Matching Mechanisms Design for Crowd-Sourced Delivery Under Supply Side Uncertainty

Shixuan Hou

A Thesis

in

The Department

of

Concordia Institute for Information Systems Engineering

Presented in Partial Fulfillment of the Requirements

for the Degree of

Doctor of Philosophy (Information and System Engineering) at

Concordia University

Montréal, Québec, Canada

March 2024

CONCORDIA UNIVERSITY

School of Graduate Studies

This is to certify that the thesis prepared

By: **Shixuan Hou**

Entitled: **Matching Mechanisms Design for Crowd-Sourced Delivery Under Supply Side Uncertainty**

and submitted in partial fulfillment of the requirements for the degree of

Doctor of Philosophy (Information and System Engineering)

complies with the regulations of this University and meets the accepted standards with respect to originality and quality.

Signed by the Final Examining Committee:

_____ Chair
Dr. Manar Amayri

_____ External Examiner
Dr. Jean-Marc Frayret

_____ Examiner
Dr. Yong Zeng

_____ Examiner
Dr. Fereshteh Mafakheri

_____ Examiner
Dr. Ciprian Alecsandru

_____ Supervisor
Dr. Chun Wang

Approved by _____
Dr. Manar Amayri, Chair
Department of Concordia Institute for Information Systems Engineering

22/3/2024

_____
Dr. Mourad Debbabi, Dean
Faculty of Engineering and Computer Science

Abstract

Matching Mechanisms Design for Crowd-Sourced Delivery Under Supply Side Uncertainty

Shixuan Hou, Ph.D.

Concordia University, 2024

Crowd-sourced delivery services represent an innovative urban logistics solution that has garnered considerable attention in recent years due to its superior economic, environmental, and social benefits. Distinct from conventional freight transportation, the crowd-sourced delivery model employs ordinary people as either a supplement or an alternative to professional delivery personnel, integrating them into the delivery process. Due to the inherent uncertainties associated with these non-professional participants, manifested in their strategic refusal of orders, aiming for obtaining better matching outcomes. Furthermore, the prevalent issue of frequent order refusals leads to challenges such as repeated matching and delivery failures, posing significant challenges for the current state of crowd-sourced delivery services.

Addressing these challenges, this dissertation introduces sophisticated matching mechanisms and compensation schemes for crowd-sourced delivery systems, aiming to optimize outcomes in complex scenarios. The research begins with an empirical investigation into the determinants of driver decisions via a series of stated preference surveys. This foundational work enables the development of an accurate predictive model for driver behavior. Integrating this model into an advanced optimization framework, the study then assesses various matching and compensation strategies, considering factors like acceptance probability and decision-making processes. Further innovation is demonstrated through the proposal of an order-postponement mechanism, informed by the urgency value of deliveries. This approach aims to increase the efficiency of crowd-sourced delivery, accommodating more orders within given time windows. A key contribution of this dissertation is the introduction of the concept of reinforced matching stability. Building on this notion, a novel

algorithm is proposed, demonstrably reducing order refusal rates to as low as 1% and achieving operational cost savings of up to 18%. This research not only addresses the immediate challenges of crowd-sourced delivery services but also contributes significantly to the broader discourse in urban logistics and transportation planning.

Acknowledgments

Completing this thesis has been a journey that I could not have embarked upon without the support, guidance, and encouragement of many. I am deeply grateful to several individuals whose contributions were invaluable to my academic and personal growth during this period.

First and foremost, I wish to express my profound gratitude to my supervisor, Dr. Chun Wang, for his unwavering support, insightful feedback, and invaluable guidance throughout the research and writing process. Your expertise and dedication have been a constant source of inspiration and have significantly contributed to the completion of this thesis.

I am also immensely thankful to Dr. Jie Gao, my senior colleague, for her assistance and mentorship in the writing process. Jie's constructive critiques, detailed feedback, and encouraging words have greatly enhanced the quality of my work. Her willingness to share her knowledge and experience has been instrumental in overcoming the challenges I encountered during my writing journey.

To my family—my parents and my wife—I owe a debt of gratitude that words cannot fully express. Your unconditional love, endless patience, and unwavering support have been my pillar of strength. Your belief in my abilities and your emotional support have been my guiding light through the ups and downs of this academic endeavor.

In closing, I wish to extend my appreciation to all who have directly or indirectly contributed to this thesis. Your support has been a vital part of this journey, and for that, I am eternally grateful.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Crowd-sourced delivery: definition and importance

Crowd-sourced delivery service (CDS) refers to a distribution model that leverages the general public or a large group of individuals, often referred to as "the crowd," to fulfill delivery tasks. These individuals use their own means of transportation to deliver packages from businesses or distribution centers directly to customers (Sampaio, Savelsbergh, Veelenturf, & Van Woensel, 2020).

The CDS is emerged by the increasing digital marketplace, the projection indicates that the global e-commerce market is projected to expand at an annual growth rate of 11.17%, culminating in a market valuation of $5.56 trillion by the year 2027 [1]. To secure a large market share, retailers often seek to attract online customers by providing rapid and reliable delivery services, even when online customers are not willing to pay large fees for these delivery services (Dayarian & Savelsbergh, 2020). Meanwhile, it is also shown that by around 2050, approximately 66% of the global population will reside in urban areas [2]. The concomitant growth of e-commerce and urban populace pose significant challenges to "last-mile delivery", refers to all logistics activities related to the delivery of shipments to private customer households in urban areas (Boysen, Fedtke, & Schwerdfeger, 2021). Particularly, when confronted with issues like urban traffic congestion,

[1] https://www.statista.com/outlook/dmo/ecommerce/worldwide
[2] https://www.worldbank.org/en/topic/urbandevelopment/overview

carbon emissions, and stringent city access regulations, the traditional logistics models, which predominantly rely on truck or van transportation, are increasingly proving to be neither cost-effective nor sustainable for efficient logistics delivery (Seghezzi & Mangiaracina, 2022). This necessitates a paradigm shift towards more innovative, environmentally friendly, and adaptive delivery methods to meet the complex demands of modern urban logistics.

Inspired by the shared economy, refers to an IT-facilitated peer-to-peer model for commercial or noncommercial sharing of under utilized goods and service capacity through an intermediary without a transfer of ownership (Schlagwein, Schoder, & Spindeldreher, 2020), CDS effectively utilizes existing transportation resources, this model has yielded notable economic, social, and environmental benefits (Le, Stathopoulos, Van Woensel, & Ukkusuri, 2019). For example, according to the company You2you [3], with the use of the crowd-shipping the costs of delivery are 40% to 60% lower than traditional transport providers. Gatta, Marcucci, Nigro, Patella, and Serafini (2018) assess the adoption of crowd-shipping in Rome, Italy, and suggest that the implement of crowd-shipping produces a total savings of 239 kg of particulates per year.

The compelling advantages of the CDS system have attracted widespread recognition, leading to its broad adoption across major e-commerce platforms, retail giants, and logistics enterprises, including Amazon Flex[4], and Walmart Spark[5]. Specifically, AmazonFlex drivers pick up parcels from central stations and deliver them to online customers, while the Walmart's business idea is to use in-store customers who are willing to deliver orders on their way home.

1.2 Differences between CDS and traditional delivery system

The uniqueness of crowd-sourced delivery systems lies in the incorporation of freelancers for the distribution of goods, bringing inherent uncertainties including behavioral patterns, route selection, and arrival rates (refer to Table. 1.1).

These factors introduce novel challenges to commercial operations and management. "Matching", as the core functionality of CDS systems, directly affect the utilization of the resource capacity,

[3] https://www.you2you.de/
[4] https://flex.amazon.com/
[5] https://drive4spark.walmart.com/

Aspect	CDS	Traditional delivery system
Reliability	Uncertainty	Highly reliable
Quality Control	Less standardized, varies per individual	Standardized, adheres to industry norms
Availability	Highly variable	Consistent availability
Order Acceptance	May not accept assigned orders, based on personal convenience	Generally, accepts all orders within service capability
Route	Depends on individual's travel plans, highly variable	Designed, highly predictable

Table 1.1: Difference between CDS and traditional system

the amount of cost savings to the retailer and the level of customer satisfaction. Empirical data indicate that platforms such as Uber (Said, 2021) and DiDi (Xu, Sun, Liu, & Wang, 2018) experience a notably high rejection rate among crowd-sourced drivers. High order rejection rates can significantly influence the retailer's delivery costs due to frequent reassignments and shifting the orders to more expensive professional fleet. Furthermore, as key participants in the crowd-sourced delivery ecosystem, drivers are often allocated a substantial number of unsuitable orders. This misalignment in order assignment can lead to a diminution in drivers' willingness to participate in delivery activities, subsequently impinging upon the overall quality of service. The phenomenon observed indicates the current matching mechanism's inadequacy in the domain of crowd-sourced delivery.

Therefore, it is worth highlighting that the main difference between the Crowdsourced Delivery System (CDS) and traditional delivery systems lies in whether the matching process takes into consideration the supply side, especially the uncertainty of order acceptance by crowd-drivers.

1.3 Challenges

1.3.1 Challenge 1: Matching mechanism design for optimization objectives

In recent years, the academic community has conducted extensive exploration and practical experimentation to address these challenges. Such as, Archetti, Savelsbergh, and Speranza (2016) employ occasional drivers to supplement capacitated vehicles to make deliveries. A multi-start heuristic algorithm is proposed to produce cost-minimum matching and routing solutions. Arslan,

Agatz, Kroon, and Zuidwijk (2019) propose a heuristic algorithm to solve the large-scale order-driver matching and routing problem, aiming to minimize the total system costs. Dayarian and Savelsbergh (2020) propose two dynamic matching and routing approaches for in-store customers and online orders. The objective of this paper is to minimize the total due time and the total delivery costs. Sampaio et al. (2020) discuss a crowd-sourced delivery system where drivers indicate their availability for tasks, with the focus on enhancing efficiency through transfers at designated locations, and presents an adaptive algorithm to minimize the total operation costs. Also aiming to maximize the total successfully matched rate and minimize the total delivery costs, Boysen, Emde, and Schwerdfeger (2022) propose an efficient exact solution procedure based on Benders decomposition, which considers the employees' minimum expected earnings per time unit.

Nevertheless, the employment of freelance hinders the direct application of traditional mechanisms to CDS, as well as the realization of the optimization objectives initially designed by the mechanism. For example, most mechanisms aims to compute the optimal order-driver matching with the objective of minimize the total delivery cost. However, when there are crowd-shippers who reject these matches, it renders the original solution infeasible. The phenomenon brings about the first challenge, matching mechanism design considering supply side uncertainty, aiming to achieve system optimization objectives.

1.3.2 Challenge 2: matching mechanism design for stability objectives

In addition to systemic optimization objectives such as minimizing costs, maximizing profits, and reducing carbon emissions, prevalent metrics for evaluating matching performance include "matching stability", a concept introduced by Gale and Shapley (1962). This concept refers to a condition in the system where no pair of elements within a matched set would prefer each other over their assigned partners. This theory has been extensively applied in the design of matching algorithms for bilateral markets. However, its application in the domain of crowd-sourced delivery remains scarce; to our knowledge, only N. Zhang, Liu, Li, Xu, and Chen (2023) have explored this area. This paper formulates the matching problem between pre-planned trips and delivery requests as a non-cooperative game and propose two exact algorithms to find stable matchings for CDS matching problems.

However, the aforementioned papers make unrealistic assumptions such as stochastic preference, external characteristics, and the agent rationality. Moreover, it fails to account for the possibility that crowd-shippers might refuse to accept orders assigned by the system. Therefore, the second challenge is to design matching mechanisms aimed at achieving matching stability, taking into account the uncertainty on the supply side.

1.4 Contributions and outline of the dissertation

This dissertation will address the three aforementioned challenges in the design of matching mechanisms within crowd-sourced delivery systems through four main chapters, each dedicated to tackling specific aspects of these issues.

Chapter 2 focuses on the uncertainty of driver behavior, emphasizing the operational decision-making level of the system. Through a series of surveys, I identify the primary factors influencing drivers' order acceptance behaviors. Subsequently, by comparing the accuracy and interpretability of various commonly used machine learning classifiers in predicting drivers' order acceptance, I construct a choice behavior prediction model to be utilized in subsequent research. Building on the foundation of Chapter 2, Chapter 3 proposes a two-stage optimization framework aimed at minimizing the system's total expected delivery cost. In the first stage, the goal is to balance maximizing the order matching rate and minimizing drivers' detour distance, resulting in an optimal pre-matching outcome. Based on this, the second stage involves calculating the most cost-effective compensation plan, integrating a binomial logit discrete choice model to predict the probability of drivers accepting orders. The proposed framework, while considering the uncertainty of driver behavior, achieves the system's optimality objectives and also reduces the order rejection rate compared to commonly used matching and compensation mechanisms in the CDS market. Following this, in Chapter 4, considering the dynamic nature of Crowd-sourced Delivery Systems, I propose a dynamic matching framework to achieve driver-order matching in a dynamic environment, with the optimization objectives of minimizing system costs and maximizing the successful matching rate. Taking into account the value of time and marginal cost of orders, the framework achieves optimal matching results in a dynamic environment by postponing the decision-making time for matching orders. This

approach allows for more flexibility and efficiency in responding to the ever-changing dynamics of CDS. Chapter 5, recognizing the uniqueness of Crowdsourced Delivery Systems where drivers may not be able to peruse all available orders to provide a complete and strict preference list, and acknowledging potential irrational behaviors, introduces the concept of reinforced stable matching. Building upon classical stable matching theory and utilizing the behavioral prediction model learned in Chapter 2, I aim to minimize system costs. A stochastic programming model is employed to calculate tailored compensation schems that increase the probability of each driver accepting system-assigned orders, thereby reinforcing the stability of the match. This matching mechanism has been proven to effectively balance system optimality and matching stability. It provides a theoretical and practical foundation for addressing matching issues in similar bilateral markets, considering individual interests and system optimization, representing our main contribution. Chapter 6 summarizes our work and proposes potential future research directions in this field.

Chapter 2

Order acceptance choice modeling of crowd-sourced delivery services: a systematic comparative study

The efficiency of crowd-sourced delivery services (CDS) like UberEats and AmazonFlex highly depends on the decisions of individual shippers. Operating as freelancers, these shippers have the freedom to accept or decline orders from the CDS platform. Their decisions not only affect their earnings and the waiting times for orders but also influence the platforms' overall revenue and reputation. Understanding the factors that shape these decisions is thus crucial. In this chapter, I gather data from CDS shippers in Shanghai, China, using stated preference surveys. We then design a discrete choice model to predict shippers' behaviors and compare its accuracy, computational efficiency, and interpretability with five commonly used machine learning methods. Our analyses reveal that the Extreme Gradient Boosting (XGB) model and Random Forests (RFs) model outperform other models in prediction accuracy, achieving f1 scores of 69.3% and 65% respectively. Notably, our permutation importance analysis indicate that the shipper's age, income, and the compensation awarded per order are the most influential determinants in their decision to accept or decline orders.

2.1 Introduction

As e-commerce and online shopping continue to expand, there is a growing demand for cost-effective, environmentally sustainable, and efficient last-mile delivery solutions. An emerging trend in this domain is the adoption of crowd-sourced delivery services (CDS) (Le et al., 2019). In this model, individuals, whether they are regular shoppers or daily commuters, use their own vehicles to undertake delivery tasks. These 'crowd-shippers' modify their usual routes, delivering products to e-shoppers and earning compensation for their service. This type of service provides a solution that not only lowers operational costs for retailers by reducing their dependence on full-time drivers, but also maximizes the efficiency of current transportation resources (Pugliese, Ferone, Festa, Guerriero, & Macrina, 2022; Simoni, Marcucci, Gatta, & Claudel, 2020). This efficient utilization can help alleviate urban traffic and decrease greenhouse gas emissions. Recognizing its potential, many companies, ranging from retail giants such as Walmart[1], Amazon[2], and JD.com[3], to startups like Piggybee[4], as well as food delivery companies like UberEats[5] and DoorDash[6], are integrating CDS strategies into their logistical operations.

The operation of crowd-sourced delivery services requires three level of decisions: *strategic*, *tactical*, and *operational*. Strategic decisions outline the business model, choosing between intra-city and inter-city delivery formats. These decisions also involve identifying target customers and leveraging potential transport resources, such as in-store patrons, daily commuters, and occasional travelers. Notably, studies like Le and Ukkusuri (2019b) and Punel, Ermagun, and Stathopoulos (2018) have explored the factors influencing crowd-shipping service adoption. Additionally, Upadhyay, Tiwari, and Tiwari (2022) delves into Generation Z's willingness to engage with these services. Tactical decisions, on the other hand, deal with medium-term management policies. These include decisions about which areas to service (Tao, Dai, Jiang, & Chen, 2021; Yildiz & Savelsbergh, 2019), setting limits on detour times (Arslan et al., 2019; Ghaderi, Tsai, Zhang, & Moayedikia, 2022), and shaping delivery pricing strategies (Ermagun & Stathopoulos, 2018; Huq,

[1] https://drive4spark.walmart.com/
[2] https://flex.amazon.ca/
[3] https://www.jdl.com/
[4] https://www.piggybee.com/
[5] https://www.uber.com/deliver/
[6] https://dasher.doordash.com

Sultana, Sarkar, Razzaque, & Tushar, 2019). Operational decisions are immediate. They concern matching packages to crowd-sourced drivers and deciding their compensation (Hou, Gao, & Wang, 2022; Hou & Wang, 2021). However, it's worth noting that crowd-sourced delivery is inherently more unpredictable than traditional urban logistics, as highlighted by Savelsbergh and Ulmer (2022). The motivations and criteria behind a crowd-sourced driver's order acceptance are complex and not entirely understood. Gdowska, Viana, and Pedroso (2018) suggest that drivers follow certain patterns in accepting orders, while Hou and Wang (2021) believe it based on the perceived benefits for the drivers. Despite these insights, a comprehensive understanding of these behaviors is still limited in this domain. To address this gap, our contributions are as follows:

- We design a set of stated preference survey questionnaires to obtain data on crowd-shipper acceptance choice;
- We design a binomial logit discrete choice model to reveal the main factors that influence crowd-shipper acceptance choice and construct an intuitive choice model;
- We compare the performance of DCM and ML classifiers in terms of prediction performance and computational efficiency;
- We clarify the impact of different features on crowd-shipper acceptance choice and provide decision support for the operation of the crowd-sourced delivery platform through feature importance analysis.

The remainder of this chapter is organized as follows. Section 2.2 reviews the related work. Section 2.3 introduces the commonly used classifiers that I will compare in this paper. Section 2.4 presents the questionnaire design and parameter estimation. The comparative study of the machine learning and discrete choice modeling approach is given in Section 2.5. Finally, future research directions are discussed in Section 2.6.

2.2 Related work

In this section, I first review several survey papers and choices modeling papers on crowd-shipping problems, summarizing the problems they addressed and the primary behavioral modeling

methods they employed. Subsequently, I review some related works in other fields that are similar to our work.

The majority of relevant studies in the field of crowd-shipping are focused on the demand side. Some papers analyze existing transaction data, such as Le et al. Le and Ukkusuri (2019a) examine shipping behaviors, potential crowd-shipping driver-partners, and stakeholder characteristics in the US, using revealed and stated preference surveys to inform logistics improvements, driver recruitment, and business strategies that align with the requester and driver expectations.

Furthermore, some researchers design their own stated preference survey questionnaires to obtain the information, as demonstrated by Le and Ukkusuri (2019b), who investigate factors influencing senders' choice of shipping services. The survey results show that shipping costs and real-time services such as courier reputation, tracking, and customization significantly impact decisions, with senders willing to pay more for crowd-shipping groceries. And this paper uses Random Utility Maximization (RUM) and Random Regret Minimization (RRM) to be the behavior modeling methods. Moreover, Galkin, Schlosser, Capayova, Takacs, and Kopytkov (2021) study Bratislava citizens' attitudes toward working as occasional crowd-shipping couriers, finding that socio-demographic factors and fee value significantly influence participation. A regression relationship model between these factors and maximum parcel weight is built to model citizens' behaviors. Al-Saudi and Himpel (2020) investigates consumers' attitudes toward crowd logistics in Qatar, finding that package insurance is the most valued attribute, followed by flexible delivery and transparent profiles. In Punel, Ermagun, and Stathopoulos (2019), to understand factors influencing crowd-shipping adoption, a structural equation modeling method is used on a US survey. Survey results show that men, younger, full-time employed individuals in areas with high population and low employment density have more willingness to be crowd-shippers.

Regarding the supply side, some studies have also made their contributions. Specifically, Ermagun and Stathopoulos (2018) develop a binomial regression model to understand the bidding behavior of the supply side in crowd-shipping services. Thereafter, Ermagun, Punel, and Stathopoulos (2020) use the random forest algorithm to predict crowd-shipping delivery performance across bidding, acceptance, and delivery stages in the US, finding that context, reward, and timing significantly impact the process, and demonstrating the potential to improve delivery probability through

pricing and request timing adjustments. Regrettably, this paper lacks a comparative learning process and intuitively employs the Random Forest method as the approach to predict shipment status. Zehtabian, Larsen, and Wøhlk (2022) model crowd-shipping pickup and delivery time estimation as a Markov decision process, proposing two look-ahead policies to improve accuracy.

To our knowledge, there is no literature specifically examining driver acceptance choice in crowd-shipping, but I found some similar studies in ride-sharing. Ashkrof, de Almeida Correia, Cats, and Van Arem (2020) use the focus group method to collect and analyze the decision behaviors of drivers in the ride-sharing market, which is similar to the crowd-shipping market. They also develop a conceptual model of tactical and operational decisions of ride-sourcing drivers. The model reveals that factors such as the rider's pickup location, drop-off location, and the size of the luggage have a significant impact on whether a driver accepts a ride request. Two years later, Ashkrof, de Almeida Correia, Cats, and van Arem (2022) use a set of stated preference survey questionnaires to explore the key factors that determine a driver's acceptance behavior. This study employs a discrete choice modeling approach.

Our study also fills the gap in the literature on the choices modeling of crowd-shippers, which has not been explored systematically. While discrete choice models and machine learning methods have been widely applied to model and predict the behavior of customers or drivers in the transportation field, there is a lack of systematic comparison and analysis of these methods in the context of shared mobility, especially in the domain of crowd-shipping. Thus, unlike other data analysis and behavior modeling studies, our main contribution lies in collecting data on the crowd-shipper's acceptance choices through a stated preference survey and comparing the accuracy, computational efficiency, and interpretability of discrete choice models and commonly used machine learning methods.

2.3 Methods

In this section, I will briefly introduce the logit model, and five commonly used ML classifiers.

2.3.1 Binomial logit model

The binomial logit model, developed by McFadden (1977), is one of the most commonly used discrete choice models to predict binary classification problems. The basic elements of the model are given as follows:

- Decision makers: the subject who makes the choice to act, in the crowd-sourced delivery is the self-employed courier.
- Alternative: there are usually multiple options for decision makers to choose from (e.g. couriers can choose orders among the given order menu, or choose to accept or reject the assigned orders)
- Attributes: the factors influencing the decision makers' choice behaviors (e.g. the distance between couriers' current locations and parcel pick-up locations)
- Decision rules: defaulted decision makers' behavioral guidelines when making a choice. In this thesis, I assume each courier is rational and obey the utility-maximization principle.

The utility function, given by Eq. (1), determines whether a courier n accepts or rejects an assigned order j.

$$U_{nj} = \alpha_j + \boldsymbol{\beta}_j \boldsymbol{X}_{nj} + \epsilon_{nj} \tag{1}$$

Where α_j can be seen as in intercept for the j^{th} alternatives; while $\boldsymbol{beta}_j$ is a vector of model parameters (coefficients), $\boldsymbol{X}_{nj}$ is a vector that captures all observable characteristics that influence the choice behaviors of decision makers. And let ϵ_{nj} be the unobservable component for the specific decision maker and the respective alternative. In usual, the classic logit model assumes the stochastic term ϵ_{nj} are independent and identically Gumbel distributed. Therefore, the probability that a decision maker n choose to accept an assigned order j is defined as Eq. (2):

$$P_n(i) = \frac{e^{\beta' X_{in}}}{e^{\beta' X_{in}} + e^{\beta' X_{jn}}} \tag{2}$$

Since We assume the error term ϵ_{nj} follows Gumbel distribution, and the coefficient vector $\boldsymbol{\beta}$ is given, the likelihood function can be defined as Eq. (3)

$$LL = \prod_{n=1}^{N} [P_n(i)^{\lambda_{in}} \cdot P_n(j)^{\lambda_{jn}}] \tag{3}$$

where λ_{ni} is equal to 1 if the individual n choose to accept the alternative i and 0 otherwise. A common practice is to take the natural logarithm of Eq. (3) to simplify the math and computations, the resulting equation known as the log-likelihood function in Eq.(4)

$$lnLL = \sum_{n=1}^{N} [\lambda_{in} \cdot ln(\frac{e^{\beta' X_{in}}}{e^{\beta' X_{in}} + e^{\beta' X_{jn}}}) + \lambda_{jn} \cdot ln(\frac{e^{\beta' X_{jn}}}{e^{\beta' X_{in}} + e^{\beta' X_{jn}}})] \tag{4}$$

Then, to estimate how likely is that the observed data follows the proposed functional form, the maximum likelihood method is used to compute the vector $\boldsymbol{\beta} = \arg\max_{\boldsymbol{\beta}} lnLL$ that maximizes the joint-density of the samples. Finally, replacing the estimated $\boldsymbol{\beta}$ values in Eq. (2), it is possible to predict an individual's acceptance on assigned orders, knowing only the values of the observed characteristics. We have some mature tools to predict couriers' behaviors. The requested data records the choice decisions of couriers. In this case, I meet several challenges and opportunities.

We assume that there are 7 attributes that may influence the drivers' behaviors, and the values of the attributes are standardized to $N(0, 1)$. The parameter values are given based on the Salas, De la Fuente, Astroza, and Carrasco (2022).

2.3.2 Decision Trees and Random forests

Decision trees (DTs) are a non-parametric supervised learning method for classification and regression. Its purpose is to create a model that predicts the value of a target variable by learning simple decision rules inferred from data features. Its main advantages are:

- Interpretability: Since the tree structure can be visualized and closely resembles the human decision-making process, it can be easily understood by non-experts after a simple description.
- Requires little data preparation: Other methods often require data normalization. Since trees can handle qualitative predictors, there is no need to create dummy variables
- In built feature selection. The additional irrelevant feature will be less used so that they can

be removed on subsequent runs. The hierarchy of attributes in a decision tree reflects the significance of attributes. It means that the features on top are the most informative.

However, there are many limitations of DTs.

- unstable: changes to the training set may have a significant impact on the structure of the tree
- NP-complete: each node is generated based on some heuristic algorithms such as greedy algorithms, which can cause the prediction results to not achieve the global optimum
- Overfitting: deeper and more complex tree structures may cause overfitting

The common algorithms for solving decision trees are ID3, C4.5, and CART. The main solution idea is to calculate the information entropy (information gain rate) and Gini impurity. Random forests (RFs) (Breiman, 2001) are an ensemble learning method for classification, regression, and other tasks that operates by constructing a multitude of decision trees at training time. For classification tasks, the output of the random forest is the class selected by most trees. RF is an extended variant of Bagging (Friedman, 2001), which further introduces random feature selection in the training process of decision trees based on the decision tree as the base learner to build Bagging integration, so it can be summarized that RF consists of four parts: random sample selection (put-back sampling); random feature selection; decision tree construction; and random forest voting (averaging).

The advantages of RFs are: It can come out with very high dimensional (many features) data and without dimensionality reduction, no need to do feature selection it can determine the importance of features can determine the interaction between different features is not easy to overfit training is faster and easy to make parallel method is relatively simple to implement for unbalanced data sets, it can balance the error. If a large portion of the features is missing, the accuracy can still be maintained. signal data without having to do the feature selection.

While random forests often achieve higher accuracy than a single decision tree, they sacrifice the intrinsic interpretability present in decision trees. And Random forests have been shown to overfit certain noisy classification or regression problems. For data with attributes that have different values, attributes with more divided values will have a greater impact on the random forest, so the random forest output on such data is not credible for attribute weights

2.3.3 K-Nearest Neighbor

The k-nearest neighbors algorithm (KNN) is a non-parametric supervised learning method, developed by (Fix & Hodges, 1989), which can be used for classification and regression. In both cases, the input consists of the k-closest training examples in a data set. The output of KNN classification is a class. An object is classified by a plurality vote of its neighbors, with the object being assigned to the class most common among its k nearest neighbors (k is a positive integer, typically small). And the neighbors are usually defined by Euclidean distance. The advantage of the KNN algorithm is apparently being simple, however, the computing cost is increasing when the data dimension is very high. In addition, samples that are close together may not belong to the same category.

2.3.4 Extreme Gradient boosting

Extreme Gradient Boosting (XGBoost) is a powerful and widely-used machine learning algorithm for regression and classification problems. It was developed by T. Chen et al. (2015) and is now maintained by a team of developers at DMLC. XGBoost is a gradient-boosting algorithm that combines the strengths of tree-based algorithms and gradient boosting to produce high-quality, accurate models.

Gradient boosting is a technique that involves iteratively improving a weak model by fitting it to the residual errors of the previous iteration. This process continues until the model achieves its optimal performance. XGBoost extends this approach by adding additional regularization and parallel processing capabilities, which allows it to achieve high accuracy and performance even with large datasets.

One of the key features of XGBoost is its ability to handle missing data. It uses a technique called gradient-based imputation to fill in missing values, which involves fitting a model to the observed data and then using that model to predict the missing values. This approach can produce more accurate imputations than traditional imputation methods.

Another important feature of XGBoost is its ability to handle both numerical and categorical data. It can convert categorical variables into numerical variables using techniques such as one-hot encoding, which creates a separate binary variable for each possible value of the categorical

variable. This allows the algorithm to accurately capture the relationships between the categorical variables and the target variable.

XGBoost also includes a number of regularization techniques to prevent overfitting and improve the generalization performance of the model. These include L1 and L2 regularization, which add penalty terms to the objective function to discourage large coefficients, and tree pruning, which removes branches of the tree that do not contribute to the model's overall accuracy.

One of the key advantages of XGBoost is its speed and scalability. It is designed to be highly optimized and can be run in parallel on multi-core CPUs or GPUs. It also includes a number of optimizations, such as approximate greedy algorithm and caching, that can speed up the training process.

XGBoost has become a popular algorithm for a wide range of applications, including image and speech recognition, natural language processing, and financial modeling. Its versatility and speed make it a valuable tool for data scientists and machine learning practitioners.

2.3.5 Support vector machine

Support vector machines (SVMs, also known as support vector networks) are one of the most robust supervised learning models with associated learning algorithms that analyze data for classification and regression analysis. Developed at AT&T Bell Laboratories by Cortes and Vapnik (1995). Given training samples, categorized to be two classes, an SVM training algorithm builds a model that assigns new instances to one class or the other, making it a non-probabilistic binary linear classifier. SVM maps training examples to points in space so as to maximize the width of the gap between the two classes. New examples are then mapped into that same space and predicted to belong to a class based on which side of the gap they fall. In addition to performing linear classification, SVMs can efficiently perform a non-linear classification using what is called the kernel trick, implicitly mapping their inputs into high-dimensional feature spaces.

The advantage of support vector machines is that:

- Since SVM is a convex optimization problem, the solution obtained must be globally optimal rather than locally optimal.

- It is not only applicable to linear problems but also to nonlinear problems (using kernel tricks).
- Data with high-dimensional sample space can also be used with SVM because the complexity of the data set depends only on the support vector and not on the dimensionality of the data set, which in a sense avoids the ”dimensionality disaster”.
- The theoretical basis is better (e.g., neural networks are more like a black box).

Nevertheless, the disadvantage of support vector machines is that the solution of a quadratic programming problem will involve the computation of a matrix of order m (m being the number of samples), so SVMs are not suitable for very large data sets. It is only applicable to binary classification problems.

2.3.6 Artificial neural network

Artificial neural networks (ANNs, also known as neural networks (NN)), are inspired by the biological neural networks that constitute animal brains. ANNs work by successive transformations of the feature space to smooth out non-linearities as much as possible, in order to find an information-rich basis to fit a basic model (Lee, Derrible, & Pereira, 2018). The key insight is that an ANNs learns its own features and fit the classification model, all at once.

A common multilayer structured feed-forward network (Multilayer Feedforward Network (Schneider, 1987)) consists of three components :

- Input layer: where neurons receive a large number of non-linear input data
- Output layer: where data are transmitted, analyzed and weighed in the neuron links to form the output result.
- Hidden layer: is the layer of neurons and links between the input and output layers. The hidden layer can have one or more layers. The number of nodes (neurons) in the hidden layer is variable, but the greater the number, the more significant the nonlinearity of the neural network, and thus the more significant the robustness of the neural network

2.4 Questionnaire design and data analysis

In this section, I present the process of designing our questionnaires and provide descriptive statistics of the collected data.

2.4.1 Questionnaire design

Our questionnaire is divided into two parts. The first part involves providing basic information, including age, gender, and income, which are the three fundamental personal details. Additionally, respondents are asked to indicate their primary mode of transportation, with four basic options: car, public transit, bicycle, and walking. The second part contains order information. Before designing this section of the questionnaire, I considered the following aspects:

- **Detour time:** Longer detour time greatly reduces crowd-shippers' willingness to deliver packages
- **Payment:** Payment is one of the attributes that directly influences the probability of acceptance of delivery requests by crowd-shipper.
- **Weather and season condition:** Different weather/season conditions will have a greater impact on crowd-shippers, they will no longer be willing to travel the extra distance to deliver orders in the rain.
- **Parcel size**: Taking into account that larger packages may not be deliverable by crowd-shippers using modes of transportation other than cars and that females might be reluctant to carry overly heavy packages.

Some parts of attributes and corresponding descriptions are summarized in Table. 3.3. The generation of weather, season, and package size data follows a discrete uniform distribution. We utilized the "geopy" Python library to randomly select four locations on the map: the origin (O), destination (A), order pickup location (B), and order drop-off location (D), all within Shanghai and its surrounding areas. Subsequently, I obtained the routes and durations between the four points by calling the Google Maps API. The original route is denoted by OD, and the new route is defined as

Table 2.1: Alternatives' attributes and corresponding descriptions

Attributes	Descriptions
Weather	Sunny, rainy (2 levels)
Season	Summer, winter (2 levels)
Parcel size	Small, medium, large (3 levels)

the path taken by the crowd-shipper from point O to completing the order (AB) and reaching the final destination (D), the detour distance is defined as $D' = D_{OA} + D_{AB} + D_{BD} - D_{OD}$. Additionally, for the payment amount, I employ the prevalent detour time-tiered compensation mechanisms currently in use, refer to Eq. 5.

$$payment = \begin{cases} 5 + D' & if\ D' \leq 5 \\ 10 + 1.5 * D' & otherwise \end{cases} \tag{5}$$

Each questionnaire includes 10 randomly generated orders and was distributed via multiple channels, such as WeChat, QQ, and Weibo. Respondents had the link from different social media, but they accessed the unique questionnaire. Each respondent was requested to answer the question shown in Fig. 3.3, and provide us with their personal information mentioned above.

Figure 2.1: Sample questionnaire

2.4.2 Survey data description and parameter estimation

	$coef$	$std\ err$	z	$P > \|z\|$
intercept	1.79	0.338	5.319	0.000
AB	0.00	0.00	0.83	0.41
BD	-0.01	0.00	-3.67	0.00
OA	0.01	0.00	1.25	0.21
OD	0.01	0.00	3.31	0.00
age	0.01	0.01	2.58	0.01
gender_m	0.06	0.10	0.53	0.60
income	-0.04	0.01	-5.60	0.00
mode_bus	-0.40	0.15	-2.67	0.01
mode_car	-0.05	0.11	-0.43	0.67
mode_walk	-1.14	0.53	-2.14	0.03
parcel_size_medium	-0.01	0.12	-0.10	0.92
parcel_size_small	0.14	0.12	1.13	0.26
payment	-0.11	0.01	-12.02	0.00
season_winter	-0.12	0.10	-1.28	0.20
weather_sunny	-0.06	0.09	-0.65	0.51

Table 2.2: Parameter estimation results

The final survey data includes 308 respondents, and 3080 order samples from Shanghai, China. Table 2.3 summarizes the descriptive statistics of collected data in China between **December 2022 and April 2023**. In the survey, 1723 orders (55.94%) are rejected by respondents, and 1357 orders (44.06%) are accepted.

Then I use the maximum log-likelihood estimation method to estimate the parameters of the DCM. The parameter estimation is executed on a computer with an Intel Core i7 6-core CPU with 16 GB of RAM, running at 2.6 GHz, using Mac OS X version 11.0.1. The model is implemented in Python version 3.8.5, using "statsmodels" module.

We summarize the estimated parameters in Table. 2.2 alongside the corresponding standard errors, Z-values, and p-values. As shown in Table V, BD, OD, income, and payment have a relatively large impact on crowd-shippers' behaviors. The probability that a crowd-shipper accepts a delivery request decreases as the BD increase ($\beta_{BD} = -0.01, p < 0.01$), income ($\beta_{income} = -0.04, p <$ 001), $mode_bus$ ($\beta_{mode\_bus} = -1.14, p < 0.015$) and payment ($\beta_{paymemt} = -0.11, p < 0.01$), while the probability decreases as the OD ($\beta_{OD} = 0.01, p < 0.01$), and age ($\beta_{age} = 0.01, p <$ 0.015). The value of the intercept, as known as the "Alternative Specific Constant" is 1.79, and

$p < 0.01$. And I found that since the p-values of the rest of the attributes are greater than 0.01, I briefly think they may not have significant impacts on the decisions of crowd-shippers.

2.5 Comparative study

In this section, I primarily compare the differences in predictive performance, efficiency, and interpretability between various machine learning (ML) methods and discrete choice model (DCM) approaches. Additionally, the tuning of different ML hyper-parameters and the interpretation of the results are also presented.

2.5.1 Hyper-parameter tuning

Before entering the training stage, it is essential to identify a collection of hyper-parameter values that yield optimal performance for each model on the given data within a reasonable time frame. This procedure is referred to as hyper-parameter optimization or tuning and significantly impacts the predictive accuracy of machine learning algorithms **?**. For the sampling strategy, the Repeated Stratified K-fold cross-validation method is used to effectively manage the imbalance in the number of individuals selecting each mode. Additionally, statistical inference tools are utilized to assess the differences in prediction accuracy between the methods. Validation and estimation of machine learning models are carried out using Python's Scikit-Learn library, while hyper-parameter selection for each machine learning classifier is achieved via Python's Scikit-Learn library *RandomizedSearchCV* package. To guarantee the reproducibility of the results, a random seed is established. The hyper-parameter selection results are presented as follows, refer to Table. 2.6.

2.5.2 Classification performance

To evaluate the predictive capabilities of our models, I utilize a well-regarded tool for binary classifier assessment, the Receiver Operating Characteristic (ROC) curve. The ROC curve's horizontal axis represents the false positive rate (FPR), which, in this study, is the ratio of delivery requests inaccurately predicted as "acceptance" among all "rejection" requests. On the other hand, the vertical axis signifies the true positive rate (TPR), which is the proportion of delivery requests

accurately predicted as ”acceptance” among all ”acceptance” requests. Consequently, a lower FPR and a higher TPR suggest superior predictive power. Our model's ROC curve, as depicted in Fig 5.6, is situated near the upper left corner, indicating that the model possesses strong predictive capabilities.

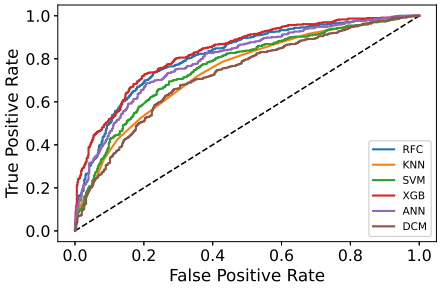


Figure 2.2: **The ROC Curve**: FPR represents the false positive rate, and TPR is the true positive rate. The ROC curve is close to the upper left corner, indicating that the model's predictive ability is good.

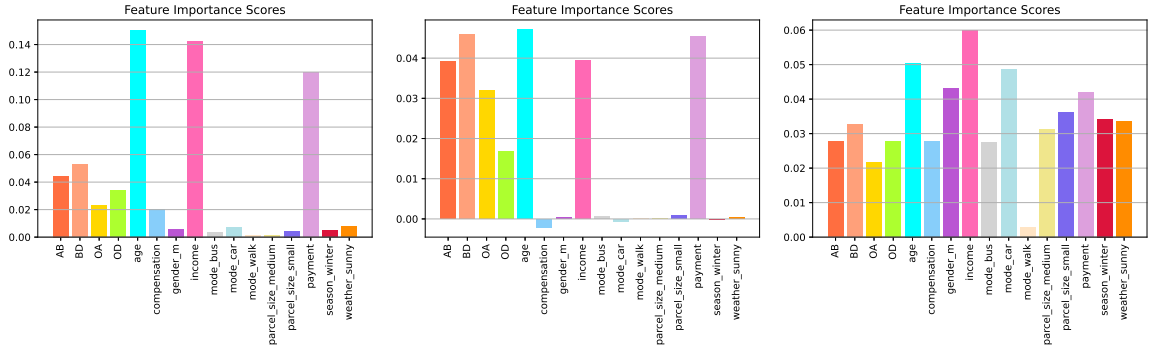
Furthermore, to prevent overfitting, I randomly split the data into training data (comprising 80% of the survey results) and testing data (consisting of 20% of the observations). The training data is utilized to estimate parameters, while the testing data serves to evaluate the model's predictive power. To enhance the reliability of the model assessment, I repeated this procedure 100 times and discovered that the results are robust. The F1-score comparative results are given in Table. 5.4

Based on our observations, I find that the XGBoost (XGB) algorithm outperforms other algorithms in terms of both the ROC curve and F1-score performance. Moreover, in terms of runtime, as shown in Table.2.5, XGB also exhibits a prominent performance, only surpassed by the simpler K-Nearest Neighbors (KNN) and Binomial Logit Regression models.

2.5.3 Feature importance

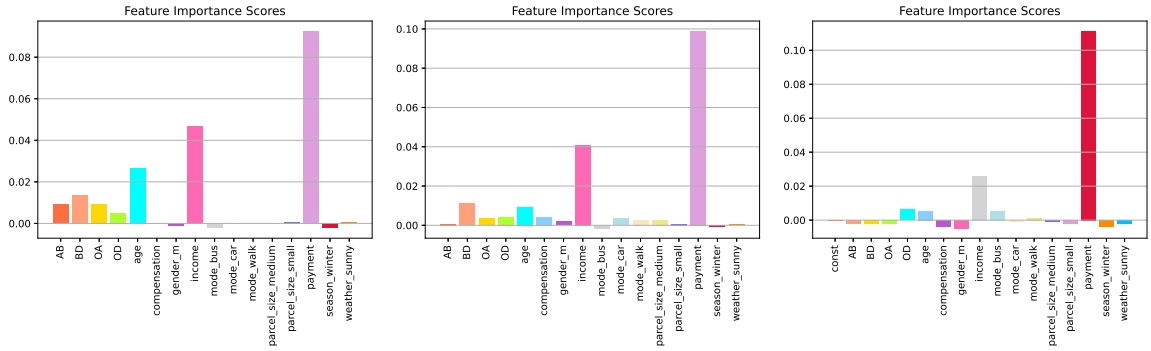
Taking into account the interpretability of different classification methods and the desire to ascertain the impact of specific features on crowd-shipper order acceptance behavior, I employ the Permutation Importance method from Python's Scikit-Learn library which is a technique used to determine the feature importance in a machine learning model by evaluating the change in model performance when the values of a particular feature are randomly shuffled. Through this approach,

I calculate the influence of various features on the outcomes in different ML methods. Please refer to Fig.2.3 and Fig. 2.4 for details.



(a) Extreme gradient boosting (b) K-neighbors (c) Random forest

Figure 2.3: Feature importance



(a) Support vector machine (b) Artificial neuron network (c) Binomial logit model

Figure 2.4: Feature importance

Through the feature importance analysis, I can observe that age, income, and payment are the three most influential factors affecting drivers' order acceptance behavior. All six different classification methods demonstrate relatively high importance scores for these features. Furthermore, I find that the interpretability of the prediction results from K-Nearest Neighbors (KNN) and Random Forest (RF) classifiers are relatively poor. The primary reason for this is that the KNN algorithm underperforms when dealing with features exhibiting high correlations. In our data generation process, the payment value is influenced by the distances between the four locations, resulting in high feature correlations. As for the RF method, besides sharing the same issue as the KNN algorithm, it is better suited for large-scale data. Our survey data has a relatively small scale, which may not be

sufficient to identify the differences in feature importance.

2.6 Summary

This chapter through a series of stated preference survey to collect driver acceptance choice information in a CDS system. By using the binomial logit model, this chapter reveals the main factors influencing the driver's acceptance choice and estimate a driver acceptance utility function. Furthermore, this chapter compare1 the logit model with five other commonly used machine learning methods and found that the XGB method outperforms the others on the collected data set in terms of prediction accuracy, computation efficiency, and interpretability. Our computational results reveal that age, income, and payment price per order are the main factors that influence drivers' decision-making. In addition, the driver's original route and the delivery route of the order also have a certain impact on the driver's acceptance choice. Moreover, the results of our experiments can provide decision support for the design of CDS order assignment systems in the industry. For the academic community, I quantified the uncertainty of drivers' decision-making choice and found suitable behavioral modeling methods based on real data, laying the foundation for future research in this area. Next chapter, I will try to integrate the choice behavior model with the optimization framework, in order to achieve system optimality objectives.

	age	income	OA	AB	BD	OD	payment
count	308.000000	308.000000	3080.000000	3080.000000	3080.000000	3080.000000	3080.000000
mean	39.207792	11.149675	11.772240	14.658377	12.854213	11.281797	17.957470
std	9.876221	11.743334	10.618881	15.026887	13.605458	10.969861	6.485460
min	24.000000	2.500000	0.241589	0.256864	0.216165	0.152365	5.888399
25%	29.000000	6.000000	3.905701	4.808093	4.194143	3.535413	13.260929
50%	39.000000	8.000000	6.720585	8.055234	7.058950	5.765039	17.153939
75%	47.000000	10.000000	19.119591	18.930769	16.607304	18.621595	22.014940
max	58.000000	90.000000	39.362480	70.961143	67.682159	41.481250	35.511317

Table 2.3: Descriptive statistics of personal information

	mean	std
RF	0.65	0.018
ANN	0.621	0.026
KNN	0.626	0.016
XGB	0.693	0.015
SVM	0.628	0.015
DCM	0.608	0.019

Table 2.4: F1 score

	mean	std
RF	0.265	0.008
ANN	1.069	0.042
KNN	0.057	0.005
XGB	0.139	0.086
SVM	0.157	0.1
DCM	0.062	0.014

Table 2.5: Running time

Model	Parameter name	Parameter in Python	Range of possible values	Best value
Random forest	The function to measure the quality of a split	criterion	$\{gini, entropy\}$	gini
	The maximum depth of the tree	max_depth	$\{1, 2, \ldots, 30\}$	10
	The number of trees in the forest	n_estimators	$\{20, 21, \ldots, 200\}$	159
	The number of features to consider when looking for the best split	max_features	$\{auto, sqrt, log2\}$	auto
	The minimum number of samples required to split an internal node	min_samples_split	$\{1, 2, ..., 10\}$	4
	The minimum number of samples required to be at a leaf node	min_samples_leaf	$\{1, 2, ..., 10\}$	2
K-Neighbors	Number of neighbors to use	n_neighbors	$\{1, 2, ..., 100\}$	25
Support vector machine	Regularization parameter	C	$\{0, 1, 2, ..., 20\}$	3
	Specifies the kernel type to be used in the algorithm	kernel	$\{linear, poly, rbf, sigmoid\}$	rbf
	Kernel coefficient	gamma	$\{scale, auto\}$	scale
Extreme Gradient Boosting	Boosting learning rate	learning_rate	$[0, 1]$	0.1
	Maximum tree depth for base learners	max_depth	$\{1, 2, ..., 30\}$	5
	Number of boosting rounds	n_estimators	$\{20, 21, ..., 100\}$	57
	Subsample ratio of the training instance	subsample	$[0.1, 1]$	0.8
Artificial neuron network	Batch size	batch_size	$\{32, 64, 128, 256\}$	256
	Learning rate	lr	$\{0.001, 0.01, 0.1\}$	0.01
	Dropout Rate	drop_out	$\{0.1, 0.3, 0.5\}$	0.3
	Activation Function	activation	$\{relu, sigmoid\}$	relu

Table 2.6: ML classifiers parameters tuning

Chapter 3

Optimization Framework for Crowd-Sourced Delivery Services With the Consideration of Shippers' Acceptance Uncertainties

In the previous chapter, using a set of real data, I revealed the main factors affecting the order acceptance behavior of crowdsourced drivers, including detour distance and compensation amount. Moreover, I modeled the order acceptance behavior of crowdsourced drivers. In this chapter, I will integrate the behavioral model with the matching mechanism to achieve the system's optimality objectives.

Crowd-Sourced Delivery Services (CDS) use in-store customers, as crowd-shippers, to deliver online orders directly to other customers. As independent contractors, the crowd-shippers are free to decide whether to accept or reject the online orders assigned by the retailer. High order rejection rates can significantly influence the retailer's delivery costs due to frequent reassignments and shifting the orders to more expensive professional fleet. To incentivize crowd-shippers to accept the matched orders, in this Chapter, I propose a two-stage optimization framework that integrates bipartite matching with an individual compensation scheme. The first stage of the optimization

framework computes the optimal matching between crowd-shippers and online orders to minimize the delivery detours and unassigned orders. Given the matching solutions as inputs, the second stage computes personal compensation for each crowd-shipper based on the characteristic of the matched order and his or her acceptance behavior uncertainty, with the goal of minimizing the expected total delivery cost of the retailer.

3.1 Introduction

The rapid growth in e-commerce and online shopping demands fast, cost-effective, and yet sustainable last-mile delivery solutions. Riding on the wave of the sharing economy, in recent years, enlisting in-store customers as crowd-shippers for delivering smaller items that can be carried by personal vehicles is becoming an emerging trend in retail delivery services (Le et al., 2019). In a typical Crowd-sourced Delivery Service (CDS), non-professional couriers (i.e., in-store customers) who have space in their own vehicles decide to make a deviation from their regular routes for carrying items to other people (i.e., customers) for a small compensation (Pugliese et al., 2022; Simoni et al., 2020). Compared with traditional professional delivery, retailers can get online orders to their customers faster, using in-store customers to deliver online orders while eliminating the additional time and costs involved with multiple delivery attempts. As a result, by adopting CDS, retailers can achieve on average 20% of delivery cost savings (Pakarti & Starita, 2019). Given the promising benefits of adopting crowd-sourced delivery, large scale e-retailers such as Walmart[1], Amazon Flex[2], and multiple start-ups such as DoorDash[3], Hitch[4] and Postmates[5] have added last-mile crowd-sourced delivery to their traditional van-based home delivery services to save costs and increase their last-mile capacities.

The core functionality provided by a CDS platform is to match in-store customers who are willing to be crowd-shippers and orders placed by online customers. Since the quality of matching will directly affect the utilization of the resource capacity, the amount of cost savings to the retailer

[1] https://corporate.walmart.com/newsroom/2018/09/05/walmart-tests-new-last-mile-grocery-delivery-service
[2] https://flex.amazon.com/
[3] https://get.doordash.com/en-ca/products/drive
[4] https://www.crunchbase.com/organization/hitch-crowdsourced-delivery
[5] https://postmates.com/

and the level of customer satisfaction, designing an effective matching framework is the key to the successful implementation of CDS for retailers. In recent years, a considerable amount of research has been devoted to the design and analysis of optimization approaches to shipper-order matching in CDS (Alnaggar, Gzara, & Bookbinder, 2021; Ermagun, Shamshiripour, & Stathopoulos, 2020; Pourrahmani & Jaller, 2021). These approaches usually focus on the settings where deliveries are performed by some combinations of crowd-shippers and the professional fleet. A retailer's objective is to minimize the total delivery costs, i.e. the delivery costs incurred by the professional fleet and the compensation paid to the crowd-shippers. In general, compensations are calculated based on one fixed fee per parcel or a base fee plus variable delivery costs for the extra time and/or distance of the detour made by crowd-shippers (Archetti et al., 2016; van Cooten, 2016; Y. Wang, Zhang, Liu, Shen, & Lee, 2016).

Despite the diversity of optimization modeling and compensation scheme design, the above-mentioned approaches are deterministic in the sense that they all assume that, given the prescribed compensation scheme, a crowd-shipper will not reject the assigned delivery order. This assumption is an unrealistic simplification to real-world CDS settings (Gdowska et al., 2018). As freelancers, crowd-shippers may behave strategically in accepting the delivery orders (Gdowska et al., 2018; Hou & Wang, 2021). For example, they may cheery-pick more desirable orders which are convenient and profitable for them and reject undesirable orders. Shippers' order rejection behaviors will render the computed matching solutions infeasible, which leads to frequent reassignments and shifting the orders to more expensive professional fleet. In many cases, these changes will also delay the delivery and jeopardize service quality. Stochastic approaches to crowd-shipping optimization are rare in the literature. The one proposed in Gdowska et al. (2018) models the stochastic process of crowd-shipper acceptance and incorporate it into the existing VRPOC model proposed in Archetti and Bertazzi (2021). As acknowledged by the authors their model is limited and simplified by assuming crowd-shipper acceptance probabilities are randomly and uniformly distributed in the range of [0,1]. In addition, the approach does not model the influence of the compensation on a crowd-shipper's acceptance probability, which is an important factor that impacts the quality and feasibility of matching solutions. A recent stochastic approach proposed in Mousavi, Bodur, and Roorda (2022), optimally selects mobile depot locations in advance of full information about the

availability of crowd-shippers. The focus of the paper is to incorporate the uncertainty in crowd-shipper availability by modeling the problem as a two-stage stochastic integer program. It does not address crowd-shipper acceptance uncertainties, but assumes 100% acceptance probability.

In this paper, I propose an optimization framework taking into consideration the crowd-shipper acceptance uncertainty by integrating a learning-based crowd-shipper acceptance probability model into the compensation optimization model. In addition to computing optimal shipper-order matching solutions which minimize delivery detours and unassigned orders, the framework computes the optimal compensation schemes to maximize the expected order acceptance rate and, therefore, minimize the retailer's costs incurred by using the more expensive professional fleet. The contribution of this paper is twofold: first, I propose a binomial logit Discrete Choice Model (DCM) for modeling crowd-shipper acceptance uncertainties. The model is trained using the data set collected by a survey I conducted. Compared with other machine learning-based prediction models, such as SVM and Decision Tree, the logit DCM has better interpretability and can be conveniently calibrated for specific individuals **?**; second, I present a stochastic optimization framework for computing optimal shipper-order matching solutions and shipper compensations. By leveraging the acceptance probability information provided by the logit DCM, the proposed optimization framework enables a CDS platform to explicitly model the influence of the amount of compensation on individual crowd-shipper's acceptance probability and compute optimal individual compensations for all crowd-shippers to minimize the expected delivery cost at system level.

The rest of this paper is organized as follows. Section II reviews the related work. Section III describes the CDS problem and formulates the dynamics of order matching by using a discrete-event system model. Section IV presents the designed matching and compensation optimization models. The performance of the proposed approach is evaluated through a computational study in Section V. Finally, Section VI concludes the paper and states future research directions.

3.2 Literature Review

CDS has attracted a lot of attention in recent years due to its great economic, social, and environmental benefits. For recent overviews, I refer to the readers to Le et al. (2019), and Pourrahmani

and Jaller (2021). Various optimization models have been proposed in the literature for assigning online orders to crowd-shippers with the objective of minimizing the total delivery cost or time. These approaches can be classified into deterministic and stochastic optimization approaches.

Assuming a deterministic environment, Archetti et al. (2016) consider a classical crowd-shipping system in which occasional drivers, as the supplement of professional drivers, deliver parcels to the locations which are not far from their destinations. The assignment problem between parcels and the occasional driver is formulated as an integer programming model that minimizes the total delivery cost. Similarly, Macrina, Pugliese, Guerriero, and Laporte (2020) formulate a cost-minimum assignment problem between occasional drivers and customers into a non-linear mixed-integer programming with considering the presence of intermediate depots. Le, Ukkusuri, Xue, and Van Woensel (2021) develop a sender-shipper matching model to maximize the profit of the platform provider by considering different supply-demand levels. W. Chen, Mes, and Schutten (2018) propose a many-to-many parcel-driver matching model to obtain an optimal matching result by taking the maximum detour distance into account to minimize the overall delivery cost of the system. Basık, Gedik, Ferhatosmanoğlu, and Wu (2018) aim to maximize the number of allocated tasks for a worker by designing a fairness matching strategy for both delivery tasks and crowd-sourced workers. Furthermore, Y. Wang et al. (2016) creatively use citizen workers to perform the last-mile delivery. This paper models the assignment problem between parcels and citizen workers as a network min-cost flow problem. Kafle, Zou, and Lin (2017) use cyclists and pedestrians as crowd-shippers to carry parcels from a truck carrier and deliver the parcels to the final destinations. The matching problem between crowd-shippers and truck relay points is formulated as a mixed-integer non-linear program that minimizes the total delivery costs. Distinguish from the matching problems between occasional couriers and delivery tasks, Behrend and Meisel (2018) consider a joint decision-making problem that matches supply-request and trip-delivery in an integrated system of item-sharing and crowd-shipping. This problem is formulated as an integer programming model that maximizes the total profits of the system. Behrend, Meisel, Fagerholt, and Andersson (2019) extend their work Behrend and Meisel (2018) in 2019. This paper considers a multi-to-one matching problem between items and drivers. An integer programming model is proposed to maximize the total platform profits.

Cheng et al. (2022) use taxi drivers as crowd-shipper to deliver packages and pointed out that previous research on crowd-shipping rarely considers the drivers' perceived value. Therefore, this paper proposes a two-phase optimization framework for optimizing delivery route planning and scheduling, and an integer linear programming model is involved in minimizing the total detour distance. Similarly, C. Chen, Pan, Wang, and Zhong (2017) also use taxis as the carrier for crowd-sourced delivery. In this paper, authors consider a reverse logistics environment that taxi drivers pick up returned orders and passengers and drop off them respectively to their destinations with as little disruption as possible to the passenger experience. A Destination-orientated spreading strategy is proposed to solve the related taxi routing and scheduling problem.

Although static and deterministic models can capture intrinsic characteristics of the matching problems in CDS, they do not consider the uncertainties of the dramatically changing environment, which hinders their applicability and feasibility in real-world settings. To overcome these limitations, researchers have proposed stochastic optimization models. Gdowska et al. (2018) formulate the matching problem between crowd-shippers and packages as a stochastic integer programming that minimizes the total delivery costs under the condition that the acceptance of delivering packages is uncertain. Raviv and Tenzer (2018) consider a crowd-shipping system that has a network of automatic service points. Occasional couriers can pick up, drop off, and intermediate transfer parcels at these points. The matching and routing problem for the couriers is formulated as a stochastic dynamic program considering the uncertainties of packages and couriers arrivals. Sadilek, Krumm, and Horvitz (2013) consider a crowd-shipping system that uses tweets to carry and deliver packages to specific destinations. This paper formulates the task delivery problems as a stochastic graph-planning problem in which agents' locations are uncertain.

Another stream of work deals with the environment uncertainties by using event-driven rolling horizon approaches, which can convert the dynamic and stochastic problem into a repeatedly solved deterministic problem. Arslan et al. (2019) formulate a matching problem between ad hoc drivers and parcel delivery tasks to an integer programming model that minimizes the total delivery cost. Considering the uncertainties of ad hoc driver arrival time and arrivals of delivery tasks, they propose a rolling horizon approach to repeatedly solve an offline matching problem. Also, Dayarian and Savelsbergh (2020) considers the uncertainties of tasks arrival rate and delivery capacity, and

they propose two rolling horizon approaches to handle different market settings. Allahviranloo and Baghestani (2019) propose a rolling horizon approach to capture the uncertainty of delivery tasks announcement time. In this system, requesters submit delivery tasks and compensation. Carriers select the most desirable and compatible tasks based on their situation. This paper formulates the assignment problem as a mixed-integer programming model. The objective is to balance the total travel time spent by the carrier and the total deviation from the planned activity attributes. Soto Setzke et al. (2018) use the rolling horizon approach and develop a dynamic matching algorithm that can match packages to registered trips based on the transportation routes and time constraints. Tu et al. (2019) propose an online dynamic optimization framework for matching food delivery tasks and crowd-sourced drivers. The objective of the matching procedure is to minimize the total travel cost.

The above-mentioned CDS-related papers mainly tackle the matching and routing problems, assuming that crowd-shippers always accept the assigned orders. They either do not take the impact of compensation on shippers' acceptance probability into consideration or adopting fixed compensation schemes. However, as pointed out by Gdowska et al. (2018), crowd-shippers are free to decide whether to accept or reject the assigned orders. To fill these gaps, in this work, I propose an optimization framework to compute matching and compensation solutions by considering the uncertainties of crowd-shippers' acceptance behaviors.

3.3 Problem statement

In this section, I introduce a CDS system, for which I design the matching and compensation optimization models. I first present an overview of the system by introducing its main components and their relationships. After that, I describe the dynamics of the CDS using a Discrete Event System (DES) model.

3.3.1 The Crowd-sourced Delivery Service: an overview

As shown in Fig. 5.1, the CDS consists of a retailer, a set of crowd-shippers, and a set of online orders.

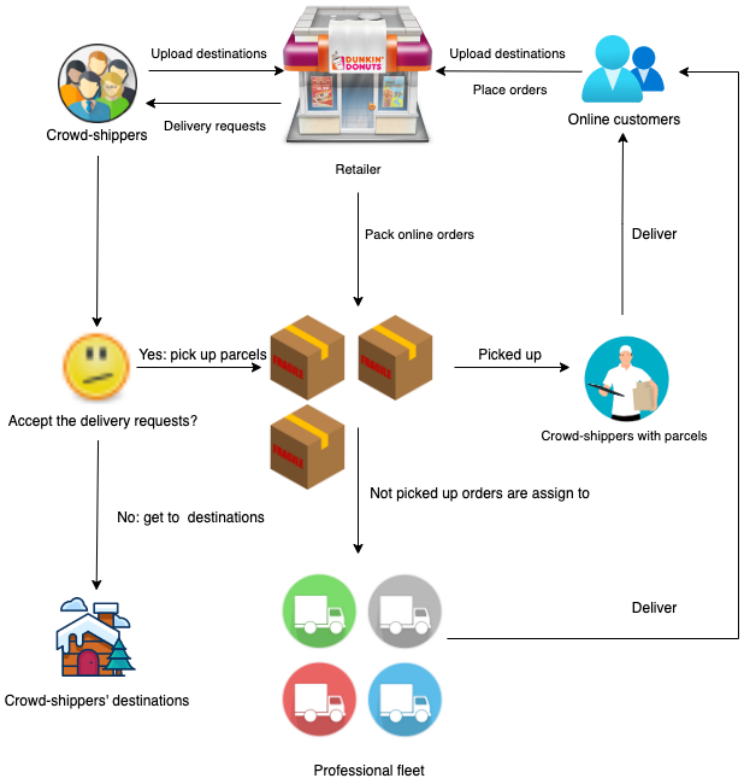


Figure 3.1: Components of the CDS system and their relationships

Retailer

The retailer provides goods to online and in-store customers in the surrounding areas and uses crowd-shippers as a supplement to the professional fleet to deliver online orders. The retailer attracts in-store customers to become registered crowd-shippers through various promotional means. Each registered crowd-shipper should upload and share location and destination information in advance. The retailer records these dynamics and updates the set of crowd-shippers and online orders when a crowd-shipper or an online order arrives. In addition, when a crowd-shipper checks out, CDS is run to get the appropriate online order-crowdshipper matching result and compensation scheme so that the crowd-shipper has the highest probability of accepting the delivery request while minimizing the total delivery cost.

Crowd-shippers and online orders

Each crowd-shipper is required to share and upload their location and destination information in advance. I assume that each driver only delivers one online order. This assumption is based on crowd-sourced delivery literature such as C. Chen et al. (2017); Y. Wang et al. (2016), where they assume in-store customers usually carry some of their own packages and may not be willing to take too many other items even for compensation. And it is assumed that crowd-shippers are rational. Each crowd-shipper can choose to accept or reject a delivery request based on the utilities when he checks out. If he accepts the request, he picks up the online order package and delivers it before getting to his original destination. If he rejects the request, he directly goes to his destination. For each online order, in this paper, for the sake of simplicity, I assume that each online order has the same size and weight. The destination information of each online order is uploaded when the order is placed.

3.3.2 DES modeling of CDS

The Crowd-sourced Delivery Services (CDS) is modeled as a Discrete Event System (DES). Table 5.2 lists the notations in the DES model. For a crowd-shipper i, the main *events* that affect his or her states are e_i^v, e_i^c, e_i^p, e_i^d, e_i^r, and e_i^l, where e_i^v represent the event that crowd-shipper i *arrives*

Table 3.1: Descriptions of Notations in the DES model

Notations	Descriptions
I	A set of crowd-shippers, indexed by i
e_i^v	Arrival event of crowd-shipper i
e_i^c	Check-out event of crowd-shipper i
e_i^p	Delivery request receiving event of crowd-shipper i
e_i^d	Delivery request acceptance event of crowd-shipper i
e_i^r	Delivery request rejection event of crowd-shipper i
e_i^l	Departure event of crowd-shipper i

at the retailer. e_i^c denotes the event that crowd-shipper i *checks out*. e_i^p means that crowd-shipper i receive a *delivery request* of one online order from the retailer. Moreover, e_i^r and e_i^d represent the *delivery request rejection* and *delivery request acceptance* events, respectively. Also the e_i^d denotes the event that crowd-shipper i delivers the online order of the request. Finally, the *departure* event of crowd-shipper i is denoted by e_i^l, which means the crowd-shipper i leaves the destination of the online order or the retailer and gets to his or her own destination.

Let Y be the state space which consists of the set of possible values of the vector $\boldsymbol{y}(k) = [y_1(k) \ldots y_i(k) \ldots y_{|I|}(k)]^T$, where $y_i(k)$ represents the state of the crowd-shipper i after the occurrence of the k^{th} event. The value of the state variable $y_i(k)$ indicates if the crowd-shipper i:

- is at home: $y_i(k) = y_0$;
- is in the store: $y_i(k) = y_1$;
- is waiting in a line: $y_i(k) = y_2$;
- is evaluating a delivery request: $y_i(k) = y_3$;
- has already delivered a package: $y_i(k) = y_4$.

The state transitions diagram of a crowd-shipper i is shown in Fig.5.2.

Among all the events, delivery request event e_i^p is one of the most important events, which triggers the solution of the first-stage optimization problem described in Sec. 3.4.1. After the retailer proposes a delivery request to a crowd-shipper. The crowd-shipper is free to choose whether to accept the request or reject it. In the considered model, this freedom is defined as the acceptance probability p_i^d, determined by the solution of the second-stage optimization problem described in

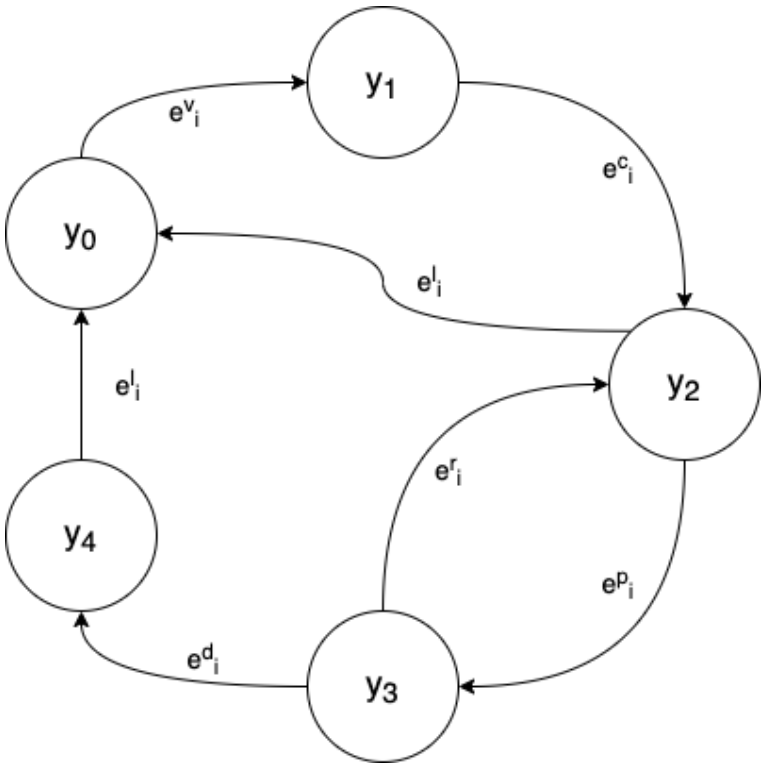


Figure 3.2: State Transition Diagram of Each crowd-shipper

Sec. 3.4.2 Meanwhile, the event e_i^r is determined with probability $p_i^r = 1 - p_i^d$, where p_i^r is the probability that a crowd-shipper i reject an assigned delivery request.

3.4 Matching and Compensation Optimization

The matching and compensation problem in the CDS is equivalent to the ride-sharing problem assuming only one pick-up point, which has been shown to be NP-hard Bei and Zhang (2018). To deal with the computation complexity of the problem, researchers usually adopt the approaches that split a master problem into two sub-problems (C. Chen et al., 2018; Jin & Ma, 2019; Wu, Chu, Saidani, Chen, & Zhou, 2020). In this section, with the same consideration, I propose an optimization framework that contains two stages: matching optimization and compensation optimization. In the first stage, online orders are matched with the crowd-shippers with the objective of minimizing delivery detours and the number of unassigned orders. Given the matching solution as input, the

optimization framework computes optimal compensation crowd-shipper compensations at the second stage to enhance the order acceptance rate and minimize the expected total delivery costs. The matching and compensation optimization procedures are triggered by the event e_i^p defined in Sec. 3.3.2. TABLE II lists the notations used in this paper.

Table 3.2: Notations used in the proposed optimization framework

Notations	Descriptions
I	Set of all crowd-shippers, indexed by i
$\hat{I}$	Set of matched crowd-shippers
J	Set of all online order, indexed by j
$\hat{J}$	Set of matched online order
$\widetilde{J}$	Set of unmatched online order
$D_{i,j}$	Extra distance i has to travel to deliver j
$\mathcal{D}(a,b)$	Distance between area a and area b
o	Retailer address
$x_{i,j}$	Binary variable that indicates the matching between i and j
d_i	Destination of crowd-shipper i
d_j	Destination of order j
s_i	The compensation paid to crowd-shipper i
c_j	Original delivery cost of online order j
$c_{\mu(i)}$	Original delivery cost of the order assigned to the crowd-shipper i
p_i^d	The probability that a crowd-shipper i accepts to deliver an assigned online order
p_i^r	The probability that a crowd-shipper i rejects to deliver an assigned online order

3.4.1 Matching Optimization

Given a set of crowd-shippers I, and a set of online orders J, the goal of the first stage optimization problem is to match online orders and crowd-shippers such that the number of unsatisfied orders and the total detour distances of crowd-shippers can be minimized. Let G_1 be the number of unsatisfied orders. Let G_2 be the total detour distances of crowd-shippers. The two objectives are weighted via two coefficients ω_1 and ω_2 to make the terms measures compatible. Let $x_{i,j}$ be the first stage decision variable which denotes the matching between crowd-shipper i and online order j. And Idenote the distance between two addresses as $D(a,b)$, where a denotes the departure location, and b denotes the arrival location. For simplification, I denotes the detour distance for crowd-shipper i to deliver online order j as $D_{i,j}$, defined as Eq. (9). In addition, let o be the location of the retailer, d_i, and d_j be the destinations of crowd-shipper i and online order j. The

optimal matching problem can be formulated as:

$$\min_{\boldsymbol{x}} \omega_1 G_1(\boldsymbol{x}) + \omega_2 G_2(\boldsymbol{x}) \tag{6}$$

$$G_1(\boldsymbol{x}) = \frac{|J| - \sum_{\forall i \in I} \sum_{\forall j \in J} x_{i,j}}{|J|} \tag{7}$$

$$G_2(\boldsymbol{x}) = \sum_{\forall i \in I} \sum_{\forall j \in J} x_{i,j} \frac{D_{i,j}}{\mathcal{D}(o, d_i)} \tag{8}$$

subject to

$$D_{i,j} = \mathcal{D}(o, d_j) + \mathcal{D}(d_j, d_i) - \mathcal{D}(o, d_i) \tag{9}$$

$$\sum_{\forall j \in J} x_{i,j} \leq 1, \ \ \forall i \in I \tag{10}$$

$$\sum_{\forall i \in I} x_{i,j} \leq 1, \ \ \forall j \in J \tag{11}$$

$$x_{i,j} = \{0, 1\}, \ \begin{cases} \forall i \in I \\ \forall j \in J \end{cases} \tag{12}$$

In this formulation, the objective function Eq. (7) is to assign crowd-shippers to satisfy as many online orders as possible. The second part of the objective function in Eq. (8) minimizes the detour distance rate for all crowd-shippers; The two parts are weighted through changing the values of coefficients ω_1 and ω_2, based on specific market requirements. Eq. (9) defines the detour distance for a crowd-shipper i delivers online order j; Constraint (10) guarantees that each crowd-shipper can only deliver no more than one online order; constraint (11) ensures one online order can only be delivered by one crowd-shipper; Finally, the constraints in Eq. (12) define the problem variables.

Solving the optimization problem determines the optimal shipper-order matchings, reported by a mapping $\mu : \hat{I} \longrightarrow \hat{J}$, where $\hat{I} \subseteq I$ is the set of crowd-shippers who actually deliver orders. In addition, $\hat{J} \subseteq J$ denotes the set of the orders to actually be delivered by crowd-shipper. For

example, if an order $j \in \hat{J}$ is determined to be delivered by an crowd-shipper $i \in \hat{I}$, then $\mu(i) = j$. If a crowd-shipper is not assigned with any online orders, $\mu(i) = \emptyset$.

3.4.2 Compensation Optimization

Since crowd-shippers would strategically accept or reject the assigned online orders prescribed in the matching stage, the goal of the second optimization stage is to compute compensations for the assigned crowd-shippers to incentivize the crowd-shippers to accept the online orders. Let s_i be the decision variable which denotes the compensation given to crowd-shipper i. If a crowd-shipper rejects the assigned online order, the retailer has to reassign this order to a professional courier. Let c_j be the original delivery cost of online order j by a professional courier, defined in Eq (28). Given the matching results μ between crowd-shippers and online orders generated by the matching optimization model, the goal of the compensation optimization model is to determine the optimal compensation scheme that minimizes the total delivery costs of the retailer. Since crowd-shippers are allowed to choose whether to accept the matched delivery requests or not. In this framework, I define the probability that a crowd-shipper i accepts a matched online order as p_i^d, while the rejection probability of the matched online order is denoted by $p_i^r = 1 - p_i^d$. The probabilities are computed based on a Logit DCM, presented in Sec. 3.4.3. Given the probabilities, the optimal matching problem can be formulated as:

$$\min_{\boldsymbol{s}} \sum_{\forall i \in \hat{I}} (s_i p_i^d + c_{\mu(i)} p_i^r) + \sum_{\forall j \in \widetilde{J}} c_j \tag{13}$$

subject to:

$$c_{\mu(i)} = c_0 + \alpha_0 \mathcal{D}(o, d_{\mu(i)}), \ \ \forall i \in \hat{I} \tag{14}$$

$$c_j = c_0 + \alpha_0 \mathcal{D}(o, d_j), \ \ \forall j \in \widetilde{J} \tag{15}$$

$$s_i \leq \omega_3 c_{\mu(i)}, \ \ \forall i \in \hat{I} \tag{16}$$

$$s_i \geq 0, \ \ \forall i \in \hat{I} \tag{17}$$

In the formulation, the objective function (13) minimize the total delivery cost of the retailer. Eq.

(14) defines the original delivery cost of the order assigned to crowd-shipper i, and Eq. (15) defines the original delivery cost of the order j not assigned to any crowd-shippers, where the delivery costs are assumed to be proportional, via the parameter α_0, to the detour distance plus a fix value c_0. Constraint (16) guarantees the value of compensation can not be over a percentage ω_3 of the original delivery cost of the order. While Constraint (17) defines the decision variables.

3.4.3 Logit DCM

In this work, I utilize a logit DCM to reveal factors that influence the decisions made by crowd-shippers and compute the probability that crowd-shippers accept assigned online orders.

Table 3.3: Alternatives' attributes and corresponding descriptions

Attributes	Descriptions
Weather	Sunny, rainy (2 levels)
Traffic	Good, bad (2 levels)
detour Distance	Short (0-2km), short medium(2-4km), medium(4-6km), long medium (6-8km) ,long (8-10km) (5 levels)
Compensation	Low (0-3 ¥), low medium(3-6 ¥), medium(6-9 ¥), high medium (9-12 ¥), high (12-15 ¥) (5 levels)

Attribute selection and questionnaire design

This study will firstly design a set of questionnaires to capture the possible attributes that influence crowd-shippers' decisions to accept assigned online orders. I preset several attributes, such as weather, traffic conditions, compensation, and detour distance based on some considerations:

- **detour distance:** Longer detour distances greatly reduce the willingness of crowd-shippers to deliver packages
- **Compensation:** Compensation scheme is one of the attributes that directly influences the probability of acceptance of delivery requests by crowd-shipper.
- **Traffic:** Poor traffic conditions can cause crowd-shippers to be reluctant to waste time delivering online orders.

- **Weather condition:** Different weather conditions will have a greater impact on crowd-shippers, they will no longer be willing to travel the extra distance to deliver orders in the rain.

The attributes and corresponding descriptions are summarized in Table 3.3.

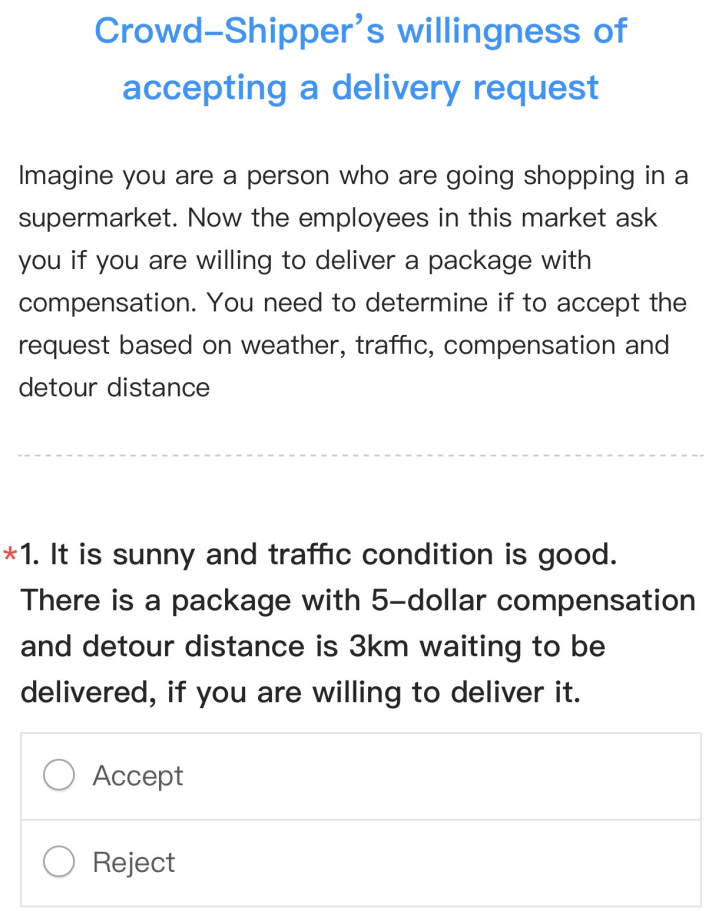


Figure 3.3: Questionnaire Sample

The survey was created in WJX [6] which is an online survey platform. The link to the online survey was distributed via multiple channels, such as WeChat, QQ, and Weibo. Respondents had the link from different social media, but they equally accessed the same online survey. Each respondent was requested to answer the question shown in Fig. 3.3.

[6] http://www.wjx.cn

Discrete choice modeling

I have treated the crowd-shippers as the decision-makers. And the alternatives are acceptance and rejection of online order. The attributes consist of the detour distance, compensation, weather, and traffic. The decision rule follows the utility maximization rule. In this paper, this DCM is of the following form:

$$Logit\left\{p_i^d\right\} = ln(\frac{p_i^d}{1-p_i^d}) = \boldsymbol{\beta}^T \boldsymbol{X_i} \tag{18}$$

Where $\boldsymbol{\beta} = [\beta_{weather}, \beta_{traffic}, \beta_{compensation}, \beta_{distance}, \beta_0]$,
and $\boldsymbol{X}_i = [X_{weather}, X_{traffic}, X_{compensation}, X_{distance}, 1]$.

3.5 Computational Study

In this section, I conduct a numerical study to verify the performance of the proposed framework in terms of the cost reduction rate, rejection rate, and computational time. First I present the data set used to conduct the experiment. Then I use the maximum likelihood estimation (MLE) method to estimate the parameters of the DCM. Finally, I evaluate the framework by comparing its performance with two alternative optimization approaches. The experiment were executed on a computer with an Intel Core i7 6-core CPU with 16 GB of RAM, running at 2.6 GHz, using Mac OS X version 11.0.1. The model was implemented in Python version 3.8.5, using "statsmodels" module. The two-stage optimization model was implemented in Gurobi 9.1.2.

3.5.1 Survey data description and parameter estimation

The final survey data includes 500 respondents from China. Table 3.5 summarizes the descriptive statistics of collected data in China between **July 2021 and September 2021**. In the survey, 370 respondents reject delivering packages, and 130 of them accept delivering packages on their way to their destinations.

I summarized the estimated coefficients in Table. 5.5 alongside the corresponding standard errors, Z-values, and p-values. As shown in Table 5.5, distance and compensation have a relatively large impact on crowd-shippers' behaviors. The probability that a crowd-shipper accepts a delivery

Descriptive statistics	Willingness	Weather	Traffic	detour distance	Compensation
Count	500	500	500	500	500
Mean	0.26	0.49	0.49	4.95	7.27
Standard Deviation	0.44	0.50	0.50	2.91	4.29
Minimum	0.00	0.00	0.00	0.02	0.02
25th percentile	0.00	0.00	0.00	2.42	3.39
Median	0.00	0.00	0.00	4.86	7.04
75th percentile	1.00	1.00	1.00	7.43	10.67
Maximum	1.00	1.00	1.00	9.97	14.96

Table 3.4: Statistics of collected data in China between July 2021 and September 2021

	coeff	std err	z	$P > \|z\|$
weather	0.5787	0.621	0.932	0.351
distance	-0.8522	0.201	-4.235	0.000
compensation	0.7337	0.122	6.028	0.000
traffic_0	1.2188	0.686	1.776	0.076
intercept	-4.2953	0.866	-4.961	0.000

Table 3.5: Parameter estimation results

request increases as the compensation increase ($\hat{\beta}_{compensation} = 0.7337$, $p < 0.01$), while the probability decreases as the detour distance increase ($\hat{\beta}_{distance} = -0.8522$, $p < 0.01$). The value of the intercept, as known as "Alternative Specific Constant" is -4.2953, and $p < 0.01$. And I found that since the p-values of weather and traffic are greater than 0.01, I briefly think they may not have significant impacts on the decisions of crowd-shippers.

To assess the predictive power of my model, I adopt a valuable tool for binary classifier evaluation, the Receiver Operating Characteristic (ROC) curve. The horizontal axis of the ROC curve represents the false positive rate (FPR). In this paper, FPR is the proportion of delivery requests incorrectly predicted to be "acceptance" among all the "rejection" requests. The vertical axis is the true positive rate (TPR), and TPR is the proportion of delivery requests correctly predicted to be "acceptance" among all the "acceptance" requests. Hence, a smaller FPR and a larger TPR indicate higher predictive power. Fig 5.6 shows the ROC curve of my model, which is close to the upper left corner, implying that the predictive ability of the model is good.

In addition, to avoid overfitting, I randomly divided the data into training data (80% of the survey results) and testing data (20% of the observations). I use the training data to estimate the parameters and use the testing data to evaluate the prediction power. To make the model assessment more reliable, I repeated this process 100 times and found that the results are robust. The average f1-score is 0.88 fairly showing a good regression prediction.

3.5.2 Parameter settings and scenarios

In order not to lose generality, I try to simulate a natural CDS environment. Firstly, I randomly choose one retailer and 11 surrounding communications in Google Map. Furthermore, the assumed

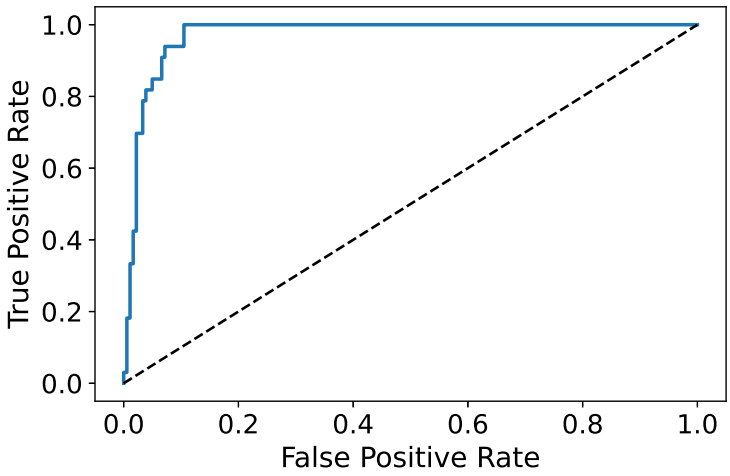


Figure 3.4: **The ROC Curve**: FPR represents the false positive rate, and TPR is the true positive rate. The ROC curve is close to the upper left corner, indicating that the model's predictive ability is good.

values of some parameters mentioned in Sec. 3.4 are shown in Table. 3.6, the delivery cost of professional fleet and the compensation of crowd-shippers are inspired by the wage calculator of Meituan, which is a Chinese food delivery company [7], similar to DoorDash and UberEats.

Parameter	Description	Value
ω_1	Weight of order satisfaction	10
ω_2	Weight of detour distance	0.1
C_0	Base delivery cost of professional fleet	5$
C_1	Base compensation of crowd-shipper	4$
α_0	Delivery cost of professional fleet per kilometer	1$
α_1	Compensation of crowd-shipper per extra kilometer	1.1$

Table 3.6: Parameter values

Moreover, I conclude three different scenarios as follows.

- **Scenarios 1:** The drop-off locations for online orders are concentrated in the same areas as the crowd-shipper destinations.
- **Scenarios 2:** The drop-off locations of online orders and crowd-shippers' destinations are uniformly distributed over all areas.

[7] Meituan https://about.meituan.com/en

- **Scenarios 3:** The drop-off locations for online orders are concentrated in the exact opposite area of crowd-shipper destinations. For example, for all online order groups, about 20% of online order drop-off locations are concentrated in area A, while only 5% of crowd-shippers' destinations are in this area.

3.5.3 Cost reduction rate comparison in three scenarios

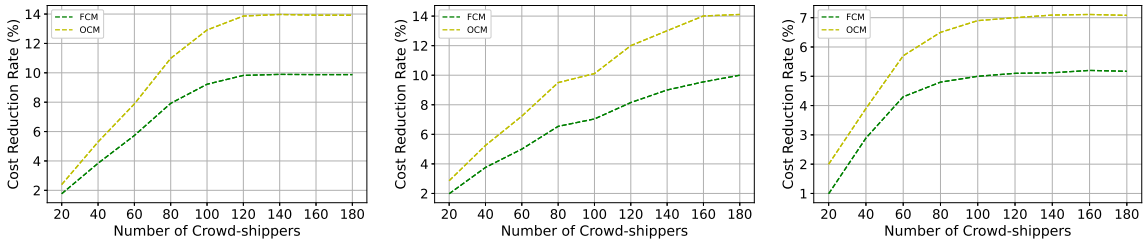
In this subsection, I will focus on analyzing the impact of the number of crowd-shippers on the Cost Reduction Rate (CRR), defined as Eq. (19) in different scenarios when the number of online orders is fixed as 100.

$$CRR = \frac{\sum_{\forall i \in \hat{I}} (c_{\mu(i)} - s_i) p_i^d}{\sum_{\forall j \in J} c_j} \tag{19}$$

I use the traditional delivery system (TDS) that satisfies all delivery requests by the professional fleet as a benchmark. In addition to comparing the cost-effectiveness and time-efficiency of two compensation mechanisms, the proposed optimal compensation mechanism (OCM) and fixed compensation mechanism (FCM), I compare the impact of the two compensation mechanisms on the acceptance rate of the crowd-shipper. Compare the performance of the two compensation mechanisms, OCM and FCM, in the same condition. As shown in Fig. 4.4, I can clearly find that my proposed OCM has a more outstanding performance in terms of cost-effectiveness, intuitively shown as a cost reduction rate, than FCM in three different scenarios.

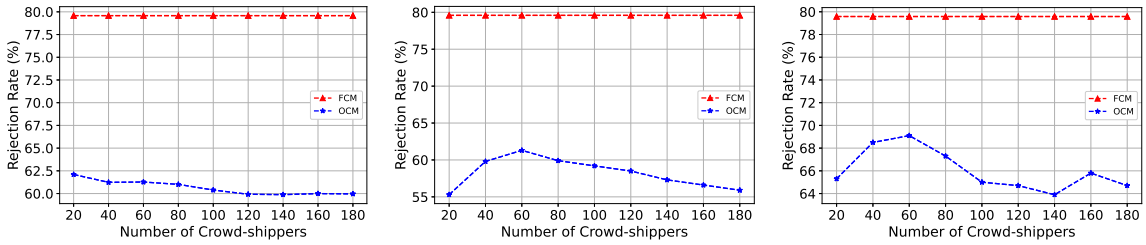
For scenario 1, under the premise that the number of online orders is inevitable when the number of crowd-shippers is less than 120, the cost reduction rate increases, and when the number of crowd-shippers is greater than 120, the value of cost reduction rate remains the same. This phenomenon is because the probability of distribution of drop-off locations of online orders within 11 regions is assumed to be precisely the same as the probability of distribution of crowd-shipper destinations in 11 regions. This allows a large proportion of crowd-shippers not to need to consider the detour distance, greatly increasing the likelihood that they will accept to deliver the online order. For Scenario 2, the cost reduction rate increases slightly less than Scenario 1 as the number of crowd-shippers grows, and after the number of crowd-shippers reaches about 180, the cost reduction rate for OCM

and FCM levels off to an approximate rate of about 14%. This phenomenon occurs because crowd-shipper destinations and online order drop-off locations are uniformly distributed over 11 areas. The crowd-shipper population needs to be greater to find more crowd-shipper individuals appropriate for the deliveries of online orders. For scenario 3, since the distribution of crowd-shipper destinations and online order drop-off locations are opposite, it is difficult to find matches that online orders can be delivered without detours, which directly leads to a decrease in the probability of crowd-shippers accepting delivery requests. The probability of accepting delivery requests decreases, leading to higher delivery costs. As shown in Fig. 4.4 (c), the maximum cost reduction rate is only 7%, which is half of the cost reduction rate of the first two scenarios. In addition, the cost reduction rate tends to level off when the number of crowd-shippers reaches 100.



(a) Scenario 1 (b) Scenario 2 (c) Scenario 3

Figure 3.5: Comparison of cost reduction rate between two compensation strategies in three scenarios



(a) Scenario 1 (b) Scenario 2 (c) Scenario 3

Figure 3.6: Comparison of rejection rate between two compensation strategies in three scenarios

3.5.4 Rejection rate comparison

Rejection Rate (RR) is one of the main criteria to measure the feasibility of CDS, which is defined as Eq (20), and its impact on the total delivery cost of the system can be observed indirectly

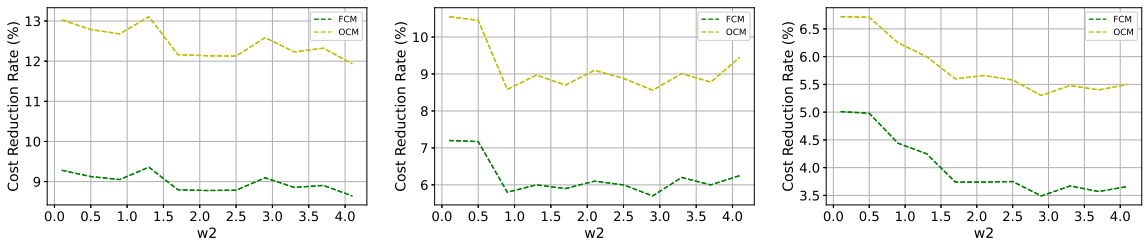
while reflecting the number of crowd-shippers, accepting delivery requests.

$$RR = \sum_{\forall i \in \hat{I}} p_i^r \tag{20}$$

By comparing the performance of FCM and OCM on RR in three scenarios, I observe that the feasibility of CDS with FCM is low, as shown in Fig. 3.6, the highest probability, reaching about 80%, of crowd-shippers are not willing to accept the delivery requests. OCM, on the other hand, fluctuates from 55% to 69% depending on the randomly generated crowd-shippers' destinations, showing better feasibility. Moreover, as I mentioned in the previous subsection, due to the different scenarios, crowd-shippers have different detour distances for the same online orders, which is intuitively reflected in the Fig. 3.6, in addition, scenario 1 has the lowest rejection rate, scenario 3 has the highest rejection rate, and scenario 2 has the middle rejection rate.

3.5.5 Parameter analysis

As what Imentioned in Sec. 3.4 that there are two parameters ω_1 and ω_2 that affect the weight of order satisfaction rate and detour distance, respectively, in my system. To observe the effect of these two parameters on my two metrics, CRR and RR, Iset up several sets of experiments, keeping ω_1 constant, and obtain some conclusions by observing the variation of CRR and RR with the change of ω_2 values in three scenarios.

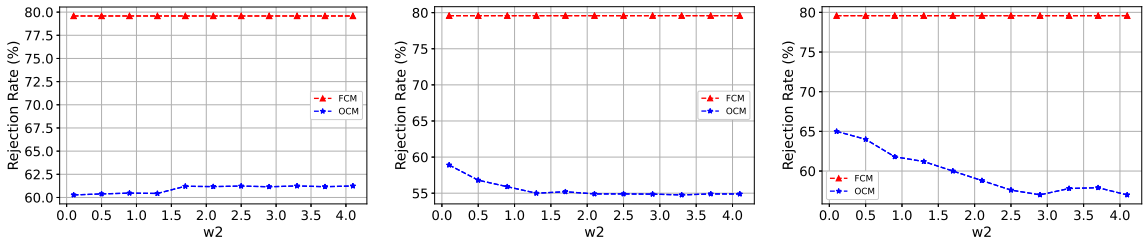


(a) Scenario 1 (b) Scenario 2 (c) Scenario 3

Figure 3.7: Comparison of cost reduction rate between two compensation strategies with different ω_2

To observe the influence of the value of ω_2 on CRR, Iset up ten groups of experiments in three scenarios. The experiments results are shown in Fig. 3.7.

In Scenarios 1, I can compare that OCM has a higher CCR than FCM no matter how I change the value of ω_2. In addition, in Scenario 1, since the location distribution of online customer drop-off locations and crowd-shipper destinations is highly consistent, the value of the weight ω_2 that affects the detour distance has little effect on the results. In Scenario 2, I can find that when $\omega_2 \in [0.5, 1.0]$, the value of CRR decreases sharply, and when $\omega_2 \leq 1.0$, the value of CRR tends to level off. Moreover, in Scenario 2, OCM outperforms FCM in terms of cost-effectiveness regardless of the value of ω_2. In Scenario 3, I can find that the value of CRR decreases gradually when $\omega_2 \in [0.5, 2.0]$ and stabilizes when $\omega_2 \leq 2.0$. In addition, in Scenario 2, OCM outperforms FCM in terms of cost-effectiveness regardless of the value of ω_2.



(a) Scenario 1 (b) Scenario 2 (c) Scenario 3

Figure 3.8: Comparison of rejection rate between two compensation strategies with different ω_2

To observe the influence of the value of ω_2 on RR, Iset up ten groups of experiments in three scenarios. The experiments results are shown in Fig. 3.8 In Scenario 1, the RR of FCM is much higher than that of OCM, and the RR of FCM is not affected by the change of ω_2 value, while the RR of OCM increases slightly with the increase of ω_2 value, but the effect is not significant enough to be negligible. In Scenario 2, the FCM performs much worse than the OCM in RR. The RR of the OCM decreases sharply when $\omega_2 \in [0, 1.3]$, after which the curve flattens out. In Scenario 3, the RR value of the FCM remains constant, and due to its characteristics, the probability of the crowd-shipper accepting a delivery request is directly affected by the detour distance. Moreover, the RR of OCM decreases slowly when $\omega_2 \in [0, 2.5]$, and when $\omega_2 \leq 2.5$, the RR of OCM tends to a stable value of about 56%.

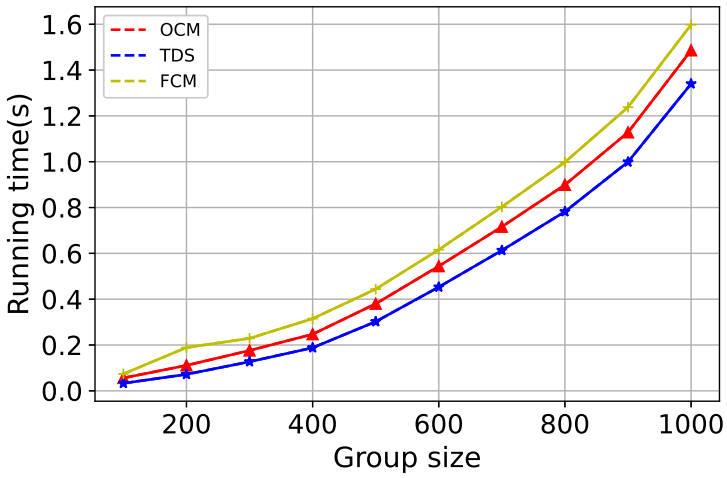


Figure 3.9: Scalability comparison

3.5.6 Time efficiency

To test the scalability of my model and strategy, I randomly generated ten groups of online orders and in-store customers with different quantities,recorded and compared the running time of different delivery strategies and compensation schemes, and the results are shown in the following Fig. 3.9. It was found that the operational efficiency of all three increased exponentially. Nevertheless, the running time is still within 2s at the scale of 1000 data sets (1000 crowd-shippers and 1000 online orders), showing good scalability of the system. In addition, the running time of my strategy is between the running time of TDS and OCM. my OCM performs better than FCM in terms of scalability. Moreover, I analyze the impact of ω_2, the weight parameter that considers the detour distance in the matching process, on CRR and RR, to provide a reference for commercialization practice, which can choose the appropriate value of ω_2 according to the actual situation to meet different realistic needs. In summary, the proposed CCS with OCM outperforms TDS and FCM both in cost-effectiveness and time-efficiency. Furthermore, it is flexible to do modifications to the parameter ω_2 based on different market demands.

3.6 Summary

This chapter proposes an optimization framework to optimize the CDS system's order matching and crowd-shipper compensation. I use a binomial logit DCM for predicting crowd-shipper acceptance probabilities. By leveraging the provided probability prediction, the proposed stochastic optimization model can explicitly take the influence of the amount of compensation on individual crowd-shippers' acceptance probability into the decision-making process and compute optimal individual compensations for crowd-shippers to minimize the expected system-level delivery costs. Given the performance on delivery cost reduction and participation incentives brought by the dynamic compensation scheme, the proposed approach enables retailers to attract and maintain a stable base of crowd-shippers, therefore, provides the opportunity for long-term delivery service optimization.

However, considering that in the crowdsourced delivery system, the demand side, namely the arrival of orders, also varies with time. In a dynamic environment, without adequate data to predict future demand, I attempt to approach the issue from the perspective of order matching time, proposing to achieve the system's optimality objective in a dynamic setting.

Chapter 4

Dynamic Order-driver Matching Framework Design for Crowd-Sourced Delivery Considering the Drivers' Acceptance Uncertainties

This chapter consider a system operating in a dynamic and stochastic setting: both in-store customers and online orders arrive continuously throughout the day, and the in-store customers are free to decide whether to accept or reject the assigned online orders. To address the inherent matching challenges presented in this system, I propose a two-stage dynamic order-driver matching framework, which processes the a order postponing algorithm integrating the concept of "Value of Time", and a formal matching and compensation decision process employing a well designed binomial logit model to predict drivers' order acceptance probability.

4.1 Introduction

Impacted by the Covid-19 pandemic, the work-from-home model, introduced to mitigate virus spread, has profoundly altered people's modes of production and daily life. One notable effect is

that customers have more likelihood purchasing goods online, resulting in a significant surge in e-commerce growth (Sardjono, Selviyanti, Mukhlis, & Tohir, 2021). These online consumers are in pursuit of cost-effective, flexible, and rapid urban logistics solutions. Concurrently, due to escalating costs associated with employing full-time delivery drivers [1] and the global economic downturn, enterprises are striving to curtail expenses, seeking affordable and efficient delivery business models. Inspired by the shared economy paradigm, the crowd-sourced delivery system, which leverages ordinary individuals either as an adjunct or replacement to professional fleets for order deliveries, has received widespread attention due to the economic, social, and environmental benefits it provides (Le et al., 2019). Additionally, it significantly reduces delivery times by tapping into a readily available workforce of drivers who can deliver goods as they go about their daily routines. Furthermore, it presents a flexible and accessible source of extra income for those drivers, which could be crucial in times of economic hardship (Mckinnon, 2016).

In recent years, many scholars have devoted their efforts to the major problems in crowd-sourced delivery, such as driver and order matching (Boysen et al., 2022), route planning (Macrina, Di Puglia Pugliese, Guerriero, & Laganà, 2017), mobile depot settings (Macrina et al., 2020), or exploring the use of different types of drivers for delivery, such as taxi drivers (Raviv & Tenzer, 2018), commuters (Lin, Nishiki, & Tavasszy, 2020), drivers (Dahle, Andersson, Christiansen, & Speranza, 2019), etc. However, it is worth noting that most of their work is based on an assumption that drivers will accept matched orders, or through the implementation of specific mechanism designs, such as, Archetti et al. assume that an in-store customer announces his willingness to deliver goods in advance, Dahle et al. set up personal compensation threshold to restrict the lower bound of acceptable compensation, and **?** use trust-generating mechanisms, scoring, and bonus systems to increase the reliability of crowd-shippers. For some other works, like considering the driver's service area (Arslan et al., 2019), reducing the driver's detour distance (P. Chen & Chankov, 2017), setting up transfer stations (Macrina et al., 2020), etc., are the primary methods considering drivers' preference.

With the consideration of driver acceptance behaviors, based on (Hou & Wang, 2021)'s work, to capture the behavioral logic of drivers accepting delivery requests, (Hou et al., 2022) uses a set

[1] https://www.bls.gov/news.release/eci.nr0.htm

of real data and employs a binomial logit model to predict whether drivers are willing to deliver the orders assigned to them. This paper proposes a two-stage optimization model, which separately calculates the optimal matching and compensation results. Nevertheless, the above-mentioned papers do not consider the dynamic problems existing in the crowd-sourced delivery systems.

In this work, I introduce a two-stage dynamic order-driver matching framework to optimize crowd-sourced delivery systems. The preliminary matching stage selectively postpones non-urgent orders with high marginal delivery costs to a subsequent decision epoch. Subsequently, the formal matching stage systematically performs iterative computations for both matching and compensation within each decision epoch, aiming to achieve optimal order-driver matches and compensation outcomes with the goal of minimizing the system's overall expected delivery costs. To accurately quantify urgency related to the order marginal costs, I define a "Value of Time" function. This function takes into account that as the decision-making time approaches the expected delivery time, the urgency level rises (Dayarian & Savelsbergh, 2020).

The organization of the rest of this paper is as follows: A discrete-event system (DES) model to formulate a CDS system is introduced in Sec.4.2. Sec.4.3 presents the designed order-driver matching framework. The performance of the proposed approach is evaluated through a computational study in Sec.4.4. Finally, Sec.4.5 concludes the paper and states future research directions.

4.2 Discrete event system modeling of a CDS system

In this section, I primarily introduce an in-store customer-based crowd-sourced delivery service, including the components and their relationships. Also, I utilize a discrete event system (DES) model to capture the dynamics influencing system operations.

The major components and their inner-relation in the CDS system are presented in Fig.5.1. And the components details are given follows:

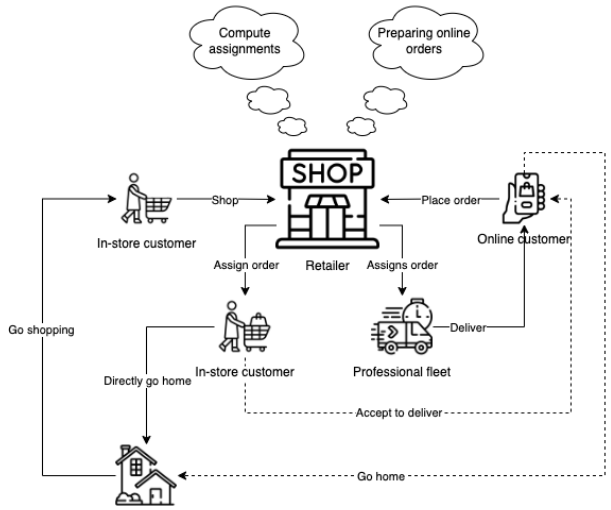


Figure 4.1: Overview of the CDS system

The system primarily comprises three integral components: retailers, in-store customers, and online orders. Retailers provide shopping and delivery services to the surrounding areas, catering to both in-store and online customers. Those in-store customers willing to assist retailers with package deliveries register as drivers through mobile devices. Employing various matching and compensation algorithms, retailers allocate orders placed by online customers to different drivers, accompanied by diverse forms of compensation. Orders accepted by drivers are successfully delivered, while those rejected are assigned to professional Fleet (PF). Furthermore, considering the time-sensitivity of orders, the system mandates that deliveries must be completed within a time frame of T after the order placement. Failing to do so classifies the order as late, adversely affecting service quality.

Given that the choice behavior of drivers significantly impacts the operation of the system, I have chosen each driver as the primary entity in the DES (Discrete Event Simulation) model. The events influencing system operations are e^a, e^r, e^d, e^n, e^p, and e^l, representing the drivers' arrival, request receive, request acceptance, rejection, drop-off, and departure events, respectively. Let Y be the state space which consists of the set of possible values of the vector $\boldsymbol{y}(k) = [y_1(k) \dots y_i(k) \dots y_{|I|}(k)]^T$, where $y_i(k)$ represents the state of the driver i after the occurrence of the k^{th} event. The value of the state variable $y_i(k)$ indicates if the driver i:

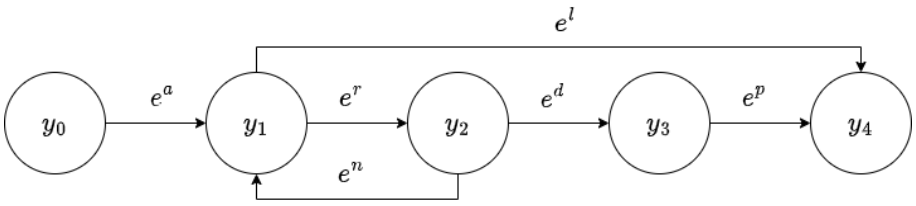


Figure 4.2: State transition diagram

- is free: $y_i(k) = y_0$;
- is available to be matched: $y_i(k) = y_1$;
- is evaluating a delivery request $y_i(k) = y_2$;
- is delivering: $y_i(k) = y_3$;
- is going to her destination: $y_i(k) = y_4$;

The state transitions diagram of a driver i is shown in Fig.4.2. We also give the detailed description of each event, and illustrate the simulation.

The occurrence of the driver arrival event, which is a decision epoch denoted by q, triggers the solution of the proposed dynamic matching framework, and the state of the driver is transited from y_0 to y_1. The driver is regarded as "avaialble" to be matched ($I^q = I^{q-1} \cup \{i\}$). We assume that the driver arrival follows the Poisson process, the elapsed time between an arrival at τ and the next one results to be an exponential stochastic variable with the rate λ_d per time unit. Also the occurrence rate of the driver arrival events is assumed to be time-varying and computed over a time horizon T. For the sake of simplicity and without losing generality, I assume that the horizon T keep consistent to the driver arrival and online order placement sampling period.

The occurrence of request proposal event, transferring the driver's state from y_1 to y_2. The drivers who receives the request, are free to accept or reject. If a request is accepted, the driver get the packed order from pick-up point at the retailer and deliver it before getting home, and the driver is removed from the available driver ($I^{q+1} = I^q / \{i\}$). Moreover, the accepted order is added to the successfully matched order set $\bar{\mathcal{J}}$. If a request is rejected, the driver just waits until the next request proposed to her, or at the latest departure time $\tau + \iota$, where $\iota \sim N(\mu, \sigma)$. And the rejected order is

assigned to PF.

4.3 the proposed approach

In this paper, I introduce a dynamic order-driver matching framework, as shown in the Fig.4.3. This framework consists of two distinct stages: a preliminary matching stage and a formal matching stage. In the preliminary matching stage, I initially utilize the matching model to pre-match available drivers with orders. Orders and drivers that are successfully matched are subsequently handed over to the compensation mechanism, where the compensation scheme with the lowest expected delivery cost is computed. Subsequently, employing the order postpone mechanism, orders that are not urgent and do not necessitate matching in the current decision epoch are postponed to the subsequent decision epoch. The non-postponed orders and available drivers then undergo the aforementioned matching and compensation processes again. The formal matching and compensation results are presented to the drivers to inquire about their willingness to accept the assignment.

We provide a detailed exploration of the matching model, compensation model, and the order postpone mechanism within the framework. For notations and their associated explanations employed throughout the paper, kindly refer to the Table.4.1.

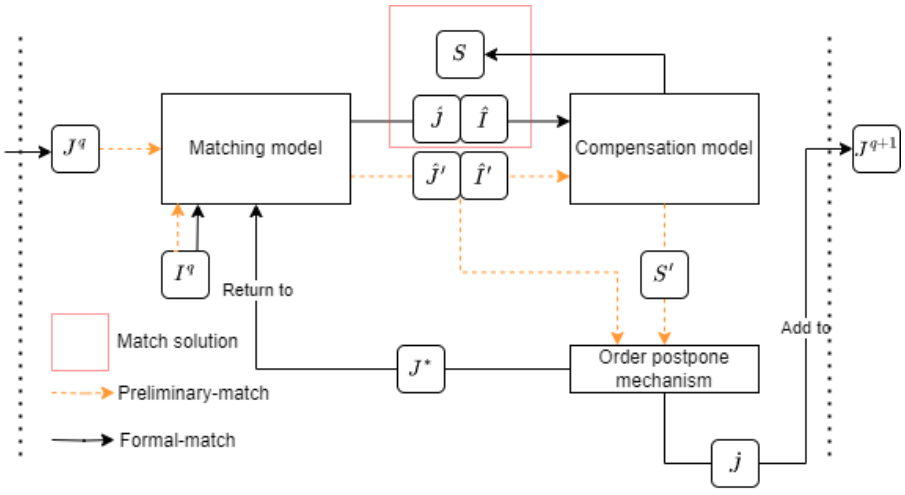


Figure 4.3: Proposed approach framework

Notation	Description
I^q	Set of available drivers at decision epoch q
J^q	Set of available orders at decision epoch q
$\hat{I}$	Set of matched drivers at current decision epoch
$\hat{J}$	Set of matched orders at current decision epoch
$\hat{I}'$	Set of pre-matched drivers at current decision epoch
$\hat{J}'$	Set of pre-matched orders at current decision epoch
$\widetilde{I}$	Set of unmatched drivers at current decision epoch
$\widetilde{J}$	Set of unmatched orders at current decision epoch
$\dot{J}$	Set of postponed orders at current decision epoch
S	Set of compensation paid to matched drivers at current decision epoch
S'	Set of compensation paid to pre-matched drivers at current decision epoch
$D_{i,j}$	Detour distance of driver i deliver order j
d_i	Distance between retailer to driver i's destination
d_j	Distance between retailer to order j's destination
$d_{i,j}$	Distance between driver i's destination to order j's destination
c_j	Original delivery costs of order j by PF
s_i	Decsion variable, indicating the compensation paid to driver i
$x_{i,j}$	Decision variable, indicating the matching between driver i and order j

Table 4.1: Notations and description

4.3.1 Matching model

In this formulation, the objective function, expressed by Eq. (21) has two terms, weighted by the parameter ω_1 and ω_2. The first term equalizes, as much as possible, the number of matched orders. The second term of the objective function avoids too much detours.

$$\min_x \omega_1 \frac{\sum_{\forall j \in J}(1 - \sum_{\forall i \in I^q} x_{i,j})}{|J|} + \omega_2 \sum_{\forall i \in I^q} \sum_{\forall j \in J^q} x_{i,j} \frac{D_{i,j}}{d_i} \tag{21}$$

subject to:

$$D_{i,j} = d_j + d_{i,j} - d_i \tag{22}$$

$$\sum_{\forall j \in J^q} x_{i,j} \leq 1, \ \ \forall i \in I^q \tag{23}$$

$$\sum_{\forall i \in I^q} x_{i,j} \leq 1, \ \ \forall j \in J^q \tag{24}$$

$$x_{i,j} = \{0, 1\}, \quad \begin{cases} \forall i \in I^q \\ \forall j \in J^q \end{cases} \tag{25}$$

The constraint in Eq. (22) compute the detour distance of driver i to deliver order j. And the constraints in Eq. (23) and Eq. (24) guarantee that each order can only be matched with at most one driver, and one driver can only be matched with at most one order. Finally, the constraints in Eq. (25) define the problem variables.

Solving the the matching model, I have the matched driver set $\hat{I}$ and matched order set $\hat{J}$. Let $\widetilde{I}$ and $\widetilde{J}$ be the unmatched driver and order set. Furthermore, I denote the matching results as a mapping $\mu : \hat{I} \longrightarrow \hat{J}$. For example, if a driver $i \in \hat{I}$ is matched to an order $j \in \hat{J}$, where $x_{i,j} = 1$, $\mu(i) = j$, and $\mu(j) = i$.

4.3.2 Compensation mechanism

Given the matching solution, I would like to determine the compensation scheme, paid to each matched driver. Also, as what I mentioned, each driver can accept or reject the assigned order, based on the consideration, I propose a compensation mechanism, to minimize the total delivery costs. We assume that the probabilities of drivers accepting or rejecting specific orders are known based on (Hou et al., 2022)'s binomial logit model. To achieve the objective of minimizing the overall delivery cost of the system, I have constructed a stochastic programming model. Here, the objective function, expressed in Eq.(42), is to minimize the total expected delivery costs $\bar{C}$.

$$\min_{\boldsymbol{s}} \overline{C} = \sum_{\forall i \in \hat{I}} (s_i p_i^d + c_{\mu(i)} p_i^r) + \sum_{\forall j \in \widetilde{J}} c_j \tag{26}$$

$$c_{\mu(i)} = c_0 + \alpha_0 d_{\mu(i)}, \quad \forall i \in \hat{I} \tag{27}$$

$$c_j = c_0 + \alpha_0 d_j, \quad \forall j \in \widetilde{J} \tag{28}$$

$$s_i \leq \omega_3 c_{\mu(i)}, \quad \forall i \in \hat{I} \tag{29}$$

$$s_i \geq 0, \quad \forall i \in \hat{I} \tag{30}$$

Constraints. (27) and constraints (28) define the PF costs to deliver unmatched and rejected orders. Constraints. (44) set up the upper bound of compensation to each driver. And Eq.(45) define the decision variables. The output of compensation mechanism is the set of compensation paid to each driver, indexed by $s_i \in S, \forall i \in \hat{I}$.

4.3.3 Order postponing mechanism

The order postponing mechanism is based on a concept known as the Value of Time (VoT), which is inspired by (Lam & Small, 2001) and primarily depicts how decisions are influenced by time. Here, I define the VoT function as a Exponential function. Given that our study involves order time windows and takes into account the uncertainty of driver acceptance behavior, I observe that the driver's waiting time is less than the order's flexible time. Therefore, postponing the decision-making time of an order might enable matching with better drivers, thus achieving low-cost, high-quality delivery service. Hence, the form of the Exponential function is defined in Eq. (31).

$$\Omega_j = e^{-\delta_j} \tag{31}$$

$$\delta_j = \rho_j - (\tau + d_j/V) \tag{32}$$

where, δ_j denotes the time interval between the decision epoch τ and the latest departure time of the order, defined as Eq.(32).

The order postponing mechanism operates the matching model to identify the optimal match results and compute the expected delivery cost for these matches. One order is arbitrarily removed from the available order list and the computation of matching results and expected delivery cost is repeated. If the difference in expected delivery costs exceeds the VoT under the current time, then that order is removed from the available list. The recalculated matching and compensation scheme is then presented to the driver for evaluation.

Algorithm 1 Postponing Mechanism

procedure PA(μ, S)
  for $\forall j \in J$ **do**
    Compute the expected delivery costs $\overline{C}$
    Compute the original delivery costs c_j
    Remove j from J
    Run the matching model
    Run the compensation mechanism
    Compute the expected delivery costs $\dot{C}$ without j,
    if $(\overline{C} - \dot{C})/c_j > \Omega_j$ **then**
      Postpone j to next decision epoch
    end if
  end for
end procedure

4.4 Computational study

In this section, I conduct a numerical study to verify the performance of the proposed Dynamic order-driver Matching Framework (DMF) in terms of the cost reduction rate, and successfully order-driver matched rate. The experiment were executed on a computer with an Intel Core i7 6-core CPU with 16 GB of RAM, running at 2.6 GHz, using Mac OS X version 14.0.1. The model was implemented in Python version 3.8.5. The optimization models were implemented in Gurobi 10.

4.4.1 Data generation

In order not to lose generality, I try to simulate a natural CDS environment. Firstly, I randomly choose one retailer and 10 surrounding communications in Google Map. Furthermore, the assumed values of some parameters mentioned in Sec.4.3 are shown in Table.4.2, the delivery cost of professional fleet and the compensation of crowd-shippers are inspired by the wage calculator of Meituan, which is a Chinese food delivery company [2], similar to DoorDash and UberEats. Moreover, I conclude three different scenarios covered three possible realistic market environments as follows.

- *Scenario 1* assumes that the distributions of destinations for orders and drivers are essentially congruent. This reflects a real-life situation where a central supermarket provides delivery and in-store shopping services for a nearby densely populated area, wherein the online and

[2] Meituan https://about.meituan.com/en

offline customers are largely the same.

- *Scenario 2* assumes that the distributions of destinations for orders and drivers are completely inverse, that is, areas with a high volume of online orders correspond to areas with fewer offline customers. This mirrors a real-life situation where a central supermarket serves different socio-economic segments of the nearby population. More affluent areas tend to opt for online shopping, while less affluent areas favor in-store selection.

- *Scenario 3* assumes that the distribution of destinations for both orders and drivers is completely random. Serving as a control group, this situation is contrasted with the previous two scenarios.

Additionally, by varying the arrival rates of drivers λ_d and orders λ_o within a unit time T, I aim to validate the feasibility and robustness of the system I proposed in a dynamic environment.

Parameter	Description	Value
μ	Mean waiting time of drivers	5 min
σ	Standard deviation of the waiting time	2
T	Order delivery time window	1 h
C_0	Fixed payment to PF	6 $/item
α_0	Payment rate to PF per mile	1.5 $
ω_1	Weight of order satisfaction	10
ω_2	Weight of detour distance	0.1
ω_3	Upper bound rate of compensation	0.9

Table 4.2: Parameter settings

4.4.2 Performance comparison

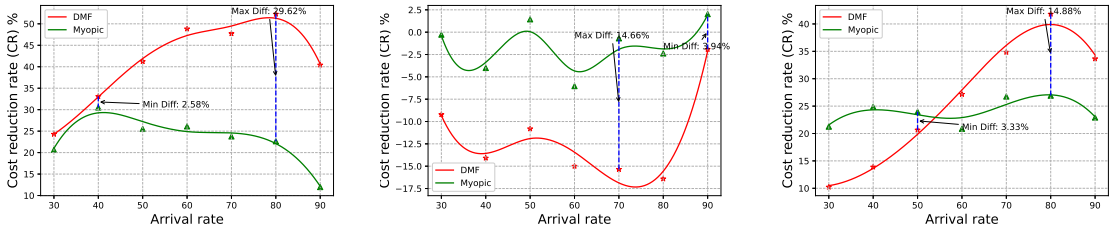
We first define two performance indicators. Subsequently, I compare the performances of two distinct matching approaches under these indicators: the dynamic order-driver matching framework I proposed, and a matching framework that does not take the order postpone mechanism into account (the Myopic approach).

Cost reduction rate (CR)

Defined as the ratio of the costs difference between having all orders delivered by PF within a day and having the driver successfully accept and deliver, to the former. This indicator is for observing the economic benefits that can be realized by our proposed matching framework.

$$CR = \frac{\sum_{\forall j \in \mathcal{J}} c_j - (\sum_{\forall j \in \bar{\mathcal{J}}} s_{\mu(j)} + \sum_{\forall j \in \tilde{\mathcal{J}}} c_j)}{\sum_{\forall j \in \mathcal{J}} c_j} \tag{33}$$

As illustrated in the Fig.4.4, I observe that the dynamic matching framework I proposed significantly outperforms the benchmark Myopic approach in Scenario 1. This disparity becomes increasingly evident, especially as the arrival rates of orders and drivers increase. However, in Scenario 2, the performance of our proposed framework is less than satisfactory. This is primarily due to Scenario 2 examining an extreme situation where the distribution of driver destinations is in direct opposition to the distribution of order destinations. Consequently, even with order delays, no suitable drivers are available for delivery. This results in many orders being assigned to drivers at a higher cost to achieve the objective of maximizing order fulfillment. As for Scenario 3, which represents a more generic situation, our method outperforms the benchmark approach when the arrival rates are higher, consistent with our initial hypothesis.



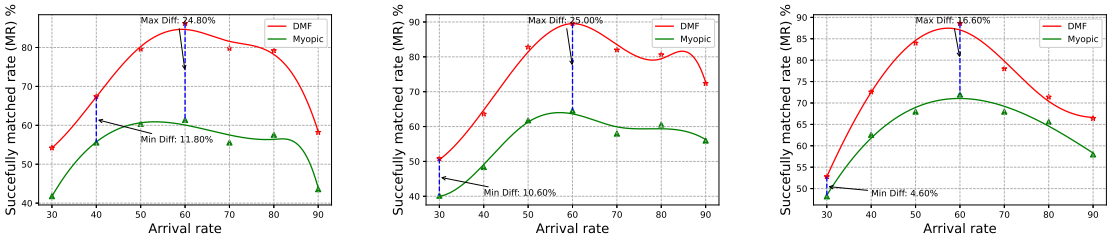
(a) Scenario 1 (b) Scenario 2 (c) Scenario 3

Figure 4.4: Comparison of cost reduction rate between the proposed approach and myopic approach

Successfully driver-order matched rate (MR)

$$MR = \frac{|\bar{\mathcal{J}}|}{|\mathcal{J}|} \tag{34}$$

Defined as the ratio of the number of orders successfully delivered by drivers ($\bar{\mathcal{J}}$) in a day to the total number of orders ($\mathcal{J}$).This indicator is for observing the feasibility of this type of business model, specifically whether it can reduce the number of back-up PF.



(a) Scenario 1 (b) Scenario 2 (c) Scenario 3

Figure 4.5: Comparison of cost reduction rate between the proposed approach and myopic approach

We observed that the trend in the order successful matching rate across the three scenarios is largely consistent. The dynamic matching framework I proposed is significantly superior to the benchmark Myopic approach. This validates our hypothesis that postponing order matching can result in a larger number of orders being assigned to crowd-sourced drivers for delivery.

4.5 Summary

This paper introduces a dynamic order-driver matching framework based on the Value of Time. The framework encompasses three main components: the matching model, the compensation model, and the order postpone mechanism. Initially, the framework pre-matches available orders, filtering out non-urgent ones with higher marginal delivery costs to be processed in the subsequent decision epoch. Subsequently, the remaining orders and available drivers are re-matched, followed by compensation calculations. The derived matching and compensation results are proposed to the drivers to ascertain their willingness to accept. Empirical tests demonstrate that incorporating the order postpone mechanism into the optimization framework can yield excellent cost benefits and societal benefits under scenarios with high order-driver arrival rates and relatively consistent order-driver destination distributions. Specifically, it allows for the allocation of more orders to crowd-sourced drivers, such as in-store customers. In the future, I plan to validate different kernel functions for the Value of Time, such as linear or polynomial functions. Additionally, I aim to gather data on the

supply side, specifically drivers' preferences regarding wait times. We will explore the implications of integrating driver wait durations into our model and their subsequent impact on system benefits.

Chapter 5

Reinforced Stable Matching for Crowd-Sourced Delivery Systems Under Stochastic Driver Acceptance Behavior

Unlike previous chapters that focus on system optimization as the objective, this chapter centers its research on individual interests. It proposes a two-stage mechanism, which first uses the classic stable matching algorithm to ensure each driver receives the best possible order they can accept. Subsequently, a customized compensation mechanism is introduced to enhance the driver's perception, increasing the likelihood of them accepting the assigned order. Experimental results show that this matching mechanism can significantly reduce the rate of order rejection while also improving the system's cost-effectiveness

5.1 Introduction

As the digital marketplace expands, the global e-commerce market is projected to grow at an annual rate of 11.17% between 2023 and 2027, reaching a market size of $5.56 trillion[1]. Within such a sizable market, there is an urgent demand for innovative urban logistics solutions capable of facilitating rapid, cost-effective, and reliable delivery options. Inspired by the sharing economy, a

[1] https://www.statista.com/outlook/dmo/ecommerce/worldwide

novel business model known as Crowd-Sourced Delivery Service (CDS) has emerged. The CDS system employs ordinary individuals, such as commuters, travelers, and shoppers, as a supplement to the professional fleet for delivering online orders (Archetti et al., 2016). By effectively utilizing existing transportation resources, this model has yielded notable economic, social, and environmental benefits (Le et al., 2019).

The compelling advantages of the CDS system have attracted widespread recognition, leading to its broad adoption across major e-commerce platforms, retail giants, and logistics enterprises, including Amazon Flex[2], Walmart Spark[3], and FedEx[4]. However, the operations of such systems are challenging. As pointed out by Said et al. (Said, 2021) and Xu et al. (Xu et al., 2018), there is a high rate of order rejection by drivers on these platforms, leading to operational issues like repeated matching, order delays, and order failures. Such phenomena, where participants are continually incentivized to alter the current allocation results, create a situation termed as *matching instability*, which significantly compromises the system's efficiency and effectiveness.

To achieve *matching stability*, in their seminal paper, Gale and Shapley (Gale & Shapley, 1962) propose a matching algorithm for school admissions and marriage markets in 1962, also known as the Gale-Shapley (GS) algorithm. In 2012, Shapley and Roth (Shapley & Roth, 2012) are awarded the Nobel Prize for their theoretical contributions to stable matching theory. Over the years, this theory has significantly evolved, finding substantial applications in various real-world markets—from labor markets (Roth & Peranson, 1999), school choice programs (Balinski & Sönmez, 1999), to kidney exchange (Roth, Sönmez, & Ünver, 2005b).

In the transportation sector, particularly within shared mobility systems, stable matching algorithms have proven to good performance in enhancing both efficiency and effectiveness. For instance, they have been applied to ride-sharing systems (X. Wang, Agatz, & Erera, 2018), ride-hailing systems (Sühr, Biega, Zehlike, Gummadi, & Chakraborty, 2019) and crowd-sourced last-mile delivery systems (N. Zhang et al., 2023). However, the existing papers have a set of matching-related assumptions. First, the assumption of *deterministic preferences*, where participants are expected

[2]https://flex.amazon.com/
[3]https://drive4spark.walmart.com/
[4]https://www.fedex.com/

to have clear and complete preferences (Roth & Peranson, 1999). Second, the assumption of *rationality*, where participants will always accept pairings that optimize their utilities (Hu & Zhou, 2022). Third, the assumption of *non-transferable utility* suggests that preferences are formed based solely on the inherent attributes of the individuals involved in the pairing, without being affected by external factors or side agreements (Bilancini & Boncinelli, 2014).

These established assumptions are not applicable to CDS systems due to their highly decentralized, stochastic, and dynamic environment. In a more realistic CDS setting, occasional drivers (are abbreviated as: drivers), also known as crowd-shippers, may not be able to accurately and completely state their preferences, and their order acceptance decisions remain uncertain (Hou et al., 2022). Moreover, external factors such as the compensation offered can influence drivers' preferences. To address these challenges, I propose a two-stage matching mechanism. In the first stage, by building on predetermined preferences, I extend the classic Gale-Shapley (GS) algorithm to derive a stable matching solution. In the second stage, equipped with this matching solution, I formulate a stochastic programming model to compute an optimal compensation scheme. This scheme aims to minimize the expected delivery costs while accounting for the uncertain behavior of drivers in accepting orders. This uncertainty is represented as the probability of drivers accepting matched orders. To obtain the probability, I train and compare various Machine Learning (ML) algorithms on a real-world survey dataset. This dataset was collected through a designed questionnaire, yielding 3080 order samples from Shanghai, China. It is observed that XGB algorithm outperforms the other ML algorithms in terms of computational efficiency and prediction accuracy. By utilizing the probability information provided by the XGB algorithm, the proposed two-stage matching mechanism can explicitly model how compensation values affect drivers acceptance behaviors, thereby generating a tailored compensation scheme to enhance the probability of order acceptance by drivers. This in turn reinforces the stable matching, which is central to addressing the challenges inherent in CDS.

The remainder of this paper proceeds as follows. In Sec. 5.2, I review the literature relevant to the matching problem in CDS, followed by a problem description in Sec. 5.3. Sec. 5.4 presents the proposed reinforced stable matching mechanism. While in Sec. 5.5, I introduce the questionnaire design and analyze the collected data. In Sec. 5.6, I conduct computational experiments and Sec. 5.7

concludes this paper.

5.2 Literature review

In this section, I first review literature related to the design of matching approaches in CDS system, then I review literature on the applications of stable matching theory in different transportation fields.

5.2.1 Matching in CDS

Effective matching, corresponding to the process of finding a proper driver to deliver orders, is the key to the success of CDS systems. Nowadays, a considerable amount of research has been devoted to the design of matching approaches for CDS systems. For recent reviews, I refer to the readers to Alnaggar et al. (2021); Pourrahmani and Jaller (2021), where matching approaches are generally classified as deterministic and stochastic.

By assuming a deterministic environment, Archetti et al. (2016) investigate a classical crowd-sourced delivery system, employing occasional drivers to deliver parcels. The matching problem is formulated as an integer programming model that minimizes the total delivery costs. Sampaio et al. (2020) introduce benefits of transfer locations to the crowd-sourced delivery system. To solve the matching problem between orders and drivers, a flow-based mixed integer programming model is proposed to minimize the total delivery costs. A similar setting of introducing intermediate depots is investigated by Macrina et al. (2020), who formulate a matching and routing problem of a crowd-sourced delivery system as an integer linear programming model. The objective is to minimize the total delivery costs and carbon emissions. Similarly, with the goal of minimizing the total costs of delivering all the parcels on time, a mixed integer programming is proposed for multi-driver-multi-order matching and scheduling (W. Chen et al., 2018). Behrend and Meisel (2018) consider a joint decision-making problem that matches supply requests and trip deliveries in an integrated system of crowd-shipping and item sharing. This problem is formulated as an integer programming model that maximizes the total profits of the system. Le et al. (2021) consider a bidding system for CDS system that order requesters declare willingness to pay (WTP) and crowd-shippers state

their expected to-be-paid (ETP). This paper proposes a mixed-integer non-linear model to minimize the total system benefits. Based on a similar bidding system, Boysen et al. (2022) propose a mixed integer programming model to compute the optimal matching between drivers and a bundle of customers, aiming to minimize the total system costs. Arslan et al. (2019) formulate a matching problem between ad hoc drivers and parcel delivery tasks to an integer programming model that minimizes the costs of ad hoc driver matches and dedicated vehicle matches. Tu et al. (2019) consider an online crowd-sourced food delivery system. A mixed-integer programming model is proposed to minimize the total delivery costs and delivery delays. This paper uses a short-sight strategy to achieve dynamically order-task allocations. Behrend et al. (2019) extend their previous work Behrend and Meisel (2018) by allowing drivers to deliver multiple items at once, for which they develop an exact solution approach. To minimize the total carrier travel time and the deviation from the original travel route, Allahviranloo and Baghestani (2019) formulate a mixed-integer programming model and propose a bidding system to handle the pricing and compensation scheme decisions.

CDS systems employ crowd-shippers to deliver online orders on their way back home, introducing a high level of uncertainty into the environment. To address this, Mousavi et al. (2022) investigate the uncertainty surrounding the availability of crowd-shippers and propose a two-stage stochastic integer program to minimize the sum of mobile depot operational costs, penalty costs for postponing deliveries, and crowd-shippers' compensation. Dayarian and Savelsbergh (2020) propose two rolling horizon dispatching mechanisms to dynamically match orders and crowd-shippers, taking into consideration probabilistic information about future online orders and crowd-shippers arrivals. To manage the uncertainty of crowd-shippers' availability, a matching and routing problem is formulated as a two-stage stochastic programming model to minimize total expected delivery costs (Torres et al., 2022). The occasional availability of crowd-shippers is estimated based on historical data. However, crowd-shippers in CDS may not always accept the orders assigned by the system. Acknowledging this acceptance uncertainty, Gdowska et al. (2018) formulate the matching problem between crowd-shippers and packages as a stochastic integer programming model that minimizes the total delivery costs. Recently, Hou et al. (2022) model the uncertainty of crowd-shippers' order acceptance as a discrete choice model, incorporated into a two-stage optimization

framework, aiming to compute a cost-effective optimal matching and compensation scheme.

The aforementioned approaches compute efficient system-wide matching results in CDS, such as cost minimization, profit maximization, and carbon emission reduction. However, they do not consider individual preferences, which may lead to a scenario where system participants, drivers in our context, strategically reject matches in the hope of obtaining more favorable matching outcomes. This scenario is referred to as *matching instability*, which affect the reliability and stability of the system (N. Zhang et al., 2023).

5.2.2 Stable matching in urban mobility fields

To address the issue of matching instability, stable matching theory, developed in 1962 (Gale & Shapley, 1962), has found successful applications in numerous fields, including labor markets (Hou, 2019; Roth & Peranson, 1999), school admissions programs (Balinski & Sönmez, 1999; Biró, Fleiner, Irving, & Manlove, 2010), and kidney exchanges (Roth, Sönmez, & Ünver, 2005a; Roth et al., 2005b).

The application of stable matching theory into CDS is limited. Recently, N. Zhang et al. (2023) attempt to apply stable matching theory to the matching mechanism design for the CDS system. They formulate the matching problem between pre-planned trips and delivery requests as a non-cooperative game and propose two exact algorithms to find stable matchings for CDS matching problems. Additionally, I have observed some efforts to apply stable matching to the fields like ride-sharing and other urban mobility systems. X. Wang et al. (2018) investigate a stable matching problem for ride-sharing systems with incomplete information. They formulate mathematical models that generate stable and nearly stable matching solutions. Peng et al. (2020) develop a stable matching model for ride-sharing systems and propose a stable matching mechanism that incorporates both a payment incentive mechanism and a deferred acceptance algorithm. Yan et al. (2021) design a matching and pricing mechanism to balance system-wide optimality with matching stability. Shurrab et al. (2021) address an EV-to-EV (V2V) charging energy sharing problem and propose a two-layer matching mechanism. This approach leverages the Gale-Shapley algorithm and a user-satisfaction model to efficiently match EVs. Zhao et al. (2019) propose an online stable matching model to address the preference-aware task matching problem in an on-demand taxi dispatching

environment with the goal of maximizing total profit and minimizing the number of blocking pairs. H. Zhang and Zhao (2018) highlight that while most ride-sharing algorithms focus solely on system optimization objectives, very few explicitly consider passengers' preferences for their peers as a matching criterion. Their proposed preference-based model outperforms efficiency-based models in terms of passenger preference satisfaction, with only a moderate loss at the aggregate level.

The aforementioned literature operates under the assumption that all participants are rational with non-transferable utility and have deterministic preferences, such that they will always accept the matching results that maximize their utility. This utility maximizing behavior is assumed to be consistent across different situations. However, in a more realistic CDS setting, the acceptance decisions of drivers are not so predictable, they may choose not to accept matched orders even if such orders could maximize their utilities (Hou et al., 2022). This could be due to personal preferences, external circumstances like traffic or weather conditions, or other unpredictable human factors (Ashkrof et al., 2020; Hou et al., 2022). The difference between theoretical assumption and practical behavior in CDS highlight the need to consider the behavior uncertainties when designing the stable matching mechanism in CDS systems.

Our work focus on generating stable matching results with the consideration of drivers' behavior uncertainty. We propose a matching mechanism that includes a tailored compensation scheme aimed at increasing the probability of drivers accepting matched orders, thereby reinforcing the stability of the matching results. Table. 5.1 summarizes the differences between this paper and the related works.

5.3 Problem Description

In this section, I introduce a CDS system that leverages occasional drivers for order deliveries. We begin with an overview of the system's architecture, highlighting the interrelationships among its components. Following this, I employ a Discrete Event Simulation (DES) model to describe the dynamics of the CDS system. By simulating the behavior of each occasional driver, particularly focusing on order acceptance choice behavior, the DES model assists in evaluating the impact of these behaviors on overall system performance. With the insight gained from the DES, I formulate

the assignment of orders to drivers as a classic bilateral matching problem. For detailed notations and descriptions, please refer to Table. 5.3.

5.3.1 System Overview

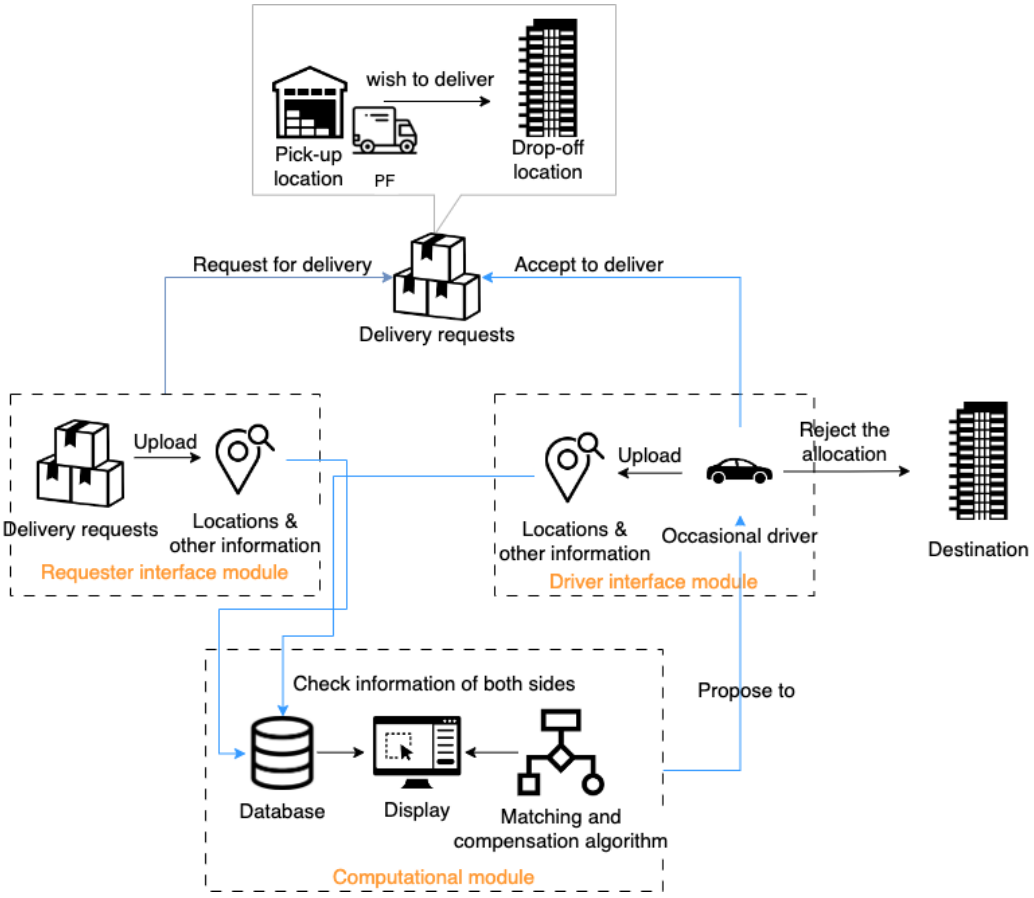


Figure 5.1: Components of the CDS system and their relationships

As illustrated in Fig. 5.1, the CDS system consists of three modules, *computational module*, *requester interface module*, and *driver interface module*. The requester, which can be either an individual or a corporate entity, seeks a proper driver for the delivery of parcels or documents (hereinafter referred to as ”orders”). Each requester uploads the order information to the system database via the requester interaction module, with the pick-up location, drop-off location, and the latest delivery time. Drivers, serving as one-time crowd-shippers, deliver the orders in exchange for

compensations. Each driver declares her availability to deliver orders and uploads her origin, destination, latest departure time and transportation mode through the driver interaction module to the system database. The computational module, leveraging the aggregated data, calculates a matching solution that aligns with both the order's and driver's preferences. This solution is proposed to drivers through the driver interface module, asking them to decide whether to accept or reject the matched orders. The accepted orders are delivered by occasional drivers, while the rejected orders, as well as the unmatched orders are assigned to professional fleets (PFs).

5.3.2 Discrete event simulation model

Let $\mathcal{I}$ be the set of all occasional drivers. For a driver $i \in \mathcal{I}$, the main *events* that affect her states are e_i^l, e_i^r, e_i^n, e_i^p, e_i^d, and e_i^s, where e_i^l represents the event that occasional driver i *logs in* the system. e_i^r denotes that an occasional driver i *receives* an order delivery request. e_i^n and e_i^p denote the *order rejection* and *order acceptance* events, respectively. Finally, let e_i^d be the event that occasional driver i *drops off* the accepted order, while e_i^s represents that the driver i *logs out* and leave the system. Please refer to Table. 5.2 for event descriptions.

Let Y be the state space which consists of the set of possible values of the vector $\boldsymbol{y}(k) = [y_1(k) \ldots y_i(k) \ldots y_{|I|}(k)]^T$, where $y_i(k)$ represents the state of the occasional driver i after the k^{th} event happened. The value of the state variable $y_i(k)$ indicates if the driver i:

- is unavailable: $y_i(k) = y_0$;
- is available to be matched: $y_i(k) = y_1$;
- is evaluating an order: $y_i(k) = y_2$;
- is in motion: $y_i(k) = y_3$;
- is at order's drop-off location: $y_i(k) = y_4$.

The state transitions diagram of an occasional driver i is shown in Fig. 5.2. And the details of some important events are illustrated below.

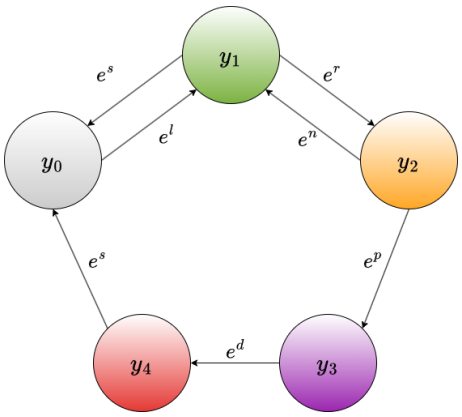


Figure 5.2: State Transition Diagram

Driver log-in and driver log-out event

For each driver $i \in \mathcal{I}$, the occurrence of her log-in event e_i^l changes her state from y_0 to y_1, meaning that the driver i declares her availability to be matched. As regards to the event timing, the log-in event e_i^l is assume to occur at τ_i^l. Let $I \subset \mathcal{I}$ be the set of available drivers whose states are y_1 at τ_i^l, and let $J \subset \mathcal{J}$ be the set of available orders that release before τ_i^l ($\rho_j \leq \tau_i^l$), where ρ_j denotes the release time of each order $j \in \mathcal{J}$. We assume that the order releases follow the Poisson process, the elapsed time between an order release and the next one results to be an exponential stochastic variable with the rate λ. And let T be a fixed time interval, the due time of each order $j \in \mathcal{J}$ is denoted by $\delta_j = \rho_j + T$. It should be noted that the occurrence of driver log-out event is scheduled in $\tau_i^s = \tau_i^l + \Delta\iota_i$, being $\Delta\iota_i \sim \mathcal{N}(\iota_i, \sigma_i^2)$, a Gaussian stochastic variable with expectation ι_i equal to the average waiting time of driver i in the system and with variance σ_i^2.

Order receive event

Given the set of available drivers I and available orders J, the proposed matching mechanism, described in Sec. 5.4.1, generates a set of matched drivers, denoted by $\hat{I}' \subseteq I$, and a set of matched orders, denoted by $\hat{J}' \subseteq J$. Accordingly, an optimal compensation scheme S, indexed by s, for all matched drivers $\hat{I}'$ is determined via an optimization model, described in the following Sec. 5.4.2. With reference to the state diagram in Fig. 5.2, τ_i^r is the instant at which the state of driver i's state changes from y_1 to y_2. Moreover, according to the solution of Algorithm. 2, let $\bar{I} \subseteq I$ be the set of

unmatched drivers and $\bar{J} \subseteq J$ be the set of unmatched orders.

Order acceptance event

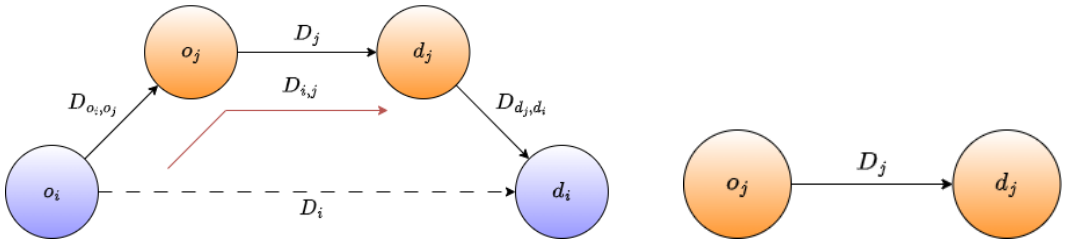
The driver $i \in \hat{I}'$ is free to choose whether to accept the proposal (matched order $j \in \hat{J}'$ with compensation $s \in S$) or not. In the considered model, this freedom is modeled as acceptance probability, which is denoted as $p^a_{i,j}(s)$. If the driver i accepts to deliver order j, meaning the occurrence of event e^p_i, the driver i's state is transferred from y_2 to y_3. The set of drivers who accept to deliver is denoted as $\hat{I} \subseteq \hat{I}'$ and the set of accepted orders is denoted as $\hat{J} \subseteq \hat{J}'$. Note that the drivers who accept to deliver orders coherently update their log-out time corresponding to the order drop-off time.

Order rejection event

For each driver $i \in \hat{I}'$, the probability of rejecting a corresponding matched order $j \in \hat{J}'$ with compensation s is defined as $p^n_{i,j}(s) = 1 - p^a_{i,j}(s)$. If the driver i rejects to deliver order j with compensation s, meaning the occurrence of event e^n_i, the driver i's state is transferred from y_3 back to y_1. Moreover, the set of rejected orders is denoted by $\dot{J}$, where $\dot{J} \cup \hat{J} = \hat{J}'$. Let $\tilde{J} = \dot{J} \cup \bar{J}$ be the orders assigned to PFs.

For simplification, I assume the number of PFs is infinite and they are always on standby at the orders' pick-up locations. Therefore, the delivery time of a PF to deliver a specific unmatched order $j \in \tilde{J}$ is assumed to be $\tau^d_j = \tau^r_i + D_j/V_0$, where D_j denotes the distance between order j's pick-up location o_j and drop-off location d_j, and V_0 denotes the average delivery speed of PFs. Let D_j/V_0 be the estimated travel time of a PF delivering an order j, the route is illustrated in Fig. 5.3(b). A late penalty C^l occurs if $\tau^d_j > \delta_j$, regarding a delayed order j by PF, denoted by $TD^o_j = 1$, otherwise $TD^o_j = 0$. The compensation paid to PFs to delivery a rejected order/unmatched order $j \in \tilde{J}$ is assumed to be proportional, via the parameter α_0, to the delivery route distance plus a fix value, and is determined by:

$$C^o_j = C_0 + \alpha_0 D_j \tag{35}$$



(a) Occasional driver routes

(b) Professional fleet routes

Figure 5.3: Delivery routes of occasional drivers and professional fleets

Order drop-off event

Let $\tau^d_{i,j}$ be the occurring time that driver $i \in \hat{I}$ drops off the accepted order $j \in \hat{J}$, the occurrence of the order drop-off event e^d_i transfers the driver i's state from y_4 to y_5. In our simulation study, I assume that a specific order drop-off event e^d_i is scheduled at $\tau^d_{i,j} = \tau^r_i + D_{i,j}/v_i$, where $D_{i,j}$ denotes the distance of driver i picks up and drops off an order j, the route refers to the Fig. 5.3(a). Let v_i be the speed of driver i depending on the driver i's transportation mode, introduced in Sec. 5.6. A late penalty C^l occurs, if $\tau^d_{i,j} > \delta_j$, regarding a delayed order j, denoted by $TD^d_{i,j} = 1$, otherwise $TD^d_{i,j} = 0$.

The driver log-in event triggers the matching between drivers and orders, which in turn facilitates the order rejection and acceptance events. These events subsequently influence the stability of the matching results. In the following section, I define this two-sided matching problem in more detail and introduce the concept of reinforced matching stability.

5.3.3 Two-sided matching formulation and reinforced matching stability definition

We formulate the CDS driver-order matching problem as a bilateral matching with two sided preferences (Gale & Shapley, 1962). The two-sided matching, originally designed for a marriage market, aims to match every man with a woman in a stable marriage. A marriage is called stable if and only if there are no agents in any match who would both prefer being matched with each other than their current counterparts **?**. Based on the similarity drawn between the CDS system and the marriage market, the problem formulation and its associated stability definition are presented as follows.

Consider a CDS platform, each available order, indexed by $j \in J$, demands a proper available driver $i \in I$ to deliver it. The two finite and disjoint sets of agents in the CDS platform are the set $I = \{1, 2, ..., |I|\}$ of drivers, and $J = \{1, 2, ..., |J|\}$ of orders. Each driver has preferences over the orders, and each order has preferences over the drivers. The preference of each driver i is represented by an ordered list, $L(i)$, on the set J. That is, a driver i's preferences can be of the form $L(i) = 1 \succ 3 \succ 2, ..$, indicating that her first choice is to deliver the order 1, her second choice is to deliver the order 3, and her third choice is to deliver order 2.

Given that drivers may not be able to evaluate all available orders to generate a comprehensive and accurate preference list, I propose that each driver's preferences among all orders are shaped by two key factors: the detour distance denoted as $\mathcal{D}^d_{i,j}$, and the expected compensation represented as $s'_{i,j}$. The detour distance factor is formulated in Eq.(36).

$$\mathcal{D}^d_{i,j} = D_{i,j} + D_{d_j,d_i} - D_i \tag{36}$$

where $D_{i,j}$ denotes the travel distance of driver i delivers order j, the delivery route can be found in Fig. 5.3(a). D_{d_j,d_i} is the distance between order j's drop-off location d_j and driver i's destination d_i, and D_i is the distance of driver i's original travel route from origin o_i to destination d_i. The expected compensation is formulated in Eq.(37).

$$s'_{i,j} = C_1 + \alpha_1 \mathcal{D}^d_{i,j} \tag{37}$$

where C_1 represents the flat fee rewarded to drivers, and α_1 is the fee rate per kilometer. Furthermore, the predetermined preferences $L(i)$ of driver i over available orders J are given by the descending order of the utility value, as formulated in Eq.(38).

$$U_{i,j} = \beta_0 + \beta_1 \mathcal{D}^d_{i,j} + \beta_2 s'_{i,j} \tag{38}$$

In this equation, β_0, β_1 and β_2 are the estimated parameters, where β_0 is the alternative specific constant (known as ASC or intercept), and β_1 and β_2 are the weights of detour distance $\mathcal{D}^d_{i,j}$ and compensation $s'_{i,j}$, respectively.

Similarly, each order j has a preference list $L(j)$ over the driver set I. We assume that each order i's preference is arranged in ascending sort of the estimated travel time $D_{i,j}/v_i$, where $D_{i,j}$ denotes the travel distance of driver i picks up and drops of a given order j, while v_i be the travel speed of driver i, given her transportation mode.

In addition, let $i \succ_j i'$ be the fact that order j prefers i to i', and similarly, I write $j \succ_i j'$ for driver i. And if an individual b is not indifferent between any two alternatives a and a', denoted by $a \sim_b a'$, she has unstrict preferences. For simplification, in this paper, I assume all the preferences list are strict. Formally, I define a CDS system as a tuple $< I, J, \mathcal{L} >$, where $\mathcal{L}$ be the set of all preference lists of both drivers and orders, and the outcome of this CDS system is defined as:

Definition 1. *A matching function μ is a one-to-one correspondence from the set $I \cup J$ onto the set $I \cup J$, that is, $\mu^2(i) = i$ such that if $\mu(i) \neq i$, then $\mu(i)$ is in J and if $\mu(j) \neq j$, then $\mu(j)$ is in I. We refer to $\mu(i)$ as the counterpart of i.*

Note that $\mu^2(i) = i$ means that if order $\mu(i)$ is matched to driver i, then the driver i is matched to order $\mu(i)$. And $\mu(i) = i$ means the driver i is not matched, while $\mu(j) = j$ denotes the unmatched order j. It is also noteworthy to mention that the GS marriage market (Gale & Shapley, 1962) and related work in the transportation domain, such as ride-sharing market (X. Wang et al., 2018), taxi sharing market (Peng et al., 2020), EV charging sharing (Shurrab et al., 2021), there is a prevalent assumption of individual rationality. That is, the preference list of an agent is presumed to be truthful, and every element within this list is deemed acceptable to the agent. In this paper, I consider the inherent uncertainty in a driver's order acceptance behavior and potential irrational decision-making. Consequently, building upon the groundwork of previous research, I introduce a concept of reinforced stable matching that incorporates these considerations.

Definition 2. *A matching μ between occasional drivers and orders is reinforced stable if and only if there is no blocking compensation s and blocking pair (i, j) satisfying the following condition*

- *If a driver k prefers being unmatched to being matched with assigned order $\mu(k)$ with given compensation s, it can be represented as follows:*

$$k \succ_k^s \mu(k) \tag{39}$$

- *There are two agents i and j in CDS system, which belong to the set of drivers and set of orders respectively. Both of them form a new pair, denoted by (i, j). Each of them has been assigned before, the matched mates are denoted by $\mu(i)$ and $\mu(j)$, but for i it prefers being matched with j to $\mu(i)$, while j prefers being matched with i to being matched with $\mu(j)$, the relationships are represented as two in-equations as below, the case is called blocked by pair (i, j).*

$$i \succ_j \mu(j) \tag{40}$$

$$j \succ_i \mu(i) \tag{41}$$

Therefore, Definition. 2 essentially suggests that at reinforced stable, no driver can accept an order with an inappropriate compensation, and no driver can further increase her utility by unilaterally switching to another order. Given these notations and definitions, in the following section, I will introduce the designed reinforced stable matching mechanism.

5.4 Reinforced stable matching mechanism

In this section, I present a two-stage mechanism to achieve reinforced stable matching within the CDS. The first stage aims to generate an initial matching solution ensuring there are no blocking pairs, while the second stage is centered on establishing a compensation scheme for drivers. This scheme considers the acceptance probabilities of drivers to reinforce the matching. Detailed explorations of each stage are provided in Sec. 5.4.1 and Sec. 5.4.2, respectively. Additionally, in Theorem. 5.4.3, I offer a formal proof confirming the reinforced stability of the matching outcomes produced by this mechanism.

5.4.1 First stage: Gale-Shapley matching algorithm

Given a set of available drivers I, and a set of available orders J, with their preference lists $\mathcal{L}$, the goal of the first stage is to generate a matching solution that there is no blocking pairs, refer to Algorithm. 2. The proof of no blocking pairs is given in Sec. 5.4.3

The Algorithm. 2 starts by initializing the sets of matched drivers $\hat{I}' = \emptyset$, matched orders

Algorithm 2 Gale-Shapley stable matching algorithm (GS)

1: **procedure** GS(I, J, $\mathcal{L}$)
2: Initialize the matched drivers and orders $\hat{I}' = \hat{J}' = \emptyset$
3: Initialize the unmatched drivers and orders $\tilde{I} = I$ and $\tilde{J} = J$
4: Initialize the matching $\mu : \emptyset \longrightarrow \emptyset$
5: **while** $\exists j \in \tilde{J}$ & $L(j) \neq \emptyset$ **do**
6: $i :=$ highest-ranked driver in $L(j)$, j is assigned to i, remove i from $L(j)$
7: **if** $i \notin \hat{I}$ **then**
8: $\mu(i) = j$, add i and j into $\hat{I}'$, and $\hat{J}'$, and remove i and j from $\tilde{I}$ and $\tilde{J}$
9: **end if**
10: **if** $i \in \hat{I}$ **then**
11: $j' = \mu(i)$
12: **if** $j \succ_i j'$ **then**
13: $\mu(i) = j$, remove j' from $\hat{J}'$, add j' into $\tilde{J}$, remove j from $\tilde{J}$, and add j into $\hat{J}'$
14: **end if**
15: **end if**
16: **end while**
17: **return** μ, $\hat{I}'$, $\hat{J}'$, $\tilde{I}$, and $\tilde{J}$
18: **end procedure**

$\hat{J}' = \emptyset$, unmatched drivers $\tilde{I} = I$, and unmatched orders $\tilde{J} = J$ (Line 2 and line 3). And I initialize the matching correspondence by $\mu : \emptyset \longrightarrow \emptyset$ (Line 4). The loops work while there exists an order $j \in \tilde{J}$ that its preference list $L(j)$ is not empty (Line 5). In each loop, each order $j \in \tilde{J}$ propose to its preferred driver i in its preference list $L(j)$ and remove i from $L(j)$ (Line 6). Then if the driver i has not been matched $i \in \tilde{I}$, matching builds $\mu(i) = j$, and add i and j into matched sets I and J, respectively, meanwhile, remove i and j from unmatched sets $\tilde{I}$ and $\tilde{J}$ as well (Line 8). If driver i is matched with an order $j' = \mu(i)$ before, and order j is preferred to j based on driver i's preference list $L(i)$, the driver i will accept the preferred order j, and rejects the order j'. The rejected order j' is put back to unmatched order set $\tilde{J}$, and remove from matched order set $\hat{J}'$. The accepted order j is added into $\hat{J}'$, and remove from $\tilde{J}$ (Line 13). The output of the algorithm is a matching function μ, and the sets of matched drivers $\hat{I}'$, matched orders $\hat{J}'$, unmatched drivers $\tilde{I}$, and unmatched orders $\tilde{J}$.

5.4.2 Second stage: Compensation model

Given matched drivers, represented as $\hat{I}'$, matched orders, denoted by $\hat{J}'$, unmatched orders, represented as $\tilde{J}$ and a function μ that maps matched drivers to their respective matched orders,

our aim is to determine the best compensation scheme for these matched drivers using a Stochastic Linear Programming model (SLP). For every matched driver $i \in \hat{I}'$, there is a probability that they will either accept or decline their matched order $\mu(i)$. The cost of delivering this order can be visualized as a binary stochastic variable. This variable follows a Bernoulli distribution $\sim \mathcal{B}(p^a_{i,\mu(i)})$. If a driver declines an order, which happens with a probability of $p^n_{i,j}$, the delivery cost becomes $C^o_{\mu(i)}$. However, if a driver accepts the order (which happens with a probability $p^a_{i,\mu(i)}$), the delivery cost is denoted by a decision variable s_i. This s_i is essentially the compensation given to driver i for the delivery. The objective function Eq. (42) aims to minimize the expected delivery cost of orders that have been matched.

$$\min_{\boldsymbol{s}} \sum_{\forall i \in \hat{I}'} (s_i p^a_{i,\mu(i)} + C^o_{\mu(i)} p^n_{i,\mu(i)}) \tag{42}$$

subject to:

$$C^o_{\mu(i)} = C_0 + \alpha_0 D_{\mu(i)}, \quad \forall i \in \hat{I}' \tag{43}$$

$$\sum_{\forall i \in \hat{I}} s_i \leq \sum_{\forall i \in \hat{I}} \omega C^o_{\mu(i)}, \quad \forall i \in \hat{I}' \tag{44}$$

$$s_i \geq 0, \quad \forall i \in \hat{I}' \tag{45}$$

Eq.(43) define the compensation paid to PFs to deliver rejected orders. Eq.(44) ensure the compensation budget, which must lower than a rate w of original delivery costs of PFs. Eq.(45) define the decision variable. Solving the optimization problem determines the optimal compensation scheme for each matched driver, reported by $s_i \in S$, where $i \in \hat{I}'$.

5.4.3 Reinforced stable matching mechanism: algorithm and proof

The reinforced stable matching mechanism, through integrating the above two stages, is given in Algorithm. 3. Given available drivers I, available orders J, and preference lists $\mathcal{L}$ as input (Line 1). We first initialize the set of drivers who accept to deliver assigned orders as empty ($\hat{I} = \emptyset$), and initialize the set of accepted orders as empty as well ($\hat{J} = \emptyset$) (Line 2). Then I operate the first stage

Algorithm 3 Reinforced Gale-Shapley stable matching mechanism (R-GS)

1: **procedure** R-GS(I, J, $\mathcal{L}$)
2: Initialize the set of drivers $\hat{I} = \emptyset$ who accept to deliver, and accepted orders $\hat{J} = \emptyset$
3: $\mu, \hat{I}', \hat{J}', \tilde{I}, \tilde{J} \longleftarrow GS(I, J, \mathcal{L})$
4: $S \longleftarrow SLP(\mu, \hat{I}', \hat{J}', \tilde{J})$
5: **for** $j \in \hat{J}'$ **do**
6: Propose $(j, s_{\mu(j)})$ to driver $\mu(j)$
7: **if** $\mu(j)$ accepts the order j given compensation $s_{\mu(j)}$ **then**
8: Add driver $\mu(j)$ to set $\hat{I}$, add order j to $\hat{J}$
9: **else**
10: Assign order j to PF
11: **end if**
12: **end for**
13: **return** Matched drivers $\hat{I}$, and matched orders $\hat{J}$
14: **end procedure**

GS algorithm (Algorithm. 2) to compute a initial matching solution, consisting of a matching function μ, matched drivers $\hat{I}'$ and orders $\hat{J}'$, as well as the unmatched drivers $\tilde{I}$ and unmatched orders $\tilde{J}$ (Line 3). Given this matching solution as input, SLP model computes an optimal compensation scheme for each matched drivers $\hat{I}'$ (Line 4). Each order $j \in \hat{J}'$ with computed compensation $s_{\mu(j)}$ is proposed to the matched driver $\mu(j) \in \hat{I}'$, if the driver accepts the order with compensation, I add driver $\mu(j)$ into set $\hat{I}$, and add order j into set $\hat{J}$ (Line 7 and line 8). If the driver $\mu(j)$ rejects the delivery request $(j, s_{\mu(j)})$, the order j is assigned to PFs (Line 9 and line 10). The output of the algorithm is the set of drivers who accept the delivery requests, $\hat{I}$, and the set of accepted orders, $\hat{J}$.

Based on our above definition of reinforced matching stability, in Theorem 5.4.3, I proof that the order-driver matching outcome produced by the R-GS mechanism is reinforced stable.

The matching result of R-GS is reinforced stable.

Proof. To demonstrate the reinforced stability of our final assignment, I need to prove the absence of blocking compensation and blocking pairs.

Assuming there exists a driver k, if she prefers not to be matched rather than being paired with partner $\mu(k)$ under a given compensation value s, then according to line 9 of Algorithm. 3, she will reject the order. Consequently, the matching outcome would shift from $\mu(k)$ to k. As such, I do not encounter a blocking compensation.

Assuming there exists a matching pair (i, j) where driver i and order j prefer each other instead

of their respective assigned matches $\mu(i)$ and $\mu(j)$. For the order j, it must have requested $\mu(j)$ and been rejected based on line 13 of Algorithm. 2. Hence, there are no blocking pairs. Therefore, I can conclude that our R-GS mechanism generates stable matching solution. □

5.5 Questionnaire design and data analysis

To obtain the driver acceptance probability mentioned in the R-GS mechanism in Sec. 5.4, I conduct a series of surveys to gather data on different types of drivers' acceptance behaviors towards various orders, aiming to identify the optimal prediction model. In this section, I discuss how I design a questionnaire to gather relevant data on the order acceptance behavior of occasional drivers. Through comparative analysis, I identify the optimal model for predicting the probability of drivers accepting orders. In addition, validation and estimation of ML algorithms are carried out using Python's Scikit-Learn library, while hyperparameter selection for each ML algorithm is achieved via Python's Scikit-Learn library RandomizedSearchCV package. For more details, please refer to Hou (2023).

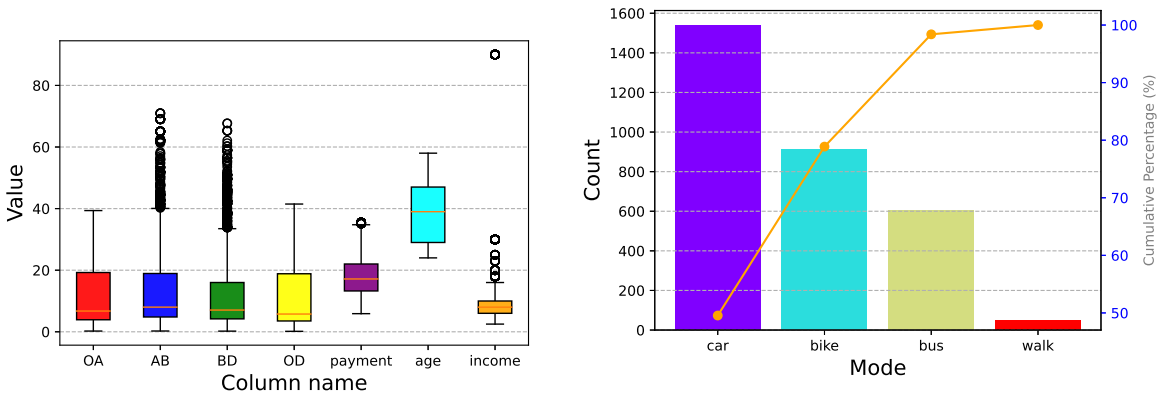
5.5.1 Questionnaire design

From our preliminary and intuitive observations, the main factors influencing drivers' order acceptance can be the distance of the order from their current location (presented by OA), the travel distance of the order (AB), the detour distance for the driver (BD), the original trip distance of driver (OD), and the compensation amount provided by the system (payment). Other factors might include the size (three levels: small, medium, and large) of the package, the season (two levels: winter and summer) and weather conditions (two levels: sunny and rainy) of the day. Additionally, personal attributes of the driver such as their income, gender, and mode of transportation might also influence their decision to accept or decline an order. Based on these insights, I design a questionnaire with 5,000 different sample scenarios, as illustrated in the following Fig. 5.4, and distribute them among various individuals through multiple channels.

5.5.2 Data description and analysis

After discarding invalid or incomplete surveys, I obtained 308 valid questionnaires, including 3080 order samples from Shanghai, China. Fig. 5.5(a) and Fig. 5.5(b) summarizes the descriptive statistics of collected data in China between **December 2022 and April 2023**. In the survey, 47.7% of the respondents were female, while 52.6% were male, and 55.94% are rejected by respondents, and 44.06% are accepted.

Figure 5.4: Questionnaire sample



(a) Numerical data statistics

(b) Categorical data statistics

Figure 5.5: Descriptive Statistics

To predict whether drivers would be willing to accept assigned orders, I systematically compare

several commonly used ML algorithms, consisting of Random Forests (RFC), K-neighbor (KNN), Support Vector Machine (SVM), Extreme Gradient Boosting (XGB), Artificial Neural Network (ANN) and the classic binomial logit model (DCM). We employ the Receiver Operating Characteristic (ROC) curve, to measure the predictive performance and capability of the various behavioral models discussed in this study in forecasting the order acceptance choices of crowd-shippers. The ROC curve of our model, as illustrated in Fig. 5.6, lies proximate to the upper-left corner, underscoring the model's robust predictive capacity.

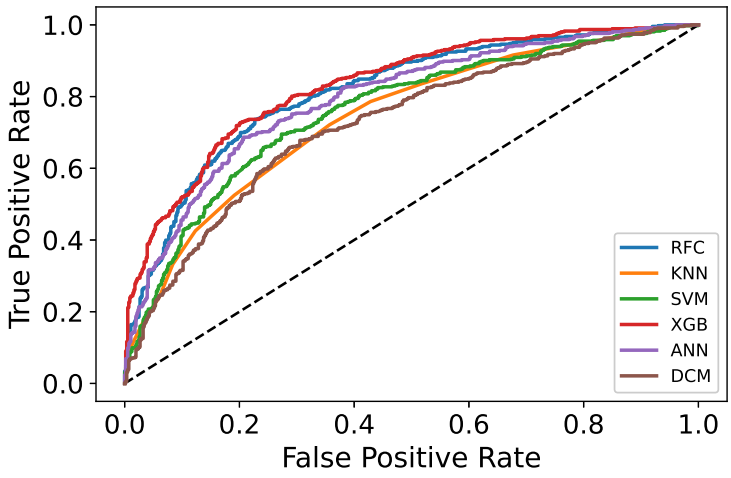


Figure 5.6: The ROC Curve

Furthermore, to prevent overfitting, I randomly split 80% of the survey results as training data and the rest of the observations are regarded as testing data. The training data is utilized to estimate parameters, while the testing data serves to evaluate the model's predictive capacity. To enhance the reliability of the model assessment, I repeated this procedure 200 times and discovered that the results are robust. The F1-score comparative results are given in Table. 5.4

Upon comparison, I find that the XGB algorithm outperforms the other algorithms in terms of prediction accuracy. Therefore, in the computational study, I adopt the XGB mechanism as the behavior model for predicting drivers' order acceptance probability.

5.6 Computational study

In this section, I compare the performance of the proposed reinforced stable matching mechanism (R-GS) with two benchmark mechanisms (GS and OPT, presented in Appendix). This section provides some insights for research in the crowd-sourced delivery and even more in urban mobility field.

5.6.1 Experimental setup

The experiments are executed on a computer with an Intel Core i7 6-core CPU with 16 GB of RAM, running at 2.6 GHz, using Mac OS X version 11.0.1. The GS algorithm is coded in Python version 3.8.5. The optimization models, including the SLP model and OPT model, were implemented in Gurobi 10.1.0. And the aforementioned parameters, including their descriptions and values, are illustrated in Table. 5.5.

For the sake of generalizability and authenticity, within a circular area centered on the downtown coordinates (31.208366, 121.468460) of **Shanghai city** and with a radius of 40km, I uniformly generate 20 location coordinates. The creation of map data utilize the "geopy" module in Python. To simplify the computation of distances between locations, Euclidean distance is employed. Additionally, I uniformly categorize the 20 locations into four equal parts, each marked with a distinct color, to represent drivers' origins, destinations, and the pickup and drop-off locations for orders. This categorization is depicted in Fig. 5.7.

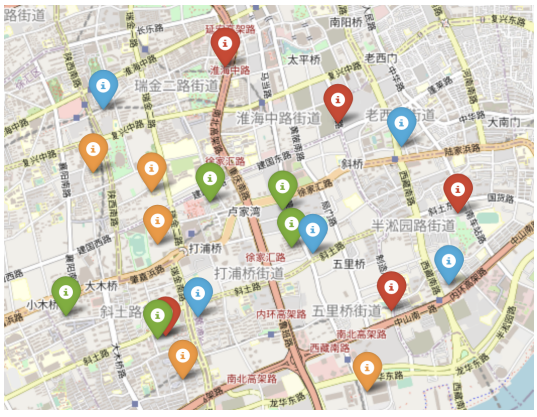


Figure 5.7: Locations distribution: Driver origins: blue points, driver destination: green points, order pick-up location: red points, order drop-off location: orange points

Experiments are conducted on randomly generated instances. For the drivers, their personal details, including age (years old), following a normal distribution $\mathcal{N}(39, 10)$, transportation mode (car, bus, bike, and walk), drawn from uniform distribution, and gender (male and female), drawn from uniform distribution. Also, income (thousands Chinese Yuan ¥) is drawn from a normal distribution $\mathcal{N}(40, 10)$. Their origins and destinations are uniformly spread across the blue and green points, respectively, as marked on the map. As for the orders, their parcel sizes are uniformly chosen from three distinct levels: large, medium, and small. Their pickup and drop-off locations are uniformly distributed across the red and orange points on the map. Pertaining to weather and seasonal information, each instance is associated with a distinct combination of weather and season. These combinations are uniformly generated from pairs of (summer, winter) and (rainy, sunny) conditions. In addition, I assume order arrivals follows Poisson process with rate λ.

Subsequently, to understand the influence of both symmetrical and asymmetrical order and driver quantities on the metrics, I established sets for the number of drivers ($|I| = \{20, 30, 40\}$) and the number of orders ($|J| = \{20, 40, 60, 80, 100\}$). During the subsequent experimental phase, I combine these data scales at random. For every such combination, I randomly generate 10 instances. The results presented are the averaged values from these 10 instances.

5.6.2 Metrics definitions

To compare the performance in system benefits of the proposed mechanism and two benchmark mechanisms, I define the following metrics:

- Order rejection rate:

  As the motivation for this paper, matching stability is the primary subject of investigation. In this regard, I employ the order rejection rate (RR) as the evaluation metric to quantify stability of different matching mechanisms. It is defined as the total number of orders rejected by drivers to the number of orders proposed to drivers.

$$RR = \frac{|\dot{J}|}{|\hat{J}'|} * 100\% \tag{46}$$

- Cost reduction rate:

This paper adopts the cost reduction rate (CR) as the evaluation metric to manifest cost-effectiveness, which is defined as the ratio of the difference in system costs between adopting CDS and not adopting CDS to the cost of the system without employing CDS.

$$CR = \frac{\sum_{\forall j \in J}(C_j^o + C^l TD_j^o) - [\sum_{\forall j \in \hat{J}}(s_{\mu(j)} + C^l TD_{\mu(j),j}^d) + \sum_{\forall j \in \tilde{J}}(C_j^o + C^l TD_j^o))]}{\sum_{\forall j \in J}(C_j^o + C^l TD_j^o)} * 100\% \tag{47}$$

- Order delay rate:

  We employ the order delay rate (DR) as the evaluation metric to gauge the delivery service quality of the matching mechanisms. It is defined as the total number of late orders, delivered by occasional drivers, to the total number of matched orders.

$$DR = \frac{\sum_{\forall j \in \hat{J}} TD_{\mu(j),j}}{|\hat{J}'|} * 100\% \tag{48}$$

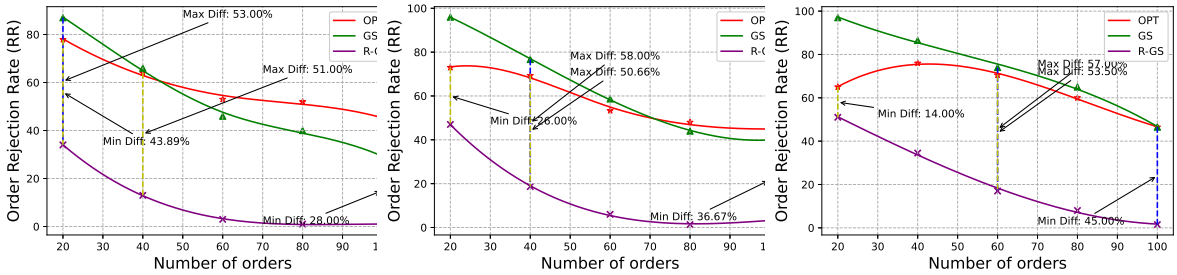
5.6.3 Performance evaluation

Since I do not find an existing approach in the literature that integrates stable matching theory with the consideration of drivers' acceptance uncertainty, I show the benefits of the proposed reinforced stable matching mechanism (R-GS) by comparing its results with those generated by traditional GS algorithm without considering drivers' uncertainty and stochastic mixed integer programming model without considering stable matching. We call these two benchmark approaches as GS and OPT approaches, where GS focus on the matching stability, and OPT approach reflects the traditional view of emphasizing on cost reduction in operation.

Fig. 5.8 shows the comparison results in terms of the average value of order rejection rate (RR) over the proposed R-GS, and two benchmark approaches GS, and OPT, which is regarded as a primary numerical indicator for measuring stable matching performance. Taking $|I| = 30$, and $|J| = 100$ as an example, the proposed R-GS mechanism can achieve 3.33 % average RR while the benchmark approaches OPT and GS achieve 44.67% and 40% average RR respectively. Especially, when $|J| = 40$ and $|I| = 30$, the three approaches show the greatest difference in RR. Compared to the benchmark approaches OPT and GS, R-GS reduce the RR by 58% and 50.66% respectively,

showing a significant advantage. In general, our proposed R-GS mechanism significantly outperforms the other two benchmark approaches across various combinations of driver order quantities. The primary reason is that I not only take into account the individual preferences of the drivers but also introduce a tailored compensation mechanism that reinforces the driver's acceptance of orders, which is directly reflected in the RR rate.

Furthermore, I observe that, while keeping the number of drivers constant and increasing the number of orders, the RR rates of all three approaches exhibit a noticeable declining trend. The rationale behind this is that, whether it's the GS algorithm that considers drivers' order acceptance preferences, the OPT approach that only considers the probability of drivers accepting orders, or the hybrid R-GS mechanism, expanding the pool of available orders for matching makes it easier to find orders that are more suitable for each driver. Additionally, I observe that when the order quantity is approximately over twice the number of drivers ($|J| > 2|I|$), the RR curves of the GS and OPT approaches intersect. Subsequently, the performance of the GS method surpasses that of the OPT approach. The underlying reason for this phenomenon is that when the number of orders is relatively small, the OPT approach tends to find the system's optimal match. Some drivers with longer delivery distances and higher costs might not receive matching requests, leading to a lower RR rate. However, as the number of orders increases, the GS algorithm allows drivers to match with their most preferred orders, thereby achieving a better RR performance.



(a) $|I| = 20$ (b) $|I| = 30$ (c) $|I| = 40$

Figure 5.8: Order rejection rates comparison of three mechanism

Fig. 5.9 shows the comparison results in terms of the average value of cost reduction rate (CR)

over the proposed R-GS, and two benchmark approaches GS, and OPT. Specifically, when the number of available drivers is $|I| = 30$ and the number of available orders is $|J| = 40$, the cost reduction rate of the R-GS mechanism reaches nearly 18%, while the GS algorithm and OPT approach have cost reduction rates of approximately 8% and 12%, respectively. In general, the proposed R-GS mechanism significantly outperforms the two benchmark approaches on CR. Specifically, I observe that R-GS exhibits the best CR when the number of orders is slightly greater than the number of drivers.

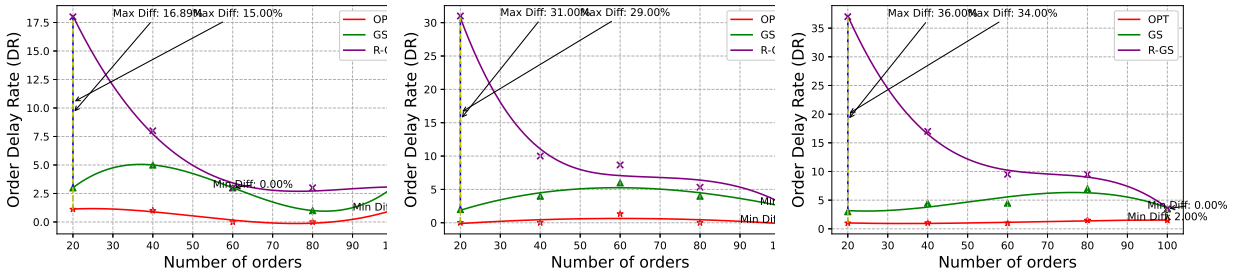
Moreover, I observe that the OPT approach outperforms the GS algorithm when the number of orders is fewer, and the cost-effectiveness of the three approaches tends to converge as the number of orders continues to grow, keeping the number of drivers constant. The primary reason is that when the order quantity is less than the number of drivers $|J| \leq |I|$, the R-GS mechanism, to ensure that drivers accept the matching results calculated by the GS algorithm, will increase the compensation amount for the drivers, thereby elevating the costs. Moreover, when the order quantity is sufficiently large, even if all drivers agree to deliver an order, the CR rate will not exhibit significant variations.

(a) $|I| = 20$ (b) $|I| = 30$ (c) $|I| = 40$

Figure 5.9: Cost reduction rates comparison of three mechanism

Fig. 5.10 shows the comparison results in terms of the average value of order delayed rate (DR) over the proposed R-GS, and two benchmark approaches GS, and OPT. Taking $|I| = 40$, and $|J| = 20$ as an example, R-GS has an average DR of up to 37%, while the other two approaches, OPT and GS have RR of only 1% and 3% respectively. The main reason for this phenomenon is that when the number of drivers exceeds the number of orders $|I| > |J|$, due to our R-GS mechanism prioritizing the acceptance situation of drivers, more compensation is paid to ensure drivers accept the assigned orders (this is also reflected in the comparison of CR). Furthermore, because drivers

have varying speeds and need additional distance to pick up orders, compared to the OPT and GS approaches, which don't prioritize increasing the likelihood of drivers accepting orders, most of these assigned orders get rejected and are quickly and directly delivered by the backup PF, resulting in better performance in terms of DR. It's worth noting that when the number of orders exceeds the number of drivers $|I| \leq |J|$, R-GS shows a clear downward trend in terms of DR. When the number of orders is sufficiently large, for instance, $|I| = 20$, and $|J| = 100$, the DR of R-GS drops to the same level as GS at 3%, and the difference with OPT is only 2%.



(a) $|I| = 20$ (b) $|I| = 30$ (c) $|I| = 40$

Figure 5.10: Order delay rates comparison of three mechanism

These phenomenons aligns with our original intentions, assumptions, and considerations when designing the R-GS mechanism. Because during the design of the mechanism, by integrating the driver behavior prediction model with the optimization model, I generate a tailored compensation scheme, maximizing the probability of drivers accepting orders. Achieving the lowest order rejection rate is thus unsurprising. At the same time, due to the use of cost effective occasional drivers instead of PFs for deliveries, the overall system cost is also reduced.

5.7 Conclusion and future work

In this study, I propose an reinforced stable matching mechanism which integrates stable matching theory with a driver behavior model to realize an efficient and stable driver-order matching outcome in the CDS system. This mechanism has two stages: in the first stage, I extend the classical Gale-Shapley (GS) algorithm to derive an initial stable matching result. Subsequently, in the second

phase, I introduce a stochastic linear programming model designed to generate an optimal compensation scheme while minimizing the expected delivery costs. By implementing this compensation mechanism, I augment the probability of drivers accepting the matching outcome, thereby achieving a superior stable match, termed as 'Reinforced Stability'.

Considering that certain drivers might be willing to deliver multiple orders, the stable matching mechanism in this one-to-many delivery model could be a good research direction. Specifically, when a driver has already accepted a certain order, their behavior towards accepting the subsequent order emerges as an intriguing domain for exploration. Moreover, given that a driver's order acceptance behavior could be influenced by certain temporal factors, the associated data collection, analysis, and the consequent design of the matching mechanism also stand out as interesting research subjects.

5.8 Appendix

We propose a mixed-integer linear programming model employed for computing the optimal order-driver matching outcomes. This model also takes into account the uncertainty associated with drivers' order acceptance behavior, which serves as a benchmark mechanism and is compared against the R-GS mechanism mentioned in the main text.

The objective function seeks to find the optimal driver-request matching results that minimize the overall expected cost for the system. Let G_1 represent the expected cost of deliveries performed by occasional drivers, consisting of the compensation paid to each driver and the penalty costs, with probability $p^a_{i,j}$, G_2 represent the expected cost of deliveries performed by backup PF, consisting of the compensation paid to each PF and the penalty costs, while the orders are rejected by the occasional driver with probability $p^n_{i,j} = 1 - p^a_{i,j}$, and G_3 represent the cost of unmatched orders delivered by PF, also consisting of the compensation paid to each PF and penalty costs. The decision variable $x_{i,j}$, within the matching model, represents the matching results between drivers and requests.

$$\min_{\boldsymbol{x}} G_1(x) + G_2(x) + G_3(x) \tag{49}$$

$$G_1(x) = \sum_{\forall i \in I} \sum_{\forall j \in J} (s'_{i,j} + C^l TD^d_{i,j}) p^a_{i,j} x_{i,j} \tag{50}$$

$$G_2(x) = \sum_{\forall i \in I} \sum_{\forall j \in J} (C^o_j + C^l TD^o_j) p^n_{i,j} x_{i,j} \tag{51}$$

$$G_3(x) = \sum_{\forall j \in J} (C^o_j + C^l TD^o_j)(1 - \sum_{\forall i \in I} x_{i,j}) \tag{52}$$

subject to:

$$\sum_{\forall i \in I} x_{i,j} \leq 1, \ \ \forall j \in J \tag{53}$$

$$\sum_{\forall j \in J} x_{i,j} \leq 1, \ \ \forall i \in I \tag{54}$$

$$x_{i,j} = \{0, 1\}, \ \begin{cases} \forall i \in I \\ \forall j \in J \end{cases} \tag{55}$$

Eq. (53) ensure each order can only be assigned to as much as one driver, while Eq. (54) ensure each driver can only deliver no more than one order. And Eq. (55) define the decision variable.

Solving the optimization problem determines the optimal driver-order matchings, reported by a mapping $\mu : I \cup J \longrightarrow I \cup J$, if an order j is assigned to a driver $x_{i,j} = 1$ let i be in set $\hat{I}'$, and let j be in set $\hat{J}'$, $\mu(i) = j$, and $\mu(j) = i$. The orders, not assigned to drivers, are proceed by PF. Let the orders be in set $\tilde{J}$, if $\sum_{\forall i \in I} x_{i,j} = 0, \ \forall j \in J$.

Studies	Service platform	Stable matching	Acceptance uncertainty
Peng et al. (2020)Yan, Lee, Chu, Chen, and Luo (2021) X. Wang et al. (2018)H. Zhang and Zhao (2018)	Ride-sharing	✓	✗
Shurrab et al. (2021)	EV charging-sharing	✓	✗
Zhao et al. (2019)	On-demand taxi	✓	✗
N. Zhang et al. (2023)	Crowd-sourced delivery	✓	✗
Arslan et al. (2019)Tu et al. (2019) Behrend et al. (2019)Dayarian and Savelsbergh (2020)Macrina et al. (2020) Torres, Gendreau, and Rei (2022)Behrend and Meisel (2018) Sampaio et al. (2020)W. Chen et al. (2018) Mousavi et al. (2022) Le et al. (2021)	Crowd-sourced delivery	✗	✗
Allahviranloo and Baghestani (2019) Boysen et al. (2022)Archetti et al. (2016) Hou et al. (2022)Gdowska et al. (2018)	Crowd-sourced delivery	✗	✓
This study	Crowd-sourced delivery	✓	✓

Table 5.1: The differences between this paper and the most relevant literature

Table 5.2: Descriptions of Events in the DES model

Events	Descriptions
e_i^l	Log-in event of occasional driver i, occurs at τ_i^l
e_i^r	Order receive event of occasional driver i, occurs at τ_i^r
e_i^p	Order acceptance event of occasional driver i, occurs at τ_i^p
e_i^n	Order rejection event of occasional driver i, occurs at τ_i^n
e_i^d	Order drop-off event of occasional driver i, occurs at τ_i^d
e_i^s	Log-out event of occasional driver i, occurs at τ_i^s

Notation	Description
$\mathcal{I}$, $\mathcal{J}$	Set of all occasional drivers, indexed by i, and set of all online orders, indexed by j
I, J	Set of available occasional drivers $\subset \mathcal{I}$, and set of available orders $\subset \mathcal{J}$
$\hat{I}'$,$\hat{J}'$	Set of matched drivers $\subset I$, and set of matched orders $\subset J$
$\hat{I}$,$\hat{J}$	Set of drivers accepting to deliver orders $\subset \hat{I}'$, and set of accepted orders $\subset \hat{J}'$
$\bar{I}$, $\bar{J}$	Set of unmatched drivers $\subset I$, and set of unmatched orders $\subset J$
$\dot{I}$, $\dot{J}$	Set of drivers rejecting to deliver orders $\subset \hat{I}'$, and set of rejected orders $\subset \hat{J}'$
$\tilde{J}$	Set of orders assigned to PFs $= \bar{J} \cup \dot{J}$
S	Set of compensation proposed to matched drivers $\hat{I}'$, indexed by s
o_i, d_i	Origin and destination of driver $i \in \mathcal{I}$
o_j, d_j	Pick-up location and drop-off location of order $j \in \mathcal{J}$
ρ_j, δ_j	Release time and due time of order $j \in \mathcal{J}$
T	A fixed time interval indicating the time window of orders
$\Delta\iota$	A fixed interval indicating the patience (waiting time) of drivers in CDS
$L(a)$	Preference list of an agent a
D_i	Distance of original travel route of occasional driver $i \in \mathcal{I}$
D_j	Distance between pick-up location and drop-off location of order $j \in \mathcal{J}$
D_{o_i,o_j}	Distance between origin o_i of driver $i \in \mathcal{I}$ and pick-up location o_j of order $j \in \mathcal{J}$
D_{d_j,d_i}	Distance between drop-off location d_j of order $j \in \mathcal{J}$ and destination d_i of driver $i \in \mathcal{I}$
$D_{i,j}$	Distance of driver $i \in \mathcal{I}$ to deliver order $j \in \mathcal{J}$
$\mathcal{D}_{i,j}^d$	Detour distance of driver $i \in \mathcal{I}$ to deliver order $j \in \mathcal{J}$
V_0	Delivery speed of PF
v_i	Delivery speed of driver $i \in \mathcal{I}$
C^l	Penalty for delayed order
C_j^o	Delivery cost of delivering order $j \in \tilde{J}$ by PF
TD_j^o	Delayed order, 1 if PF deliver order $j \in \tilde{J}$ late, 0 otherwise
$TD_{i,j}^d$	Delayed order, 1 if driver $i \in \hat{I}$ deliver order $j \in \hat{J}$ late, 0 otherwise
$U_{i,j}$	Utility of driver i deliver order j
$p_{i,j}^{a(n)}(s)$	Probability that driver i accepts (rejects) to deliver order j given compensation s
μ	A mapping indicting the matching result
$x_{i,j}$	Decision variable, 1 if driver i is matched to order j, 0 otherwise
s_i	Decision variable, indicating the compensation paid to driver i
$s'_{i,j}$	Driver i's expected compensation of delivering order j

Table 5.3: Notation and description

	mean	std
RF	0.65	0.018
ANN	0.621	0.026
KNN	0.626	0.016
XGB	0.693	0.015
SVM	0.628	0.015
DCM	0.608	0.019

Table 5.4: F1 score comparison of different ML algorithms

Parameter	Description	Value
λ	Order arrival rate	20
ι	Average waiting time of each driver	30min
σ^2	Variance waiting time of each driver	5
C_0	Base delivery cost of professional fleet	¥10
C_1	Base compensation of drivers	¥6
α_0	Delivery cost of professional fleet per kilometer	¥1
α_1	Compensation of drivers per extra kilometer	¥1.1
$\mathcal{V}$	Velocity of professional fleet	40km/h
v_{car}	Velocity of private car	40km/h
v_{bus}	Velocity of public bus	20km/h
v_{bike}	Velocity of bicycle	10km/h
v_{walk}	Velocity of pedestrian	5km/h
β_c	Estimated parameter for compensation	0.73
β_d	Estimated parameter for detour distance	0.85
β_0	Intercept value	- 4.29
C^l	Penalty for late order	¥3
ω	highest compensation rate	0.9

Table 5.5: Parameter settings

Chapter 6

Conclusion and future research plan

This dissertation takes a typical two-sided market, the crowd-sourced delivery system, as an example. Through several different matching mechanism designs, I achieve system optimality and stability under the condition of supply-side behavioral uncertainty. In this chapter, I summarize the main contributions of this dissertation and present some future research directions.

6.1 Summaries of contributions

The main contribution of this dissertation lies in the design of the matching mechanism under the conditions of supply-side uncertainty, taking into account different market settings. Specifically, the contributions are as follows:

6.1.1 Supply side behavior modeling (Chapter 2)

In this chapter, I have practiced how to obtain decision-related data from the supply side in a two-sided market within the context of the crowd-sourced delivery system. This includes questionnaire design, data processing, hyper-parameter selection, model training, validation, testing, as well as a comparison of feature importance. This lays a solid foundation for subsequent and related works. These processes can be applied to other similar two-sided markets.

6.1.2 Optimality of Matching mechanism design in static environment (Chapter 3)

The main pain point of our graduation thesis lies in the uncertainty of supply-side behavior. When designing a matching mechanism, how to take this uncertainty into account becomes an urgent issue to be resolved. Therefore, in this chapter, I design an optimization framework as the foundation for subsequent work. Firstly, considering the satisfaction rate of the demand side and some supply-side preferences (but not individual preferences), I calculate the preliminary matching results. Then, by combining the well-trained behavioral model with the compensation mechanism, I incentivize the supply side through compensation to achieve the expected system objectives.

6.1.3 Optimality of Matching mechanism design in dynamic environment (Chapter 4)

Building on the foundation of Chapter 3, I recognize the dynamic characteristics present in two-sided markets, where participants from both sides continuously enter the market. The static optimal results might not be the best choices in a dynamic setting. Furthermore, considering that in such markets, the participation of the supply side plays a decisive role in the successful operation of the system, I contemplate postpone decisions on the demand side to provide a broader selection pool for the supply side. At the same time, I introduce a ”value of time” function based on the definition of time urgency. By calculating the marginal cost-benefit and time value of each order in the current decision phase, I determine whether it should be postponed.

6.1.4 Stability of matching mechanism design considering supply side uncertainty (Chapter 5)

In this chapter, starting from another important system objective in two-sided markets, matching stability, I propose a definition of reinforced stable matching and a related algorithm. By integrating it with the behavioral model learned in Chapter 2, I suggest using a customized compensation scheme to reinforce the weakly stable matching results under the condition of supply-side behavioral uncertainty. Experimental results show that, under the premise of supply-side behavioral uncertainty, considering that the supply side cannot traverse all participants on the demand side and

provide complete preferences, the algorithm I propose achieves significant improvements in stability without causing much degradation in optimality. This provides a reliable theoretical foundation and implementation method for achieving matching stability in similar two-sided markets.

6.2 Future research plan

The summarized future research directions for my study on Crowd-sourced delivery, one of the two-sided markets, are comprehensive and multi-faceted. Let's break them down into key themes:

- **Improving Computational Efficiency**

  As discussed in Chapter 2 regarding computational efficiency experiments, I find that the time complexity of our proposed two-stage framework is $O(n^2)$. During the solution process, heuristic methods are employed to find a good initial solution. Metaheuristic algorithms, such as Simulated Annealing and Genetic Algorithms, are utilized to rapidly explore the solution space. In terms of decomposition methods: The use of decomposition techniques, such as Lagrangian Relaxation and Benders Decomposition, is applied to break down large-scale problems into smaller, more manageable sub-problems. This approach enables efficient problem-solving by simplifying complex issues and focusing on individual components sequentially or in a manner that facilitates parallel processing.

- **Advanced Time-Series Forecasting**

  In Chapter 4, I address the challenge of a cold start in the dynamic matching framework, where there is a lack of data support. In future work, I plan to acquire historical data on order and driver arrival rates either through collaboration with companies or via survey methods. Advanced time series forecasting methods, such as ARIMA (AutoRegressive Integrated Moving Average) and Convolutional Neural Networks (CNNs), will be employed to predict future arrivals of drivers or orders. This predictive insight will guide current matching decisions and future strategies, enhancing the efficacy and efficiency of the dynamic matching framework in crowd-sourced delivery systems.

- **Dynamic matching with reinforcement learning**

The application of reinforcement learning in crowd-sourced delivery for matching and route planning involves creating a simulated environment where states such as delivery orders and driver locations are defined. The algorithm learns to optimize objectives, such as reducing delivery time or cost, through a series of actions (e.g., accepting orders, planning routes). A reward function provides positive or negative feedback based on the outcomes of these actions (like timely delivery), guiding the model to make more effective decisions. This approach enables continuous learning and adaptation in the ever-changing real-world environment, optimizing the delivery process. Reinforcement learning's adaptability makes it particularly suitable for dynamic and complex systems like crowd-sourced delivery, where conditions and requirements can change rapidly.

References

Allahviranloo, M., & Baghestani, A. (2019). A dynamic crowdshipping model and daily travel behavior. *Transportation Research Part E: Logistics and Transportation Review*, *128*, 175–190.

Alnaggar, A., Gzara, F., & Bookbinder, J. H. (2021). Crowdsourced delivery: A review of platforms and academic literature. *Omega*, *98*, 102139.

Al-Saudi, A., & Himpel, F. (2020). Crowd logistics delivery determinants: A stated-preference survey.

Archetti, C., & Bertazzi, L. (2021). Recent challenges in routing and inventory routing: E-commerce and last-mile delivery. *Networks*, *77*(2), 255–268.

Archetti, C., Savelsbergh, M., & Speranza, M. G. (2016). The vehicle routing problem with occasional drivers. *European Journal of Operational Research*, *254*(2), 472–480.

Arslan, A. M., Agatz, N., Kroon, L., & Zuidwijk, R. (2019). Crowdsourced delivery—a dynamic pickup and delivery problem with ad hoc drivers. *Transportation Science*, *53*(1), 222–235.

Ashkrof, P., de Almeida Correia, G. H., Cats, O., & Van Arem, B. (2020). Understanding ride-sourcing drivers' behaviour and preferences: Insights from focus groups analysis. *Research in Transportation Business & Management*, *37*, 100516.

Ashkrof, P., de Almeida Correia, G. H., Cats, O., & van Arem, B. (2022). Ride acceptance behaviour of ride-sourcing drivers. *Transportation Research Part C: Emerging Technologies*, *142*, 103783.

Balinski, M., & Sönmez, T. (1999). A tale of two mechanisms: student placement. *Journal of Economic theory*, *84*(1), 73–94.

Basık, F., Gedik, B., Ferhatosmanoğlu, H., & Wu, K.-L. (2018). Fair task allocation in crowdsourced delivery. *IEEE Transactions on Services Computing*, *14*(4), 1040–1053.

Behrend, M., & Meisel, F. (2018). The integration of item-sharing and crowdshipping: Can collaborative consumption be pushed by delivering through the crowd? *Transportation Research Part B: Methodological*, *111*, 227–243.

Behrend, M., Meisel, F., Fagerholt, K., & Andersson, H. (2019). An exact solution method for the capacitated item-sharing and crowdshipping problem. *European Journal of Operational Research*, *279*(2), 589–604.

Bei, X., & Zhang, S. (2018). Algorithms for trip-vehicle assignment in ride-sharing. In *Proceedings of the aaai conference on artificial intelligence* (Vol. 32).

Bilancini, E., & Boncinelli, L. (2014). Instrumental cardinal concerns for social status in two-sided matching with non-transferable utility. *European Economic Review*, *67*, 174–189.

Biró, P., Fleiner, T., Irving, R. W., & Manlove, D. F. (2010). The college admissions problem with lower and common quotas. *Theoretical Computer Science*, *411*(34-36), 3136–3153.

Boysen, N., Emde, S., & Schwerdfeger, S. (2022). Crowdshipping by employees of distribution centers: Optimization approaches for matching supply and demand. *European Journal of Operational Research*, *296*(2), 539–556.

Boysen, N., Fedtke, S., & Schwerdfeger, S. (2021). Last-mile delivery concepts: a survey from an operational research perspective. *Or Spectrum*, *43*, 1–58.

Breiman, L. (2001). Random forests. *Machine learning*, *45*(1), 5–32.

Chen, C., Jiao, S., Zhang, S., Liu, W., Feng, L., & Wang, Y. (2018). Tripimputor: Real-time imputing taxi trip purpose leveraging multi-sourced urban data. *IEEE Transactions on Intelligent Transportation Systems*, *19*(10), 3292–3304.

Chen, C., Pan, S., Wang, Z., & Zhong, R. Y. (2017). Using taxis to collect citywide e-commerce reverse flows: a crowdsourcing solution. *International Journal of Production Research*, *55*(7), 1833–1844.

Chen, P., & Chankov, S. M. (2017). Crowdsourced delivery for last-mile distribution: An agent-based modelling and simulation approach. In *2017 ieee international conference on industrial engineering and engineering management (ieem)* (pp. 1271–1275).

Chen, T., He, T., Benesty, M., Khotilovich, V., Tang, Y., Cho, H., ... others (2015). Xgboost: extreme gradient boosting. *R package version 0.4-2*, *1*(4), 1–4.

Chen, W., Mes, M., & Schutten, M. (2018). Multi-hop driver-parcel matching problem with time windows. *Flexible services and manufacturing journal*, *30*(3), 517–553.

Cheng, S., Chen, C., Pan, S., Huang, H., Zhang, W., & Feng, Y. (2022). Citywide package deliveries via crowdshipping: minimizing the efforts from crowdsourcers. *Frontiers of Computer Science*, *16*(5), 1–13.

Cortes, C., & Vapnik, V. (1995). Support-vector networks. *Machine learning*, *20*(3), 273–297.

Dahle, L., Andersson, H., Christiansen, M., & Speranza, M. G. (2019). The pickup and delivery problem with time windows and occasional drivers. *Computers & Operations Research*, *109*, 122–133.

Dayarian, I., & Savelsbergh, M. (2020). Crowdshipping and same-day delivery: Employing in-store customers to deliver online orders. *Production and Operations Management*, *29*(9), 2153–2174.

Ermagun, A., Punel, A., & Stathopoulos, A. (2020). Shipment status prediction in online crowdsourced shipping platforms. *Sustainable Cities and Society*, *53*, 101950.

Ermagun, A., Shamshiripour, A., & Stathopoulos, A. (2020). Performance analysis of crowdshipping in urban and suburban areas. *Transportation*, *47*(4), 1955–1985.

Ermagun, A., & Stathopoulos, A. (2018). To bid or not to bid: An empirical study of the supply determinants of crowd-shipping. *Transportation Research Part A: Policy and Practice*, *116*, 468–483.

Fix, E., & Hodges, J. L. (1989). Discriminatory analysis. nonparametric discrimination: Consistency properties. *International Statistical Review/Revue Internationale de Statistique*, *57*(3), 238–247.

Friedman, J. H. (2001). Greedy function approximation: a gradient boosting machine. *Annals of statistics*, 1189–1232.

Gale, D., & Shapley, L. S. (1962). College admissions and the stability of marriage. *The American Mathematical Monthly*, *69*(1), 9–15.

Galkin, A., Schlosser, T., Capayova, S., Takacs, J., & Kopytkov, D. (2021). Attitudes of bratislava

citizens to be a crowd-shipping non-professional courier. *Transportation Research Procedia*, *55*, 152–158.

Gatta, V., Marcucci, E., Nigro, M., Patella, S. M., & Serafini, S. (2018). Public transport-based crowdshipping for sustainable city logistics: Assessing economic and environmental impacts. *Sustainability*, *11*(1), 145.

Gdowska, K., Viana, A., & Pedroso, J. P. (2018). Stochastic last-mile delivery with crowdshipping. *Transportation research procedia*, *30*, 90–100.

Ghaderi, H., Tsai, P.-W., Zhang, L., & Moayedikia, A. (2022). An integrated crowdshipping framework for green last mile delivery. *Sustainable Cities and Society*, *78*, 103552.

Hou, S. (2019). *A two-sided matching system design for dynamic labor markets* (Unpublished doctoral dissertation). Concordia Institution for information systems engineering.

Hou, S. (2023). Order acceptance choice modeling of crowd-sourced delivery services: a systematic comparative study.

Hou, S., Gao, J., & Wang, C. (2022). Optimization framework for crowd-sourced delivery services with the consideration of shippers' acceptance uncertainties. *IEEE Transactions on Intelligent Transportation Systems*.

Hou, S., & Wang, C. (2021). Matching models for crowd-shipping considering shipper's acceptance uncertainty. In *2021 ieee international conference on autonomous systems (icas)* (pp. 1–6).

Hu, M., & Zhou, Y. (2022). Dynamic type matching. *Manufacturing & Service Operations Management*, *24*(1), 125–142.

Huq, F., Sultana, N., Sarkar, S., Razzaque, M. A., & Tushar, M. H. K. (2019). Optimal worker selection for maximizing quality-of-service of online food delivery system. In *2019 international conference on sustainable technologies for industry 4.0 (sti)* (pp. 1–6).

Jin, J., & Ma, X. (2019). A multi-objective agent-based control approach with application in intelligent traffic signal system. *IEEE Transactions on Intelligent Transportation Systems*, *20*(10), 3900–3912.

Kafle, N., Zou, B., & Lin, J. (2017). Design and modeling of a crowdsource-enabled system for urban parcel relay and delivery. *Transportation research part B: methodological*, *99*, 62–82.

Lam, T. C., & Small, K. A. (2001). The value of time and reliability: measurement from a value

pricing experiment. *Transportation Research Part E: Logistics and Transportation Review*, *37*(2-3), 231–251.

Le, T. V., Stathopoulos, A., Van Woensel, T., & Ukkusuri, S. V. (2019). Supply, demand, operations, and management of crowd-shipping services: A review and empirical evidence. *Transportation Research Part C: Emerging Technologies*, *103*, 83–103.

Le, T. V., & Ukkusuri, S. V. (2019a). Crowd-shipping services for last mile delivery: Analysis from american survey data. *Transportation Research Interdisciplinary Perspectives*, *1*, 100008.

Le, T. V., & Ukkusuri, S. V. (2019b). Influencing factors that determine the usage of the crowd-shipping services. *Transportation Research Record*, *2673*(7), 550–566.

Le, T. V., Ukkusuri, S. V., Xue, J., & Van Woensel, T. (2021). Designing pricing and compensation schemes by integrating matching and routing models for crowd-shipping systems. *Transportation Research Part E: Logistics and Transportation Review*, *149*, 102209.

Lee, D., Derrible, S., & Pereira, F. C. (2018). Comparison of four types of artificial neural network and a multinomial logit model for travel mode choice modeling. *Transportation Research Record*, *2672*(49), 101–112.

Lin, X., Nishiki, Y., & Tavasszy, L. A. (2020). Performance and intrusiveness of crowdshipping systems: An experiment with commuting cyclists in the netherlands. *Sustainability*, *12*(17), 7208.

Macrina, G., Di Puglia Pugliese, L., Guerriero, F., & Laganà, D. (2017). The vehicle routing problem with occasional drivers and time windows. In *Optimization and decision science: Methodologies and applications: Ods, sorrento, italy, september 4-7, 2017 47* (pp. 577–587).

Macrina, G., Pugliese, L. D. P., Guerriero, F., & Laporte, G. (2020). Crowd-shipping with time windows and transshipment nodes. *Computers & Operations Research*, *113*, 104806.

McFadden, D. (1977). Modelling the choice of residential location.

Mckinnon, A. C. (2016). A communal approach to reducing urban traffic levels. *Kuehne logistics university, logistics white paper*.

Mousavi, K., Bodur, M., & Roorda, M. J. (2022). Stochastic last-mile delivery with crowd-shipping and mobile depots. *Transportation Science*, *56*(3), 612–630.

Pakarti, C. R., & Starita, S. (2019). Minimizing urban logistics cost using crowd-shipping. In

Proceedings of the 3rd international conference on vision, image and signal processing (pp. 1–6).

Peng, Z., Shan, W., Jia, P., Yu, B., Jiang, Y., & Yao, B. (2020). Stable ride-sharing matching for the commuters with payment design. *Transportation*, *47*, 1–21.

Pourrahmani, E., & Jaller, M. (2021). Crowdshipping in last mile deliveries: Operational challenges and research opportunities. *Socio-Economic Planning Sciences*, *78*, 101063.

Pugliese, L. D. P., Ferone, D., Festa, P., Guerriero, F., & Macrina, G. (2022). Combining variable neighborhood search and machine learning to solve the vehicle routing problem with crowd-shipping. *Optimization Letters*, 1–23.

Punel, A., Ermagun, A., & Stathopoulos, A. (2018). Studying determinants of crowd-shipping use. *Travel Behaviour and Society*, *12*, 30–40.

Punel, A., Ermagun, A., & Stathopoulos, A. (2019). Push and pull factors in adopting a crowd-sourced delivery system. *Transportation Research Record*, *2673*(7), 529–540.

Raviv, T., & Tenzer, E. Z. (2018). Crowd-shipping of small parcels in a physical internet. *URL: http://www. eng. tau. ac. il/˜ talraviv/Publications/Crowd-shipping% 2of% 20small% 20parcels% 20in% 20a% 20physical% 20internet. pdf. preprint*.

Roth, A. E., & Peranson, E. (1999). The redesign of the matching market for american physicians: Some engineering aspects of economic design. *American economic review*, *89*(4), 748–780.

Roth, A. E., Sönmez, T., & Ünver, M. U. (2005a). A kidney exchange clearinghouse in new england. *American Economic Review*, *95*(2), 376–380.

Roth, A. E., Sönmez, T., & Ünver, M. U. (2005b). Pairwise kidney exchange. *Journal of Economic theory*, *125*(2), 151–188.

Sadilek, A., Krumm, J., & Horvitz, E. (2013). Crowdphysics: Planned and opportunistic crowd-sourcing for physical tasks. In *Proceedings of the international aaai conference on web and social media* (Vol. 7, pp. 536–545).

Said, C. (2021). Uber may stop letting drivers see destinations and name prices. *San Francisco Chronicle. https://www. sfchronicle. com/business/article/Uber-may-stop-letting-drivers-see-destinations-16078491. php*.

Salas, P., De la Fuente, R., Astroza, S., & Carrasco, J. A. (2022). A systematic comparative

evaluation of machine learning classifiers and discrete choice models for travel mode choice in the presence of response heterogeneity. *Expert Systems with Applications*, *193*, 116253.

Sampaio, A., Savelsbergh, M., Veelenturf, L. P., & Van Woensel, T. (2020). Delivery systems with crowd-sourced drivers: A pickup and delivery problem with transfers. *Networks*, *76*(2), 232–255.

Sardjono, W., Selviyanti, E., Mukhlis, M., & Tohir, M. (2021). Global issues: utilization of e-commerce and increased use of mobile commerce application as a result of the covid-19 pandemic. In *Journal of physics: Conference series* (Vol. 1832, p. 012024).

Savelsbergh, M. W., & Ulmer, M. W. (2022). Challenges and opportunities in crowdsourced delivery planning and operations. *4OR*, *20*(1), 1–21.

Schlagwein, D., Schoder, D., & Spindeldreher, K. (2020). Consolidated, systemic conceptualization, and definition of the "sharing economy". *Journal of the Association for Information Science and Technology*, *71*(7), 817–838.

Schneider, W. (1987). Connectionism: Is it a paradigm shift for psychology? *Behavior Research Methods, Instruments, & Computers*, *19*(2), 73–83.

Seghezzi, A., & Mangiaracina, R. (2022). Investigating multi-parcel crowdsourcing logistics for b2c e-commerce last-mile deliveries. *International Journal of Logistics Research and Applications*, *25*(3), 260–277.

Shapley, L., & Roth, A. (2012). Stable matching: Theory, evidence, and practical design. *URL: http://www. nobelprize. org/nobel _ prizes/economic-sciences/laureates/2012/popular-economicsciences2012. pdf*.

Shurrab, M., Singh, S., Otrok, H., Mizouni, R., Khadkikar, V., & Zeineldin, H. (2021). A stable matching game for v2v energy sharing–a user satisfaction framework. *IEEE Transactions on Intelligent Transportation Systems*, *23*(7), 7601–7613.

Simoni, M. D., Marcucci, E., Gatta, V., & Claudel, C. G. (2020). Potential last-mile impacts of crowdshipping services: A simulation-based evaluation. *Transportation*, *47*(4), 1933–1954.

Soto Setzke, D., Pflügler, C., Schreieck, M., Fröhlich, S., Wiesche, M., & Krcmar, H. (2018). Crowdsourced delivery. In *Management digitaler plattformen* (pp. 269–283). Springer.

Sühr, T., Biega, A. J., Zehlike, M., Gummadi, K. P., & Chakraborty, A. (2019). Two-sided fairness

for repeated matchings in two-sided markets: A case study of a ride-hailing platform. In *Proceedings of the 25th acm sigkdd international conference on knowledge discovery & data mining* (pp. 3082–3092).

Tao, J., Dai, H., Jiang, H., & Chen, W. (2021). Dispatch optimisation in o2o on-demand service with crowd-sourced and in-house drivers. *International Journal of Production Research*, *59*(20), 6054–6068.

Torres, F., Gendreau, M., & Rei, W. (2022). Vehicle routing with stochastic supply of crowd vehicles and time windows. *Transportation Science*, *56*(3), 631–653.

Tu, W., Zhao, T., Zhou, B., Jiang, J., Xia, J., & Li, Q. (2019). Ocd: Online crowdsourced delivery for on-demand food. *IEEE Internet of Things Journal*, *7*(8), 6842–6854.

Upadhyay, C. K., Tiwari, V., & Tiwari, V. (2022). Generation "z" willingness to participate in crowdshipping services to achieve sustainable last-mile delivery in emerging market. *International Journal of Emerging Markets*.

van Cooten, C. (2016). Crowdsourced delivery—the traditional delivery method reinvented. *Eindhoven University of Technology¡ https://pure. tue. nl/ws/files/46946377/855853-1. pdf*.

Wang, X., Agatz, N., & Erera, A. (2018). Stable matching for dynamic ride-sharing systems. *Transportation Science*, *52*(4), 850–867.

Wang, Y., Zhang, D., Liu, Q., Shen, F., & Lee, L. H. (2016). Towards enhancing the last-mile delivery: An effective crowd-tasking model with scalable solutions. *Transportation Research Part E: Logistics and Transportation Review*, *93*, 279–293.

Wu, P., Chu, F., Saidani, N., Chen, H., & Zhou, W. (2020). Iot-based location and quality decision-making in emerging shared parking facilities with competition. *Decision Support Systems*, *134*, 113301.

Xu, K., Sun, L., Liu, J., & Wang, H. (2018). An empirical investigation of taxi driver response behavior to ride-hailing requests: A spatio-temporal perspective. *PloS one*, *13*(6), e0198605.

Yan, P., Lee, C.-Y., Chu, C., Chen, C., & Luo, Z. (2021). Matching and pricing in ride-sharing: Optimality, stability, and financial sustainability. *Omega*, *102*, 102351.

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

Zehtabian, S., Larsen, C., & Wøhlk, S. (2022). Estimation of the arrival time of deliveries by occasional drivers in a crowd-shipping setting. *European Journal of Operational Research*.

Zhang, H., & Zhao, J. (2018). Mobility sharing as a preference matching problem. *IEEE Transactions on Intelligent Transportation Systems*, *20*(7), 2584–2592.

Zhang, N., Liu, Z., Li, F., Xu, Z., & Chen, Z. (2023). Stable matching for crowdsourcing last-mile delivery. *IEEE Transactions on Intelligent Transportation Systems*.

Zhao, B., Xu, P., Shi, Y., Tong, Y., Zhou, Z., & Zeng, Y. (2019). Preference-aware task assignment in on-demand taxi dispatching: An online stable matching approach. In *Proceedings of the aaai conference on artificial intelligence* (Vol. 33, pp. 2245–2252).