_{max}$ denotes the minimal and maximal delivery cost obtained for collaborative routing with alliances after run the tabu search alone 5 times. Columns $\mathcal{S}_1$ is the cost savings in percentage achieved by collaborative routing with alliance compare to self-routing.

As shown in Table 4.7, we observe that both collaboration with alliances and full collaboration consistently result in lower delivery costs compared to self-routing. Column $\mathcal{S}_{min}$ and $\mathcal{S}_{max}$ are the minimum and maximum savings that the learning framework can achieve compare to the direct use heuristic method (tabu search) alone. For small-sized instances (No.1 to No.5), our learning and optimization framework can obtain solutions comparable to tabu search. For moderate or larger test instances with denser alliance structures (No.6 to No.18), our learning framework outperforms the heuristic method (tabu search) alone by approximately 2% to 10% and can achieve cost savings of up to 17%. We also compare the running times of our proposed learning and optimization framework and directly using heuristic method (tabu search), as shown in Figure 4.5. It illustrates that our new approach requires fewer computing resources, especially in large-scale cases.

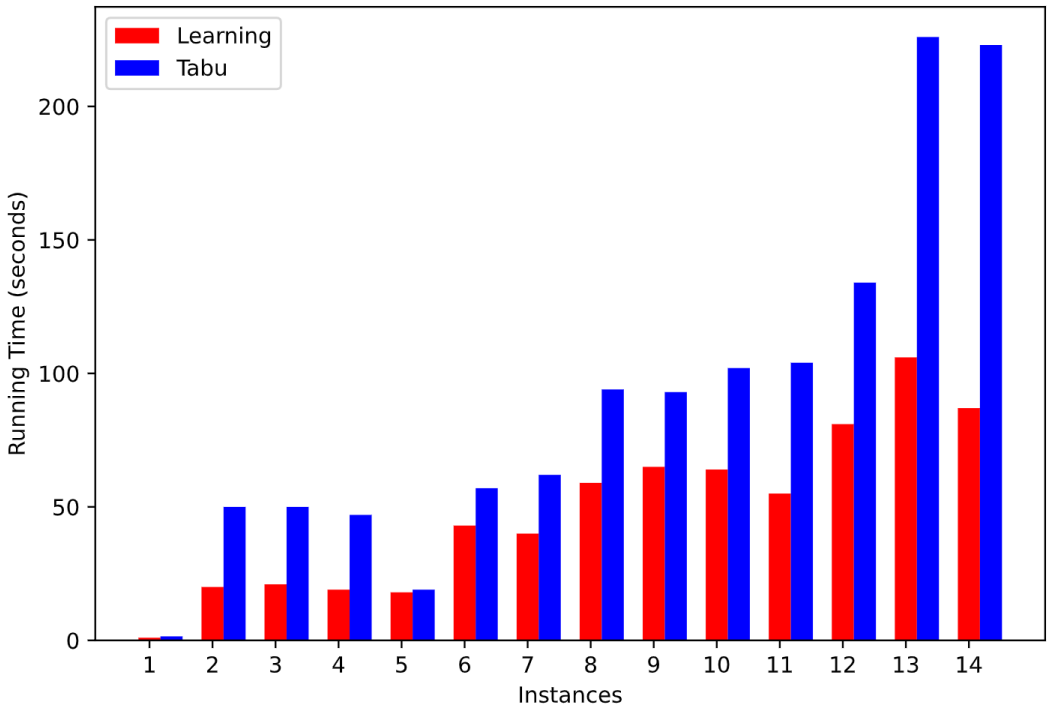


Figure 4.5: Comparison of learning-based methods and Tabu search methods in terms of algorithm running time.

4.5 Conclusion

This chapter addresses an emerging concept in collaborative urban delivery problems involving multiple alliance structures. Our experiments demonstrate that compared to individual optimal planning by LSPs themselves, centralized collaborative routing has the potential to reduce the total operating cost by approximately 20%. Furthermore, in comparison to centralized collaborative routing using a direct heuristic algorithm, our learning-based optimization approach can achieve cost reductions of up to 17%, with the added benefit of reduced computational time. Moreover, the learning-based approach provides a methodological framework, allowing for the replacement of tabu search with other heuristic methods to potentially enhance results. We have observed two key factors: (1) increased participation of LSPs in alliances generally leads to greater cost savings; (2) the alliance structure significantly impacts cost savings, with denser alliance structures resulting in more substantial savings. This suggests that overlapping alliance structures can promote more sustainable logistics practices. These cost savings can potentially be

translated into profit-sharing schemes among participating LSPs, thereby providing incentives for their involvement in such alliance structures. Profit-sharing mechanisms are a topic worthy of future research, although they fall outside the scope of this chapter. In the future, we also aim to provide a robust optimization model to handle errors in cost prediction during the first stage of our framework.

Chapter 5

Optimization of Customer Service and Driver Dispatch Areas for On-Demand Food Delivery

5.1 Motivation and Background

Advanced technologies such as smartphones and wireless communications are transforming transportation-enabled urban services in many ways at a rapid pace. The emergence and success of on-demand passenger and logistics service platforms is one of the most notable innovations. As one of the key innovations, on-demand food delivery platforms such as Uber Eats, DoorDash, Grab Food and Meituan that provide door-to-door food delivery services have achieved great success in past few years, especially accelerated by the COVID-19 pandemic. For example, Grab Food, Southeast Asia's largest food delivery service provider in 480 cities in 8 countries, has a GMV of $7.6 billion with a 29% annual increase in 2021. Meituan, the largest on-demand food delivery platform in China, serves more than 30 million orders daily and generates a profit of 4.71 billion rmb in 2020. According to a report by [84], the food delivery market has been doubled during the COVID-19 pandemic with a market value of over $150 billion in the US.

As shown in Figure 5.1, there are three parties other than the platform in the on-

demand food delivery market: customers, restaurants, and drivers. Customers choose and order food from a restaurant nearby listed on the platform. Once an order is placed, the platform notifies the restaurant to prepare food and broadcasts/dispatches the order and delivery task information to drivers waiting nearby. A driver then picks up food from the restaurant and delivers it to the customer.

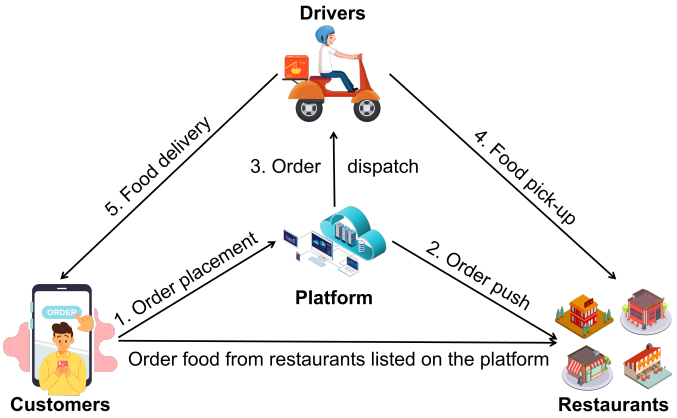


Figure 5.1: Illustration of the on-demand food delivery service.

In a typical food delivery service, demand and supply are both time-dependent, erratic, and uncertain. For example, using the real data from a crowd-sourcing food delivery platform, Figure 5.2 demonstrates the distributions of hourly orders from $09:00$ to $24:00$. We can see that the order volume increases greatly during the peak period at noon and evening, resulting in a lack of drivers and longer delivery time. Customers still have high expectations for the delivery time and may abandon the platform and seek for alternatives if the delivery time is too long. For such platforms, it is very challenging and requires great efforts to balance time-dependent supply and demand.

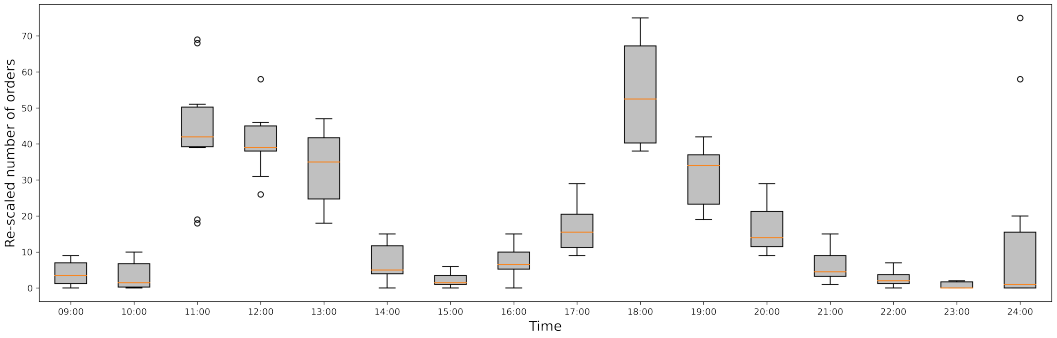


Figure 5.2: The distribution of (re-scaled) number of orders over hours.

To coordinate the balance between supply and demand in on-demand platforms, a common approach is dynamic pricing [115, 36, 141]. When a restaurant is busy with a large number of orders, the platform could raise the delivery fee to discourage customers from ordering from that restaurant and also encourage drivers to deliver orders for that restaurant. Another approach is to adjust the restaurant's customer service area [143, 120, 29]. If the number of orders is too high and the delivery time is too long (i.e., delivery supply is less than demand), the platform could decrease the restaurant's service area to reduce demand; in contrast, if the number of orders is too low and the delivery time is short, (i.e., delivery supply is more than demand), then the service area can be enlarged to serve more customers.

Both of the aforementioned approaches only concentrate on the demand side—the management of customer orders, and ignore the supply side (the drivers). In this chapter, we propose and focus on a new approach to balance supply and demand—by adjusting the customer service area and driver dispatch area simultaneously. Specifically, for each restaurant in a specified operating time horizon, the platform decides the (1) customer service area (CSA), i.e., the radius of the surrounding area within which customers can see the restaurant's information and order food from it; and (2) driver dispatch area (DDA), i.e., the radius of the surrounding area within which drivers can see the restaurant's information and deliver orders from it. Dynamic modifications to the driver dispatch area (DDA) of a restaurant hold significant business importance. Long pickup distances adversely impact both the order delivery time and driver utilization. In practical scenarios, the restaurant's order demand fluctuates dynamically, particularly during midday and evening peak hours. Ensuring timely delivery necessitates the potential expansion of the DDA, which could attract additional drivers to prevent delays and ensure timely deliveries. On the contrary, during off-peak hours, the adjustment of a restaurant's DDA allows for effective regulation of the driver count allocated to serve that establishment. Assigning orders to drivers within a limited DDA radius aids in controlling the distance covered by drivers. When a driver is assigned a delivery significantly distant from their current location, it may result in extended delivery times and heighten the prospect of order delays,

ultimately casting a negative effect on customer experience.

This consideration motivates us to consider an optimization problem to coordinate the supply and demand simultaneously, which we refer to in this study as the Restaurant Area Size Optimization (RASO) problem. We propose a data-driven optimization framework that determines the optimal radius of the customer service area and driver dispatch area. The objective is to maximize the total number of orders served while ensuring a service level requirement on order delivery time.

Finding the optimal radii of CSA and DDA is challenging. As shown in Figure 5.3, while the radius of CSA affects the number of orders (i.e., demand side) and the radius of DDA affects the number of potential delivery drivers (i.e., supply side) for each restaurant, the interactions of demand and supply will collectively affect the order delivery time in a complicated way, owing to the large complexity in dynamic order dispatching and driver routings. To address these challenges, we first study the relationship between the radius of customer service area and the number of orders for each restaurant. Second, we examine the number of drivers, which depends on the radius of the driver dispatch area. Third and more importantly, we explore how various factors related to demand, supply, and others, affect the order delivery time. Moreover, in practical scenarios, due to the flexibility of drivers in delivering orders from various restaurants, those operating within the driver dispatch area (DDA) of one restaurant may not exclusively serve that restaurant, especially if they are also within the DDA area for other restaurants. This introduces additional complexity to the problem. In the context of this research, our focus is on treating restaurants within a geographical neighborhood as a cluster, each with a common size for both CSA and DDA.

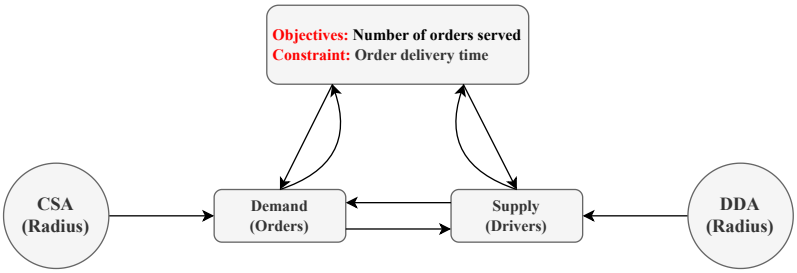


Figure 5.3: RASO problem in on-demand food delivery service.

In summary, the main contributions of this chapter are summarized as follows:

- We propose and solve an innovative operational problem, i.e., the Restaurant Area Size Optimization problem (RASO) for on-demand food delivery services, which determines the optimal radii of CSA and DDA for restaurants simultaneously to balance supply and demand, with the objective of maximizing the number of orders served.

- Specifically, we propose a data-driven optimization framework to solve the RASO problem. On the demand side, a discrete choice model is developed to characterize the relationship between customer order behavior and customer-restaurant distance, and then estimate the number of orders for the restaurants. On the supply side, several machine learning models are proposed to predict the order delivery time with varying sizes of CSA and DDA.

- We following the integrated predictive-and-prescriptive modeling framework introduced in Chapter 1, integrate a model tree prediction of order delivery time and formulate the RASO problem as a Mixed Integer Quadratically Constrained Program (MIQCP), which can be solved efficiently.

- We perform a set of extensive numerical experiments using a real-world dataset. The computational study demonstrates that the proposed framework can significantly improve the number of orders served and outperforms benchmark methods.

The reminder of the chapter is organized as follows. Section 5.2 describes the research problem. Section 5.3 presents the data-driven framework for joint optimization of CSA and DDA. Section 5.4 presents the real data, simulator, and the performance of order delivery time prediction. Section 5.5 presents experimental results of the proposed model and other benchmark methods. Finally, we conclude and discuss future research in Section 5.6.

5.2 Problem Description

In this section, we present a formal description of the RASO problem for on-demand food delivery service, then present a simple example for illustration.

5.2.1 Problem Statement

In the context of food delivery services, each restaurant is associated with a customer service area (CSA), denoted as A_c, which represents the geographical region within which customers can view the restaurant's information and place food orders. Each restaurant is also associated with a driver dispatch area (DDA), denoted as A_d, wherein drivers can access the restaurant's information and fulfill delivery orders. Both the CSA and DDA are determined by the platform and can be adjusted dynamically. To begin, we consider a food delivery system consisting of a cluster of restaurants $\mathcal{R}$, wherein a fleet of capacitated and homogeneous drivers $\mathcal{D} = \{d_1, d_2, ..., d_m, ...\}$ delivers a set of orders $\mathcal{O} = \{o_1, o_2, ..., o_n, ...\}$ that arrive starting from the initiation of the operating horizon $\mathcal{T}$.

Customer Orders: Each order $o \in \mathcal{O}$ can be described as a tuple (o^+, o^-, L^o), where o^+ denotes the pickup location (restaurant), o^- represents the delivery location (customer), and L^o indicates the promised delivery time (all orders have a uniform guaranteed delivery time, e.g., 45 minutes). We assume that the order preparation time varies based on the restaurant and follows a Gamma distribution, a topic further discussed in Section 5.4.2. Customers have the option to cancel their orders if they fail to secure a driver within a specified time frame.

Drivers: The system comprises a total number of $\mathcal{D}$ homogeneous drivers. These drivers are strategically positioned around the restaurants, and each individual driver $d \in \mathcal{D}$ possesses a service capacity denoted as p (representing the maximum number of orders they can carry).

Operating Horizon, CSA and DDA: The initiation of the operating horizon $\mathcal{T}$ is denoted by $t = 1$, and its conclusion, signifying the end of the operating, is indicated

as $t = |T|$. At the commencement of each time period $t \in \mathcal{T}$, the platform is required to make three decisions: (1) determining the radius of the CSA, denoted as ρ_c, which directly influences the demand for each restaurant; (2) specifying the radius of the DDA, denoted as ρ_d, which significantly impacts the delivery supply for each restaurant; and (3) establishing the method for dispatching orders to available drivers for delivery. For each time period $t \in \mathcal{T}$, the platform has the option to establish the radii ρ_c and ρ_d and subsequently perform specific multiple order dispatching in batches.

The entire decision-making process in the Restaurant Area Size Optimization (RASO) can be summarized as follows: At the initiation of each RASO decision time t, the platform first determines the radius ρ_c for the CSA, represented as A_c, and the radius ρ_d for the DDA, denoted as A_d. During time period t, at each order dispatching decision time, the platform receives $\mathcal{O}_t$ orders within A_c for each restaurant and observes a set of drivers $\mathcal{D}_t$ within A_d for each restaurant. Subsequently, the platform makes decisions regarding the dispatch of orders to drivers. These dispatched decisions lead each driver $d \in \mathcal{D}$ to implement a delivery route RT_d, defined as a sequence of visiting locations, based on their ongoing carrying orders Ω_d and newly dispatched orders O_d. In the event that an order $o \in \mathcal{O}$ cannot find an available driver, it will be held until the next dispatching time unless it is cancelled by the customer. These decisions are made iteratively until the end time $|T|$ of the operating horizon $\mathcal{T}$.

5.2.2 Example

The impact of the radius ρ_c of the CSA on demand, exemplified by the number of orders received by a restaurant, and the influence of the radius ρ_d of the DDA on supply, demonstrated by the number of available drivers for the restaurant, is illustrated in Figure 5.4. In Figure 5.4 (a), when the CSA radius is ρ_c, only 2 orders are expected to arrive within the CSA (colored in grey), while 6 drivers are available within the DDA (colored in yellow). The estimated demand is considerably lower than the available service capacity, suggesting a potential benefit in increasing the CSA radius to attract more orders for that

restaurant, for instance, expanding it to ρ'_c to accommodate 9 expected orders. In Figure 5.4 (b), when the DDA radius is ρ_d, 9 orders are expected within the CSA, but only 1 driver is available within the DDA. The service capacity is significantly lower than the estimated demand, indicating the platform's advantage in increasing the DDA radius to attract more available drivers for that restaurant, for example, extending it to ρ'_d to include 6 drivers.

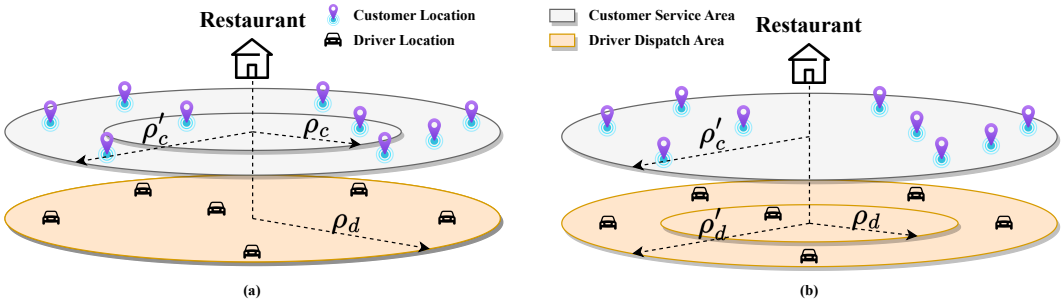


Figure 5.4: An example of a restaurant's (a) customer service area and (b) driver dispatch area.

5.3 Solution Method

This section presents the optimization framework for the RASO problem. We begin by introducing the main steps and the master optimization model in Section 5.3.1. Subsequently, we delve into customer order estimation on the demand side in Section 5.3.2, order dispatching algorithms, and service operation in Section 5.3.3, and order delivery time prediction in Section 5.3.4. Lastly, we present a specific Mixed-integer Quadratically Constrained Programming (MIQCP) model in Section 5.3.5, incorporating a model tree prediction for order delivery time. The notation is summarized in Table 5.1.

5.3.1 Framework and Optimization Model

We present a data-driven optimization framework encompassing customer demand estimation, order dispatching, order delivery time prediction, and optimization of customer service and driver dispatch areas. The framework is depicted in Figure 5.5.

Table 5.1: Notation for RASO problem.

Sets:	
$\mathcal{O}$	Set of orders
$\mathcal{D}$	Set of drivers
$\mathcal{T}$	Planning horizon; $\mathcal{T} = \{1, 2, ..., \lvert T \rvert\}$
Input parameters:	
L^o	Promised delivery time for order o
$\epsilon_{\max}$	A threshold predetermined as upper bound for average order delay
$\rho_c^{\min}, \rho_c^{\max}$	Minimum and maximum allowed radius of restaurant's customer service area (CSA)
$\rho_d^{\min}, \rho_d^{\max}$	Minimum and maximum allowed radius of restaurant's driver dispath area (DDA)
Decision variables:	
ρ_c	Radius of customer service area (CSA) for a restaurant
ρ_d	Radius of driver dispatch area (DDA) for a restaurant
Intermediate variables:	
$O(\rho_c)$	Function, the output is the number of prospective orders
$O(\rho_c, \rho_d)$	Function, the output is the total number of orders served
$L(\rho_c, \rho_d)$	Function, the output is the average delivery time for orders served

First, delivery time prediction is depicted on the right. Based on real-world data concerning historical orders and deliveries, the key steps for delivery time prediction are as follows: (1) generating instances with various CSA and DDA radii, (2) extracting relevant features from instances to predict delivery times, (3) simulating order deliveries using a predefined dispatch algorithm to obtain order delivery times as label data, and (4) training supervised machine learning models for delivery time prediction.

Next, customer order estimation is presented on the left. Initially, we calculate the distance between customers and restaurants, using it as an input feature to develop a customer choice model. This model allows us to investigate the relationship between customer order behavior and the distance to the restaurant. Subsequently, we compute the probability of a customer placing an order and estimate the total number of prospective orders for each restaurant.

The central illustration of the main optimization framework follows. At the beginning of time period t, we initiate the process with the collection of essential information, such as the locations of customers and drivers. Subsequently, we utilize our proposed data-driven approaches to develop two models: one for estimating customer order de-

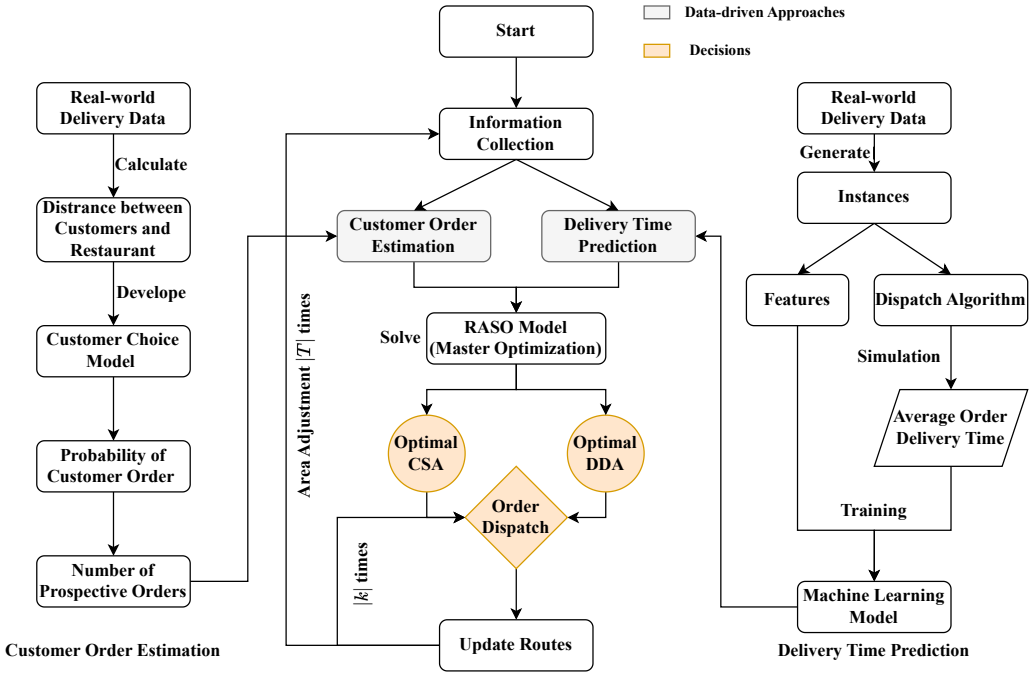


Figure 5.5: Optimization framework of restaurant area size optimization problem.

mand and the other for predicting delivery times. Through the integration of the estimation and prediction models with our proposed optimization model, we can effectively solve the RASO problem. Consequently, we can determine the optimal radii of the CSA and DDA for restaurants, thereby achieving a balance between supply and demand. The dispatch decisions result in updated delivery routes for each driver, with each driver having their designated route.

Specifically, we propose a master optimization formulation for the RASO problem. To enhance manageability, the planning horizon is segmented into shorter periods (e.g., 45 minutes), rather than planning for the entire duration at once. As each period concludes and time progresses, we 'roll' the planning window forward. At the onset of each new period, and prior to addressing the RASO problem, it is crucial to consider the most recent data, such as drivers' current locations, existing delivery routes, and the estimated incoming orders. This approach ensures access to accurate and current information, leading to more informed decision-making in subsequent segments. To facilitate this rolling horizon framework works well in practice, we have designed a simulator to simulate the order dispatching and driver routing process using real industrial data. Our formulation

focuses on a single time period, and it can be easily extended to multiple time periods using the proposed rolling horizon methodology. The primary goal from the platform's perspective, as expressed in objective function (5.1), is to maximize the number of served orders while guaranteeing a satisfactory level of order delivery time. More precisely, the platform aims to ensure that the order delivery delay for each restaurant in any time period $t \in T$ remains below a predetermined threshold ϵ_{max} (which could be 0 by default), as stipulated in constraint (5.2). The selection of the CSA and DDA radii occurs within practically permissible ranges, as indicated by constraints (5.3)-(5.5).

$$\textbf{[RASO]} \quad \max_{\rho_c, \rho_d} \quad O(\rho_c, \rho_d), \tag{5.1}$$

$$\textbf{s.t.} \quad L(\rho_c, \rho_d) - \frac{1}{|\mathcal{O}|} \sum_{o=1}^{|\mathcal{O}|} L^o \leq \epsilon_{\max}, \tag{5.2}$$

$$\rho_c^{\min} \leq \rho_c \leq \rho_c^{\max}, \tag{5.3}$$

$$\rho_d^{\min} \leq \rho_d \leq \rho_d^{\max}, \tag{5.4}$$

$$\rho_c, \rho_d \in \mathbb{R}^+. \tag{5.5}$$

In each time period $t \in \mathcal{T}$, the decision variables ρ_c and ρ_d represent the radii of the restaurant's CSA and DDA, respectively. The function $O(\rho_c, \rho_d)$ quantifies the total number of successfully delivered orders from restaurant r (the index r is omitted when clear) and represents the count of served orders. The function $L(\rho_c, \rho_d)$ calculates the average delivery time for the served orders. The primary challenges in solving the RASO problem are twofold: firstly, achieving accurate approximation of the objective function $O(\rho_c, \rho_d)$ through a closed-form formula, and secondly, accurately approximating the delivery time $L(\rho_c, \rho_d)$ and effectively integrating it into the optimization formulation. In reality, both $O(\rho_c, \rho_d)$ and $L(\rho_c, \rho_d)$ are challenging to estimate or express mathematically, as they depend on the complex interplay between supply and demand, and are undoubtedly influenced by decisions ρ_c and ρ_d simultaneously. To address these challenges, we introduce a data-driven approach to approximate function $O(\rho_c, \rho_d)$ based on real-world data. Additionally, we utilize machine learning models to derive the opti-

mal representation of function $L(\rho_c, \rho_d)$. The details of our approach are elaborated in the subsequent subsections.

5.3.2 Customer Order Demand Estimation

We begin by discussing the estimation of customer order demand. Customers are notably sensitive to the restaurant's proximity on the demand side. Longer distances from the restaurant tend to lead to extended delivery times, thereby reducing customers' willingness to place orders. The objective is to understand the relationship between the number of prospective orders $O(\rho_c)$ and the radius of CSA ρ_c while taking into account customers' order behavior. In particular, we employ a multinomial logit (MNL) model to investigate how the distance between a customer's location and the restaurant influences their order behavior.

Customer choice set: From a customer's perspective, they have three choices: to order from restaurant r, to order from any other restaurant, or not to order on the platform at all. To represent this customer choice, we introduce a binary decision variable $z_{i,r}$, which signifies the order behavior for each restaurant:

$$z_{i,r} = \begin{cases} 1, & \text{if customer } i \text{ place an order at restaurant } r, \\ 0, & \text{if customer } i \text{ does not place an order at restaurant } r. \end{cases} \tag{5.6}$$

The variable $U_{i,r}$ denotes the utility experienced by customer i when placing an order at restaurant r. In real-world scenarios, $U_{i,r}$ relies on several factors, including the restaurant's cuisine style, meal price, delivery fee, rating, customer preferences, and the distance between the customer and the restaurant. Here, we assumes the following expression for $U_{i,r}$:

$$U_{i,r} = V_{i,r} + \boldsymbol{\beta}_r \cdot \mathbf{x}_{i,r} + \epsilon_{i,r}, \tag{5.7}$$

where $\mathbf{x}_{i,r} = (x_{i,r,1}, x_{i,r,2}, ..., x_{i,r,|M|})$ represents the factors that relative to the cus-

tomer restaurant selection, $\boldsymbol{\beta}_r = (\beta_{r1}, \beta_{r2}, ..., \beta_{r,|M|})^T$ are the restaurant choice parameters, the dimension is $|M|$. The value $V_{i,r}$ is a constant representing the utility when the distance $x_{i,r} = 0$, while $\epsilon_{i,r}$ captures numerous unknown or unobservable factors treated as random variables following the Gumbel distribution. Based on the latest industry study conducted by [89], 77% of surveyed customers identified delivery time as the most crucial factor when selecting an online restaurant for delivery. The primary objective of this section is to examine how a restaurant's CSA (radius) affects the volume of restaurant orders (demand). Generally, the distance between a customer and a restaurant plays a significant role in food delivery time and subsequently influences customer preferences for restaurant selection. To simplify the representation, we substitute the vector of factors $\boldsymbol{x}_{i,r}$ and $\boldsymbol{\beta}_r$ with variable $x_{i,r}$ representing the distance between the restaurant and the customer, along with a corresponding constant β_r, and a constant variable $v_{i,r}$ representing the location utility [1], to model this influence. Intuitively, the farther the customer is from the restaurant and the longer the delivery time, the less inclined the customer will be to place an order (i.e., $\beta_r < 0$).

Then, the probability that customer i will place an order from restaurant r is

$$\begin{aligned}
\Pr(z_{i,r} = 1) &= \Pr(U_{i,r} > U_{i,r'}, \forall r' \neq r) \\
&= \Pr(V_{i,r} + \beta_r \cdot x_{i,r} + v_{i,r} + \epsilon_{i,r} > V_{i,r'} + \beta_r \cdot x_{i,r'} + v_{i,r'} + \epsilon_{i,r'}) \\
&= \Pr(\epsilon_{i,r'} < \beta_r \cdot (x_{i,r} - x_{i,r'}) + V_{i,r} + v_{i,r} + \epsilon_{i,r} - V_{i,r'} - v_{i,r'}) \\
&= \frac{e^{\beta_r \cdot x_{i,r} + V_{i,r} + v_{i,r}}}{e^{\beta_r \cdot x_{i,r} + V_{i,r} + v_{i,r}} + \sum_{r'} e^{v_{i,r'} + V_{i,r'} + \beta_r \cdot x_{i,r'}}}.
\end{aligned} \tag{5.8}$$

The food delivery platform offers a vast number of restaurants, providing customers with a wide range of choices. Additionally, the utility $U_{i,r}$ is influenced by numerous unobservable variables, making it challenging to precisely determine its value. In such cases where explicitly computing the denominator for each individual and alternative combi-

[1] The city of Singapore is divided into five districts. In our analysis, we make the assumption that restaurants within the same district share the same location utility $v_{i,r}$.

nation is infeasible due to the large number of alternatives ([119]), a large constant value can be employed to approximate the denominator. Consequently, we assume a reduced form of the probability as follows:

$$\Pr(z_{i,r} = 1) \approx \alpha_r \cdot e^{\beta_r \cdot x_{i,r}}, \tag{5.9}$$

where α_r and β_r are restaurant-specific parameters. In this context, the probability is considered in its reduced form when other factors, such as price and ratings, remain unobservable. With access to additional data, we could incorporate a greater number of variables into both the utility function and the probability function. Assuming that the number of customers per unit area (e.g., km^2) is n_r, the number of orders from customers at a distance x from restaurant r can be expressed as follows:

$$O_r(x) = 2\pi n_r \alpha_r e^{\beta_r x} x. \tag{5.10}$$

Finally, defining $\alpha'_r = 2\pi n_r \alpha_r$, the number of prospective orders for restaurant r can receive within its CSA (a certain with radius ρ_c) can be given as:

$$\begin{aligned} O_r(\rho_c) &= \int P(z_{i,r} = 1) \cdot n_r \cdot 2\pi x \mathrm{d}x \\ &= \int 2\pi n_r \alpha_r \cdot e^{\beta_r x} x \mathrm{d}x \\ &= \int \alpha'_r \cdot e^{\beta_r x} x \mathrm{d}x \end{aligned} \tag{5.11}$$

5.3.3 Order Dispatching and Service Operation

While this study does not specifically focus on order-dispatching techniques, it acknowledges the significance of dispatching decisions throughout the entire service process. This subsection provides a brief description of the order-dispatching algorithm. The notation for the order-dispatching algorithm is provided in Table 5.2.

The following dispatching rules and assumptions are applied:

- **Dispatch fairness**: If the number of orders carried by driver d is large/small (i.e., Ω_d is large/small), s/he will be less/more likely dispatched new orders;

Algorithm 2 Order Dispatching Algorithm

Input: $\mathcal{O}, \mathcal{D}, \mathcal{R}, \theta_d, \Omega_d, \rho_d, p$
Output: RT'_d

procedure DISPATCH($\mathcal{O}, \mathcal{D}$)
  for $d \in \mathcal{D}$ **do**
    if $\Omega_d \neq \emptyset$ **then**
      Get driver d current location l_d and delivery location o^- for order $o \in \Omega_d$
      Re-schedule the planned route RT_d by solving a TSP problem with unvisited locations for carried orders Ω_d
      Update the delivery route RT_d
    end if
  end for
  Initialize dispatched and being picked up orders $\mathcal{O}_{\text{pick}} = \emptyset$
  for $d \in \mathcal{D}$ **do**
    if d is going to pick up order $o \in \mathcal{O}$ **then**
      Update $\mathcal{O}_{\text{pick}} \leftarrow \mathcal{O}_{\text{pick}} \cup o$
    end if
  end for
  for $o \in \mathcal{O}$ **do**
    if $o \in \mathcal{O}_{\text{pick}}$ **then**
      Continue
    end if
    Initialize the available drivers set $\mathcal{D}_{\text{avail}} = \emptyset$
    for $d \in \mathcal{D}$ **do**
      Calculate the distance $c_{o,d}$ between order o pickup location o^+ and driver d current location l_d
      Calculate the remaining capacity p_d^{remain} for driver d
      if $c_{o,d} \leq \rho_d$ and $p_d^{\text{remain}} \geq 1$ **then**
        $\mathcal{D}_{\text{avail}} \leftarrow \mathcal{D}_{\text{avail}} \cup d$
      end if
    end for
    Assign order o to driver $d \in \mathcal{D}_{\text{avail}}$ with maximum p_d^{remain}
    do greedy insertion
    insert pickup node o^+ and delivery node o^- to route RT_d at positions with minimum distance increased
    update $\mathsf{RT}_d \leftarrow \mathsf{RT}_d \cup (o^+, o^-)$
  end for
  return new delivery plan route RT'_d for $d \in \mathcal{D}$
end procedure

Table 5.2: Notation for order dispatch.

Sets and Variables	
$\mathcal{R}$	Set of restaurants
l_d	Current location of driver d
Ω_d	Ongoing carrying orders by driver d
p_d^{remain}	Remaining capacity for driver d
$c_{o,d}$	Distance between pickup location o^+ and driver's current location l_d
RT_d	Planned delivery route for driver d
$\mathcal{O}_{\text{pick}}$	Orders that have been dispatched and are being picked up by the drivers
$\mathcal{D}_{\text{avail}}$	Drivers with available capacity and in the restaurant's DDA

- **Order re-dispatch**: Orders that have not been picked up within a specified time frame (e.g., 10 minutes) but have been assigned a driver can be re-dispatched to a new driver;

- **Driver routing behavior**: Following order dispatch, drivers deliver their carried orders by solving an open Traveling Salesman Problem (TSP) to optimize their route;

- **Order cancellation**: If a newly generated order cannot find an available driver within a certain time (e.g., 15 minutes), it is assumed that the customer becomes impatient and cancels the order.

The first rule is aimed at ensuring fairness in dispatching by preventing certain drivers from being overloaded with a disproportionately large number of orders, while others only receive a few. The second rule serves to enhance the flexibility of the dispatch algorithm by allowing for order re-dispatch, which has the potential to reduce order pickup time. Third, we assume that drivers always follow the shortest one-way delivery travel route from the restaurant to one of the customer locations. This route is derived by solving an open TSP, where the driver does not return to the starting location. [2] Lastly, customers have the option to cancel orders if the platform fails to find a driver within a specific time period.

[2] It is important to note that in real-world scenarios, due to complexities such as left turns in intersections, the actual travel distance for order delivery tends to be equal to or greater than the travel distance derived from the open TSP solution.

To summarize, we present the rules for order dispatching and service operations using the pseudo-codes in Algorithm 2. Specifically, Lines $2-6$ reconstruct each driver's delivery route based on the orders they are carrying. Lines $8-10$ list the orders that have already been dispatched and are currently being picked up by the designated driver. Any remaining orders are dispatched to drivers in the DDA with the largest available capacity, as demonstrated in Lines $11-20$. Additionally, Lines $21-23$ implement a greedy insertion approach to incorporate new orders into the driver's current route with minimal increase in distance. Notably, the value of DDA radius ρ_d significantly impacts the number of available drivers on the supply side, subsequently affecting order delivery times, as evident from Lines 18 and 19 in Algorithm 2. An increase in the DDA radius ρ_d while keeping the restaurant's CSA unchanged results in more available drivers on the supply side, potentially reducing order delivery times. However, a larger DDA may also lead to longer pickup travel distances for orders, potentially increasing the overall delivery time. The estimation of order delivery time will be discussed in the subsequent subsection.

5.3.4 Delivery time prediction

This subsection focuses on estimating the order delivery time. Specifically, given the radii ρ_c and ρ_d of the CSA and DDA for a restaurant, our objective is to derive a mathematical representation of $L(\rho_c, \rho_d)$ that effectively approximates the average order delivery time for served orders. To achieve this, we adopt a data-driven approach utilizing a customized simulator and dataset (as detailed in Section 5.4). By employing various supervised machine learning models, we aim to obtain closed-form formulas for order delivery time, which can then be integrated into the RASO formulation introduced in Section 5.3.1.

Feature generation

For each restaurant r, Table 5.3 presents the features that are considered for predicting the average order delivery time.

For ensuring both interpretability and accuracy of the prediction model, we propose five simple features. The CSA radius ($\rho_{c,r}$) and DDA radius ($\rho_{d,r}$) for each restaurant

Table 5.3: Features in average delivery time prediction.

Features	Definitions
$\rho_{c,r}$	Radius of CSA of restaurant r
$\rho_{d,r}$	Radius of DDA of restaurant r
$D_r(\rho_{d,r})$	Number of available drivers in DDA of restaurant r
$O_r(\rho_{c,r})$	Number of prospective orders in CSA of restaurant r

r are considered. $D_r(\rho_{d,r})$ represents the platform's supply, where a larger DDA allows more available drivers for order dispatch, potentially reducing the order delivery time. On the other hand, $O_r(\rho_{c,r})$ represents the platform's demand, where a larger CSA allows more customers to place orders from the restaurant, which may increase the order delivery times.

5.3.5 A Mixed-Integer Quadratically Constrained Programming Model

After conducting feature engineering and building prediction models, the subsequent step involves selecting an appropriate prediction model that aligns with the proposed RASO model. In their study, [69] explored various machine learning models, encompassing both linear and nonlinear (tree-based) models, for delivery time predictions. In this section, we comprehensively evaluate several regression models, including ordinary least squares (OLS), ridge regression (Ridge), linear support vector regression (SVR), model tree ([130]), classification and regression trees ([74]), and XGBoost. The selection process and experimental details are presented in the following section, specifically in Section 4.4.2.

Model Tree

In this subsection, we show how the RASO model proposed in Section 5.3.1 can embed a trained machine learning model for delivery time prediction. We explore both linear and nonlinear (tree-based) models and select one that exhibits both good prediction performance and optimization compatibility. Specifically, based on our numerical experiments

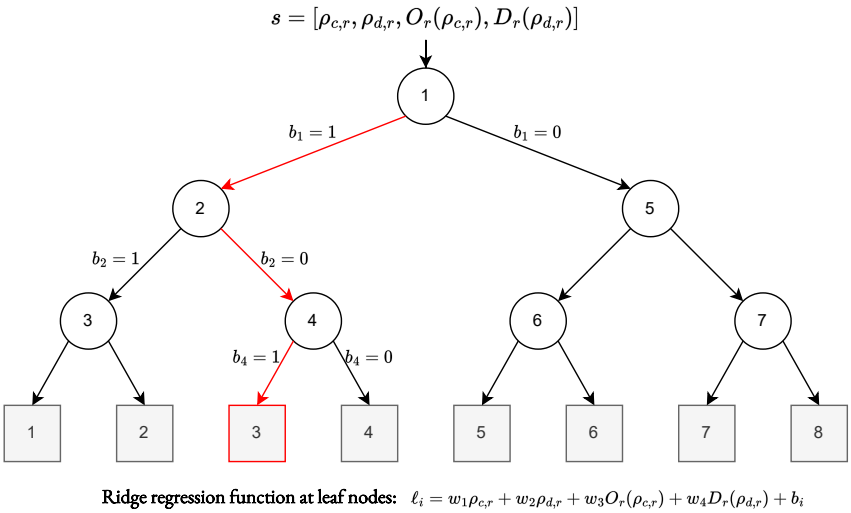


Figure 5.6: A model tree example with ridge regression functions in the leaf nodes.

with prediction models, presented in detail in Section 5.4, we choose the model tree as the prediction model. Unlike linear models, which assume linear functional relationships between ground truth labels and features (e.g., ordinary least squares and ridge regression), and some tree-based models that predict feature vector values through data feature splitting based on decision rules (e.g., classification and regression trees), the model tree combines linear regression models (e.g., ridge regression) for data feature splitting in a tree structure. Unlike classification and regression trees, which usually predict the mean or median value of labels in each leaf node, the model tree can fit any regression models in the leaf nodes. Formally, the model tree offers good interpretability and can examine both linear and nonlinear relationships in the data, making it a suitable choice with a limited amount of training data [101, 39, 40, 96].

Linearization of Model Tree

Next, we demonstrate the integration of the model tree into the RASO formulation, resulting in a mixed-integer quadratically constrained programming (MIQCP) model. The updated model is efficiently solvable using commercial optimization solvers like Gurobi and CPLEX within a reasonable timeframe.

Figure 5.6 depicts a model tree with ridge regression functions assigned to its leaves.

The tree's depth is 3, containing 7 branch nodes and 8 leaf nodes. The feature vector $\boldsymbol{s}$ characterizes the status based on decision variables. At each branch node, a decision rule based on the split feature's value is applied. For instance, in node 1 (the root), if the rule is true, the feature vector $\boldsymbol{s}$ is routed to the left; otherwise, it is routed to the right. Each leaf node corresponds to a ridge regression function ℓ_j, where j represents the index of leaf nodes. In this example, the feature vector $\boldsymbol{s}$ reaches leaf node 3 as the final destination.

Generally, a model tree is defined by (1) its tree structure, (2) decision rules at each branch node, and (3) linear regression functions assigned to each leaf node. We can represent such a model tree as a mixed-integer linear program (MILP). In the following sections, we introduce additional notation, which can be found in Table 5.4.

Table 5.4: Notation for the MIQCP.

Sets:	
$\mathcal{B}$:	branch nodes set;
$\mathcal{F}$:	leaf nodes set;
$\mathcal{B}_j^{\text{left}}(\boldsymbol{s})$:	for $\boldsymbol{s}$, set of branch nodes where the **left branch** following leaf node j;
$\mathcal{B}_j^{\text{right}}(\boldsymbol{s})$:	for $\boldsymbol{s}$, set of branch nodes where the **right branch** following leaf node j;
Input parameters:	
μ:	a small constant value;
M:	a large constant value;
$\boldsymbol{s}$:	a feature vector characterize the status based on decision variables;
ℓ_j:	linear function at leaf node $j, \forall j \in \mathcal{B}$;
$\boldsymbol{s}_k$:	value of feature k in vector $\boldsymbol{s}$;
v_{ik}:	value of feature k if it is selected for splitting at the branch node i;
Decision variables:	
$b_i(\boldsymbol{s})$:	binary variable, 1 if $\boldsymbol{s}$ branches left at node $i, \forall i \in \mathcal{B}$;
$c_j(\boldsymbol{s})$:	binary variable, 1 if $\boldsymbol{s}$ is located to the leaf node $j, \forall j \in \mathcal{B}$;
$v_j(\boldsymbol{s})$:	prediction value of $\boldsymbol{s}$ at leaf node $j, \forall j \in \mathcal{B}$;

Following the decision rule at branch node $i \in \mathcal{B}$, we introduce the following set of constraints relevant to the delivery time predictions:

$$\boldsymbol{s}_k \geq v_{ik} + \mu - M b_i(\boldsymbol{s}), \tag{5.12}$$

$$\boldsymbol{s}_k \leq v_{ik} + M(1 - b_i(\boldsymbol{s})), \tag{5.13}$$

$$b_i(\boldsymbol{s}) \in \{0, 1\}, \tag{5.14}$$

where $b_i(\boldsymbol{s}) = 1$ indicates $\boldsymbol{s}_k$ less than v_{ik}, feature vector $\boldsymbol{s}$ will be routed to the left branch, and $b_i(\boldsymbol{s}) = 0$ indicates $\boldsymbol{s}_k$ greater than v_{ik} and will be routed to the right branch. To locate the leaf node j where feature vector $\boldsymbol{s}$ belongs, we introduce two sets, $\mathcal{B}_j^{\text{left}}(\boldsymbol{s})$ and $\mathcal{B}_j^{\text{right}}(\boldsymbol{s})$, and we have:

$$\mathcal{B} = \mathcal{B}_j^{\text{left}}(\boldsymbol{s}) \cup \mathcal{B}_j^{\text{right}}(\boldsymbol{s}), \quad \forall j \in \mathcal{F}. \tag{5.15}$$

Accordingly, we establish the following constraints for each leaf node $j \in \mathcal{F}$:

$$c_j(\boldsymbol{s}) \leq b_i(\boldsymbol{s}), \quad \forall j \in \mathcal{F}, i \in \mathcal{B}_j^{\text{left}}, \tag{5.16}$$

$$c_j(\boldsymbol{s}) \leq 1 - b_i(\boldsymbol{s}), \quad \forall j \in \mathcal{F}, i \in \mathcal{B}_j^{\text{right}}, \tag{5.17}$$

$$\sum_{j \in \mathcal{F}} c_j(\boldsymbol{s}) = 1, \tag{5.18}$$

where $c_j(\boldsymbol{s}) = 1$ indicates that feature vector $\boldsymbol{s}$ will be routed to leaf node j. Constraints (5.16)-(5.17) are responsible for enforcing the branch path, while constraint (5.18) ensures that each feature vector $\boldsymbol{s}$ can be routed to only one leaf node. Ultimately, the predicted value of the delivery time for feature vector $\boldsymbol{s}$ is given by:

$$v_j(\boldsymbol{s}) = \ell_j(\boldsymbol{s}) c_j(\boldsymbol{s}), \quad \forall j \in \mathcal{F}. \tag{5.19}$$

Revised RASO Model

Finally, the revised RASO with the embedded model tree predictor for delivery time is formulated as follows:

$$[\textbf{RASO-LT}] \quad \max_{\rho_{c,r},\rho_{d,r}} \quad \sum_{r\in\mathcal{R}} O_r(\rho_{c,r}), \tag{5.20}$$

$$\textbf{s.t.} \quad \text{Constraints} \quad (5.3)-(5.5), (5.11)-(5.19),$$

$$D_r(\rho_{d,r}) = PWL(\rho_{d,r}), \quad \forall r \in \mathcal{R}, \tag{5.21}$$

$$\boldsymbol{s} = [\rho_{c,r}, \rho_{d,r}, O_r(\rho_{c,r}), D_r(\rho_{d,r})], \quad \forall r \in \mathcal{R}, \tag{5.22}$$

$$L_r(\rho_{c,r}, \rho_{d,r}) = \sum_{j\in\mathcal{F}} v_j(\boldsymbol{s}), \quad \forall r \in \mathcal{R}, \tag{5.23}$$

$$L_r(\rho_{c,r}, \rho_{d,r}) - \frac{1}{|\mathcal{O}|}\sum_{o=1}^{|\mathcal{O}|} L^o \leq \epsilon_{\max}, \quad \forall r \in \mathcal{R}. \tag{5.24}$$

The objective function (5.20) is designed to maximize the total number of prospective orders $O_r(\rho_{c,r})$, which will be discussed in the following paragraph. Constraints (5.3)-(5.5) restrict the feasible domains of the decision variables, and constraints (5.11)-(5.19) show the integration of the order demand estimation and delivery time prediction with the optimization tool. Constraint (5.22) represents $\boldsymbol{s}$ as a feature vector. In the decision time, the platform can observe the drivers' current locations and compute the distance between the restaurant and them. In real-world scenarios, the total number of drivers $D_r(\rho_{d,r})$ in the DDA for restaurant r with radius $\rho_{r,d}$ can be approximated using a piecewise-linear function (Constraint (5.21)). This modeling approach is widely adopted in practice and is supported by popular commercial optimization solvers like Gurobi and CPLEX. Constraint (5.23) indicates that the average order delivery time equals the predicted value at the selected leaf node j. Constraint (5.24) is the delivery time constraint.

In the following, we provide a brief discussion on the prospective orders $O_r(\rho_{c,r})$ and served orders $O_r(\rho_{c,r}, \rho_{d,r})$. Suppose a customer place an order from a restaurant and the platform accepts the order and proceeds to dispatch a driver for delivery. Each order can result in two outcomes: (1) if an available driver is nearby, the order is successfully

delivered and classified as "served"; (2) if the platform fails to locate an available driver within a certain waiting time (e.g., 15 minutes), the customer cancels the order, and it is marked as "expired." In the extreme case where the DDA radius $\rho_{d,r} = 0$, all orders will be expired as no drivers are available for the restaurant. From the perspective of the platform, since the aim is to maximize the number of orders served, in other words, we want each order can be delivered successfully within the promised delivery time and the number of expired orders to be as low as possible (ideally zero). We have observed that expired orders tend to occur when the service capacity is much smaller than the number of prospective orders. For instance, this can happen when the CSA radius is extremely large (e.g., 10 kilometers) and the DDA radius is very small (e.g., 1 kilometers). To address this issue, we impose a penalty during data preparation for training machine learning models used in delivery time prediction. This is achieved by setting a large delivery time Δt for expired orders. Consequently, in case expired orders occur during the solution of the RASO problem, the corresponding predicted average delivery time will not satisfy the time constraint (5.24). This approach ensures that when solving the RASO problem, the primary focus is on maximizing the total number of orders that can be successfully delivered, represented by the sum of $O_r(\rho_{c,r})$ for all restaurants in the set $\mathcal{R}$.

In the next two sections, we present experimental setups for the estimation of the number of orders and prediction of the average order delivery time. We conduct comprehensive experiments to evaluate the performance of the proposed data-driven optimization framework.

5.4 Simulator, Dataset, and Prediction Models

In this section, we first introduce the simulator we developed to simulate the process of food delivery services in Section 5.4.1, including decisions on the radii of CSA and DDA for restaurants, order generation, dispatching, pickup, and delivery. Then in Section 5.4.2, we present a real-world delivery dataset from a crowd-sourcing food delivery platform. The performance of customer demand estimation is discussed in Section 5.4.3,

followed by a discussion on the performance of order delivery time prediction models in Section 5.4.4.

5.4.1 Simulator for On-Demand Food Delivery Service

To better simulate the real-world environment in food delivery service and also to generate training samples for order delivery time prediction depicted in Figure 5.5, we develop a simulator that is calibrated using real data. The simulator contains several components: service area decisions, order generation, information collection, order dispatching algorithm, and status updates of orders and drivers. In Algorithm 3, we show the entire simulation procedure. As can be seen, we solve the RASO problem for each time period $t \in \mathcal{T}$ to get the optimal radius for CSA and DDA for each restaurant, then for each order dispatching interval $k \in [t, t+1)$, we proceed the following steps: (1) generating new order requests for each restaurant within the CSA; (2) gathering information about all dispatched and newly generated orders, drivers, and restaurants, such as each order's pickup and delivery locations, driver's location, driver's carried orders, restaurant dispatch area, and so on; (3) implementing the order-dispatching algorithm to dispatch orders to drivers who have available capacity in the DDA; (4) updating orders' status for orders that have been successfully dispatched, picked up and delivered, also for orders that have been canceled by customers; and (5) updating the drivers' planned delivery routes based on newly dispatched orders.

We also account for the random ready time for orders in our simulations. In food delivery services, the preparation time (which includes cooking the meal and making it ready for pickup) for restaurants is often highly uncertain. Accurately estimating the food preparation time for each order can yield significant benefits, both for delivery drivers and the customers' experience. In this study, we make the assumption that the food preparation time for each restaurant follows distinct gamma distributions, characterized by different shape parameters α and a scalar parameter β. This choice is motivated by the common use of gamma distributions for modeling waiting times in on-demand food delivery

Algorithm 3 Simulation

Input: information of orders $\mathcal{O}$, drivers $\mathcal{D}$ and restaurants $\mathcal{R}$

procedure SIMULATION($\mathcal{O}, \mathcal{D}, \mathcal{R}$)
  for each area optimization time period $t \in \mathcal{T}$ **do**
    Area optimization: Solve the RASO model and get the optimal customer service area $A_{c,r}$ and driver dispatch area $A_{d,r}$.
    for each order dispatch time interval $k \in t_k$ **do**
      Order generation: generate new order requests for each restaurant according to the orders arrival rate within the customer service area $A_{c,r}$
      Information collection: collect information of all dispatched and newly generated orders, drivers, and restaurants;
      Order dispatch: dispatch remaining and newly generated orders to drivers in the driver dispatch area who have available capacity using Algorithm 2;
      Update orders' status: update the status of orders that have been successfully dispatched, picked up and delivered, also for orders that have been canceled by customers;
      Update drivers' status: update drivers' planned delivery route based on newly dispatched orders;
    end for
    end for
  end for
  end for
end procedure

platforms [121, 45].

5.4.2 Real-world Dataset

The dataset used in the numerical experiments is provided by a crowd-sourcing food delivery platform in Singapore. The data contains a sample of around $80,000$ order records for over $2,000$ restaurants and over $30,000$ customers over 8 months (October 2020 to May 2021). Each delivery record includes order and driver information, including order pickup and delivery locations, delivery distance, order accept time, driver ID, and fee paid. This study focuses on a cluster of restaurants known as "hawker centre" (or "food centre"), which represents an open-air complex commonly found in Singapore. The "hawker centre" is a popular dining destination featuring various food stalls or vendors, offering a wide variety of affordable and delicious local dishes. Figure 5.7 depicts customer locations (i.e, order delivery locations) from four well-known hawker centres [3] in Singapore, one

[3]Due to space limitations, we will concentrate on presenting the results of four representative hawker centres in this chapter. The remaining five hawker centres will be included in the appendix for reference to ensure the readability.

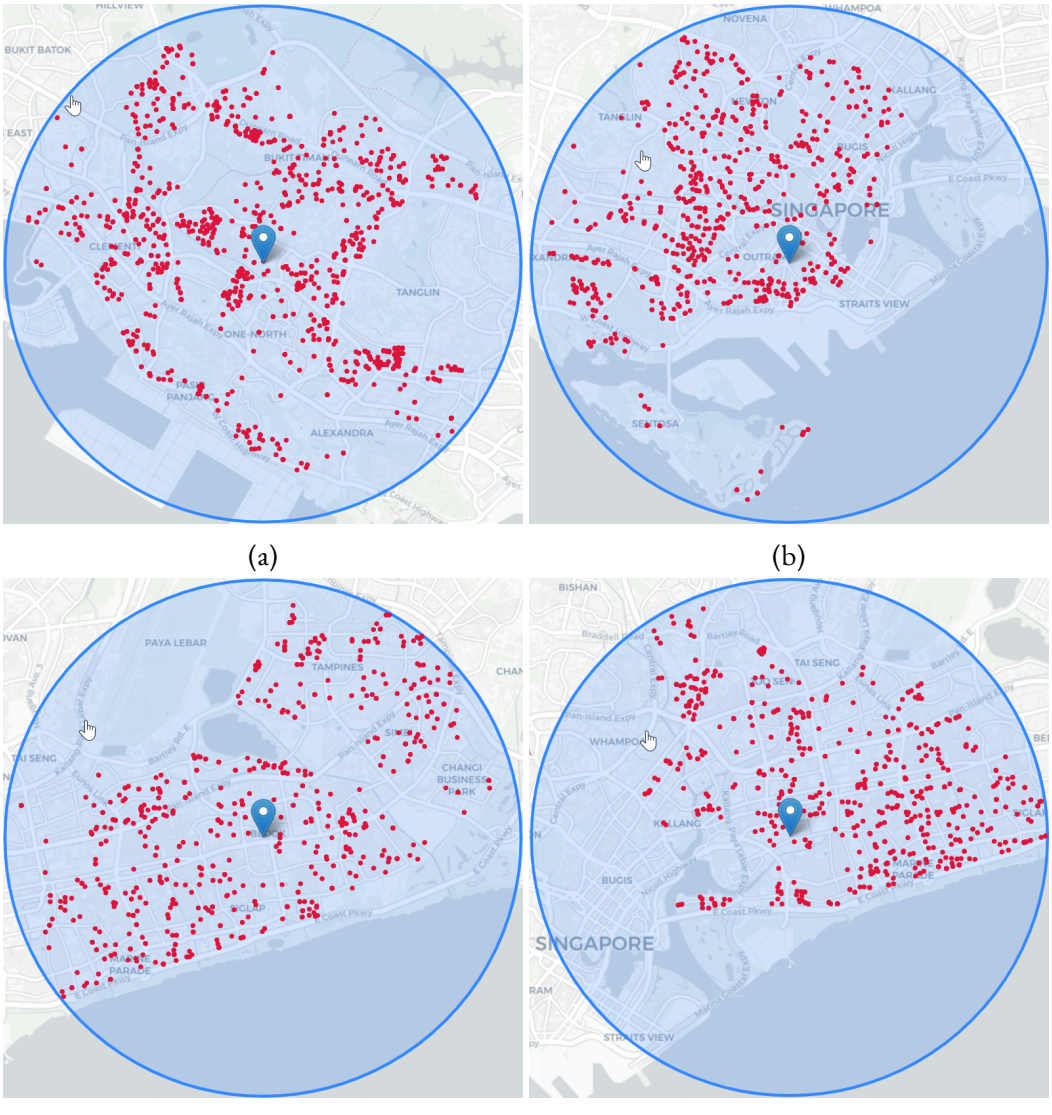


Figure 5.7: Geographical distribution of restaurants (a) Ghim Moh Market & Food Centre, (b) Maxwell Food Centre, (c) Bedok Interchange Hawker Centre, and (d) Old Airport Road Food Centre and order delivery locations (blue circle represents the CSA).

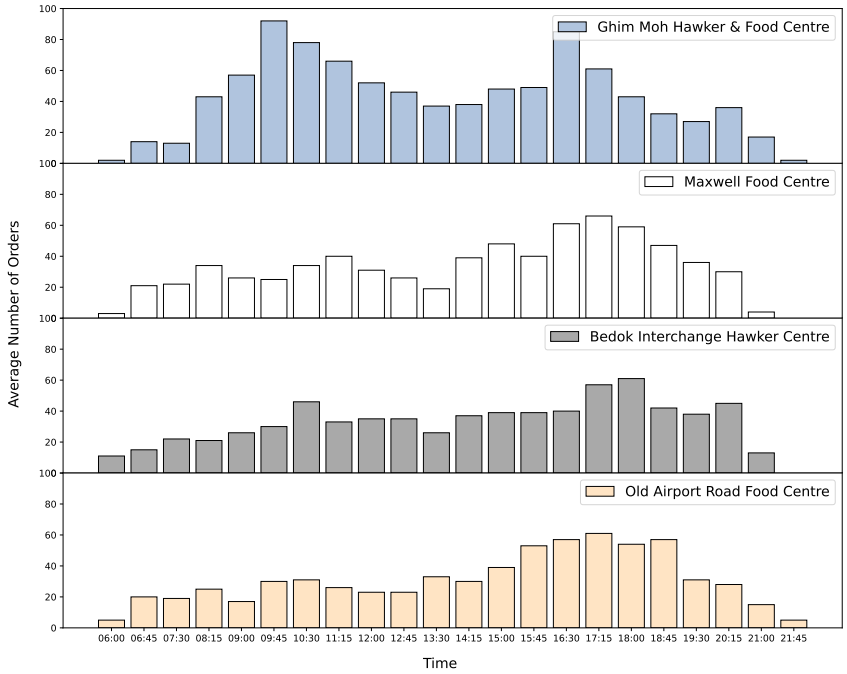


Figure 5.8: (Re-scaled) Average number of orders for restaurants over time (each 45 minutes).

located in the central business district, and others in the residential area. Customer locations are depicted by red dots and hawker centre locations are depicted by blue markers. After cleaning the data and removing some missing values, we choose the top 9 restaurants based on the number of orders in the dataset for further investigation. Figure 5.8 illustrates the average number of orders during various time periods throughout the day, ranging from $6:00$ to $21:45$, for the four selected restaurants. The x-axis represents the time period; and y-axis shows the (re-scaled) average number of orders. Notably, both restaurants exhibit substantial peaks during the noon hours ($9:45$ to $12:00$) and the evening hours ($16:30$ to $18:00$). The distributions of order preparation time are visualized in Figure 5.9.

5.4.3 Results on Customer Demand Estimation

Using the real-world order data, we conduct numerical experiments to evaluate the proposed estimation of customer order demand based on the discrete choice model proposed in Section 5.3.2. Figure 5.10 displays histograms representing the total number of cus-

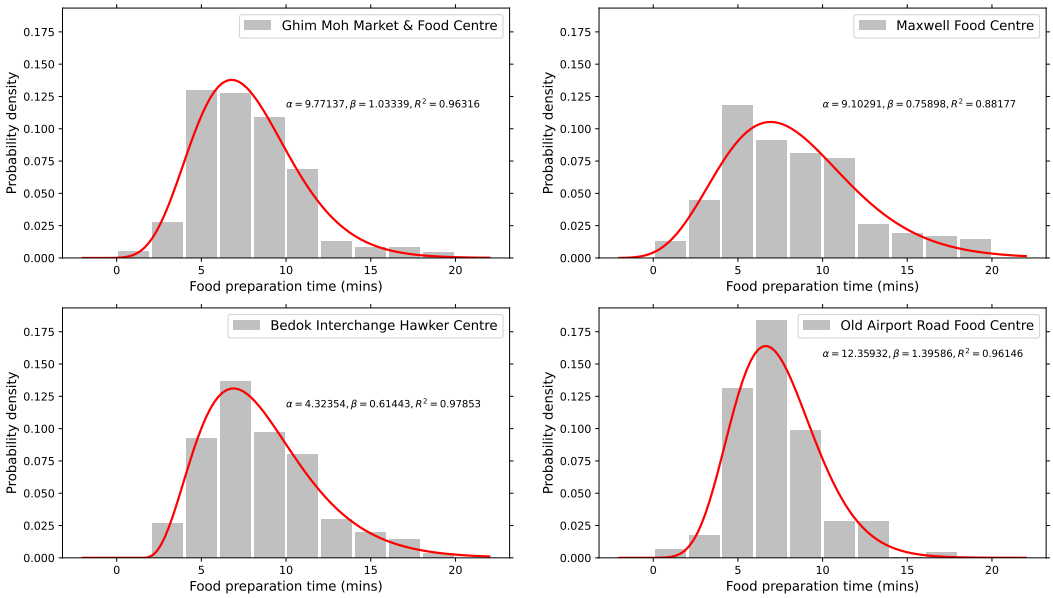


Figure 5.9: Histogram of customer restaurants' orders preparation time.

tomer orders in relation to the distance between the restaurant and the customer for selected hawker centres. The x-axis indicates the customer's distance (in kilometers) from the restaurant, while the y-axis represents the number of orders placed for each restaurant. It shows that as the distance from the restaurant increases, the number of orders exhibits a steep initial increase, followed by a gradual decrease. This observation aligns with the number of orders described by the expression $\alpha'_r e^{\beta r x} x = 2\pi n_r \alpha_r e^{\beta_r x} x$, which is dependent on the distance x from the restaurant, as indicated in equation (5.10). Notably, a greater distance from the restaurant leads to an increase in the marginal service area (represented by $2\pi x n_r$). However, it also results in a reduction in customers' willingness to place orders (demonstrated by $\alpha_r e^{\beta_r x}$ with a negative β_r). To estimate the prospective orders approximation function (5.11) for each restaurant, nonlinear least squares is employed. The coefficient of determination (R^2) is utilized as a performance metric to determine the values of α'_r and β_r for each restaurant:

$$\bar{y} = \frac{1}{n}\sum_{i=1}^{n} y_i, \quad SS_{\text{res}} = \sum_{i=1}^{n}(y_i - \hat{y}_i)^2, \quad SS_{\text{tot}} = \sum_{i=1}^{n}(y_i - \bar{y})^2, \quad R^2 = 1 - \frac{SS_{\text{res}}}{SS_{\text{tot}}}, \tag{5.25}$$

where n is the total number of estimated values, and y_i and $\hat{y}_i$ are the actual value and its estimate, respectively. If the fitted value is exactly the same as the actual value, $R^2 = 1$. The fitting function of each restaurant and its R^2 value are displayed at the top right in each subfigure. We observe that the estimation works well for all hawker centres, with R^2 value ranging from 0.827 (Ghim Moh Market & Food Centre) to 0.892 (Maxwell Food Centre). This demonstrates the effectiveness of the customers' choice model for estimating the number of prospective orders with different service radii. In the appendix, we show the estimation performance for all nine hawker centres.

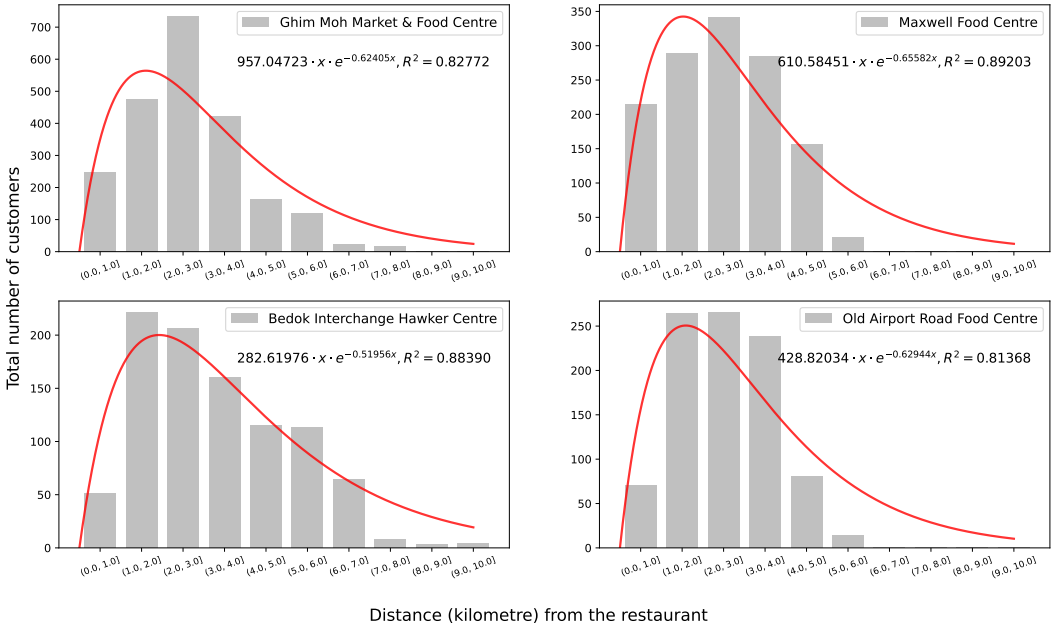


Figure 5.10: Histogram of customers orders in terms of customer-restaurant distance.

5.4.4 Model Selection for Delivery Time Prediction

For order delivery time prediction, we evaluate six machine learning models: ordinary least squares (OLS), ridge regression (Ridge), linear support vector regression (SVR), model tree, regression tree, and XGBoost (Gradient Boosting decision tree). We implement 5-fold cross-validation to select the best parameters (e.g., coefficient value for the regulation term, maximum depth of the tree in tree-based models) for all models. In the experiment, nine instances were generated for different restaurants (with different α'_r and β_r) and different numbers of orders and drivers. The information on instances is shown

in Table 5.5. We simulate deliveries for each restaurant from $11:00$ to $12:00$—which generates over $8,000$ data points for each instance—with 80% as the training set and 20% as the test set. All training, validation, and testing are implemented in Python 3.9.

Table 5.5: Information of instances.

Instance	Restaurant	Orders	Drivers	α'_r	β_r
1	Ghim Moh Market & Food Centre	$[50, 80]$	$[20, 30]$	957.04623	-0.62405
2	Maxwell Food Centre	$[20, 50]$	$[10, 20]$	610.58451	-0.65582
3	Bedok Interchange Hawker Centre	$[15, 35]$	$[5, 15]$	282.61976	-0.51956
4	Old Airport Road Food Centre	$[20, 40]$	$[5, 15]$	428.82034	-0.62944
5	Kovan Food Centre	$[60, 90]$	$[20, 30]$	217.19399	-0.44593
6	Bukit Merah View Market & Food Centre	$[30, 60]$	$[15, 25]$	935.43501	-0.77316
7	Hong Lim Food Centre	$[25, 45]$	$[5, 15]$	499.51778	-0.6803
8	Bukit Timah Market & Food Centre	$[80, 100]$	$[25, 45]$	185.56244	-0.39559
9	Alexandra Village Food Centre	$[35, 55]$	$[10, 20]$	252.13369	-0.52519

We use the mean absolute percentage error (MAPE) and R^2 score as performance metrics for the prediction models:

$$\text{MAPE} = \frac{1}{n}\sum_{i=1}^{n}\frac{|y_i - \hat{y}_i|}{y_i} \times 100\%. \tag{5.26}$$

The results of the machine learning (ML) models' performance in predicting average delivery time are presented in Table 5.6. Based on the R^2 score, all tree-based ML models demonstrate commendable predictive capability for average delivery time. Conversely, the linear models (i.e., OLS, Ridge, and SVR) exhibit significantly lower R^2 scores and higher MAPE values when compared to the tree-based models (i.e., model tree, regression tree, and XGBoost). Particularly noteworthy is the model tree, which consistently achieves the lowest MAPE across all instances. Furthermore, the model tree outperforms XGBoost, as well as all linear models and the regression tree, in both R^2 score and MAPE. Considering prediction accuracy, compatibility, and computational performance, we opt for the model tree as the delivery time prediction model and seamlessly integrate it into the RASO-LT model using the linearization method introduced in Section 5.3.5.

5.5 Experiments and Discussion

In this section, we conduct a set of numerical experiments to evaluate the performance of the **RASO-LT** model comparing it with several benchmarks in practice. Then, we

Table 5.6: Performance of ML methods on average delivery time prediction.

(a) Ghim Moh Market & Food Centre

ML Methods	R^2	MAPE
Ordinal Least Square	0.578	0.567
Ridge Regression	0.577	0.566
Linear SVR	0.344	0.325
Regression Tree	0.900	0.161
Model Tree	**0.912**	**0.147**
XGBoost	0.893	0.176

(b) Maxwell Food Centre

ML Methods	R^2	MAPE
Ordinal Least Square	0.483	0.659
Ridge Regression	0.483	0.658
Linear SVR	0.198	0.351
Regression Tree	0.883	0.199
Model Tree	**0.888**	**0.176**
XGBoost	0.848	0.248

(c) Bedok Interchange Hawker Centre

ML Methods	R^2	MAPE
Ordinal Least Square	0.485	0.626
Ridge Regression	0.485	0.625
Linear SVR	0.251	0.345
Regression Tree	0.914	0.167
Model Tree	**0.922**	**0.154**
XGBoost	0.882	0.212

(d) Old Airport Road Food Centre

ML Methods	R^2	MAPE
Ordinal Least Square	0.489	0.699
Ridge Regression	0.489	0.698
Linear SVR	0.275	0.393
Regression Tree	0.903	0.170
Model Tree	**0.910**	**0.158**
XGBoost	0.834	0.264

discuss the impact of some important factors and variables on the performance of the food delivery services, such as the threshold of order delay and temporal distribution of customer order demand. We run the experiments in Python 3.9 using Gurobi 9.5.1, on a 2.5GHz Xeon CPU.

Table 5.7: Growth rate of served orders for hawker centres.

Hawker Centre	Ghim Moh	Maxwell	Bedok Interchange	Old Airport Road
Growth Rate	17.0%	19.0%	29.6%	19.4%

5.5.1 Model performance.

We report the performance of the proposed **RASO-LT** model in terms of the following four metrics: (1) growth rate of orders served; (2) average delivery time of served orders; (3) average travel distance of each driver; and (4) on-time rate, which is the ratio of orders delivered in promised time to total orders arriving in a specific period of time. We use **Fixed-CSA-DDA** as a baseline, where both the restaurant's CSA and DDA are fixed, with radii set to 5 kilometers (commonly used choices by practitioners).

We conducted experiments during lunchtime ($11:00$ to $12:00$), varying the number of drivers in the system from small to medium and larger scales. We record the metric for results before and after the deployment of the **RASO-LT** as $\mathbb{M}_{\text{Fix}}$ and $\mathbb{M}_{\text{RASO-LT}}$, re-

spectively, which were used to verify the effectiveness of our model. Then we define the $\mathbb{M}_{\text{Rat}}$ as the changing ratio between $\mathbb{M}_{\text{Fix}}$ and $\mathbb{M}_{\text{RASO-LT}}$ as follows:

$$\mathbb{M}_{\text{Rat}} = \frac{\mathbb{M}_{\text{RASO-LT}} - \mathbb{M}_{\text{Fix}}}{\mathbb{M}_{\text{Fix}}} \tag{5.27}$$

The results are presented in Table 5.7, demonstrating that the **RASO-LT** model's newly introduced restaurant's CSA and DDA lead to significant improvements ranging from 17.4% to 29.6% for the selected four prominent hawker centres when compared to the conventional fixed CSA and DDA approach. Detailed experimental outcomes are depicted in Figure 5.11. Specifically, the average travel distance of drivers increased by 16.5% to 20.3%, 11.1% to 26.0%, 15.1% to 27.0%, and 5.6% to 13.7% for the four hawker centres, respectively. It is also evident that there was a decline in delivery efficiency to a certain degree. The average order delivery time experienced increases of 2.4% to 6.3%, 6.0% to 9.7%, 4.3% to 27.7%, and 4.2% to 15.3% for the corresponding centres. These findings indicate that the proposed method demonstrates superior performance with a sufficient number of drivers (e.g., large instance), as the corresponding rise in the average order delivery time remains comparatively modest. As the number of orders served grows, the on-time delivery rate is observed to decrease, with reductions of up to 2.5%, 7.6%, 6.9%, and 7.0%, respectively (Figure 5.11c). Full details of the evaluation results are provided in Table 7.2 in the Appendix.

To comprehensively evaluate our proposed area sizing optimization strategy, we also assess its performance against additional benchmark policies for longer service times, ranging from $09:00$ to $15:00$. Those two new benchmarks are:

- **Fixed-CSA**: The restaurant's CSA is fixed with a radius of 5 kilometers, while the DDA is optimized.
- **Fixed-DDA**: The restaurant's DDA is fixed with a radius of 5 kilometers, while the CSA is optimized. [4]

[4] The radius for a restaurant's CSA is set to around 4 kilometers in [29]. However, in our work, we set the fixed radius of restaurant's CSA or DDA slightly higher to 5 kilometersto account for the higher maximum delivery distance of 10 kilometers in our dataset compared to [29].

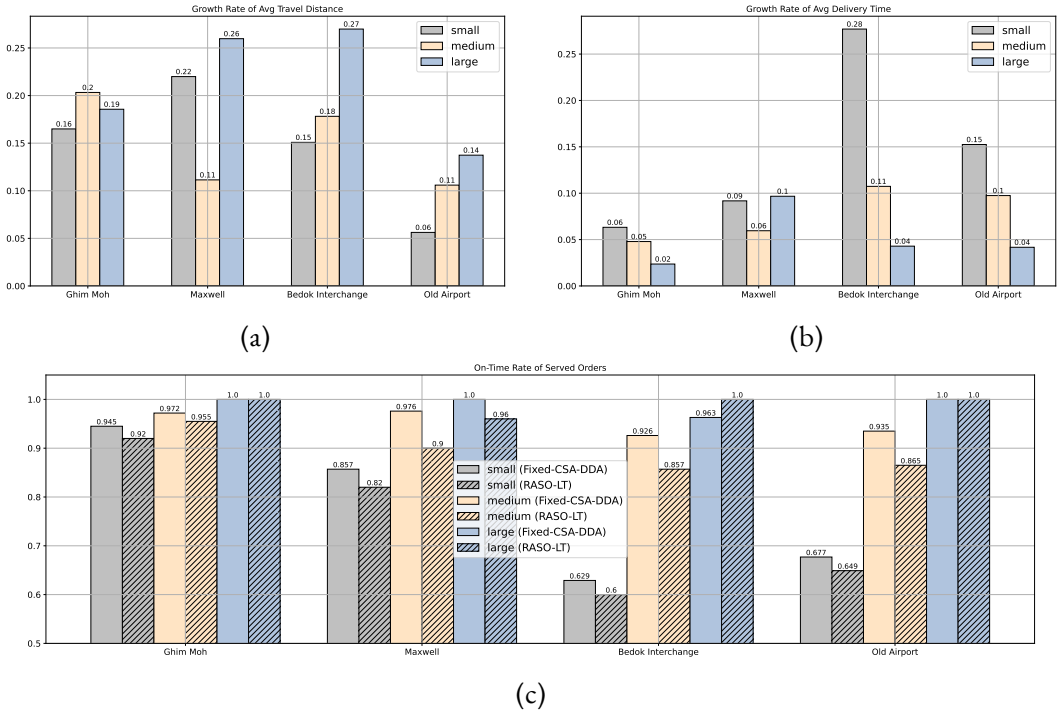


Figure 5.11: Experiment results on four big food centres: (a) growth rate of average travel distance, (b) growth rate of order average delivery time, and (c) on-time rate of served orders.

We have selected Ghim Moh Market & Food Centre with learned parameter $\alpha'_r = 957.04723$ and $\beta_r = -0.62405$ to conduct a comprehensive and in-depth analysis. Additionally, we also analyze the effects of different parameter settings, including the threshold of delay and order arrival rate. Moreover, we set the promised delivery time for orders to 50 minutes with the threshold for order delay $\epsilon_{\max} = 0$, which is commensurate with common customer expectations (e.g., [69]). Additionally, we assume the number of drivers in the system equals to 15 for all time periods $t \in \mathcal{T}$, and these drivers are randomly distributed in the area. Each driver is assumed to have a service capacity of 10 orders ($p = 10$). The maximum and lowest radius for both restaurants' CSA ($\rho_c^{\max}$, $\rho_c^{\min}$) and DDA ($\rho_d^{\max}$, $\rho_d^{\min}$) are set to be 2 kilometers and 10 kilometers, respectively.

Table 5.8 shows the number of orders served and the average order delivery time for the proposed **RASO-LT** model over other benchmark policies. To compare the policies, we calculate the relative improvement in terms of the number of orders served $\mathbb{M}^o_{\text{Rat}}$, and the actual average order delivery time for each time period $t \in \mathcal{T}$. Compared with poli-

cies **Fixed-DDA**, **Fixed-CSA** and **Fixed-CSA-DDA**, we find that the **RASO-LT** model serves the most orders with 16.9%, 19.2% and 19.2% improvements, and also leads to the average delivery time increased by 5.7%, 8.0% and 8.9%, respectively. The results indicate that adjusting the CSA and DDA simultaneously for restaurants can result in substantial service improvements at the cost of slightly longer delivery times.

Table 5.8: Number of orders served and average order delivery time for different polices.

Policy	RASO-LT	Fixed-DDA	Fixed-CSA	Fixed-CSA-DDA
Number of Orders Served	**802**	686	673	673
Order Delivery Time (minute)	**42.7**	40.38	39.5	39.2

5.5.2 The impact of the threshold for order delay.

To understand the impact of the threshold for order delay on the performance, we calculate the number of orders served with the varying values of $\epsilon_{\max}$ from -10 minutes (orders are required to deliver within 40 minutes) to 10 minutes (orders are required to deliver within 60 minutes) for all policies[5]. However, the delivery time constraint may be violated when we have a negative value of $\epsilon_{\max}$ for **Fixed-CSA-DDA**, since the fixed radius of CSA $\rho_c = 5$ kilometers and radius of DDA $\rho_d = 5$ kilometers are were specially selected to allow orders to be delivered within 50 minutes. To be fair in comparison, we decrease the radius of CSA ρ_c for **Fixed-CSA-DDA** from 5.0 kilometers to 4.5 and 4.0 kilometers when $\epsilon_{\max} = -10$ and $\epsilon_{\max} = -5$, respectively, to satisfy the delivery time constraint (5.24).

As shown in Figure 5.12 (a), with $\epsilon_{\max}$ becomes larger, both **RASO-LT** and **Fixed-DDA** can serve more orders. We also observe that the **RASO-LT** policy can provide the best solution quality with the largest number of orders served. When $\epsilon_{\max}$ is set to -10, -5, 0, 5, and 10 minutes, **RASO-LT** serves 3.1%, 13.4%, 19.2%, 20.6% and 20.6% more orders comparing to **Fixed-CSA**, and serves 11.0%, 16.6%, 16.9%, 10.5% and

[5]In industry practice, the platforms usually require drivers an ETA which is different from the ETA promised to the customers. This difference can be represented by the parameter $\epsilon_{\max}$. From another perspective, the impact of $\epsilon_{\max}$ can also be interpreted as the impact of ETA (i.e., L^o) when $\epsilon_{\max} = 0$.

6.8% more orders comparing to **Fixed-DDA**. Furthermore, compared with policies with a fixed radius of CSA ρ_c, the relative improvement in the number of orders served by our **RASO-LT** will gradually increase with longer allowed delays, until it reaches the maximum value 20.6% (ρ_c equals to 10 kilometers). This indicates that our **RASO-LT** is more advantageous when customers are more patient with delivery times. Finally, for **RASO-LT**, we see that the increased amount of orders served are 69, 39, 10, and 0 with each 5 minutes increment in the value of ϵ_{max} from -10 to 10. Those observations imply that allowing more time for delivery does assist achieve more orders, while the benefits progressively decrease.

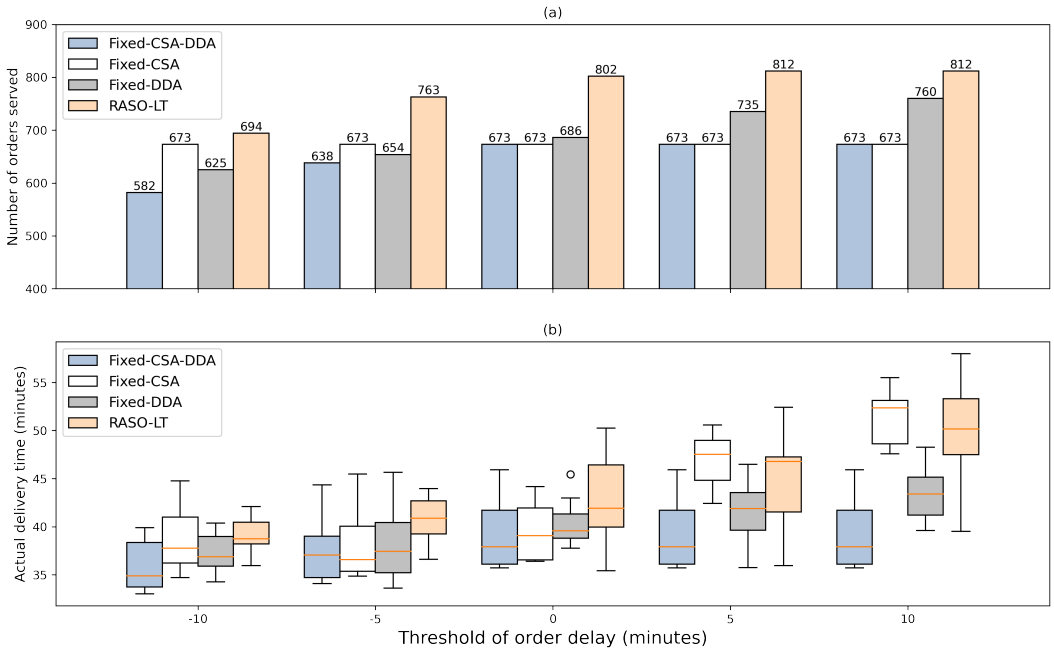


Figure 5.12: Impact of the threshold for order delay ($\epsilon_{\max}$) on (a) the number of orders served and (b) the actual delivery time of orders.

We also calculate the actual order delivery time in each time period $t \in \mathcal{T}$ with different $\epsilon_{\max}$, as depicted as box plots in Figure 5.12 (b). As can be observed, for policies that can optimize the radius of CSA ρ_c (i.e., **RASO-LT** and **Fixed-DDA**) to serve more orders, the average order delivery time grows strictly as the number of orders served. For example, we can find that the average order delivery time of **RASO-LT** is always larger than **Fixed-DDA** across all $\epsilon_{\max}$, since **RASO-LT** can always serve more orders than **Fixed-DDA**. However, for **Fixed-CSA** that can only optimize the radius of DDA ρ_d, we also

find that as the $\epsilon_{\max}$ increases, so does the actual average order delivery time. The explanation for this is a larger $\epsilon_{\max}$ will make the policy to presume that customers are more lenient with the order delivery time, resulting in a smaller radius of DDA with fewer available drivers and a longer actual delivery time.

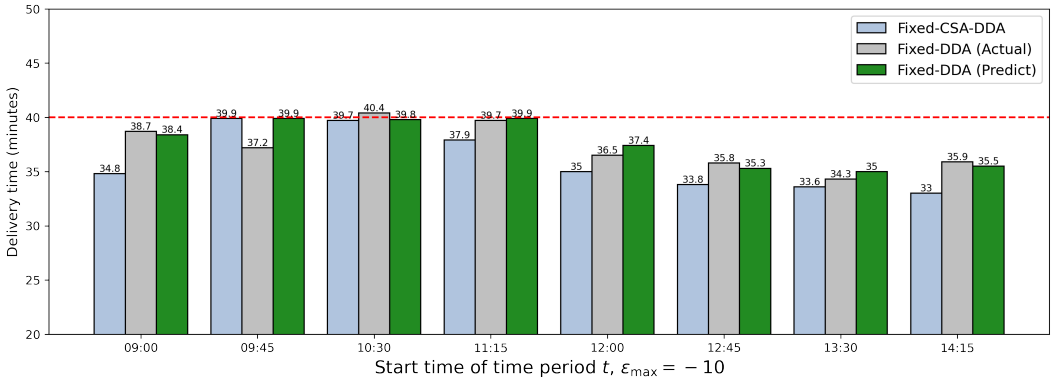


Figure 5.13: Order delivery time comparisons with $\epsilon_{\max}$ equals to -5.

It is worth noting that **Fixed-CSA-DDA** serves fewer orders served at $\epsilon_{\max}$ equals to -10 and -5, which are 528 and 638, respectively, than **Fixed-CSA**, which serves 673 orders. This happens because we decrease the radius of CSA ρ_c from 5 kilometers to 4 kilometers ($\epsilon_{\max} = -10$) and 4.5 kilometers ($\epsilon_{\max} = -5$) to ensure the delivery time constraint (5.24) that the predicted average order delivery time $L_r(\rho_c, \rho_d)$ must be less than or equal to $L^o + \epsilon_{\max}$ (by default, the promised delivery time $L^o = 50$) minutes in each time period $t \in \mathcal{T}$. To better comprehend the importance of the delivery time constraint, we compare **Fixed-CSA-DDA** and **Fixed-DDA**, the two policies with the fewest served orders. We compare the actual order delivery time of **Fixed-CSA-DDA**, **Fixed-DDA** and the predicted order delivery time of **Fixed-CSA-DDA** with $\epsilon_{\max}$ values -10 for $t \in \mathcal{T}$ from $09:00$ to $15:00$. The results can be found in Figure 5.13. For **Fixed-DDA**, we observe that the predicted order delivery time is always less than 40 minutes ($L^o + \epsilon_{\max}$), and the actual order delivery time only exceed 40 minutes slightly in time period $10:30$ to $11:15$ due to the prediction errors. The actual delivery time of **Fixed-CSA-DDA** never exceeds 40 minutes even in both $09:45$ to $10:30$ and $10:30$ to $11:15$ when the peak of orders occurs. We also find that the actual delivery time of **Fixed-DDA** is greater than that of **Fixed-CSA-DDA** in every time period t. This is due

to the fact that the more orders served the longer the average order delivery time will be, and **Fixed-DDA** can serve more orders by adjusting ρ_c.

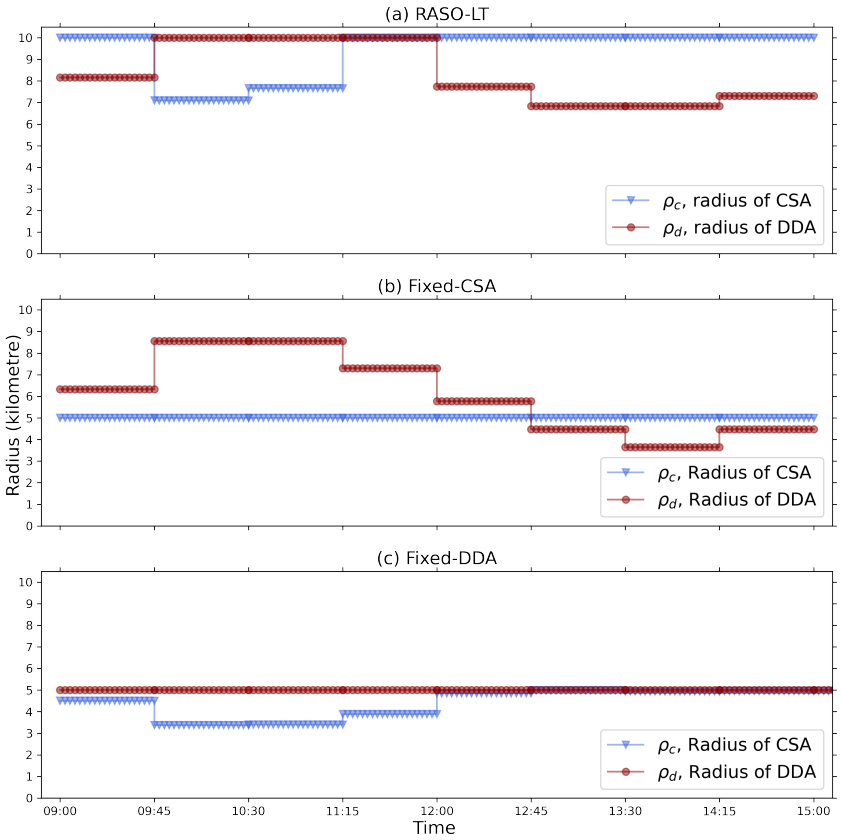


Figure 5.14: Radius over time for policy (a) RASO-LT, (b) Fixed-CSA, and (c) Fixed-DDA.

5.5.3 Comparison of radius adjustments over time.

Next, we investigate how those policies generate the radii of the restaurant's CSA ρ_c and DDA ρ_d over time $t \in \mathcal{T}$, except for **Fixed-CSA-DDA**. Figure 5.14 shows the radius adjustments for the three policies. When only one of the areas can be adjusted, we find that both policies **Fixed-CSA** and **Fixed-DDA** react to the peaks (from $09:45$ to $12:00$) by setting a smaller radius of CSA ρ_c and a larger radius of DDA ρ_d, respectively, and to the off-peaks (from $09:00$ to $09:45$ and $12:00$ to $15:00$) by setting a larger radius of CSA ρ_c and a smaller radius of DDA ρ_d, respectively. In addition, it is interesting to observe that **RASO-LT** can adjust the CSA and DDA simultaneously by setting the radius of DDA ρ_d to the maximum value and a relative moderate radius of

CSA ρ_c during the peak periods. While in the off-peak periods, since the platform aims to serve more orders, the radius of CSA ρ_c is set to the maximum value and the radius of DDA ρ_d can be smaller to meet the demand. In general, we observe that it is always beneficial to maintain a large radius of CSA ρ_c when the restaurant has sufficient service capacity, which can bring more drivers to satisfy the order demand, and the radius of CSA ρ_c will only be decreased to ensure orders will not be delayed when there are no more drivers can be found (ρ_d reaches its maximum value, 10 kilometers).

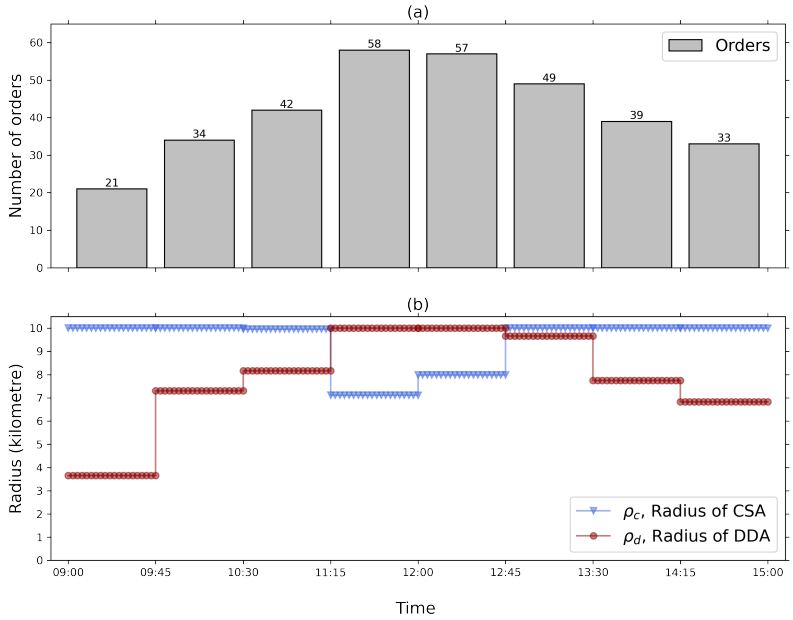


Figure 5.15: Radius over time (RASO-LT) with customer orders arrival distribution.

5.5.4 Comparison with different customer order arrival rates.

Last, we analyze the radius adjustment decisions over time $t \in \mathcal{T}$ for different customer order arrival rates. In an extreme peak scenario, where the customer order arrival rates are more than twice that of off-peak periods, we illustrate the radius adjustment decisions using **RASO-LT** in Figure 5.15. The results demonstrate that the DDA radius adjustments align with the changes in customer order arrival rates. Moreover, the CSA radius ρ_c only decreases to around 7 kilometers during peak periods from $11:15$ to $12:45$, which corresponds to the occurrence of the highest order peak.

By combining the results presented in Figures 5.14 and 5.15, we observe that our

RASO-LT approach primarily adjusts the radius of the restaurant's DDA ρ_d first and subsequently adapts the CSA radius ρ_c to meet the order's delivery time constraint. This adjustment allows us to reduce the CSA when the restaurant encounters a shortage of service capacity during peak periods.

5.6 Conclusion

This chapter introduces the restaurant area sizing optimization problem (RASO) as a new operational problem for managing supply and demand in on-demand food delivery services. The main objective is to maximize the total number of orders served while ensuring a required service level for order delivery time. Initially, we examine the relationship between the radius of the customer service area and the number of customer orders received by the restaurant, focusing on the demand side. Subsequently, we analyze the number of drivers, which depends on the radius of the driver dispatch area, and explore various factors to predict the average delivery time on the supply side. To model the problem, we integrate closed-form formulations for order estimation based on a customer choice model and delivery time prediction using a model tree model. The resulting integrated model is formulated as an MIQCP (Mixed-Integer Quadratically Constrained Program) and can be efficiently solved using Gurobi as the optimization solver.

We utilize a customized simulator capable of simulating order generation, placement, and dispatch, along with real-world food delivery data provided by our industry partner, to conduct extensive experiments. The objective is to evaluate the performance of our proposed RASO-LT model and compare it with other benchmark area sizing policies. The results show a significant performance improvement achieved by our approach, indicating that simultaneous adjustments of the radii of the customer service area (CSA) and driver dispatch area (DDA) can substantially increase the total number of orders served within an acceptable delivery time. Furthermore, we investigate the impact of various factors related to demand, supply, and the threshold for order delay on the radius adjustment decisions over time.

Several future research directions can be explored. A natural avenue involves modeling the RASO problem as a Markov decision process and utilizing reinforcement learning approaches to learn a policy for sequential decision-making on CSA and DDA from end to end. Another intriguing direction is to synergize our area sizing optimization with other powerful tools, such as surge pricing. For instance, in scenarios with extreme undersupply where increasing the DDA fails to bring sufficient supply to serve orders placed in the CSA with a radius $\rho_c^{\min}$, an integrated optimization method combining area sizing and surge pricing could prove valuable.

Chapter 6

Learning Order-Level Dispatch for On-Demand Food Delivery Service

6.1 Motivation and Background

One of the significant transformations witnessed in the logistics domain in recent years pertains to the emergence and triumph of on-demand passenger and logistics service platforms. Notably, on-demand food delivery platforms such as Uber Eats, DoorDash, Grab Food, and Meituan have played a pivotal role by offering efficient door-to-door food delivery services. These platforms have achieved remarkable success, with the COVID-19 pandemic further catalyzing their growth. For example, Grab Food, operating in 480 cities across eight countries and serving as the largest food delivery service provider in Southeast Asia, reported a substantial gross merchandise volume of $7.6 billion in 2021, reflecting a remarkable 29% year-on-year increase. Similarly, Meituan, China's largest on-demand food delivery platform, handles over 30 million daily orders and achieved a profit of 4.71 billion RMB in 2020. According to a report by [84], the food delivery market in the United States experienced a twofold surge during the COVID-19 pandemic, with its market value surpassing $150 billion.

Within the on-demand food delivery market, as the same illustrated in Figure 5.1, the ecosystem comprises three principal entities: customers, restaurants, and drivers (or

drivers), in addition to the platform itself. Customers utilize the platform to peruse and select food options from nearby restaurants enlisted on the platform. Upon placing an order, the platform promptly notifies the respective restaurant to commence food preparation and conveys comprehensive order details, including the delivery task, to drivers located in proximity. Subsequently, the assigned driver collects the prepared food from the restaurant and ensures its prompt delivery to the customer's location. Throughout this process, the paramount challenge lies in the effective dispatching of orders to drivers, facilitating timely pickups from restaurants and subsequent deliveries to customers, all within stringent promised delivery time constraints (e.g., 45 minutes).

Conventional order dispatching strategies implemented in practical settings (e.g., [151]) typically involve fixed time intervals (e.g., every 10 minutes), within which the platform accumulates all incoming orders and consolidates them into a matching pool. Subsequently, the orders are collectively assigned to drivers for simultaneous delivery. However, recent investigations in the realm of ridesourcing (e.g., [57, 140]) have revealed that by extending the dispatching time interval, it is possible to optimize the allocation of drivers to passengers, leading to reduced wait times for both parties involved. Inspired by the success of extended dispatching time intervals in ridesourcing, our research focuses on optimizing the dispatch times for individual orders in on-demand food delivery services. Specifically, our research aims to identify the optimal timing for each individual order to enter the matching pool, thereby initiating the dispatching process, upon its arrival on the platform.

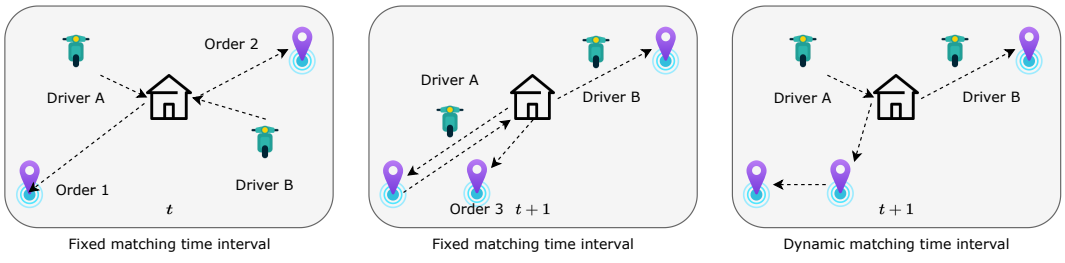


Figure 6.1: Delivery distance/time reduction with dynamic dispatching time interval.

Figure 6.1 provides an illustrative example that demonstrates the impact of optimizing dispatch times at an order-level on both delivery distance and time. The figure highlights

two distinct dispatching times, denoted as t and $t+1$. At $t+1$, three orders are received, with orders 1 and 2 arriving prior to t, and order 3 arriving within the time interval $(t, t+1]$. Initially, we consider dispatching using a fixed time interval approach, wherein all orders arriving in the interval $(t-1, t]$ enter the matching pool and are dispatched to drivers for delivery at time t. As per this strategy, orders 1 and 2 are assigned to drivers A and B respectively at t, while order 3 is dispatched to driver A at $t+1$, who is en route to deliver order 1. The delivery routes are depicted by black dotted lines with arrows. However, if we postpone the inclusion of order 1 in the matching pool until $t+1$, driver A can conveniently pick up both order 1 and 3 for delivery. This adjustment results in a reduced total travel distance, albeit at the expense of a lengthier waiting time for order 1.

Although implementing above postponement strategy at the order level can significantly reduce the total travel distance and decrease delivery times, it is crucial to acknowledge the potential trade-offs involved: the extended dispatching time may elicit impatience among customers, prompting them to cancel their orders. Thus, understanding the delicate balance between customer waiting time, driver travel distance, and order delivery time across various dispatching time intervals (or the number of intervals to wait before entering the matching pool) becomes crucial. Consequently, developing optimal dispatching strategies that yield superior system performance is of paramount importance. A promising approach to this entails dynamically determining the dispatch time for each customer order.

To address the aforementioned challenges, we propose a comprehensive model that treats the dynamic order dispatching problem in on-demand food delivery services as a multi-agent Markov decision process. In our model, each customer order is considered as an independent agent with the ability to make decisions regarding its entry into the matching pool for dispatch. Subsequently, the platform employs a driver routing algorithm to optimize the insertion of orders into drivers' delivery routes. In summary, our study makes the following key contributions:

- We investigate the dynamic order dispatching problem in the context of on-

demand food delivery services and formulate it as a Markov decision process. Specifically, we focus on the setting where each order decides its entry into the matching pool in the subsequent dispatching times, followed by the application of the driver routing algorithm.

- We propose a two-stage optimization framework based on learning techniques to address the dynamic order dispatching problem. In the first stage, we use an end-to-end RL optimization approach. This involves crafting a centralized multi-agent deep Q network that streamlines order dispatching, allowing orders to join the matching pool for pickup and delivery. For the second stage, we leverage an efficient driver routing algorithm that takes into account driver routing behavior and supports the re-dispatch of orders.

- We conduct extensive numerical experiments using a real-world dataset to evaluate the performance of the proposed framework. The computational study demonstrates that our framework outperforms benchmark methods, indicating its effectiveness in improving the efficiency of order dispatching in on-demand food delivery services.

In conclusion, our research provides a comprehensive model and a learning-based optimization framework that significantly enhance the order dispatching process in on-demand food delivery services, leading to improved operational efficiency and customer satisfaction.

The reminder of this chapter is organized as follows. Section 6.2 describes the research problem. Section 6.3 presents MDP model for the optimization of the dispatching time at an order-level as well as the driver routing algorithm. Section 6.4 introduces the real data and simulator. Section 6.5 presents experimental results of the proposed model and other benchmark methods. Finally, we conclude and discuss future research in Section 6.6.

6.2 Problem Description

In this section, we present a formal description of the order-level dispatch time optimization problem for a on-demand food delivery service platform, and present a simple example for illustration.

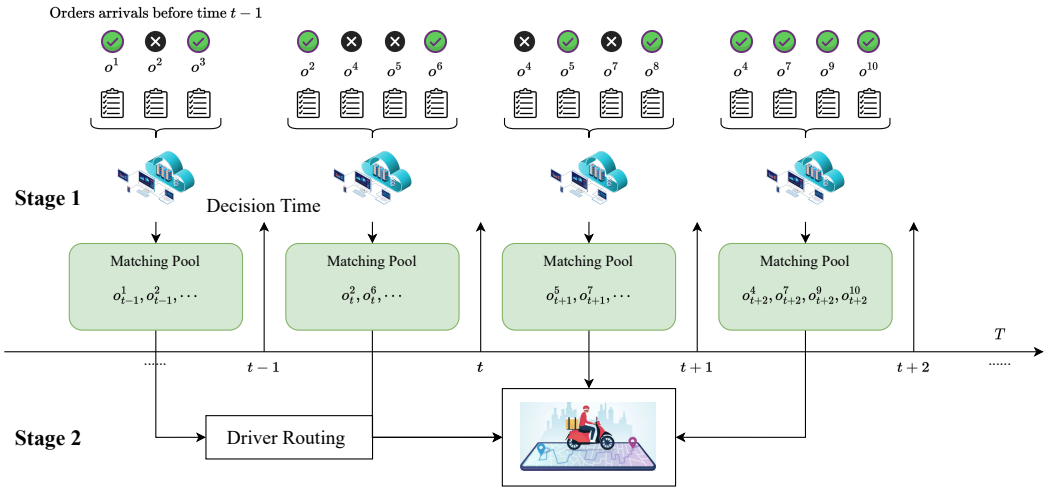


Figure 6.2: A two stage optimization framework for on-demand food delivery service.

We are given a fleet of drivers $\mathcal{D} = \{d_1, d_2, \cdots, d_m, \cdots\}$ to deliver a set of orders $\mathcal{O} = \{o^1, o^2, ..., o^n, ...\}$ that arrive dynamically in the planning horizon $\mathcal{T} = \{1, 2, ..., T\}$. Each order $o \in \mathcal{O}$ can be represented as a tuple (o_t, o_+, o_-, p^o) that contains the arrival time, pickup and delivery locations and the promised delivery time. Each driver $d \in \mathcal{D}$ has a service capacity cap (i.e., the maximum number of orders that can be carried) and drivers are distributed around the entire area. The platform must make two decisions: (1) when to release the orders to the matching pool; and (2) how to dispatch the orders in the matching pool for available drivers for delivery. More specifically, at each dispatch time $t \in \mathcal{T}$, the platform could determine which orders with $o_t \leq t$ can enter the matching pool, followed by specific order dispatching in batches. Those decisions are made sequentially over time until it reaches an ending time of T. An example of the proposed two-stage framework is demonstrated in Figure 6.2. As shown in the figure, the order dispatching decisions are made at time $t-1, t, t+1$ and $t+2$, e.g., every 5 minutes, and between consecutive decision times, we have a matching pool to hold all orders are

allowed to dispatch to available drivers for pickup and delivery. For example, before time $t-1$, there are three orders, o^1, o^2, o^3 arrive in the platform and waiting for dispatching. The platform decide order o^1 and o^3 can enter the matching pool, while order o^2 should waiting for the next decision time at t. And there are three new orders (o^4, o^5, o^6) arriving between $t-1$ and t. Order o^4 is delayed again for dispatch and only order o^6 is allowed to enter the matching pool. All orders including order o^4 which arrive in the platform before time t are allowed for dispatch at time $t+2$. In this case, a total of four orders are delayed for dispatch, with orders o^2, o^5 and o^7 waited for one more time interval, while order o^4 waited two time intervals. Following the determination of dispatchable orders, the platform will allocates each order in the dispatch pool to available nearby drivers and subsequently updates the driver's route for pickup and delivery.

Here, we are interested in optimization from a platform's perspective, i.e. we aim to minimize the total cost with the delivery constraints. To be specific, as given in the objective function below, the platform aims to minimize the total travel distance while ensuring a service level of served orders as well as order delivery time.

$$\textbf{min} \quad \sum_{d\in\mathcal{D}} c_d + \lambda \cdot \sum_{o\in\mathcal{O}} \max(0, f^o - p^o) + \mu \cdot \sum_{o\in\mathcal{O}} u^o \tag{6.1}$$

In this context, we use c_d to denote the total travel distance of driver d within the entire planning horizon $\mathcal{T}$. Furthermore, we define variable f^o to represent the time when customer o receives their order, and u^o as a binary variable with a value of 1 if order o is cancelled by the customer, and 0 otherwise. The first component computes the total travel distance for all drivers. We impose penalties for served orders with delays or orders cancelled by customers through the assignment of large values to parameters λ and μ, respectively. The entire mathematical programming model is given as follows. We minimize

the total cost given by (6.1) above subject to the following constraints:

$$\sum_{d\in\mathcal{D}}\sum_{j\in\mathcal{L}_1\cup\mathcal{D}} x_{ijd} = 1, \quad \forall i \in \mathcal{L}_1 \tag{6.2}$$

$$\sum_{i\in\mathcal{V}} x_{ijd} - \sum_{i\in\mathcal{V}} x_{jid} = 0, \quad \forall j \in \mathcal{L}_1 \cup \mathcal{L}_2, d \in \mathcal{D} \tag{6.3}$$

$$t_{id} + s_i + d_{ij} - M(1 - x_{ijd}) \leq t_{jk}, \quad \forall i, j \in \mathcal{L}_1 \cup \mathcal{L}_2, d \in \mathcal{D} \tag{6.4}$$

$$t_{o^+d} \leq t_{o^-d} \quad \forall o \in \mathcal{O}, d \in \mathcal{D} \tag{6.5}$$

$$w_{jd} \leq w_{id} + q_j + M(1 - x_{ijd}), \quad \forall i, j \in \mathcal{L}_1 \cup \mathcal{L}_2, d \in \mathcal{D} \tag{6.6}$$

$$w_{jd} \geq w_{id} + q_j - M(1 - x_{ijd}), \quad \forall i, j \in \mathcal{L}_1 \cup \mathcal{L}_2, d \in \mathcal{D} \tag{6.7}$$

$$w_{id} \leq cap, \quad \forall i \in \mathcal{L}_1 \cup \mathcal{L}_2, d \in \mathcal{D} \tag{6.8}$$

$$\sum_{d\in\mathcal{D}}\sum_{j\in\mathcal{L}_1\cup\mathcal{L}_2} x_{ijd} d_{ij} = c_d, \quad \forall d \in \mathcal{D} \tag{6.9}$$

$$x_{ijd} \in \{0, 1\} \quad \forall i, j \in \mathcal{L}_1 \cup \mathcal{L}_2, d \in \mathcal{D} \tag{6.10}$$

The binary decision variable x_{ijd} indicates whether driver d travels from location i to j. The continuous variable t_{id} represents the arrival time of driver d at location i, while variable w_{id} denotes the number of orders handled by driver d upon departing from location i. Parameters s_i and d_{ij} denote the service time at location i and the travel distance between locations i and j, respectively. Constraint (6.2) ensures that both the pickup and delivery nodes for each order are visited exactly once. Constraint (6.3) ensures that each order o be served by a single driver d. Constraints (6.4) and (6.5) pertain to the timing of location visits and ensure the practicability of order deliveries. More precisely, when consecutive locations i and j are traversed in the route of driver d, the arrival time at location j must exceed or equal the departure time from location i, accounting for the travel time between these locations and the service duration at location i. Constraints (6.6) and (6.7) compute the total weight carried by the driver after visiting each pickup or delivery node. Additionally, we impose the condition $q_{o^+} = -q_{o^-}$ for every order o. Constraint (6.8) ensures that the number of orders carried by each driver does not surpass the upper capacity limit. We summarize the notation in Table 6.1.

Table 6.1: Notation.

Sets:	
$\mathcal{O}$	Set of orders;
$\mathcal{D}$	Set of drivers;
$\mathcal{T}$	Planning horizon;
$\mathcal{L}_1$	Set of pickup locations;
$\mathcal{L}_2$	Set of delivery locations;
Input parameters:	
s_i	Service time at location i;
q_i	Number of orders to pickup or delivery at location i;
cap	driver capacity;
c_{ij}	Travel distance between location i and location j;
p^o	The promised delivery time of order o;
λ	Unit penalty of delays for served orders;
μ	Unit penalty of unserved (cancelled) orders;
Decision variables:	
x_{ijd}	Binary variable, 1 if the driver d visit location j directly after location i, 0 otherwise;
Intermediate variables:	
t_{id}	Arrival time when driver d at location i;
w_{id}	Number of carrying orders when driver d departs location i;
f^o	The time when order o is delivered;
u^o	Binary variable, 1 if order o is cancelled by the customers, and 0 otherwise;
c_d	Total travel distance of driver d;

6.3 Model Formulation

This section is devoted to introducing our multi-agent MDP model, where each individual order serves as an agent.

In the context of our problem, it seems most logical to conceive of the platform as the decision-maker. The platform, acting as a single centralized agent, determines which orders are allowed to enter the matching pool. However, the nature of this modeling approach significantly complicates reinforcement learning tasks due to the dynamic nature of action spaces. In on-demand food delivery service, the number of orders arriving in each dispatching time interval (e.g., $(t-1, t]$) changes over time. Correspondingly, we need to maintain a dynamic action space, names $\mathcal{A}_t$. However, representing a dynamic action space in the agent's learning algorithm can be challenging. Since standard RL algorithms are typically designed for static action spaces, so modifications or entirely new

approaches may be necessary to handle dynamic scenarios. Given the set of orders waiting for entering the matching pool at time t is $O_t = \{o_t^j, o_t^{j+1}, \cdots, o_t^{j+k}\}$, for example, as shown in the left side of the Figure 6.3, at the decision step t, an action of the agent is to select a subset of orders a_t from all k orders. Accordingly, we define the action space $\mathcal{A}_t = \{a_t | a_t \in O_t\}$, which contains all possible order subsets. Note that the size of the combinatorial space $\mathcal{A}_t$ is 2^k, which grows exponentially with the number of accumulated orders k in time interval $(t-1, t]$. This limitation hinders the application of general reinforcement learning algorithms to large-scale problems, as these algorithms necessitate an explicit representation of all actions and exploration across a large action space. To accommodate a large action space while maintaining tractable learning, we factorize the combinatorial action space $\mathcal{A}_t$ into elementary actions along each order dimension. Specifically, as illustrated on the right side of Figure 6.3, we conceptualize each order as an individual agent. The elementary action for order i at decision time t is denoted as a_t^i. This implies that order i has two options: either enter the matching pool for dispatching a driver for delivery, or wait until the next decision time. Using this representation, any subset of orders can be expressed as $a_t = \cup_{i=j}^{i=j+k} a_t^i$. Consequently, our task transforms into learning policies for binary decisions corresponding to each order. This approach allows for the exploration of a large action space through the traversal of binary action spaces, which increase linearly with the number of orders.

We then present the details of our Multi-Agent Deep Q-Learning Network (**Dynamic-M-DQN**) algorithm for dynamic order dispatching policy learning as well as the driver routing (**DR**) algorithm.

6.3.1 MDP Model

Our multi-agent MDP model is represented by a tuple consisting of the agents, states, state transitions, and rewards, defined as follows:

Agent. We consider each customer order as an individual agent, endowed with the decision-making capability to opt for entry into the matching pool. The planning hori-

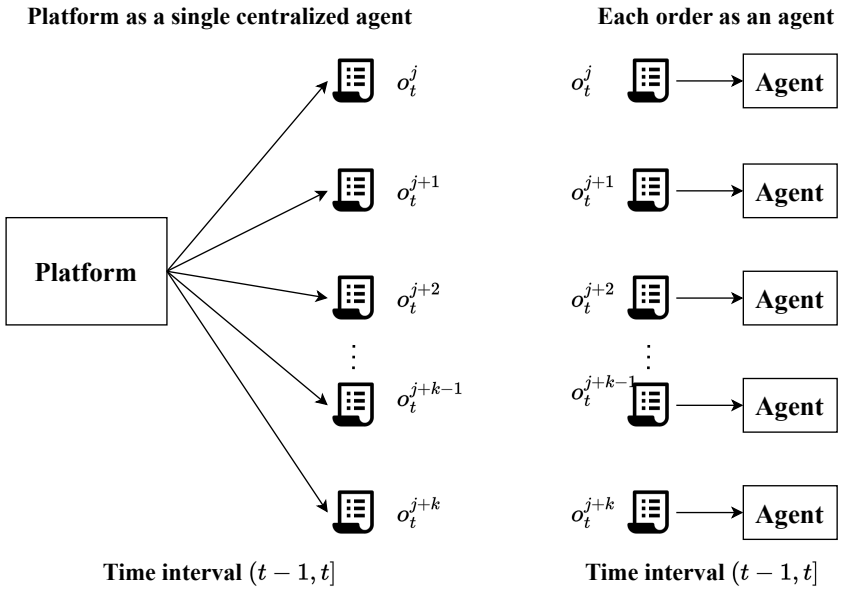


Figure 6.3: Comparison of two different action representations: (a) platform as a single centralized agent; and (2) each order as an agent.

zon, denoted as $\mathcal{T}$, encompasses a total of T dispatch times and $|N|$ agents. Agent activation occurs upon receiving a customer request from a restaurant and is randomly initiated by the environment, in accordance with the underlying demand distribution. The lifecycle of an agent commences when a customer places a request on the platform and concludes upon one of two events: (1) successful pick-up by a driver or (2) cancellation by the customer during the queueing period, resulting in non-entry into the matching pool. The count of agents in each time interval, represented as N_t for $t \in \mathcal{T}$, varies across different time intervals. Serving as a centralized meta-agent, the platform assumes the responsibility of making matching decisions on behalf of all agents.

State. At each decision time $t \in \mathcal{T}$, the state $\mathbf{s}_t$ is defined. we distinguish between two types of states: the global states $s_{t,\text{global}}$ which provide comprehensive demand and supply information for the entire environment and are shared by all agents at dispatch time t, and the local states $s_{t,\text{local}}$ which are specific to each individual agent and vary among them at the same dispatch time. Assuming the study area is divided into $|\mathcal{M}|$ small hexagons, the global states in our model encapsulate four types of spatio-temporal supply-demand features: (1) the count of pending orders in the matching pool within

each hexagon; (2) the available service capacity in each hexagon; (3) the origin-destination distribution of all pending orders within the study area; and (4) the pair of current and destination hexagons for each driver operating within the study area. Consequently, the dimension of the global state vector is $|\mathcal{M}| \times (2|\mathcal{M}| + 2)$. The local states, on the other hand, comprise three elements: (1) the origin (restaurant) and destination (customer's location) of each order; (2) the estimated increase in travel distance for dispatched drivers, as calculated by the routing algorithm; and (3) the time elapsed since the customer placed the order. It is evident that the local-view state varies among agents, each characterized by a dimensionality of $2|\mathcal{M}| + 2$. Therefore, the state for agent i at the decision time t can be expressed as $s_t^i = [s_{t,\text{global}}, s_{t,\text{local}}] \in \mathbb{R}^{(|\mathcal{M}|+1)\times(2|\mathcal{M}|+2)}$. However, if the study area is extensive (indicated by a large $|\mathcal{M}|$), the dimension of the global states $s_{t,\text{global}}$ could be further reduced. In such cases, it may be advantageous to include only features (3) and (4) in the global states. Consequently, the dimension of s_t^i would be $2 \times |\mathcal{M}|^2 + 2|\mathcal{M}| + 2$. Additionally, given the typical structure of on-demand food delivery services, where each order usually involves customer locations and restaurants in close proximity, a divide-and-conquer strategy is feasible. This approach entails initially dividing the area into smaller sub-areas based on restaurant service areas, and then applying our RL framework for order dispatching within each sub-area.

Action. At each decision time t, every agent i possesses two feasible actions represented as $a_t^i = \{0, 1\}$. Specifically, $a_t^i = 1$ signifies that agent i opts to join the matching pool, thereby becoming eligible for dispatch to a driver. Conversely, $a_t^i = 0$ indicates that agent i chooses not to participate in the matching pool and instead awaits the subsequent matching time $t + 1$.

Reward. The reward function is specifically designed to ensure alignment with the established optimization objective function, which aims to balance the trade-offs among driver travel distance, order delays, and the number of unserved orders. Let e_i denote the last dispatch time for agent i in the queue (i.e., orders that have not been picked up or cancelled). The reward for agent i at time t is denoted as r_t^i. The following conditions are considered:

- For $t < e_i$, if agent i decides not to enter the matching pool, a constant penalty is incurred: $r_t^i = -\Delta t$. This penalty is imposed due to the existence of a promised delivery time for each order, and customers generally prefer to receive their orders promptly.

- If agent i is cancelled by the customer after a prolonged waiting time, the reward is calculated as: $r_t^i = -\Delta t - r_{\text{cancel}}$. Here, $-r_{\text{cancel}}$ represents a constant negative penalty reward assigned to cancelled orders.

- When $t = e_i$, indicating that agent i enters the matching pool and is subsequently picked up by a driver, the reward is determined as: $r_t^i = r_{\text{match}} - \frac{d_{i,j}}{v}$. In this case, r_{match} represents a constant positive reward for successful matching, $d_{i,j}$ denotes the estimated increase in travel distance for the matched driver j during the order pickup and delivery process, and v represents the driver speed.

Here, the values of r_{cancel} and r_{match} may vary based on the attributes of a customer order, such as the restaurant, food value, and expected delivery time. However, to ensure fairness, all agents receive the same constant reward $-r_{\text{cancel}}$ or r_{match}, if an order is cancelled or dispatched. Intuitively, a large value of r_{match} implies that the platform prioritizes immediate order dispatch to drivers, giving less importance to the potential increase in driver travel distances and order delays. And a small $-r_{\text{cancel}}$ suggest that the platform aims to fulfill as many orders as possible.

We further consider the trade-offs among driver travel distance, order delays, and the number of unserved orders under two scenarios: (1) If an order is eligible for the matching pool, the driver's travel distance is still a crucial factor. To account for this, we set $r_t^i = r_{\text{match}} - \frac{d_{i,j}}{v}$. Here, if $d_{i,j}$ is significantly large, the benefit of dispatching order i reduces, potentially becoming negative. (2) If an order is postponed, meaning it does not enter the matching pool, we consider the trade-off between postponement and cancellation. In this scenario, we define $r_t^i = -\Delta t$ for postponement and $r_t^i = -\Delta t - r_{\text{cancel}}$ for cancellation. A large value of $-r_{\text{cancel}}/\Delta t$ suggests that the platform is striving to delay the allocation

of orders to maximize potential benefits on saving travel distance and order delays, while also attempting to minimize the likelihood of orders being canceled by customers.

6.3.2 Reinforcement Learning Dispatch Algorithm

In the field of sequential decision-making, Reinforcement Learning (RL) has emerged as a key framework, demonstrating significant proficiency in addressing complex challenges in transportation research [34].

Background

In the context of RL, an agent engages with an environment, executing actions and acquiring rewards, with the objective of maximizing the cumulative rewards $R_t = \sum_{k=t}^{T} \gamma^{k-t} r_k$ over time. In this subsection, our focus centers on the design of a RL algorithm for determining the incorporation of customer orders into the matching pool across time periods $t \in \mathcal{T}$. To accomplish this, we employ a deep RL methodology to ascertain the optimal dispatching timing for individual customer orders, leveraging real-world historical food delivery data. Our approach entails adopting a value-based, model-free reinforcement learning technique, utilizing a deep Q network (DQN) with experience replay to approximate the action-value function $Q(\mathbf{s}, \mathbf{a})$. For every dispatch decision time t, the action-value function, denoted as $Q_\pi(s_t, a_t)$, signifies the expected reward associated with selecting action a_t within state s_t under a specified policy π. The principal aim of an agent is to maximize the anticipated reward. More precisely, the agent endeavors to ascertain the optimal action-value function, labeled as $Q_*(s_t, a_t)$, that fulfills the condition:

$$Q_*(s_t, a_t) = \max_{\pi} Q_\pi(s_t, a_t) \quad \forall s_t \in \mathbf{s}, a_t \in \mathbf{a}, \tag{6.11}$$

for any policy π. Accordingly, the agent chooses the action $a_t = \textbf{argmax} Q_*(\mathbf{s}, \mathbf{a})$, aiming to optimize its decision-making process at each iteration.

Rather than training distinct DQN networks, each with separate experience replay and target networks, for individual intelligent agents (orders), we introduce a centralized

learning framework called **Dynamic-M-DQN**. This framework employs a single centralized DQN to make dispatch decisions at an order-level. There are two principal rationales for adopting a centralized training framework. Firstly, the variable number of agents presents a substantial challenge when implementing decentralized approaches that allocate an individual neural network to each agent. This challenge stems from the inherent uncertainty in predefining the count of neural networks to be created. Secondly, a centralized training framework streamlines the coordination and information exchange among agents, enabling them to collaboratively learn and enhance their decision-making procedures. Numerous precedent investigations, such as [87], [57], [103], and [90], have delved into varied methodologies to expand single-agent RL algorithms into multi-agent contexts. These methodologies encompass the inclusion of communication channels among agents and the adaptation of reward functions to accommodate the actions of fellow agents.

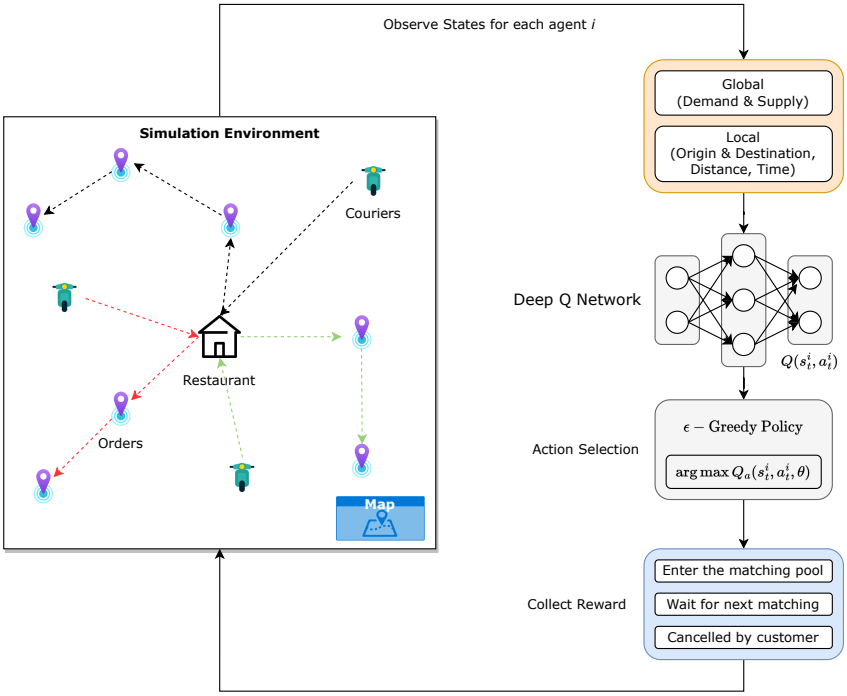


Figure 6.4: Dynamic-M-DQN algorithm for order dispatching with dynamic time intervals.

Algorithm 4 Dynamic Multi-agent Deep Q Learning Algorithm

procedure DYAMIC-M-DQN
  Initialize the agent specific replay buffer $\mathcal{B} = \{\mathcal{B}^1, \mathcal{B}^2, \cdots, \mathcal{B}^{|N|}\}$;
  Initialize the Q network with weights $\boldsymbol{\theta}$;
  for epoch $= 1 \rightarrow |E|$ **do** ▷ Start of an epoch
    Initialize the environment with joint state s_0;
    Initialize the replay buffer $\mathcal{B}^i$ for each agent i;
    for dispatch decision time $t \in \mathcal{T}$ **do**
      Collect information for each agent $i \in N_t$ that arrivals before time t;
      for $i = 1 \rightarrow N_t$ **do** ▷ Choose an action
        Observe the states s_t^i for each agent i;
        Take action a_t^i according to the ϵ-greedy policy with $Q(s_t^i, a_t^i|\boldsymbol{\theta})$;
      end for
      end for
      Run the simulation with the inputs of joint states $\mathbf{s}_t = \{s_t^1, s_t^2, \cdots, s_t^{N_t}\}$ and actions $\mathbf{a}_t = \{a_t^1, a_t^2, \cdots, a_t^{N_t}\}$, observe the joint state $\mathbf{r}_t = \{r_t^1, r_t^2, \cdots, r_t^{N_t}\}$ and new states $\mathbf{s}_{t+1} = \{s_{t+1}^1, s_{t+1}^2, \cdots, s_{t+1}^{N_t}\}$. For each agent i, we denote $e_t^i = 1$ if it reaches the terminate state (picked up or cancelled), 0 otherwise; ▷ State transition
      for agent $i = 1 \rightarrow N_t$ **do**
        Append $(s_t^i, a_t^i, r_t^i, s_{t+1}^i, d_t^i)$ to the replay buffer B^i of agent i;
        Append $\mathcal{B}^i$ to $\mathcal{B}$; ▷ Update replay buffer
      end for
      end for
    end for
    if size of the replay buffer $|\mathcal{B}| \geq b_{\min}$ **then**
      Randomly sample M complete trajectories $(s_t^i, a_t^i, r_t^i, s_{t+1}^i, e_t^i)$ for each agent i from the replay buffer $\mathcal{B}$ from all agents;
      Update the weights $\boldsymbol{\theta}$ with the minimization of target loss $\mathcal{L}(\boldsymbol{\theta})$:
      $\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} - \alpha \cdot \mathcal{L}(\boldsymbol{\theta}) \cdot \nabla_{\boldsymbol{\theta}} Q(s_t^i, a_t^i|\boldsymbol{\theta})$ ▷ Weight updates
    end if
    end for
  end for
  end for
end procedure

RL Algorithm

We initialize a neural network with parameter $\boldsymbol{\theta}$ to approximate the action value function $Q(\mathbf{s}, \mathbf{a}|\boldsymbol{\theta})$. For every decision time $t \in \mathcal{T}$, we gather the transitions $(s_t^i, a_t^i, r_t^i, s_{t+1}^i)$ for each agent i and store them within agent-specific replay buffers denoted as $\mathcal{B} = \{\mathcal{B}^1, \mathcal{B}^2, \cdots, \mathcal{B}^{|N|}\}$, where $\mathcal{B}^i$ signifies the replay buffer for agent i. The training process aims to iteratively update the action-value function $Q(\mathbf{s}, \mathbf{a}|\boldsymbol{\theta})$ in order to find the optimal action-value function $Q_*(\mathbf{s}, \mathbf{a}|\boldsymbol{\theta}^*)$ that maximize the expected long-term reward.

Table 6.2: Notation for driver routing algorithm.

Sets and Variables:	
$\mathcal{R}$	Set of restaurants
l_d	Current location of driver d
ω_d	Ongoing carrying orders by driver d
$c_{o,d}$	Distance between pickup location o^+ and driver's current location l_d
RT_d	Planned delivery route for driver d
$\mathcal{O}_{\text{pick}}$	Orders that have been dispatched and are being picked up by the drivers

The update rule of action-value function of each agent i is defined as follows:

$$Q(s_t^i, a_t^i|\boldsymbol{\theta}) \leftarrow Q(s_t^i, a_t^i|\boldsymbol{\theta}) + \alpha \cdot \left[r_t^i + \gamma \max_{a \in \mathbf{a}} Q(s_{t+1}^i, \mathbf{a}|\boldsymbol{\theta}) - Q(s_t^i, a_t^i|\boldsymbol{\theta})\right], \quad (6.12)$$

here, α stands as the learning rate, dictating the magnitude of each update's progression. Furthermore, a target network is integrated to periodically adjust the Q values, fostering greater stability in the training procedure by curbing transient fluctuations. The parameter $\boldsymbol{\theta}$ undergoes updates via the minimization of a loss function, delineated as the disparity between the anticipated Q values and the projected Q values. The formulation of the loss function $\mathcal{L}_{\boldsymbol{\theta}}$ is as follows:

$$\mathcal{L}_{\boldsymbol{\theta}} = \mathbb{E}_{(s_t^i, a_t^i, r_t^i, s_{t+1}^i)} \left[\left(r_t^i + \gamma \cdot \max_{a_{t+1}^i} Q(s_{t+1}^i, a_{t+1}^i|\boldsymbol{\theta}^-) - Q(s_t^i, a_t^i|\boldsymbol{\theta}) \right)^2 \right], \quad (6.13)$$

where $\gamma \in (0, 1]$ is a discount factor, and $\boldsymbol{\theta}^-$ denotes the network parameters of the target network.

The complete Dynamic-M-DQN algorithm for order dispatching is depicted in Figure 6.4, with detailed steps summarized in Algorithm 4.

6.3.3 Driver Routing

This study primarily centers around order dispatching, yet routing decisions play a vital role in the overall service process at a lower level. In recent years, there has been extensive attention given to the problem of route optimization in food delivery. The routing problem is first formalized as meal delivery routing problem in [104], which is a variant of dynamic pickup and delivery problem and has been studied in recent decades [97]. Given the high demand for timely food delivery, exact solution methods [142] are often

computationally intensive and impractical for addressing driver routing problems. Consequently, conventional heuristic algorithms [151, 129] are typically employed in prominent food delivery platforms (e.g., Meituan) as a means of swiftly obtaining approximate solutions.

In this subsection, we present a comprehensive account of our driver routing algorithm, as depicted in Algorithm 5. The symbols and terminology employed in this algorithm are detailed in Table 6.2.

Dispatching rules and assumptions for service operations are as follows:

- **Dispatch fairness**: If the number of orders carried by driver d is large/small (i.e., Ω_d is large/small), s/he will be less/more likely dispatched new orders;
- **Order re-dispatch**: Orders that have not been picked up within a specified time frame (e.g., 10 minutes) but have been assigned a driver can be re-dispatched to a new driver;
- **Driver routing behavior**: Following order dispatch, drivers deliver their carried orders by solving an open Traveling Salesman Problem (TSP) to optimize their route;
- **Order cancellation**: If a newly generated order cannot find an available driver within a certain time (e.g., 15 minutes), it is assumed that the customer becomes impatient and cancels the order.

The first rule is aimed at ensuring fairness in dispatching by preventing certain drivers from being overloaded with a disproportionately large number of orders, while others only receive a few. The second rule serves to enhance the flexibility of the dispatch algorithm by allowing for order re-dispatch, which has the potential to reduce order pickup time. Third, we assume that drivers always follow the shortest one-way delivery travel route from the restaurant to one of the customer locations. This route is derived by solving an open TSP, where the driver does not return to the starting location. [1] Lastly,

[1] It is important to note that in real-world scenarios, due to complexities such as left turns in intersec-

customers have the option to cancel orders if the platform fails to find a driver within a specific time period.

Algorithm 5 Driver Routing Algorithm

Input: $\mathcal{O}_{\text{pool}}, \mathcal{D}, \mathcal{R}, \omega_d$
Output: RT'_d

procedure ROUTING($\mathcal{O}_{\text{pool}}, \mathcal{D}$)
  Initialize dispatched and being picked up orders $\mathcal{O}_{\text{pick}} = \emptyset$ ▷ Update driver's orders
  for $d \in \mathcal{D}$ **do**
    if c is going to pick up order $o \in \mathcal{O}_{\text{pool}}$ **then**
      Update $\mathcal{O}_{\text{pick}} \leftarrow \mathcal{O}_{\text{pick}} \cup o$
    end if
  end for
  for $o \in \mathcal{O}_{\text{pool}}$ **do** ▷ Greedy insertion
    if $o \in \mathcal{O}_{\text{pick}}$ **then**
      Continue
    end if
    Initialize the available drivers set $\mathcal{D}_{\text{avail}} = \emptyset$
    for $d \in \mathcal{D}$ **do**
      if $|\omega_d| < cap$ (driver capacity, e.g., 10 orders) **then**
        $\mathcal{D}_{\text{avail}} \leftarrow \mathcal{D}_{\text{avail}} \cup c$
      end if
    end for
    Assign order o to driver $d \in \mathcal{D}_{\text{avail}}$ with maximum available capacity
    do greedy insertion
    Insert pickup node o_+ and delivery node o_- to route RT_d at positions with
    minimum distance increased
    Update $\mathsf{RT}_d \leftarrow \mathsf{RT}_d \cup (o_+, o_-)$
  end for
  for $d \in \mathcal{D}$ **do** ▷ Driver's delivery route reconstruct
    if $\omega_d \neq \emptyset$ **then**
      Get driver c current location l_d and delivery location o_- for order $o \in \omega_d$
      Re-schedule the planned route RT_d by solving a open TSP problem with
      unvisited locations for carried orders ω_d
      Update the delivery route RT_d
    end if
  end for
  return new delivery plan route RT'_d for $d \in \mathcal{D}$
end procedure

tions, the actual travel distance for order delivery tends to be equal to or greater than the travel distance derived from the open TSP solution.

6.4 Simulator and Dataset

In this section, we introduce our custom-designed simulator that simulates our process of the food delivery service. Subsequently, we present a real-world delivery dataset obtained from a crowd-sourcing food delivery platform in Singapore for training our Dynamic-M-DQN algorithm.

Simulator. To facilitate the training of Dynamic-M-DQN (given in Algorithm 4) and to generate representative samples that mimic real-world scenarios in food delivery service, we designed a simulator calibrated with real data. The simulator consists of multiple modules, encompassing order generation, information collection, the order dispatching algorithm, and updates for both order and driver statuses. Five distinct states have been defined for each agent and order, specifically denoted as "Generated", "Dispatched", "Ongoing", "Completed", and "Cancelled". The state denoted as "Generated" signifies that the customer has initiated an order placement, yet it remains pending for dispatch to a driver for pickup. "Dispatched" indicates that the order has entered the matching pool, though pickup by a driver has not transpired as of yet. "Ongoing" signifies that a driver has collected the order and is presently in the process of delivery. Ultimately, "Completed" denotes successful delivery to the customer, while "Cancelled" denotes the customer's decision to rescind the order. The comprehensive simulation framework is depicted in Figure 6.5, comprising five key constituents: input data, agent properties, platform operational procedures, optimization algorithms, and output outcomes.

During each order dispatching interval $(t, t+1]$, the subsequent procedures are conducted: (a) novel order requests are generated for individual restaurants and categorized as "Generated"; (b) the implementation of Dynamic-M-DQN is employed to ascertain the eligibility of orders entering the matching pool, whereby those qualifying are designated as "Dispatched"; (c) execution of the driver routing algorithm takes place, with orders collected designated as "Ongoing"; (d) orders successfully dispatched, collected, and delivered are updated to "Completed," while customer-canceled orders are identi-

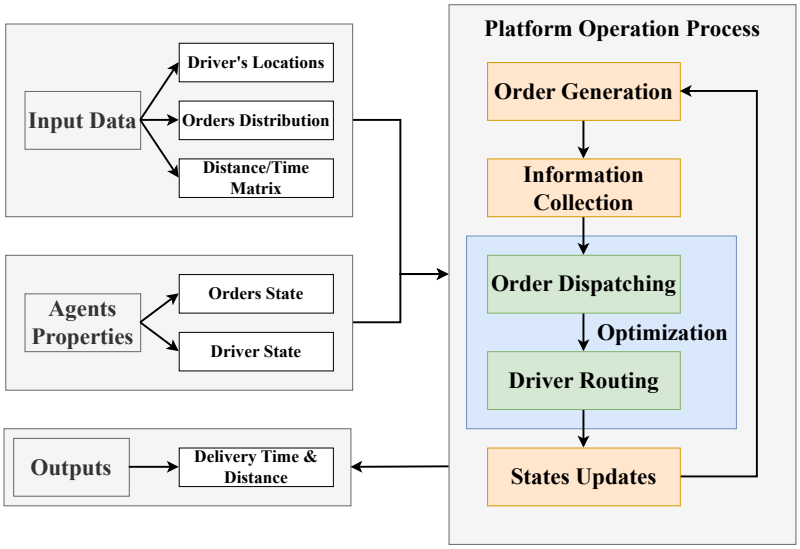


Figure 6.5: Simulation Framework.

fied as "Cancelled"; and (e) the predetermined delivery routes for drivers are revised. It's noteworthy that the collective state, encompassing both global and local states, is initialized before the execution of Dynamic-M-DQN in step (b).

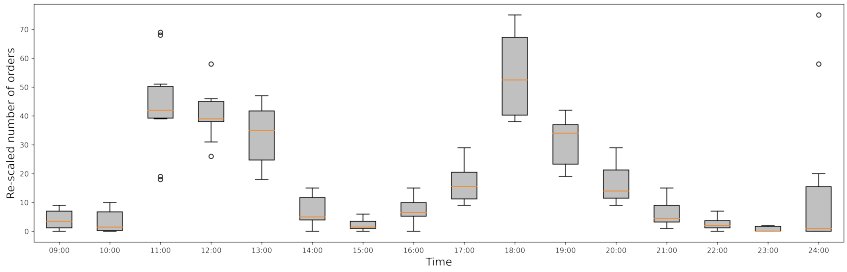


Figure 6.6: Distribution of number of orders over hours.

Real-world Dataset. The dataset employed in the numerical experiments originates from a crowd-sourced food delivery platform in Singapore [2]. The dataset encompasses approximately 80, 000 order records and more than 30, 000 customers spanning 8 months (October 2020 to May 2021). Each delivery record encompasses a wealth of order and driver details, encompassing order pickup and delivery locations, delivery distance, order acceptance time, driver identification, and the corresponding fee. Figure 6.6 illustrates the hourly order distribution spanning from $09:00$ to $24:00$. Evidently, the order volume exhibits substantial augmentation during the peak intervals at $11:00$ and $18:00$.

[2] The dataset used in this study was contributed by an industry collaborator who has chosen to remain anonymous.

Table 6.3: Hyper-parameter settings for **Dynamic-M-DQN**.

Parameters	**Values**	**Parameters**	**Values**
Learning rate	$2e-3$	Hidden layer neurons	$512, 256, 128$
Discount γ	0.99	Number of epochs	100
Epsilon ϵ	$0.1 \sim 0.9$	Dispatches in one episode	96
Buffer size $\lvert\mathcal{B}\rvert$	500	Batch size	64
Target update	10	State dimension	$2 \times \lvert\mathcal{M}\rvert^2 + 2\lvert\mathcal{M}\rvert + 2$

For the purpose of enabling in-depth exploration, we have proactively provided access to the tailored simulator and exemplar test data utilized within this research to the wider community[3].

6.5 Experiments

In this section, we present a set of experiments to evaluate the performance of our two-stage learning-based optimization approach. We generate restaurants and customer orders to be delivered following the patterns of our partner dataset. We also account for the random ready time for orders in our simulations. We make the assumption that the food preparation time for each restaurant follows distinct gamma distributions, characterized by different shape parameters α and a scalar parameter β. This choice is motivated by the common use of gamma distributions for modeling waiting times in on-demand food delivery platforms [121, 45]. We set the promised delivery time to 45 minutes and the velocity of the drivers is established at 20 km/h. The simulations are executed employing Python version 3.9 and leveraging the PyTorch library [94].

Parameters Settings. Table 6.3 presents the hyper-parameter configurations for our **Dynamic-M-DQN** algorithm. An epoch is delineated as the complete temporal span $\mathcal{T}$, ranging from $6:00$ AM to $10:00$ PM. This epoch is partitioned into 96 dispatching time intervals, each lasting 10 minutes, predicated on the order arrival rate within our dataset.

Evaluation Metrics. Our experimental focus is on the minimization of a weighted sum that encompasses the total travel distance of drivers, cumulative order delays, and

[3]https://anonymous.4open.science/r/Food-Delivery-Simulator-37DC/

the count of customer-cancelled orders. To effectively portray delivery performance, we employ three evaluation metrics: (1) total travel distance of drivers (C_{total}) in kilometers; (2) cumulative order delays (T_{delay}) in seconds; and (3) count of customer-initiated cancellations (N_{cancel}) stemming from extended waiting periods. The objective is defined as follows:

$$\text{Obj} = \frac{D_{\text{total}}}{|D|} + \frac{T_{\text{delay}}}{3600} \cdot \lambda + N_{\text{cancel}} \cdot \mu \tag{6.14}$$

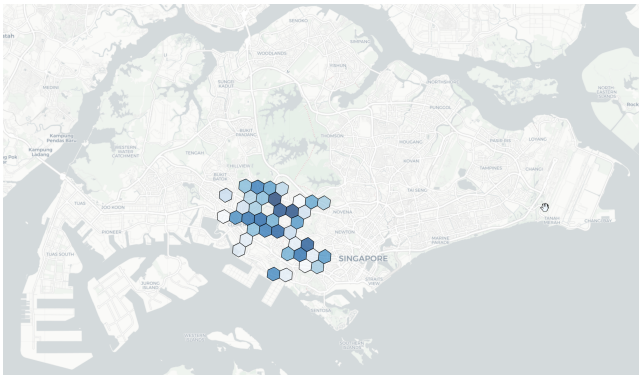
Our research focuses on a multi-objective optimization challenge, aiming to achieve a balance among three critical elements: the delivery cost for drivers, as measured by the average travel distance; the platforms' revenue, reflected by the number of unserved orders; and the user experience, demonstrated through the extent of order delays. Importantly, platforms have the flexibility to select different values for λ and μ, allowing them to tailor the evaluation of model performances based on their prioritization of these two objectives.

Here, a penalty value of $\lambda = 100$ is employed for delays, as customers frequently hold elevated delivery time expectations. Extended delays may prompt customers to depart the platform in search of alternatives. Furthermore, a significant penalty value of $\mu = 500$ is administered to mitigate the concern of customer cancellations arising from extended waiting durations for available drivers.

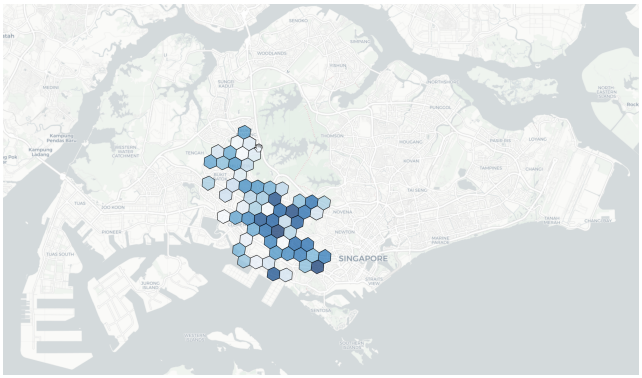
Main Results. Our investigation encompasses three datasets varying in scale: a small dataset, comprising a single restaurant with more than 450 orders and 15 drivers; a medium dataset, encompassing two restaurants with a total of over 700 orders and 40 drivers; and a large dataset, involving three restaurants and exceeding 1500 orders, with a driver pool of 100. Figure 6.7 presents a visual representation of customer order locations for each dataset. The degree of shading in the hexagonal areas corresponds to the number of orders received, with darker shades indicating higher order volumes.

In this study, we analyze the performance of the proposed **Dynamic-M-DQN** coupled with the **DR** algorithm. A comparative evaluation is conducted between this approach and the following commonly employed dispatch strategies and routing algo-

(a) Small instance (450 orders, 15 drivers)



(b) Medium instance (700 orders, 40 drivers)



(c) Large instance (1500 orders, 100 drivers)

Figure 6.7: Density of customer orders for: (a) small, (b) medium, and (c) large instances.

rithms:

- **FIX**: Dispatch strategy. In this approach, the order dispatching time interval remains constant for all orders (e.g., 5 minutes), mirroring the default dispatching mode found on numerous food delivery platforms like Meituan [151] and Grubhub [142];

- **MGI**: Routing algorithm. We adopt the Greedy heuristic ([86]), a well-regarded routing algorithm for addressing the dynamic pickup and delivery problem prevalent in the industry. Specifically, our use of **MGI** refers to the modified greedy insertion which is actively employed online by Meituan-Dianping ([151]), China's largest food delivery platform [4];

- **H-RL**: Dispatch strategy. This strategy employs a hierarchical reinforcement learning based optimization framework, as proposed by [78]. In this approach, the upper-level agent dynamically decomposes the dynamic pickup and delivery problem (DPDP) into a sequence of sub-problems for optimization. Meanwhile, the lower-level agent selects operators to enhance the resultant solution. We utilize the term **H-RL** to represent the DQN model situated in the upper-level for the purpose of order dispatch.

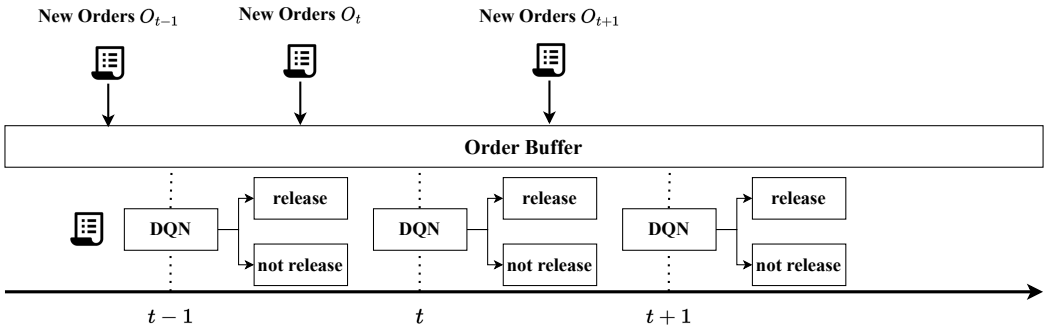


Figure 6.8: Illustration of the **H-RL** policy.

[4]Meituan-Dianping is a Chinese shopping platform in China for locally found consumer products and retail services including entertainment, dining, delivery, travel and other services. https://www.meituan.com/

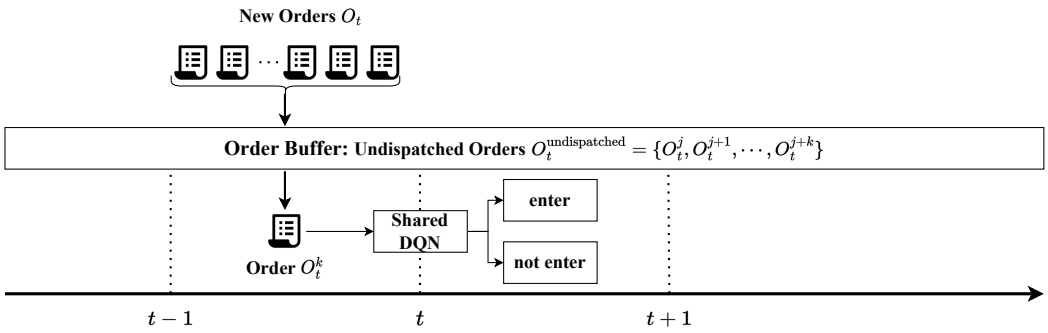


Figure 6.9: Illustration of our **Dynamic-M-DQN** policy.

To more clearly illustrate the differences between **H-RL** and our **Dynamic-M-DQN**, we refer to the examples depicted in Figure 6.8 and Figure 6.9.

The **H-RL** algorithm necessitates maintaining an order buffer to store newly generated orders, which are then dispatched in batches at predetermined intervals. As depicted in Figure 6.8, orders received before the decision time $t-1$ are labeled as O_{t-1}. The trained agent with Deep Q-Network (DQN) is responsible for deciding whether to release all these orders into the matching pool simultaneously or defer them to the next decision point, t. In other words, all accumulated orders O_{t-1} that arrive before $t-1$ are either collectively released to the matching pool at $t-1$ or uniformly postponed to the subsequent decision time t based on the DQN agent's decision. While in our **Dynamic-M-DQN** approach, illustrated in Figure 6.9, each order is considered an individual agent. These orders collectively utilize a centralized Deep Q-Network (DQN) to determine their entry timing into the matching pool. It is important to highlight that our approach offers enhanced flexibility compared to **H-RL**, as it facilitates decision-making at the individual order level, in contrast to **H-RL**'s batch-order decision-making process.

Our comparative analysis includes our proposed approach **Dynamic-M-DQN + DR** alongside the following policies: (1) **H-RL + DR**; (2) **FIX + DR**; and (3) **FIX + MGI**.

As a baseline, we will employ the **FIX + MGI** approach. We quantify several metrics, specifically the total travel distance C_{total}, cumulative delays T_{delay}, and the count of canceled orders N_{cancel}, for each policy, correspondingly. These measurements are employed

Table 6.4: Results summary of the three approaches on different sized datasets.

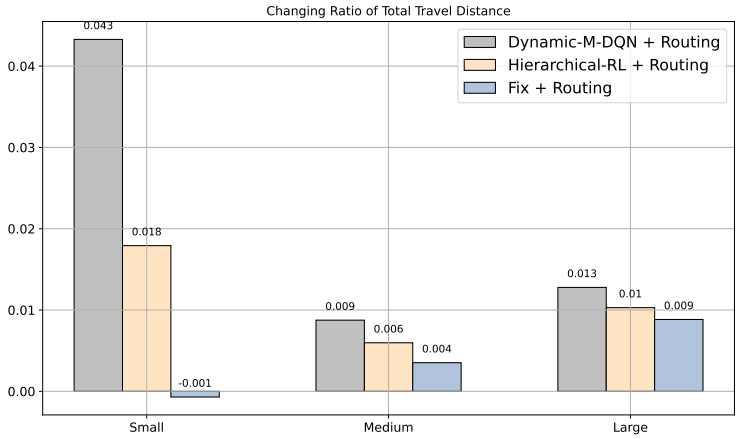
Instance	Approach	C_{total}	T_{delay}	N_{cancel}	$\mathbb{M}_{\text{Rat}}$
Small	Dynamic-M-DQN + DR	2,130.37	191	0	0.825
450 orders	H-RL + DR	2,186.85	896	0	0.243
15 drivers	FIX + DR	2228.31	1,113	0	0.064
	FIX + MGI	2226.75	1,191	0	0
Instance	**Approach**	C_{total}	T_{delay}	N_{cancel}	$\mathbb{M}_{\text{Rat}}$
Medium	Dynamic-M-DQN + DR	3,091.55	562	0	0.758
700 orders	H-RL + DR	3,036.23	1,221	0	0.474
40 drivers	FIX + DR	3,107.87	1,783	0	0.231
	FIX + MGI	3,118.85	2,320	0	0
Instance	**Approach**	C_{total}	T_{delay}	N_{cancel}	$\mathbb{M}_{\text{Rat}}$
Large	Dynamic-M-DQN + DR	6,666.08	10,645	5	0.153
1,500 orders	H-RL + DR	6,682.91	11,016	6	0.123
100 drivers	FIX + DR	6,692.72	11,136	6	0.114
	FIX + MGI	6,752.39	12,581	6	0

to ascertain the effectiveness of our proposed model. The comprehensive experimental results in Table 6.4 illustrate that employing the **Dynamic-M-DQN + DR** approach results in a substantial reduction in cumulative delays, along with a moderate decrease in total travel distance for drivers. Interestingly, the weighted sum objective value, as computed using equation (6.14), exhibited reductions of 82.5%, 24.3%, and 6.4% relative to the baseline approach for **Dynamic-M-DQN + DR**, **H-RL + DR**, and **FIX + DR**, respectively. In cases of both small and medium-sized instances, all customer orders can be fulfilled without any cancellations. In the case of large-sized instances, our **Dynamic-M-DQN + DR** approach stands out among all methods, demonstrating the lowest count of canceled orders.

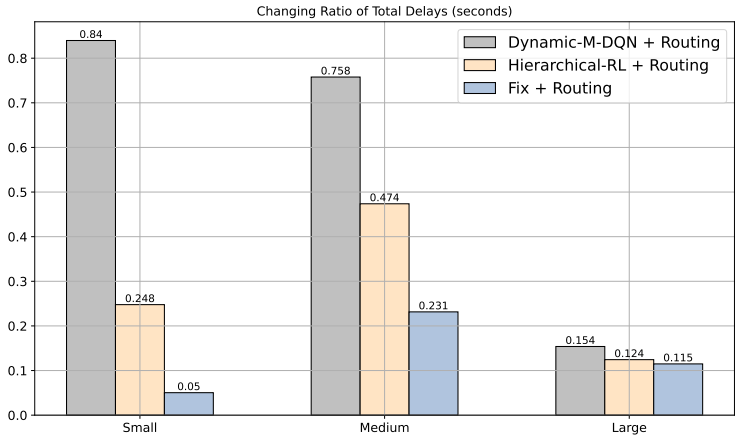
Next, we define $\mathbb{M}_{\text{approach}}$ as the changing ratio for a given evaluation metric between the selected approach and the baseline approach **FIX + MGI** as follows:

$$\mathbb{M}_{\text{Rat}} = \frac{\mathbb{M}_{\text{approach}} - \mathbb{M}_{\text{FIX + MGI}}}{\mathbb{M}_{\text{FIX + MGI}}} \tag{6.15}$$

Figure 6.10 illustrates the changing ratios of C_{total} and T_{delay} across all three instances. The results reveal the superiority of our **Dynamic-M-DQN + DR** approach over the alternative approaches for all instances. Our approach resulted in a reduction in the rate



(a)

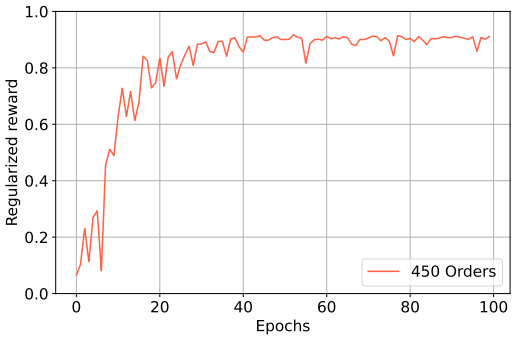


(b)

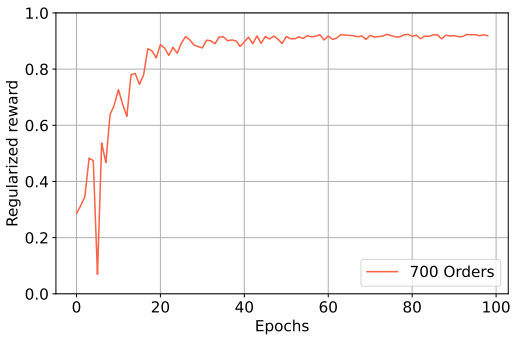
Figure 6.10: Comparison of experiments results: (a) Changing ratio of total travel distance, (b) Changing ratio of total delays against the baseline, for small, medium and large instances.

of total travel distance for drivers, with decreases ranging from 4.3% to 0.9% to 1.3%. Additionally, substantial reductions of 84.0%, 75.8%, and 15.4% were observed in total order delays for the small, medium, and large instances respectively. The findings suggest that dynamic optimization of the dispatching time interval for order dispatch can yield substantial enhancements in service. These enhancements manifest as noteworthy reductions in delays (up to 84.0%), accompanied by a slight decrease in delivery distances (up to 4.3%).

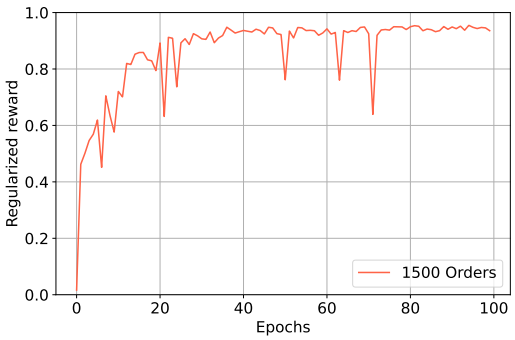
Figure 6.11 depicts the convergence patterns of the regularized mean rewards attained by our **Dynamic-M-DQN** model across datasets of varying sizes. Our observations suggest that our approach can establish a high-quality policy within around 40 epochs across



(a) Small instance (450 orders, 15 drivers)



(b) Medium instance (700 orders, 40 drivers)



(c) Large instance (1500 orders, 100 drivers)

Figure 6.11: Reward convergence curves of **Dynamic-M-DQN** on different sized datasets: (a) small, (b) medium, and (c) large instances.

all datasets, as substantiated by the converging training process. These outcomes underscore both the effectiveness and efficiency of our policy for dynamic dispatching time intervals, which effectively optimizes the order dispatching process within on-demand food delivery services.

6.5.1 Sensitivity Analysis in Terms of the Rewards and the Fix Dispatching Time

Next we present the main idea in reward design and look at the affects among different objectives. In the objective function, we consider the trade-offs among driver travel distance, order delays, and the number of unserved orders under two scenarios: (1) If an order is eligible for the matching pool, the driver's travel distance is still a crucial factor. To account for this, we set $r_t^i = r_{\text{match}} - \frac{d_{i,j}}{v}$. Here, if $d_{i,j}$ is significantly large, the benefit of dispatching order i reduces, potentially becoming negative. (2) If an order is post-

poned, meaning it does not enter the matching pool, we consider the trade-off between postponement and cancellation. In this scenario, we define $r_t^i = -\Delta t$ for postponement and $r_t^i = -\Delta t - r_{\text{cancel}}$ for cancellation. A large value of $-r_{\text{cancel}}/\Delta t$ suggests that the platform is striving to delay the allocation of orders to maximize potential benefits on saving travel distance and order delays, while also attempting to minimize the likelihood of orders being cancelled by customers.

Table 6.5: Evolution of the performance with different ρ.

Δt	r_{cancel}	ρ	C_{total}	T_{delay}	N_{cancel}
1	-100	100	3,095.00	638	0
10	-100	10	**3,091.55**	**562**	0
20	-100	5	3,143.02	1,963	0
50	-100	2	3,135.33	2,859	0

In light of these findings, we conducted additional experiments to refine the design of the reward function. Specifically, a sensitivity analysis was performed focusing on the ratio $\rho = -r_{\text{cancel}}/\Delta t$, which is pivotal in training our **Dynamic-M-DQN** method. According to the results presented in Table 6.5, our approach achieves the lowest total travel distance (C_{total}) and delay times (T_{delays}) when $\rho = 10$. Conversely, both the total travel distance and delays increase with a smaller ρ (e.g., $\rho = 5, 2$). This trend can be attributed to the fact that a lower ρ places greater emphasis on postponing rather than cancelling orders. Consequently, with a lower ρ, orders tend to enter the matching pool sooner, as postponement incurs a substantial negative reward, almost equivalent to cancellation. However, a very high ρ value (such as $\rho = 100$) does not necessarily equate to better performance. This is attributed to the tendency of orders to be excessively postponed with a high ρ, potentially missing optimal entry times into the matching pool, which could result in lower travel distances and delays. With a given λ and μ by the platform, if the ratio μ/λ is large, it indicates that the platform prioritizes serving more orders over minimizing order delays. To align with this objective function, a larger ρ should be adopted to discourage order cancellation.

Designing an optimal reward function for RL applications presents significant challenges, especially in the context of on-demand food delivery services where the environ-

ment is dynamic and complex. This complexity makes it difficult to define a reward function that is consistently effective and efficient across various scenarios. Additionally, when addressing multi-objective considerations, accurately reflecting the trade-offs among different objectives becomes an even more daunting task. For further insights into the diverse methodologies for designing reward functions in the successful application of RL algorithms, readers are referred to [24, 28, 122, 52].

At last, we evaluated the performance of the **DR** algorithm using a newly assembled dataset. This assessment encompassed various fixed time intervals between consecutive dispatch decisions. The results of this comparative analysis are presented in Table 6.6. Our observations suggest that extending the dispatching time interval correlates with an increment in the cumulative delivery distance covered by the drivers. Conversely, this adjustment leads to a substantial decrease in order delay times. Moreover, it is worth noting that all orders can be fulfilled without any cancellations in all scenarios.

Table 6.6: Comparison with different dispatching time intervals.

700 orders & 60 drivers	**Metrics**	**2 mins**	**5 mins**	**10 mins**	**15 mins**
DR	D_{total}	2,275.86	2,681.60	3,040.71	3,111.33
	T_{delay}	58,916	3,873	755	371
	N_{cancel}	0	0	0	0

6.6 Conclusion

This chapter introduces a novel two-stage framework designed for the optimization of order dispatching and driver routing within on-demand food delivery services. At the higher level, we present a novel multi-agent reinforcement learning approach named Dynamic-M-DQN. This method is employed to dynamically ascertain the optimal timing for including an order into the matching pool for pickup and delivery. At the lower level, a new routing algorithm is devised. This algorithm, while straightforward, proves to be highly efficient. It is responsible for strategically charting delivery routes for drivers, factoring in considerations such as fairness, driver behavior, and order re-dispatch. An in-depth evaluation of our proposed approach against benchmark systems is undertaken

through the customized simulator. This simulator accurately replicates processes including order generation, dispatch, and driver routing, alongside the utilization of real-world food delivery data from our industry collaborator. Substantial experimentation conducted within this simulator involves a thorough comparative analysis. Our approach is compared against simulated approaches that adhere to fixed dispatching time intervals, the prevalent MGI employed in driver routing, and hierarchical RL based approach. The novel approach we propose yields enhancements in objective metrics ranging from 15.3% to 82.5%, demonstrated across datasets of various sizes. Evaluation outcomes affirm the efficacy of our approach, showcasing a noteworthy reduction in average travel distance and substantial alleviation of delivery delays. Future endeavors involve the utilization of RL methodologies for the development of an end-to-end routing policy.

Chapter 7

Conclusion and Future Works

7.1 Concluding Remarks

Data-driven optimization for urban delivery problems has been receiving more and more attention in practice as well as in the research community. In this thesis, we discuss three important and challenging operational problems in urban logistics delivery. One concerns the collaborative vehicle routing problem in the urban last-mile delivery, while the other two focus on supply-demand management and dynamic order dispatch in on-demand food delivery service. Our research centers on the utilization of data-driven methodologies to tackle these challenges.

In Chapter 4, we propose a two-stage learning and optimization framework in which we study two practical problems arising in collaborative vehicle routing with multiple overlapping alliances: the request assignment problem, which involves assigning each request to an alliance with the lowest predicted delivery cost, and the vehicle routing problem, which involves creating the delivery plan for vehicles in each alliance. Unlike traditional approaches to managing collaborative vehicle routing problems [43], which directly use heuristic algorithms (e.g., adaptive large neighborhood search) to solve the routing problem under a centralized framework, the learning step introduces a cost prediction problem to reduce the computational complexity of the decisions in request assignment, a major concern in collaborative routing. We identify a set of promising features for de-

livery cost prediction and propose an efficient tabu search algorithm to solve the routing problem for each alliance after request assignment. Computational studies demonstrate the effectiveness of our learning and optimization framework in terms of both delivery cost and computational time.

Chapter 5 focuses on the supply and demand management problem with dynamic area size optimization. This chapter consists of designing a customer choice model for order estimation on the demand side, developing a machine learning model for predicting average order delivery times on the supply side, and integrating these two problems into the master optimization RASO model. The integrated problem is formulated as a Mixed-Integer Quadratically Constrained Programming (MIQCP) model, which can be efficiently solved by commercial solvers such as Gurobi. Computational experiments conducted with real-world data collected in Singapore have demonstrated that the proposed data-driven optimization framework can significantly improve the number of served orders and outperform benchmark methods commonly used in practice.

Chapter 6 deals with the order dispatch problem in on-demand food delivery services. In contrast to the data-driven methods employed in the previous two works, in this chapter, our aim is to optimize the dispatch time interval end-to-end with the objective of minimizing the weighted sum of the total travel distance and total delays using a reinforcement learning approach. We formulate the problem as a Markov Decision Process and propose a centralized multi-agent deep Q-network that streamlines the order dispatching process, allowing orders to join the matching pool for pickup and delivery. Computational studies demonstrate that our framework improves the efficiency of order dispatching in on-demand food delivery services.

7.2 Future Research

One promising direction for future research is to consider uncertainty explicitly when modeling and solving the proposed individual problems. In practice, urban deliveries are highly dynamic, and the input information may not always be accurately predicted by

machine learning models. Therefore, to enhance robustness, it is interesting to explore data-driven approaches that work hand-in-hand with stochastic optimization (such as [111]) and with robust optimization (such as [11, 26, 47]). For example, in robust optimization, the choice of the uncertainty set is crucial, and should be based on the specific problem, the nature of the data, and the available information about uncertainties. The shape and size of the uncertainty set can heavily influence the nature of the robust solution. A larger uncertainty set can lead to more conservative solutions, ensuring performance across a broader range of scenarios but potentially sacrificing optimal performance in nominal or expected scenarios. By utilizing the data-driven approaches, for instance, we can explore to use machine learning or deep learning models to extract hidden features from the historical data to design the uncertainty set for robust optimization.

Given the encouraging experimental outcomes in optimizing the customer service and driver dispatch areas of restaurants, as well as optimizing the order dispatching time interval, a deeper investigation into combining these strategies into a cohesive decision-making framework is merited. For instance, the area sizing issue could be conceptualized as a Markov decision process (MDP), facilitating its formulation within a dual-agent reinforcement learning (RL) framework. In this setup, Agent A_1 would address the area sizing challenge, while Agent A_2 would tackle the order dispatching task. A shared reward function and a transition kernel, influenced by their combined actions, enable these agents to collaboratively resolve the issue, as illustrated in [102].

Another interesting topic deserving attention is the use of deep (reinforcement) learning methods to solve combinatorial optimization problems in the context of urban delivery, such as the dynamic pickup and delivery problem [79]. These kinds of Learning to Optimize (L2Opt) approaches have shown great potential for finding near optimal solutions more quickly, enabling real-time decision-making for vehicle routing problems [59, 88, 75, 53, 147]. However, currently, L2Opt methods also have some limitations. For example, L2Opt models usually require large amounts of training data to learn effective optimization strategies. Gathering and curating such data can be challenging and expensive, especially for optimization problems in urban deliveries. Additionally, L2Opt

models may struggle to generalize to unseen or significantly different problem instances [12]. They are typically proficient at solving problems similar to those encountered during training but may perform poorly on novel or rare scenarios. Addressing these data dependency and generalization issues presents an interesting optimization problem that remains to be explored.

Bibliography

[1] O. Ackerman Viden, Y. Trabelsi, P. Xu, K. A. Sankararaman, O. Maksimov, and S. Kraus. Allocation problem in remote teleoperation: Online matching with offline reusable resources and delayed assignments. In *Proceedings of the 2023 International Conference on Autonomous Agents and Multiagent Systems*, pages 513–521, 2023.

[2] R. Anderson, J. Huchette, W. Ma, C. Tjandraatmadja, and J. P. Vielma. Strong mixed-integer programming formulations for trained neural networks. *Mathematical Programming*, 183(1-2):3–39, 2020.

[3] J. D. Angrist, S. Caldwell, and J. V. Hall. Uber versus taxi: A driver's eye view. *American Economic Journal: Applied Economics*, 13(3):272–308, 2021.

[4] R. Auad, A. Erera, and M. Savelsbergh. Using simple integer programs to assess capacity requirements and demand management strategies in meal delivery. *Preprint, Optimization Online*, 2020.

[5] S. Bahrami, M. Nourinejad, M. M. Nesheli, and Y. Yin. Optimal composition of solo and pool services for on-demand ride-hailing. *Transportation Research Part E: Logistics and Transportation Review*, 161:102680, 2022.

[6] J. Bai, K. C. So, C. S. Tang, X. Chen, and H. Wang. Coordinating supply and demand on an on-demand service platform with impatient customers. *Manufacturing & Service Operations Management*, 21(3):556–570, 2019.

[7] D. Banerjee, A. L. Erera, A. M. Stroh, and A. Toriello. Who has access to e-commerce and when? time-varying service regions in same-day delivery. *Transportation Research Part B: Methodological*, 170:148–168, 2023.

[8] G. Berbeglia, J.-F. Cordeau, and G. Laporte. Dynamic pickup and delivery problems. *European journal of operational research*, 202(1):8–15, 2010.

[9] S. Berger and C. Bierwirth. Solutions to the request reassignment problem in collaborative carrier networks. *Transportation Research Part E: Logistics and Transportation Review*, 46(5):627–638, 2010.

[10] D. Bergman, T. Huang, P. Brooks, A. Lodi, and A. U. Raghunathan. Janos: an integrated predictive and prescriptive modeling framework. *INFORMS Journal on Computing*, 34(2):807–816, 2022.

[11] D. Bertsimas, V. Gupta, and N. Kallus. Data-driven robust optimization. *Mathematical Programming*, 167:235–292, 2018.

[12] J. Bi, Y. Ma, J. Wang, Z. Cao, J. Chen, Y. Sun, and Y. M. Chee. Learning generalizable models for vehicle routing problems via knowledge distillation. *Advances in Neural Information Processing Systems*, 35:31226–31238, 2022.

[13] K. Bimpikis, O. Candogan, and D. Saban. Spatial pricing in ride-sharing networks. *Operations Research*, 67(3):744–769, 2019.

[14] A. Bozanta, M. Cevik, C. Kavaklioglu, E. M. Kavuk, A. Tosun, S. B. Sonuc, A. Duranel, and A. Basar. Courier routing and assignment for food delivery service using reinforcement learning. *Computers & Industrial Engineering*, 164:107871, 2022.

[15] J. Cai, Q. Zhu, Q. Lin, L. Ma, J. Li, and Z. Ming. A survey of dynamic pickup and delivery problems. *Neurocomputing*, page 126631, 2023.

[16] F. Ceccon, J. Jalving, J. Haddad, A. Thebelt, C. Tsay, C. D. Laird, and R. Misener. Omlt: Optimization & machine learning toolkit. *The Journal of Machine Learning Research*, 23(1):15829–15836, 2022.

[17] J.-F. Chen, L. Wang, H. Ren, J. Pan, S. Wang, J. Zheng, and X. Wang. An imitation learning-enhanced iterated matching algorithm for on-demand food delivery. *IEEE Transactions on Intelligent Transportation Systems*, 23(10):18603–18619, 2022.

[18] X. Chen, M. W. Ulmer, and B. W. Thomas. Deep q-learning for same-day delivery with vehicles and drones. *European Journal of Operational Research*, 298(3):939–952, 2022.

[19] Y. Chen, Y. Qian, Y. Yao, Z. Wu, R. Li, Y. Zhou, H. Hu, and Y. Xu. Can sophisticated dispatching strategy acquired by reinforcement learning? In *Proceedings of the 18th International Conference on Autonomous Agents and MultiAgent Systems*, pages 1395–1403, 2019.

[20] J.-F. Cordeau, G. Laporte, and A. Mercier. A unified tabu search heuristic for vehicle routing problems with time windows. *Journal of the Operational research society*, 52(8):928–936, 2001.

[21] F. Cruijssen, M. Cools, and W. Dullaert. Horizontal cooperation in logistics: opportunities and impediments. *Transportation Research Part E: Logistics and Transportation Review*, 43(2):129–142, 2007.

[22] S. Dahl and U. Derigs. Cooperative planning in express carrier networks—an empirical study on the effectiveness of a real-time decision support system. *Decision Support Systems*, 51(3):620–626, 2011.

[23] G. B. Dantzig and J. H. Ramser. The truck dispatching problem. *Management science*, 6(1):80–91, 1959.

[24] P. Dayan and B. W. Balleine. Reward, motivation, and reinforcement learning. *Neuron*, 36(2):285–298, 2002.

[25] E. Delage, S. Arroyo, and Y. Ye. The value of stochastic modeling in two-stage stochastic programs with cost uncertainty. *Operations Research*, 62(6):1377–1393, 2014.

[26] E. Delage and Y. Ye. Distributionally robust optimization under moment uncertainty with application to data-driven problems. *Operations research*, 58(3):595–612, 2010.

[27] A. Delarue, R. Anderson, and C. Tjandraatmadja. Reinforcement learning with combinatorial actions: An application to vehicle routing. *Advances in Neural Information Processing Systems*, 33:609–620, 2020.

[28] D. Dewey. Reinforcement learning and the reward engineering principle. In *2014 AAAI Spring Symposium Series*, 2014.

[29] X. Ding, R. Zhang, Z. Mao, K. Xing, F. Du, X. Liu, G. Wei, F. Yin, R. He, and Z. Sun. Delivery scope: A new way of restaurant retrieval for on-demand food delivery service. In *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pages 3026–3034, 2020.

[30] Y. Ding, B. Guo, L. Zheng, M. Lu, D. Zhang, S. Wang, S. H. Son, and T. He. A city-wide crowdsourcing delivery system with reinforcement learning. *Proceedings of the ACM on Interactive, Mobile, Wearable and Ubiquitous Technologies*, 5(3):1–22, 2021.

[31] L. Duan, Y. Zhan, H. Hu, Y. Gong, J. Wei, X. Zhang, and Y. Xu. Efficiently solving the practical vehicle routing problem: A novel joint learning approach. In *Proceedings of the 26th ACM SIGKDD international conference on knowledge discovery & data mining*, pages 3054–3063, 2020.

[32] O. El Balghiti, A. N. Elmachtoub, P. Grigas, and A. Tewari. Generalization bounds in the predict-then-optimize framework. *Advances in neural information processing systems*, 32, 2019.

[33] A. N. Elmachtoub and P. Grigas. Smart "predict, then optimize". *Management Science*, 68(1):9–26, 2022.

[34] N. P. Farazi, B. Zou, T. Ahamed, and L. Barua. Deep reinforcement learning in transportation research: A review. *Transportation research interdisciplinary perspectives*, 11:100425, 2021.

[35] N. P. Farazi, B. Zou, and T. Tulabandhula. Dynamic on-demand crowdshipping using constrained and heuristics-embedded double dueling deep q-network. *Transportation Research Part E: Logistics and Transportation Review*, 166:102890, 2022.

[36] P. Feldman, A. E. Frazelle, and R. Swinney. Service delivery platforms: Pricing and revenue implications. *Available at SSRN*, 3258739, 2018.

[37] E. Fernández, M. Roca-Riu, and M. G. Speranza. The shared customer collaboration vehicle routing problem. *European Journal of Operational Research*, 265(3):1078–1093, 2018.

[38] W. Ferrell, K. Ellis, P. Kaminsky, and C. Rainwater. Horizontal collaboration: opportunities for improved logistics planning. *International Journal of Production Research*, 58(14):4267–4284, 2020.

[39] E. Frank, Y. Wang, S. Inglis, G. Holmes, and I. H. Witten. Using model trees for classification. *Machine learning*, 32(1):63–76, 1998.

[40] J. Gama. Functional trees. *Machine learning*, 55(3):219–250, 2004.

[41] M. Gansterer and R. F. Hartl. Request evaluation strategies for carriers in auction-based collaborations. *OR spectrum*, 38(1):3–23, 2016.

[42] M. Gansterer and R. F. Hartl. Collaborative vehicle routing: a survey. *European Journal of Operational Research*, 268(1):1–12, 2018.

[43] M. Gansterer, R. F. Hartl, and S. Wieser. Assignment constraints in shared transportation services. *Annals of Operations Research*, 305:513–539, 2021.

[44] C. Gao, F. Zhang, G. Wu, Q. Hu, Q. Ru, J. Hao, R. He, and Z. Sun. A deep learning method for route and time prediction in food delivery service. In *Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery & Data Mining*, pages 2879–2889, 2021.

[45] C. Gao, F. Zhang, Y. Zhou, R. Feng, Q. Ru, K. Bian, R. He, and Z. Sun. Applying deep learning based probabilistic forecasting to food preparation time for on-demand delivery service. In *Proceedings of the 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, pages 2924–2934, 2022.

[46] F. Glover and M. Laguna. Tabu search. In *Handbook of combinatorial optimization*, pages 2093–2229. Springer, 1998.

[47] M. Goerigk and J. Kurtz. Data-driven robust optimization using deep neural networks. *Computers & Operations Research*, 151:106087, 2023.

[48] M. Guajardo, M. Rönnqvist, P. Flisberg, and M. Frisk. Collaborative transportation with overlapping coalitions. *European Journal of Operational Research*, 271(1):238–249, 2018.

[49] X. Guo, A. Haupt, H. Wang, R. Qadri, and J. Zhao. Understanding multi-homing and switching by platform drivers. *Transportation Research Part C: Emerging Technologies*, 154:104233, 2023.

[50] A. Haydari and Y. Yılmaz. Deep reinforcement learning for intelligent transportation systems: A survey. *IEEE Transactions on Intelligent Transportation Systems*, 23(1):11–32, 2020.

[51] F. D. Hildebrandt and M. W. Ulmer. Supervised learning for arrival time estimations in restaurant meal delivery. *Transportation Science*, 2021.

[52] R. T. Icarte, T. Q. Klassen, R. Valenzano, and S. A. McIlraith. Reward machines: Exploiting reward function structure in reinforcement learning. *Journal of Artificial Intelligence Research*, 73:173–208, 2022.

[53] J. James, W. Yu, and J. Gu. Online vehicle routing with neural combinatorial optimization and deep reinforcement learning. *IEEE Transactions on Intelligent Transportation Systems*, 20(10):3806–3817, 2019.

[54] J. Jin, M. Zhou, W. Zhang, M. Li, Z. Guo, Z. Qin, Y. Jiao, X. Tang, C. Wang, J. Wang, et al. Coride: joint order dispatching and fleet management for multi-scale ride-hailing platforms. In *Proceedings of the 28th ACM International Conference on Information and Knowledge Management*, pages 1983–1992, 2019.

[55] W. Joe and H. C. Lau. Deep reinforcement learning approach to solve dynamic vehicle routing problem with stochastic customers. In *Proceedings of the international conference on automated planning and scheduling*, volume 30, pages 394–402, 2020.

[56] J. Ke, X. M. Chen, H. Yang, and S. Li. Coordinating supply and demand in ride-sourcing markets with pre-assigned pooling service and traffic congestion externality. *Transportation Research Part E: Logistics and Transportation Review*, 166:102887, 2022.

[57] J. Ke, F. Xiao, H. Yang, and J. Ye. Learning to delay in ride-sourcing systems: a multi-agent deep reinforcement learning framework. *IEEE Transactions on Knowledge and Data Engineering*, 34(5):2280–2292, 2020.

[58] J. Ke, H. Yang, X. Li, H. Wang, and J. Ye. Pricing and equilibrium in on-demand ride-pooling markets. *Transportation Research Part B: Methodological*, 139:411–431, 2020.

[59] W. Kool, H. Van Hoof, and M. Welling. Attention, learn to solve routing problems! *arXiv preprint arXiv:1803.08475*, 2018.

[60] M. Lai, X. Cai, and Q. Hu. An iterative auction for carrier collaboration in truckload pickup and delivery. *Transportation Research Part E: Logistics and Transportation Review*, 107:60–80, 2017.

[61] H. Li and A. Lim. A metaheuristic for the pickup and delivery problem with time windows. *International Journal on Artificial Intelligence Tools*, 12(02):173–186, 2003.

[62] H.-C. Li and J.-K. Liang. Service pricing strategy of food delivery platform operators: A demand-supply interaction model. *Research in Transportation Business & Management*, 45:100904, 2022.

[63] J. Li, G. Rong, and Y. Feng. Request selection and exchange approach for carrier collaboration based on auction of a single request. *Transportation Research Part E: Logistics and Transportation Review*, 84:23–39, 2015.

[64] J. Li, L. Xin, Z. Cao, A. Lim, W. Song, and J. Zhang. Heterogeneous attentions for solving pickup and delivery problem via deep reinforcement learning. *IEEE Transactions on Intelligent Transportation Systems*, 23(3):2306–2315, 2021.

[65] X. Li, X. Li, H. Wang, J. Shi, and Y. P. Aneja. Supply regulation under the exclusion policy in a ride-sourcing market. *Transportation Research Part B: Methodological*, 166:69–94, 2022.

[66] X. Li, W. Luo, M. Yuan, J. Wang, J. Lu, J. Wang, J. Lü, and J. Zeng. Learning to optimize industry-scale dynamic pickup and delivery problems. In *2021 IEEE 37th International Conference on Data Engineering (ICDE)*, pages 2511–2522. IEEE, 2021.

[67] J. Liang, J. Ke, H. Wang, H. Ye, and J. Tang. A poisson-based distribution learning framework for short-term prediction of food delivery demand ranges. *IEEE Transactions on Intelligent Transportation Systems*, 2023.

[68] M. K. Lim, H.-Y. Mak, and Z.-J. M. Shen. Agility and proximity considerations in supply chain design. *Management Science*, 63(4):1026–1041, 2017.

[69] S. Liu, L. He, and Z.-J. Max Shen. On-time last-mile delivery: Order assignment with travel-time predictors. *Management Science*, 67(7):4095–4119, 2021.

[70] S. Liu, H. Jiang, S. Chen, J. Ye, R. He, and Z. Sun. Integrating dijkstra's algorithm into deep inverse reinforcement learning for food delivery route planning. *Transportation Research Part E: Logistics and Transportation Review*, 142:102070, 2020.

[71] T. Liu, Z. Xu, D. Vignon, Y. Yin, Q. Li, and Z. Qin. Effects of threshold-based incentives on drivers' labor supply behavior. *Transportation Research Part C: Emerging Technologies*, 152:104140, 2023.

[72] Y. Liu and Y. Ouyang. Planning ride-pooling services with detour restrictions for spatially heterogeneous demand: A multi-zone queuing network approach. *Transportation Research Part B: Methodological*, 174:102779, 2023.

[73] Z. Liu, W. Wang, J. He, J. Zhang, J. Wang, S. Li, Y. Sun, and X. Ren. A new hybrid algorithm for vehicle routing optimization. *Sustainability*, 15(14):10982, 2023.

[74] W.-Y. Loh. Classification and regression trees. *Wiley interdisciplinary reviews: data mining and knowledge discovery*, 1(1):14–23, 2011.

[75] H. Lu, X. Zhang, and S. Yang. A learning-based iterative method for solving vehicle routing problems. In *International conference on learning representations*, 2019.

[76] Q. Luo, V. Nagarajan, A. Sundt, Y. Yin, J. Vincent, and M. Shahabi. Efficient algorithms for stochastic ride-pooling assignment with mixed fleets. *Transportation Science*, 2023.

[77] G. Lyu, W. C. Cheung, C.-P. Teo, and H. Wang. Multiobjective stochastic optimization: A case of real-time matching in ride-sourcing markets. *Manufacturing & Service Operations Management*, 2023.

[78] Y. Ma, X. Hao, J. Hao, J. Lu, X. Liu, T. Xialiang, M. Yuan, Z. Li, J. Tang, and Z. Meng. A hierarchical reinforcement learning based optimization framework for large-scale dynamic pickup and delivery problems. *Advances in Neural Information Processing Systems*, 34:23609–23620, 2021.

[79] Y. Ma, J. Li, Z. Cao, W. Song, H. Guo, Y. Gong, and Y. M. Chee. Efficient neural neighborhood search for pickup and delivery problems. *Update*, 1(2):3.

[80] A. MacKay, D. Svartbäck, and A. G. Ekholm. Dynamic pricing and demand volatility: Evidence from restaurant food delivery. *Available at SSRN 4164271*, 2022.

[81] J. Mandi, P. J. Stuckey, T. Guns, et al. Smart predict-and-optimize for hard combinatorial optimization problems. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pages 1603–1610, 2020.

[82] S. Mankad, M. Shunko, and Q. Yu. How to find your most valuable service outlets: Measuring influence using network analysis. *Available at SSRN*, 2019.

[83] D. Maragno, H. Wiberg, D. Bertsimas, S. I. Birbil, D. d. Hertog, and A. Fajemisin. Mixed-integer optimization with constraint learning. *arXiv preprint arXiv:2111.04469*, 2021.

[84] McKinsey & Company. Ordering in: The rapid evolution of food delivery, 2021. `https://www.mckinsey.com/industries/technology-media-and-telecommunications/our-insights/ordering-in-the-rapid-evolution-of-food-delivery`, Last accessed on 2022-07-14.

[85] V. V. Mišić and G. Perakis. Data analytics in operations management: A review. *Manufacturing & Service Operations Management*, 22(1):158–169, 2020.

[86] S. Mitrović-Minić and G. Laporte. Waiting strategies for the dynamic pickup and delivery problem with time windows. *Transportation Research Part B: Methodological*, 38(7):635–655, 2004.

[87] I. Mordatch and P. Abbeel. Emergence of grounded compositional language in multi-agent populations. In *Proceedings of the AAAI conference on artificial intelligence*, volume 32, 2018.

[88] M. Nazari, A. Oroojlooy, L. Snyder, and M. Takác. Reinforcement learning for solving the vehicle routing problem. *Advances in neural information processing systems*, 31, 2018.

[89] Nextbite. What consumers really want from food delivery, 2021. `https://www.nrn.com/delivery-takeout-solutions/what-consumers-really-want-food-delivery`, Last accessed on 2023-06-05.

[90] T. T. Nguyen, N. D. Nguyen, and S. Nahavandi. Deep reinforcement learning for multi-agent systems: A review of challenges, solutions, and applications. *IEEE transactions on cybernetics*, 50(9):3826–3839, 2020.

[91] D. Nicola, R. Vetschera, and A. Dragomir. Total distance approximations for routing solutions. *Computers & Operations Research*, 102:67–74, 2019.

[92] G. OPTIMIZATION. Gurobi machine learning, 2023. Last accessed on 2023-09-25.

[93] Y. Ouyang and C. F. Daganzo. Discretization and validation of the continuum approximation scheme for terminal system design. *Transportation Science*, 40(1):89–98, 2006.

[94] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, et al. Pytorch: An imperative style, high-performance deep learning library. *Advances in neural information processing systems*, 32, 2019.

[95] E. Pérez-Bernabeu, A. A. Juan, J. Faulin, and B. B. Barrios. Horizontal cooperation in road transportation: a case illustrating savings in distances and greenhouse gas emissions. *International Transactions in Operational Research*, 22(3):585–606, 2015.

[96] D. Potts and C. Sammut. Incremental learning of linear model trees. *Machine Learning*, 61(1):5–48, 2005.

[97] H. N. Psaraftis, M. Wen, and C. A. Kontovas. Dynamic vehicle routing problems: Three decades and counting. *Networks*, 67(1):3–31, 2016.

[98] G. Qin, Q. Luo, Y. Yin, J. Sun, and J. Ye. Optimizing matching time intervals for ride-hailing services using reinforcement learning. *Transportation Research Part C: Emerging Technologies*, 129:103239, 2021.

[99] Z. Qin, X. Tang, Y. Jiao, F. Zhang, Z. Xu, H. Zhu, and J. Ye. Ride-hailing order dispatching at didi via reinforcement learning. *INFORMS Journal on Applied Analytics*, 50(5):272–286, 2020.

[100] Z. T. Qin, H. Zhu, and J. Ye. Reinforcement learning for ridesharing: An extended survey. *Transportation Research Part C: Emerging Technologies*, 144:103852, 2022.

[101] J. R. Quinlan et al. Learning with continuous classes. In *5th Australian joint conference on artificial intelligence*, volume 92, pages 343–348. World Scientific, 1992.

[102] G. Radanovic, R. Devidze, D. Parkes, and A. Singla. Learning to collaborate in markov decision processes. In *International Conference on Machine Learning*, pages 5261–5270. PMLR, 2019.

[103] T. Rashid, M. Samvelyan, C. S. De Witt, G. Farquhar, J. Foerster, and S. Whiteson. Monotonic value function factorisation for deep multi-agent reinforcement learning. *The Journal of Machine Learning Research*, 21(1):7234–7284, 2020.

[104] D. Reyes, A. Erera, M. Savelsbergh, S. Sahasrabudhe, and R. O'Neil. The meal delivery routing problem. *Optimization Online*, 6571, 2018.

[105] S. Sadeghi Eshkevari, X. Tang, Z. Qin, J. Mei, C. Zhang, Q. Meng, and J. Xu. Reinforcement learning in the wild: Scalable rl dispatching algorithm deployed in ridehailing marketplace. In *Proceedings of the 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, pages 3838–3848, 2022.

[106] N. Salari, S. Liu, and Z.-J. M. Shen. Real-time delivery time forecasting and promising in online retailing: when will your package arrive? *Manufacturing & Service Operations Management*, 2022.

[107] M. Savelsbergh and T. Van Woensel. 50th anniversary invited article—city logistics: Challenges and opportunities. *Transportation Science*, 50(2):579–590, 2016.

[108] M. W. Savelsbergh and M. Sol. The general pickup and delivery problem. *Transportation science*, 29(1):17–29, 1995.

[109] D. Simchi-Levi. Om forum—om research: From problem-driven to data-driven research. *Manufacturing & Service Operations Management*, 16(1):2–10, 2014.

[110] H. Sun, H. Wang, and Z. Wan. Model and analysis of labor supply for ride-sharing platforms in the presence of sample self-selection and endogeneity. *Transportation Research Part B: Methodological*, 125:76–93, 2019.

[111] T. Sutter, B. P. Van Parys, and D. Kuhn. A general framework for optimal data-driven optimization. *arXiv preprint arXiv:2010.06606*, 2020.

[112] R. S. Sutton and A. G. Barto. *Reinforcement learning: An introduction*. MIT press, 2018.

[113] É. Taillard, P. Badeau, M. Gendreau, F. Guertin, and J.-Y. Potvin. A tabu search heuristic for the vehicle routing problem with soft time windows. *Transportation science*, 31(2):170–186, 1997.

[114] X. Tang, Z. Qin, F. Zhang, Z. Wang, Z. Xu, Y. Ma, H. Zhu, and J. Ye. A deep value-network based approach for multi-driver order dispatching. In *Proceedings of the 25th ACM SIGKDD international conference on knowledge discovery & data mining*, pages 1780–1790, 2019.

[115] T. A. Taylor. On-demand service platforms. *Manufacturing & Service Operations Management*, 20(4):704–720, 2018.

[116] X. Tian, R. Yan, Y. Liu, and S. Wang. A smart predict-then-optimize method for targeted and cost-effective maritime transportation. *Transportation Research Part B: Methodological*, 172:32–52, 2023.

[117] T. Tong, H. Dai, Q. Xiao, and N. Yan. Will dynamic pricing outperform? theoretical analysis and empirical evidence from o2o on-demand food service market. *International Journal of Production Economics*, 219:375–385, 2020.

[118] P. Toth and D. Vigo. An overview of vehicle routing problems. *The vehicle routing problem*, pages 1–26, 2002.

[119] K. E. Train. *Discrete choice methods with simulation*. Cambridge university press, 2009.

[120] M. W. Ulmer, A. Erera, and M. Savelsbergh. Dynamic service area sizing in urban delivery. *OR Spectrum*, pages 1–31, 2022.

[121] M. W. Ulmer, B. W. Thomas, A. M. Campbell, and N. Woyak. The restaurant meal delivery problem: Dynamic pickup and delivery with deadlines and random ready times. *Transportation Science*, 55(1):75–100, 2021.

[122] H. Van Seijen, M. Fatemi, J. Romoff, R. Laroche, T. Barnes, and J. Tsang. Hybrid reward architecture for reinforcement learning. *Advances in Neural Information Processing Systems*, 30, 2017.

[123] L. Verdonck, A. Caris, K. Ramaekers, and G. K. Janssens. Collaborative logistics from the perspective of road transportation companies. *Transport Reviews*, 33(6):700–719, 2013.

[124] D. Vignon, Y. Yin, and J. Ke. Regulating the ride-hailing market in the age of uberization. *Transportation research part E: logistics and transportation review*, 169:102969, 2023.

[125] H. Wang. Transportation-enabled urban services: A brief discussion. *Multimodal Transportation*, 1(2):100007, 2022.

[126] H. Wang and X. Bei. Real-time driver-request assignment in ridesourcing. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 36, pages 3840–3849, 2022.

[127] H. Wang, X. Tang, Y.-H. Kuo, D. Kifer, and Z. Li. A simple baseline for travel time estimation using large-scale trip data. *ACM Transactions on Intelligent Systems and Technology (TIST)*, 10(2):1–22, 2019.

[128] H. Wang and H. Yang. Ridesourcing systems: A framework and review. *Transportation Research Part B: Methodological*, 129:122–155, 2019.

[129] X. Wang, L. Wang, S. Wang, J.-f. Chen, and C. Wu. An xgboost-enhanced fast constructive algorithm for food delivery route planning problem. *Computers & Industrial Engineering*, 152:107029, 2021.

[130] Y. Wang and I. H. Witten. Induction of model trees for predicting continuous classes. 1996.

[131] Y. Wang, Y. Zheng, and Y. Xue. Travel time estimation of a path using sparse trajectories. In *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 25–34, 2014.

[132] Z. Wang, K. Fu, and J. Ye. Learning to estimate the travel time. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pages 858–866, 2018.

[133] W. Weng and Y. Yu. Labor-right protecting dispatch of meal delivery platforms. In *60th IEEE Conference on Decision and Control, CDC 2021, Austin, TX, USA, December 14-17, 2021*, pages 1349–1355. IEEE, 2021.

[134] Y. Xie, W. Ma, and L. Xin. The benefits of delay to online decision-making. *Available at SSRN*, 2022.

[135] Z. Xu, Z. Li, Q. Guan, D. Zhang, Q. Li, J. Nan, C. Liu, W. Bian, and J. Ye. Large-scale order dispatch in on-demand ride-hailing platforms: A learning and planning approach. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pages 905–913, 2018.

[136] R. Yan, S. Wang, and K. Fagerholt. A semi-"smart predict then optimize"(semi-spo) method for efficient ship inspection. *Transportation Research Part B: Methodological*, 142:100–125, 2020.

[137] Y. Yan, A. H. Chow, C. P. Ho, Y.-H. Kuo, Q. Wu, and C. Ying. Reinforcement learning for logistics and supply chain management: Methodologies, state of the art, and future opportunities. *Transportation Research Part E: Logistics and Transportation Review*, 162:102712, 2022.

[138] Y. Yan, Y. Deng, S. Cui, Y.-H. Kuo, A. H. Chow, and C. Ying. A policy gradient approach to solving dynamic assignment problem for on-site service delivery. *Transportation Research Part E: Logistics and Transportation Review*, 178:103260, 2023.

[139] C. Yang, S. Jin, J. A. Alagbé, and C. Bai. A semi-"smart predict, then optimize" method for traffic signal control. *IEEE Intelligent Transportation Systems Magazine*, 2023.

[140] H. Yang, X. Qin, J. Ke, and J. Ye. Optimizing matching time interval and matching radius in on-demand ride-sourcing markets. *Transportation Research Part B: Methodological*, 131:84–105, 2020.

[141] H. Yang, C. Shao, H. Wang, and J. Ye. Integrated reward scheme and surge pricing in a ridesourcing market. *Transportation Research Part B: Methodological*, 134:126–142, 2020.

[142] B. Yildiz and M. Savelsbergh. Provably high-quality solutions for the meal delivery routing problem. *Transportation Science*, 53(5):1372–1388, 2019.

[143] B. Yildiz and M. Savelsbergh. Service and capacity planning in crowd-sourced delivery. *Transportation Research Part C: Emerging Technologies*, 100:177–199, 2019.

[144] K. Zhang and Y. M. Nie. To pool or not to pool: Equilibrium, pricing and regulation. *Transportation Research Part B: Methodological*, 151:59–90, 2021.

[145] L. Zhang, T. Hu, Y. Min, G. Wu, J. Zhang, P. Feng, P. Gong, and J. Ye. A taxi order dispatch model based on combinatorial optimization. In *Proceedings of the 23rd ACM SIGKDD international conference on knowledge discovery and data mining*, pages 2151–2159, 2017.

[146] M. Zhang, S. Pratap, G. Q. Huang, and Z. Zhao. Optimal collaborative transportation service trading in b2b e-commerce logistics. *International Journal of Production Research*, 55(18):5485–5501, 2017.

[147] Y. Zhang, L. Bliek, P. da Costa, R. R. Afshar, R. Reijnen, T. Catshoek, D. Vos, S. Verwer, F. Schmitt-Ulms, A. Hottung, et al. The first ai4tsp competition: Learning to solve stochastic routing problems. *Artificial Intelligence*, 319:103918, 2023.

[148] Y. Zhao, F. Papier, and C.-P. Teo. Market thickness in online food delivery platforms: The impact of food processing times. *Available at SSRN 4612561*, 2023.

[149] J. Zheng, L. Wang, J.-f. Chen, X. Wang, Y. Liang, H. Duan, Z. Li, and X. Ding. Dynamic multi-objective balancing for online food delivery via fuzzy logic system-based supply–demand relationship identification. *Computers & Industrial Engineering*, 172:108609, 2022.

[150] M. Zhou, J. Jin, W. Zhang, Z. Qin, Y. Jiao, C. Wang, G. Wu, Y. Yu, and J. Ye. Multi-agent reinforcement learning for order-dispatching via order-vehicle distribution matching. In *Proceedings of the 28th ACM international conference on information and knowledge management*, pages 2645–2653, 2019.

[151] Q. Zhou, H. Zheng, S. Wang, J. Hao, R. He, Z. Sun, X. Wang, and L. Wang. Two fast heuristics for online order dispatching. In *2020 IEEE Congress on Evolutionary Computation (CEC)*, pages 1–8. IEEE, 2020.

[152] W. Zhou, S. Zhu, P. Cao, and J. Wu. Analysis of an on-demand food delivery platform: Participatory equilibrium and two-sided pricing strategy. *Journal of the Operational Research Society*, pages 1–12, 2023.

[153] Y. Zhou, H. Yang, J. Ke, H. Wang, and X. Li. Competition and third-party platform-integration in ride-sourcing markets. *Transportation Research Part B: Methodological*, 159:76–103, 2022.

[154] Z. Zhu, J. Ke, and H. Wang. A mean-field markov decision process model for spatial-temporal subsidies in ride-sourcing markets. *Transportation Research Part B: Methodological*, 150:540–565, 2021.

[155] G. Zou, J. Tang, L. Yilmaz, and X. Kong. Online food ordering delivery strategies based on deep reinforcement learning. *Applied Intelligence*, pages 1–13, 2022.

Appendix

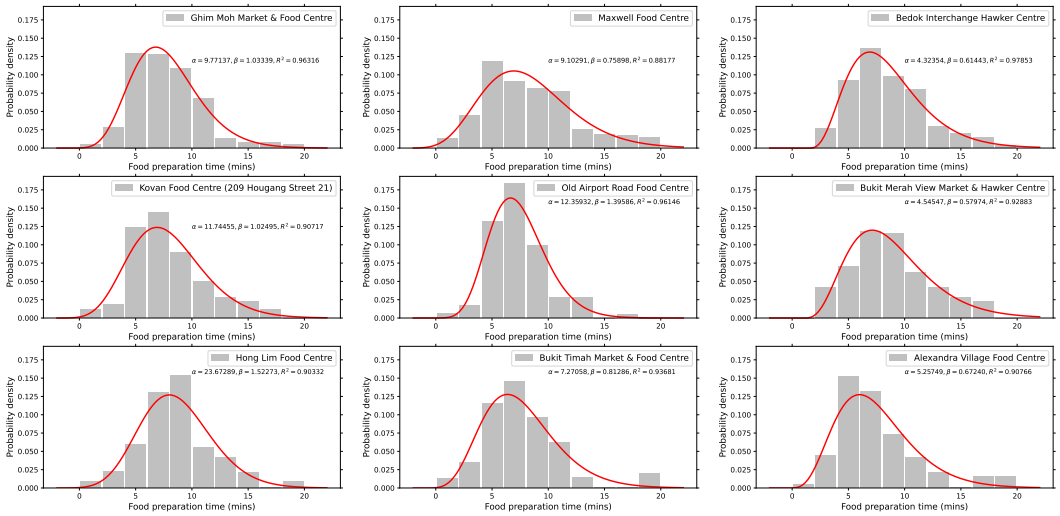


Figure 7.1: Histogram of Customer Restaurants' Orders Preparation Time for All Nine Hawker Centres.

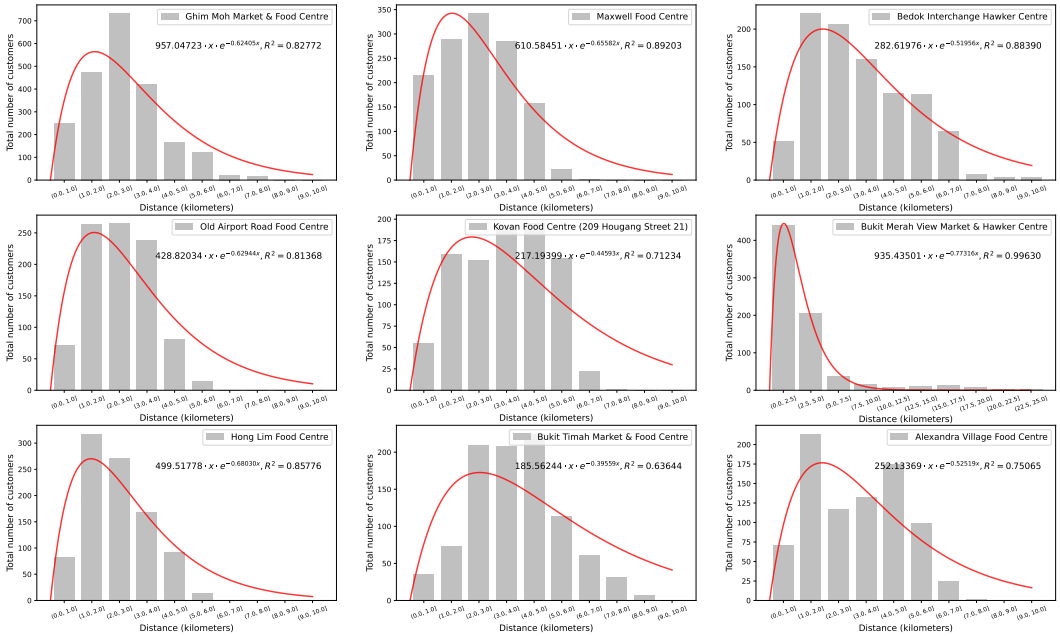


Figure 7.2: Histograms of Customer Orders Based on Customer-Restaurant Distance Segments for All Nine Hawker Centres.

Table 7.1: Performance of ML Methods on Average Delivery Time Prediction.

(a) Kovan Food Centre

ML Methods	R^2	MAPE
Ordinal Least Square	0.598	0.507
Ridge Regression	0.598	0.506
Linear SVR	0.481	0.366
Regression Tree	0.902	0.153
Model Tree	**0.922**	**0.133**
XGBoost	0.851	0.182

(b) Bukit Merah View Market & Hawker Centre

ML Methods	R^2	MAPE
Ordinal Least Square	0.429	0.524
Ridge Regression	0.428	0.523
Linear SVR	0.117	0.244
Regression Tree	0.845	0.189
Model Tree	**0.856**	**0.169**
XGBoost	0.826	0.205

(c) Hong Lim Food Centre

ML Methods	R^2	MAPE
Ordinal Least Square	0.452	0.648
Ridge Regression	0.452	0.647
Linear SVR	0.263	0.356
Regression Tree	0.890	0.195
Model Tree	**0.895**	**179**
XGBoost	0.804	0.268

(d) Bukit Timah Market & Food Centre

ML Methods	R^2	MAPE
Ordinal Least Square	0.571	0.551
Ridge Regression	0.572	0.551
Linear SVR	0.450	0.373
Regression Tree	0.919	0.139
Model Tree	**0.939**	**0.113**
XGBoost	0.900	0.171

(e) Alexandra Village Food Centre

ML Methods	R^2	MAPE
Ordinal Least Square	0.487	0.565
Ridge Regression	0.487	0.564
Linear SVR	0.270	0.315
Regression Tree	0.891	0.183
Model Tree	**0.901**	**0.162**
XGBoost	0.842	0.219

Table 7.2: Experimental Results for All Nine Hawker Centres.

(a) Ghim Moh Hawker & Food Centre

Method	Drivers	Distance (km)	Avg Distance	Delayed Orders	Delay (s)	Avg Delay	On-Time Rate	Avg Delivery Time
Fixed	20	311.52	15.58	0	0	0	0.945	1,107
	25	320.95	12.84	0	0	0	0.972	1,044
	30	329.77	10.99	0	0	0	1.000	1,015
RASO-LT	20	362.97	18.15	7	3,576	510.85	0.920	1,177
	25	386.36	15.45	4	781	195.25	0.955	1,094
	30	391.00	13.03	0	0	0	1.000	1,039

(b) Maxwell Food Centre

Method	Drivers	Distance (km)	Avg Distance	Delayed Orders	Delay (s)	Avg Delay	On-Time Rate	Avg Delivery Time
Fixed	10	170.47	17.05	6	1,930	321.66	0.857	1,308.00
	15	205.99	13.73	1	75	75	0.976	1,142.00
	20	194.77	9.74	0	0	0	1.000	1,034.00
RASO-LT	10	208.03	20.80	9	4,491	499	0.820	1,428.00
	15	228.93	15.26	5	1,332	266.4	0.900	1,210.00
	20	245.41	12.27	2	1,180	590	0.960	1,134.00

(c) Bedok Interchange Hawker Centre

Method	Drivers	Distance (km)	Avg Distance	Delayed Orders	Delay (s)	Avg Delay	On-Time Rate	Avg Delivery Time
Fixed	5	107.06	21.41	10	3,725	372.5	0.629	1,567.00
	10	135.82	13.58	2	272	136	0.926	1,192.00
	15	137.91	9.19	1	14	14	0.963	1095.00
RASO-LT	5	123.20	24.64	14	17,920	1,280	0.600	2,001.00
	10	160.00	16.00	5	1,716	343.2	0.857	1,320.00
	15	175.00	11.67	0	0	0	1.000	1,142.00

(d) Old Airport Road Food Centre

Method	Drivers	Distance (km)	Avg Distance	Delayed Orders	Delay (s)	Avg Delay	On-Time Rate	Avg Delivery Time
Fixed	5	113.77	22.75	10	7,969	796.9	0.677	1,554.00
	10	133.05	13.31	2	80	40	0.935	1,098.00
	15	145.18	9.68	0	0	0	1.000	1,031.00
RASO-LT	5	120.17	24.03	13	15,627	1,202.08	0.649	1,791.00
	10	147.16	14.72	5	2,188	437.6	0.865	1,205.00
	15	165.13	11.01	0	0	0	1.000	1,074.00

(e) Kovan Food Centre

Method	Drivers	Distance (km)	Avg Distance	Delayed Orders	Delay (s)	Avg Delay	On-Time Rate	Avg Delivery Time
Fixed	20	342.43	17.12	6	2,448	408	0.907	1,365.00
	25	335.72	13.43	3	1,367	455.67	0.953	1,295.00
	30	374.81	12.49	0	0	0	1.000	1240.00
RASO-LT	20	432.88	21.64	15	6,051	403.4	0.840	1,554.00
	25	387.63	15.51	19	7,266	382.42	0.797	1,580.00
	30	326.26	10.88	31	21,111	681	0.670	1,761.00

(f) Bukit Merah View Market & Hawker Centre

Method	Drivers	Distance (km)	Avg Distance	Delayed Orders	Delay (s)	Avg Delay	On-Time Rate	Avg Delivery Time
Fixed	15	208.17	13.88	0	0	0	1.000	954.00
	20	220.93	11.05	0	0	0	1.000	925.00
	25	208.02	8.32	0	0	0	1.000	881.00
RASO-LT	15	219.87	14.66	0	0	0	1.000	989.00
	20	220.36	11.02	0	0	0	1.000	951.00
	25	233.96	9.36	0	0	0	1.000	903.00

(g) Hong Lim Food Centre

Method	Drivers	Distance (km)	Avg Distance	Delayed Orders	Delay (s)	Avg Delay	On-Time Rate	Avg Delivery Time
Fixed	5	106.10	21.22	13	7,662	589.38	0.675	1,762.00
	10	156.67	15.67	1	75	75	0.975	1,148.00
	15	178.00	11.87	0	0	0	1.000	1,089.00
RASO-LT	5	114.36	22.87	15	15,273	1,018.2	0.681	1,848.00
	10	182.06	18.21	1	458	458	0.978	1,249.00
	15	156.15	10.41	2	292	146	0.957	1,180.00

(h) Bukit Timah Market & Food Centre

Method	Drivers	Distance (km)	Avg Distance	Delayed Orders	Delay (s)	Avg Delay	On-Time Rate	Avg Delivery Time
Fixed	25	378.47	15.14	3	1,984	661.33	0.955	1,369.00
	30	397.74	13.26	1	797	797	0.985	1,331.00
	35	412.66	11.79	0	0	0	1.000	1,311.00
RASO-LT	25	499.42	19.98	22	11,642	529.18	0.788	1,634.00
	30	457.92	15.26	26	17,869	687.26	0.750	1,731.00
	35	479.58	13.70	24	24,015	1,000.62	0.769	1,751.00

(i) Alexandra Village Food Centre

Method	Drivers	Distance (km)	Avg Distance	Delayed Orders	Delay (s)	Avg Delay	On-Time Rate	Avg Delivery Time
Fixed	10	179.20	17.92	2	1,547	773.5	0.952	1,308.00
	15	203.17	13.54	1	322	322	0.976	1,208.00
	20	227.35	11.37	0	0	0	1.000	1,166.00
RASO-LT	10	223.19	22.32	15	7,901	526.73	0.732	1,678.00
	15	255.57	17.04	2	526	263	0.964	1,221.00
	20	258.08	12.90	0	0	0	1.000	1,183.00