PREFERENCE LEARNING APPLICATION IN PEER-TO-PEER LOGISTICS PLATFORMS

by
FATEMEH GHOLIZADEHFOTOUHABADI

Submitted to the Sabancı Graduate Business School
in partial fulfilment of
the requirements for the degree of Master of Science in Business Analytics

Sabancı University
July 2021

A PREFERENCE LEARNING APPLICATION IN PEER-TO-PEER LOGISTICS PLATFORMS

Approved by:

Date of Approval: July 12, 2021

ABSTRACT

PREFERENCE LEARNING APPLICATION IN PEER-TO-PEER LOGISTICS PLATFORMS

FATEMEH GHOLIZADEHFOTOUHABADI

BUSINESS ANALYTICS M.Sc. THESIS, July 2021

Thesis Supervisors: Assoc. Prof. Raha Akhavan Tabatabaei, Asst. Prof. Ezgi Karabulut Türkseven

Keywords: Online optimization, Preference learning, Recommendation sets, Bilevel optimization, Peer-to-peer logistics platform

Peer-to-peer logistics platforms iteratively match free agents willing to offer services to the demand requests. In some settings, the platform may not have full information about the agents' request preferences. To reduce this uncertainty, the platform can offer a menu of requests to each agent. By collecting the agents' preference information in each decision period, the platform has access to the historical data of agents. The gathered information helps the platform learn the agents' utility functions over time. We simulate a ride-sharing setting, iteratively generating menus of requests for each driver and using the drivers' preference information in each period to learn the drivers' utility functions and predict their future choices. We show that as we learn the drivers' utility functions, offering too many options might cause the driver to deviate from the platform's optimal request assignment therefore, it is beneficial to decrease the menu size as we learn.

ÖZET

EŞLER-ARASI LOJISTIK PLATFORMLARINDA TERCIH OĞRENME UYGULAMASI

FATEMEH GHOLIZADEHFOTOUHABADI

İŞ ANALİTİĞİ YÜKSEK LİSANS TEZİ, TEMMUZ 2021

Tez Danışmanları: Doç. Dr. Raha Akhavan Tabatabaei, Dr. Öğr. Üyesi Ezgi Karabulut Türkseven

Anahtar Kelimeler: Çevrimiçi eniyileme, Tercih öğrenme, Öneri kümeleri, İki seviyeli eniyileme, Eşler-arası lojistik platformları

Eşler-arası lojistik platformlar, hizmet sunmaya istekli aracıları taleplerle yinelemeli olarak eşleştirir. Bazı durumlarda platform, aracıların tercihleri hakkında tam bilgiye sahip olmayabilir. Bu belirsizliği azaltmak için platform, her aracıya bir istek menüsü sunabilir. Platform, her bir karar döneminde aracıların tercih bilgilerini toplayarak, aracıların geçmiş verilerine erişim sağlar. Toplanan bilgiler, platformun zaman içinde aracıların amaç fonksiyonlarını öğrenmesine yardımcı olur. Bu tezde eşler-arası lojistik platform olarak bir araç paylaşımı sistemini simüle ediyoruz, her sürücü için yinelemeli olarak istek menüleri oluşturuyoruz ve sürücülerin fayda fonksiyonlarını öğrenmek ve gelecekteki seçimlerini tahmin etmek için her dönemde sürücülerin tercih bilgilerini kullanıyoruz. Sürücülerin yardımcı fonksiyonlarını öğrenirken, çok fazla seçenek sunmanın, sürücünün platformun optimal iş eşleştirmesinden sapmasına neden olabileceğini, bu nedenle öğrendikçe menü boyutunu küçültmenin faydalı olduğunu gösteriyoruz.

TABLE OF CONTENTS

LIST OF FIGURES

1. INTRODUCTION

Peer-to-peer logistics systems, including ride-sharing and crowdsourced delivery platforms, enable an inventive way to fulfill logistics services during the past recent years. One of the main challenges in peer-to-peer logistics systems is matching the agents to the requests, and the two popular matching approaches are centralized and decentralized. The most common approach is the centralized matching approach, where the platform assigns the requests to prioritize the demand fulfillment. Although this matching approach leads to a high benefit for the platform, it does not consider the agents' preferences, therefore it can cause lower participation. On the other hand, decentralized systems enable the agents to choose their favorite requests in the set of available requests in the system. In contrast to the first approach, the decentralized method has lower demand fulfillment, leading to multiple agents select one request while some requests may not be selected by any agent.

A sustainable system requires satisfied agents in the long run. To consider the agents' autonomy and increase their participation the platform can provide a personalized menu of requests for each driver. To prioritize both the platform's benefit and the agents' preference, a third approach was presented by Mofidi & Pazour (2019), in which the platform as the leader provides each agent a set of personalized recommendation requests based on his/her preference information. The agents (followers) can select their favorite requests in these menus. After receiving the agents' feedbacks, the platform determines the assignments. This approach benefits the platform by considering the agents' decisions, providing a quick time to match, and reducing the multiple selected or rejected requests.

Considering the agents' preferences is essential to provide recommendation sets with higher benefit and accurately predict the agents' selections. Providing preference information by the agents can be time-consuming or may not be possible due to privacy reasons or the agents' lack of knowledge. Further, these preferences may change from one day to another. In most settings, the agents participate in the system repetitively. Therefore, by interacting with the agents continuously, the platform can collect their historical information. Our objective is to use the historical

data to learn the agents' utility functions and customize our system to them.

This work focuses on a peer-to-peer logistics system that iteratively matches demand requests to the available occasional agents. The agents' utility functions are assumed to be unknown. By offering personalized recommendation sets and gathering the agent's feedback in each iteration, the platform can improve its estimate of the utility functions. Providing recommendation sets can increase the probability that the drivers are offered a request they are willing to select and speeds up the process of learning. However, providing too many options can lead to duplicate selections. More importantly, as we become more certain about the agents' utility functions, offering too many options might lead the agents to select the requests with lower platform benefits. Therefore, our contribution is to find the strategies to offer the optimal menu sizes through learning.

The remainder of this paper is structured as follows. In Section 2 we review the recent literature in peer-to-peer logistics system related to our work. In Section 3 we review our adapted optimization model and in Section 4 we introduce our methodology. In Section 5 we discuss our computational results and in Section 6 we present the conclusions.

2. Literature Review

This review is divided into two streams of literature: literature on peer-to-peer logistics systems, including ride-sharing and crowdsourced delivery platforms, and literature on predicting future preferences by learning utility functions using historical data. We will review each literature in the following subsections.

2.1 Literature on peer-to-peer logistics systems

In the peer-to-peer logistics system review, we focus on dynamic ride-sharing settings that match independent drivers to the demand requests. Agatz, Erera, Savelsbergh & Wang (2012) are one of the first to present the definition, process, and concept characteristics of a dynamic ride-sharing system in exhaustive detail. Based on their description, a dynamic ride-sharing setting is an automated system that matches independent drivers to the demand requests to share a prearranged one-time trip. They outlined the possible optimization challenges in different ride-sharing systems.

Most of the dynamic ride-sharing platforms follow two matching policies, either a centralized or decentralized approach. In settings with a centralized matching policy, the driver's autonomy is not considered. That is, the drivers accept the request assignments determined by the platform. Many centralized models assume the drivers are willing to take any request, assign to them, including Wang, Zhang, Liu, Shen & Lee (2016). Some centralized models specify the drivers' flexibility in the form of different constraints. For instance, Arslan, Agatz, Kroon & Zuidwijk (2019) introduced a centralized model with additional restrictions on the drivers' maximum travel time, maximum distance, and the maximum number of stops they are willing to make. Similarly, in the model proposed by Najmi, Rey & Rashidi (2017), a time window shows the drivers' available times. However, all of these models assume the drivers will accept any request that meets all the determined

restrictions. Therefore, in such models, the drivers' preferences are not considered. On the other hand, in the decentralized models, the drivers can choose their favorite request among the available requests in the setting, using the provided information in the system Nourinejad & Roorda (2016). However, decentralized models do not provide a quick time to match. Further, the drivers' interdependencies are not considered in such models. That is, some requests may be selected by multiple drivers while others selected by none.

Wang, Agatz & Erera (2018) are the first to consider the drivers' preferences in their presented model. The authors introduced the concept of stable matching in the domain of the ride-sharing problem. According to their description, a set of matched driver-rider pairs is defined as stable if no current matched or unmatched drivers and riders in the setting prefer to be matched together. After investigating the impact of considering the stability on the system's performance, they showed that increasing the stability slightly reduces the system's performance; still, it improves the system's sustainability in the long run.

Relevant to our paper is the discussion on driver's autonomy in Agatz et al. (2012). The authors mentioned in the case of the participants' dissatisfaction, the platform might lose its sustainability in the long run. Hence, considering the drivers' behavior is critical. Further, providing preference information is time-consuming and does not consider the interdependency among the drivers. To mitigate the uncertainty about the drivers' preferences and to consider all drivers simultaneously, the platform can offer a personalized recommendation sets to each driver. Most recent literature did not consider the driver's preference, but few works considered the drivers' decision behavior by suggesting a set of demand requests to each driver.

To the best of our knowledge, Mofidi & Pazour (2019) is the first work that has considered drivers' decisions by offering simultaneous recommendation sets to each driver. Their work presented a bilevel optimization model in which the platform leads by providing a personalized set of requests to each driver. The drivers follow by accepting their favorite requests. They converted their presented bilevel optimization problem into an equivalent single-level one, which we will adapt in our paper. Assumed the drivers' behaviors deterministic, they proved that the optimal size of the recommendation set is one. In contrast, in case of uncertainty about the driver's selection, offering more than one suggestion to each driver is beneficial.

More recently, two other works have adapted the proposed optimization model presented by Mofidi & Pazour (2019). Horner, Pazour & Mitchell (2021) developed the mentioned bilevel model by considering the drivers' stochastic behaviors directly in the optimization problem, using the Sample Average Approximation method when

drivers signal their willingness to fulfill requests offered on the recommendation set. In case of uncertainty about the driver's behavior, the authors concluded that offering multiple options to the drivers is beneficial. In contrast to Mofidi & Pazour (2019) and Horner et al. (2021) that considered a single-period setting, Ausseil, Pazour & Ulmer (Ausseil et al.) proposed a Multiple Scenario Approach repeatedly sampling drivers' selections, considering the stochastic supply and demand capacity and the drivers' stochastic behaviors in a multi-period model.

Therefore, none of the mentioned literature on peer-to-peer logistics platforms used the agents' preference information collected by the platform to learn the agents' request preferences.

2.2 Literature on preference learning

For the preference learning literature, we focus on the working paper by Bailey, Karabulut & Pazour (Bailey et al.). They considered an online setting in which freelance agents iteratively select their preference in a set of options. The system does not know the agents' utility functions, so the objective is to learn the utility functions and predict the agents' future selections using their preference information. To learn the utility functions, they have developed two preference learning methods: (1) a projection-based method and (2) a support vector machine (SVM) based method. It was shown that the projection-based algorithm either correctly predicts the future preference or leads to a new estimate strictly closer to the actual utility function. Finally, they have shown that both projection-based and SVM-based methods can learn the utility function well. Although the projection-based method is computationally faster, the SVM-based method provides higher accuracy.

Our paper considers a peer-to-peer logistics system, iteratively matching agents willing to offer services to potential demand requests. The agents' utility functions are unknown to the platform. To mitigate the platform's uncertainty about the agents' selection behaviors, the platform provides a personalized set of requests (menu of requests), so the agents' feedbacks can be received based on their preferred requests on the menus. By collecting the agents' preference information in each iteration, the platform improves its estimation of the agents' utility functions over time. We use the projection-based algorithm developed by Bailey, Karabulut &

Pazour (Bailey et al.) to improve the estimate in the learning process. Finally, we investigate the effect of varying menu sizes on the learning setting.

Therefore, limited to a few works, considered the agents' preferences in their models, but none of the literature on peer-to-peer logistics systems optimized recommendation sets using the agents' preference information gradually gathered by the platform. In contrast to the recent literature, we explicitly study the agents' behavior and use their preference information to learn the agents' utility functions in a multi-period setting over time. By learning the utility functions, the platform can have more accurate predictions from the future request selections and provide recommendation sets that benefit the platform.

3. Problem Description

We consider a peer-to-peer logistics platform that iteratively matches demand requests to the available freelance agents in the system. Each agent has a preference over a set of requests. This preference is based on their utility functions. The agents are assumed not to be controlled by the platform; therefore, the platform's knowledge of their utility functions is incomplete. The aim is to study the agents' decision behavior and learn their utility functions over time.

In our multi-period learning setting, the set of agents are always the same and a new set of requests arrive at the system at each time step. Each matched agent in a decision period will be available in the next period like all other agents, and the unmatched requests in each period will leave the system.

For illustrative purposes, the agents are drivers, and the requests are the customers in a ride-sharing platform. The agent's utility can be a function of different attributes, including ride cost, the distance between the agent's and customer's location, and congestion pricing. However, in our case, each driver's utility function is negative of the distance between the driver's planned location and the customer's pickup/drop-off origin. While the platform accesses the customers' locations, the drivers' planned locations are private and unknown. Therefore, we try to learn the drivers' planned locations and improve our estimation of their utility functions to predict future request preferences more accurately.

The dynamic between the platform and the drivers can be modeled as a Stackelberg game: Mofidi & Pazour (2019) represent a mathematical formulation to model a bilevel decision process in a single period setting. In the upper level, the platform leads by offering a personalized menu of requests (a subset of available requests in the setting) to each driver, while the drivers' selected requests are anticipated based on their estimated utility functions. The offered menus may include the same requests for multiple drivers. After releasing the recommendation sets, the drivers reveal their favorite requests on the offered menus in the lower level problem. The drivers have the option not to choose any request in the menu if the utility of all the requests

in the menu is below a certain threshold. Each request may be selected by more than one driver (collision), one driver (1-to-1 match), or no driver (rejection). The platform determines the matched driver-customer pairs to minimize the platform's loss (including the number of rejections and collisions). The authors showed if the platform knows the driver's utility function with certainty, the optimal menu size is one. Therefore, the presented bilevel problem by Mofidi & Pazour (2019) yields the optimal menus considering the drivers' estimated utility functions. However, our online optimization problem consists of two sequential decision stages in each decision period. In the first stage, the menus are obtained using the optimization problem presented by Mofidi & Pazour (2019) and in the second stage, the drivers' actual preferences are captured to improve our estimate of their utility functions. Thus, the drivers' preference information is gathered gradually to learn the utility functions over time.

3.1 Mathematical Formulation

We adapt the optimization problem presented by Mofidi & Pazour (2019) and use the same notations in our online optimization model. Our model consists of a set of decision periods $t = 1, 2, 3, \ldots, T$. In each single period, we have a set of n drivers $D = \{1, 2, 3, \ldots, n\}$ and a set of m customers, $C = \{1, 2, 3, \ldots, m\}$. In each decision period, the platform sends out simultaneous private menus based on each driver's utility function. If driver j selects customer i from the suggested recommendation menu; the driver benefits u_{ij} from fulfilling request i. We consider all drivers as rational decision-makers trying to maximize their benefit. Therefore, a driver selects the request i with the highest utility value for the driver. Each driver's preference is independent of the other drivers. In some scenarios, if the requests' utilities are below a defined threshold, the driver doesn't choose any options.

To model the defined two-stage decision process, two mathematical formulations are presented by Mofidi & Pazour (2019). First, we review the bilevel optimization framework, afterwards we bring the equivalent single-level problem.

Decision Variables

- x_{ij}: 1 if the platform recommends customer i to driver j ; 0 otherwise.
- y_{ij}: 1 if driver j selects customer i ; 0 otherwise.

- z_i: Integer number of unmatched drivers who select customer i (collision).
- w_i: 1 if customer i is not selected by any driver (rejection); 0 otherwise.

Bilevel Optimization Framework (BLF)

$$\min_{\mathbf{x}} \sum_{i\in C}\sum_{j\in D} L_{ij}y_{ij} + \sum_{i\in C} d_i z_i + \sum_{i\in C} r_i w_i \tag{3.1}$$

s.t.

$$\sum_{i\in C} x_{ij} = \theta_j \qquad \forall j \in D \tag{3.2}$$

$$\sum_{j\in D} y_{ij} \leq z_i + 1 \qquad \forall i \in C \tag{3.3}$$

$$1 - \sum_{j\in D} y_{ij} \leq w_i \qquad \forall i \in C \tag{3.4}$$

$$x_{ij} \in \{0,1\} \qquad \forall i \in C, \forall j \in D \tag{3.5}$$

$$w_i \in \{0,1\} \qquad \forall i \in C \tag{3.6}$$

$$z_i \in \{\mathbb{Z}_{\geq 0}\} \qquad \forall i \in C \tag{3.7}$$

$$\max_{\mathbf{y}} \sum_{i\in C}\sum_{j\in D} u_{ij}yij \tag{3.8}$$

s.t.

$$y_{ij} \leq x_{ij} \qquad \forall i \in C, \forall j \in D \tag{3.9}$$

$$\sum_{i\in C} y_{ij} = 1 \qquad \forall j \in D \tag{3.10}$$

$$y_{ij} \in \{0,1\} \qquad \forall i \in C, \forall j \in D \tag{3.11}$$

The bilevel optimization framework is given by (3.1)-(3.11). The leader's problem is captured by (3.1)-(3.7). The leader's objective function (3.1) minimizes platform's expected loss. In the first term, L_{ij} denotes the platform's loss if driver j selects customer i. d_i denotes the linear collision penalty per unmatched driver who selected customer i and r_i denotes the rejection penalty per customer if no driver selected customer i. Therefore, the platform's objective function captures drivers' decisions and considers the drivers' interdependencies by setting rejection and collision penalties.

Constraint (3.2) ensures driver j is recommended a menu of size θ_j. Constraints (3.3) and (3.4) captures the collision and rejection cases, respectively. Constraints

(3.5)-(3.7) define bounds for each discussed decision variable in the leader problem.

The followers' problem is given in (3.8)-(3.11). u_{ij} denotes driver j's benefit from fulfilling request i. Therefore, the followers' objective function given by (3.8) maximizes this benefit for all drivers. Constraint (3.9) ensures drivers' selections to be a subset of their personalized menu. Constraint (3.10) enforces each driver to select exactly one request. Finally, constraint (3.11) defines the bounds for the binary decision variable y_{ij}.

For the scenarios with the no-choice option, we consider a dummy no-choice customer that has L_{ij} and u_{ij} equal to the threshold, for all $j \in D$. That is, if all of the requests' utilities are less than the dummy no-choice customer's utility, the option with the maximum utility/minimum distance is the dummy no-choice request, and hence the driver does not choose any options.

A bilevel optimization model is computationally expensive. Therefore, Mofidi & Pazour (2019) convert the bilevel optimization problem to an equivalent single-level one. For the details of the reformulation, see Mofidi & Pazour (2019). The followers' objective function given in (3.8) is replaced by the linear constraints given in (3.12)-(3.14) using a new binary variable t_{ij}. In constraints (3.12)-(3.13), the drivers' selection behaviors are captured for all possible alternatives. In the mentioned constraints, the coefficients of x_{ij}, y_{ij}, and t_{ij} are represented as α_{ijk}, β_{ijk}, and γ_{ijk} respectively.

The Followers' Objective Function Reformulation

$$\sum_{i \in C} \alpha_{ijk} x_{ij} - \sum_{i \in C} \gamma_{ijk} t_{ij} \leq 0 \qquad \forall j \in D, \forall k \in K \tag{3.12}$$

$$\sum_{i \in C} \gamma_{ijk} t_{ij} - \sum_{i \in C} \beta_{ijk} y_{ij} \leq 0 \qquad \forall j \in D, \forall k \in K \tag{3.13}$$

$$t_{ij} \in \{0,1\} \qquad \forall i \in C, \forall j \in D \tag{3.14}$$

Therefore, in the first stage of our model, we solve the single-level optimization problem, given by (3.1)-(3.7), (3.9)-(3.11), and (3.12)-(3.14) to determine the personalized menus using the estimates of the drivers' utility functions. That is, the y_{ij} variables are estimated driver-customer assignments and the optimal objective value of this problem is the estimated cost of the platform. In the second stage, the menus are given to the drivers to reveal their actual selections $\bar{y}_{ij}$. Based on these actual selections, we calculate the actual collisions ($\bar{z}_i$) and rejections ($\bar{w}_i$).

3.2 Motivation

One of the main results in Mofidi & Pazour (2019) is that the optimal menu size in case of certainty about the drivers' behaviors is one, nevertheless, if there is uncertainty about the drivers' utility functions, a menu of size two or three is optimal from the platform's perspective. That is there is a value in offering choices in case of uncertainty. In our setting, we start with no prior information, indicating a high level of uncertainty about the drivers' preferences. However, as we learn the drivers' utility functions, the level of uncertainty about the drivers' behaviors decreases. Therefore, we expect that the optimal menu size to offer at the beginning should be different from the optimal menu size offered in the later iterations as we learn. The goal of this research is to investigate the effect of menu size on the learning process.

4. Methodology

Our proposed methodology consists of a series of four modules in each iteration: (1) optimization module, (2) feedback module, (3) learning module, and (4) cost calculation module, as shown in Figure 4.1. The content of each module will be explained in detail in the following subsections.

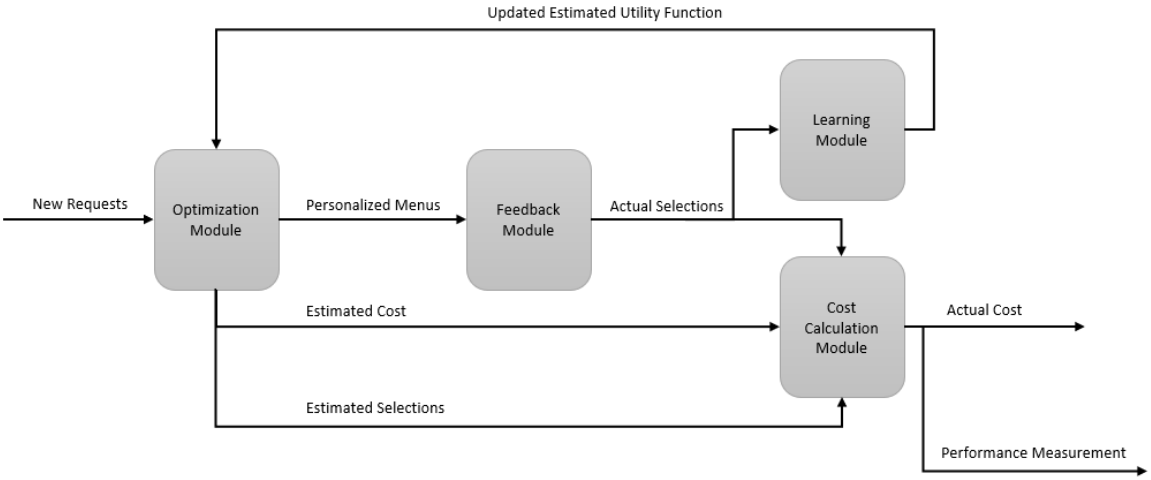

Figure 4.1 Illustration of our algorithm

4.1 Optimization module

In each iteration, the parameter L_{ij} used in the optimization problem given in (3.1)-(3.7), (3.9)-(3.11), and (3.12)-(3.14) is updated by inputting the new set of requests and the current estimate of the utility functions. We study different menu

size strategies through the choice of the parameter θ_j. We present two different strategies: (1) Fixed menu size strategy (FM), and (2) Adaptive menu size strategy (AM).

- **Fixed menu size strategy(FM):** All drivers are provided recommendation sets with the same fixed menu size in all iterations.
- **Adaptive menu size strategy (AM):** All drivers are offered the same menu size in the first iteration, and then each menu size decreases as we learn more about each driver's utility function. That is, for each driver, every time we have at least p correct estimated selections in the last q iterations, we decrease the driver's personalized menu by one until the driver is left with one or two options.

At the end of each iteration, the personalized menus are provided by the optimization module using one of these strategies. Estimated objective value and estimated selections are the outputs of the optimization module.

4.2 Feedback module

In this module, the provided menus are revealed to the drivers, and they select their preferred customer based on their true utility functions. In our case, since we assume the utility functions are distance based, the drivers select the request closest to their ideal point. The output of this module is the drivers' actual selections.

4.3 Learning module

The drivers' feedbacks are used to update the estimated utility functions in this module. We use the preference learning algorithm presented by Bailey, Karabulut & Pazour (Bailey et al.) to improve our estimate in each iteration. The authors' developed algorithm is a projection-based method for learning the Euclidean utility function using the preference information.

As displayed in Figure 4.2, suppose the driver prefers request c_1 to c_2 in the recommendation set, while the platform estimates c_2 as the preferred request. Consider the hyperplane that is equidistant to the actual selection c_1 and the estimated selection c_2. Since we estimated that the driver would prefer c_2, c_2 is closer to our estimated location than c_1, therefore our estimated location lies on the c_2 side of this hyperplane. Similarly, since the driver's preference is c_1, c_1 is closer to the driver's actual location than c_2, therefore the driver's actual location is on the c_1 side of this hyperplane. To improve our estimate, the current estimate is projected to this hyperplane and becomes our new estimate. In Figure 4.2, D, D_{t-1}, and D_t denote the driver's actual location, current estimated location and new estimated location respectively.

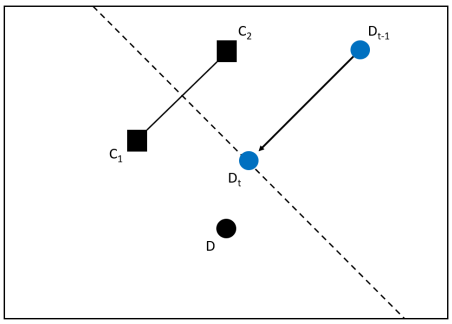

Figure 4.2 Projection-based algorithm

It is shown that the projection method either correctly predicts the future preference or each update leads to a new estimate that is strictly closer to the actual one. This method is straightforward, fast, and computationally cheap; however, the estimate updates only if mistaken, therefore cannot utilize all the information available.

Using the projection-based method in the learning module, we update our estimate to be used in the optimization module in the next iteration.

4.4 Cost calculation module

After receiving the actual selections from the feedback module, the realized objective value (actual cost) will be calculated in the cost calculation module using the equation given in (4.1).

$$\sum_{i \in C} \sum_{j \in D} \bar{L}_{ij} \bar{y}_{ij} + \sum_{i \in C} d_i \bar{z}_i + \sum_{i \in C} r_i \bar{w}_i \tag{4.1}$$

In the first term, $\bar{L}_{ij}$ and $\bar{y}_{ij}$ denote the platform's actual loss (the true distance between customer i and driver j) and the driver's actual selection ($\bar{y}_{ij} = 1$ if driver j selects customer i ; 0 otherwise) respectively. The actual collisions ($\bar{z}_i$) and rejections ($\bar{w}_i$) are calculated based on the actual selections.

To evaluate the performance of the model, we calculate three performance measures:

- The actual objective value: This measure is calculated using the equation given in (4.2),

$$\frac{1}{t'}\sum_{t=1}^{t'}\lambda^t \tag{4.2}$$

  where λ^t is the actual objective value at iteration t given in (4.1).

- The average distance between the drivers' actual and estimated locations (Average Distance). We calculate the average distance using the equation given in (4.3),

$$\frac{1}{t'}\sum_{t=1}^{t'}(\frac{\sum_{j\in D}{\Delta_j}^t}{n}) \tag{4.3}$$

  where ${\Delta_j}^t$ is the Euclidean distance between driver j's actual and estimated location at iteration t.

- The ratio of correctly estimated preferences (Accuracy). Accuracy is calculated by equation (4.4),

$$\frac{1}{t'}\sum_{t=1}^{t'}(\frac{\sum_{j\in D}{\delta_j}^t}{n}) \tag{4.4}$$

  where ${\delta_j}^t = 1$ if driver j's preference is correctly estimated, i.e. $y_{ij} = \bar{y}_{ij}$ for all $i \in C$ at iteration t ;otherwise 0.

Therefore, the performance measurement and calculating the actual costs are the outputs of the cost calculation module.

5. Experimental Results

We develop a computational study to investigate the effect of different parameters on the process of learning the driver's utility function. The ride-sharing platform we assume has the following attributes: We assume the whole system's performance occurs in the unit square, i.e. in the two-dimensional Euclidean space in $[0,1] \times [0,1]$. That is, all drivers and customers in the setting consist of two-dimensional locations (x, y) generated randomly in the unit square, i.e. $x \sim U(0,1)$ and $y \sim U(0,1)$. In all performed experiments, the number of drivers (n) and customers (m) are set as $n = m = 10$. We perform each setting for 100 iterations and replicate each scenario 200 times. We generated 200 random locations as the drivers' actual locations and initial estimated locations and 20000 locations as the customers' locations and used the same set of random numbers in each scenario. This is done to minimize the variation among different strategies caused by random variable selection.

We consider the platform's loss for each driver-customer pair is equal to the Euclidean distance between the estimation of the driver's location and the customer's pickup/drop-off location. For the collision penalty and the rejection penalty, we set, $d_i = r_i = 0.5 \ \forall i \in C$ in the experimented settings. For the scenarios in which the drivers are allowed to select the no-choice option, we define the no-choice threshold as the maximum distance a driver is willing to drive with empty seats to pick up/drop off a rider. Therefore, a driver will choose the no-choice option if the no-choice threshold is less than his utility for all customers. We set $nc = 0.3$ as the no-choice threshold.

We investigate the effect of different parameters in different scenarios with the menu size strategies (FM and AM strategy) applied. To evaluate the performances of these scenarios, we consider three performance measures: (1) actual objective value, (2) average distance, and (3) accuracy.

All mentioned measures are averaged over 200 replications and reported in the following subsection. Our experiments are implemented in Python 3.7 programming language, using Gurobi Optimization 9.1.1. as the optimization solver, on an i7-core

2.60 GHz CPU with 32 GB RAM.

5.1 Results and Insights

In this subsection, the effect of different parameters is studied in different scenarios.

5.1.1 The effect of menu size in the fixed strategy

To discuss the effect of menu size in FM strategy, all variables are fixed other than θ. We set $\theta = 2$, $\theta = 3$, and $\theta = 6$ in each scenario. By comparing the performances of different menu sizes, one can observe offering fewer options leads to higher accuracy values in earlier iterations. That is, having $\theta = 2$ and $\theta = 3$, the accuracy exceeds 98% around the 7th iteration, while for $\theta = 6$ in the 45th iteration the accuracy gets close to 95%, as shown in Figure 5.1 (A). However, $\theta = 6$ converges to lower values in terms of average distance, since offering more options increases the chance of having more mistaken estimates, therefore more updates. As shown in Figure 5.1 (C), while $\theta = 6$ converges to a value around 0.1, $\theta = 2$ and $\theta = 3$ gets close to a value around 0.25 and 0.2 ,respectively. Finally, for the actual objective values, the scenario with $\theta = 3$ reaches higher values in the earlier iterations compare to others. After the 50th iteration, all three scenarios will have an actual objective value around 0.3 with $\theta = 3$ performing slightly better, as illustrated in Figure 5.1 (B).

5.1.2 The effect of accuracy in the adaptive strategy

To investigate the effect of accuracy in AM strategy, we compare $p = 7$, $p = 8$, and $p = 9$ in three scenarios while having $q = 10$ in all scenarios. We start from $\theta = 6$ and decrease the menu size to 2. That is, every time we have at least p correct estimated selections in the last q iterations, we decrease the driver's personalized menu by one until we have 2 options. As displayed in Figure 5.2 (A), Figure 5.2 (B), and Figure 5.2 (C), no significant difference can be observed between these scenarios

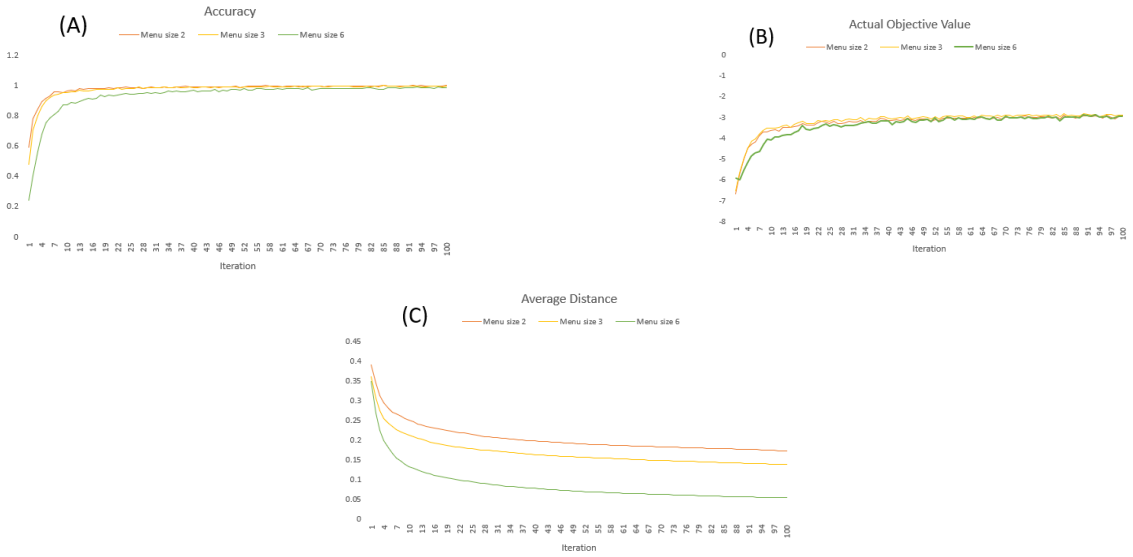

Figure 5.1 The effect of menu size in the fixed strategy

for all performance measures.

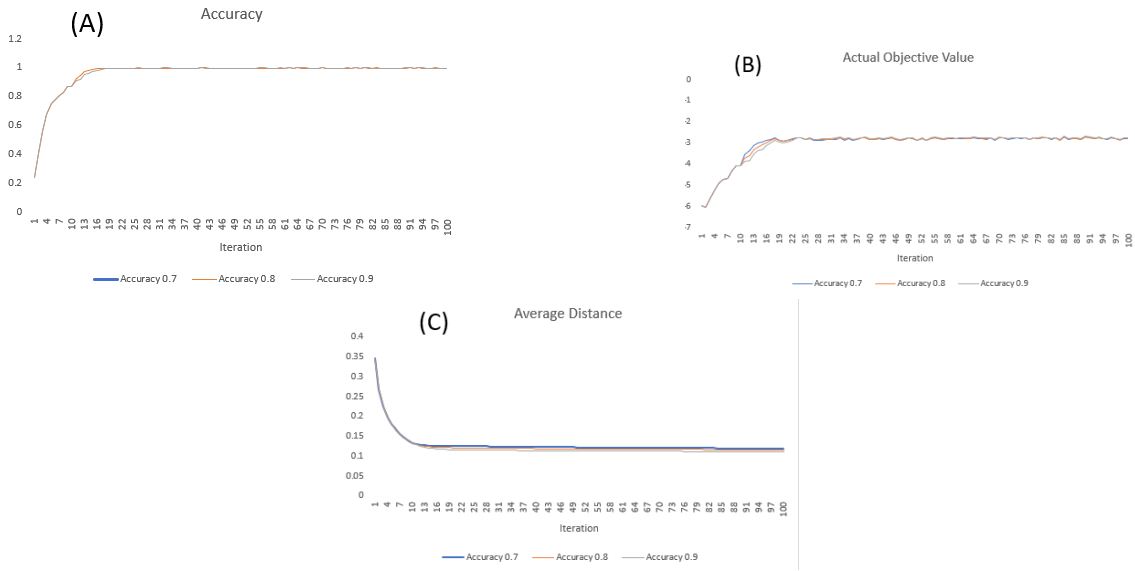

Figure 5.2 The effect of accuracy in the adaptive strategy

5.1.3 The effect of the number of steps in adaptive strategy

Two settings, where the first is with $p = 4$ and $q = 5$ and the second with $p = 8$ and $q = 10$, are compared to observe the effect of the number of steps in the adaptive strategy. Both settings start from $\theta = 6$ and decrease to $\theta = 2$ for each driver. As shown in Figure 5.3 (A), for the scenario with $p = 4$ and $q = 5$ the accuracy exceeds

98% in the 10th iteration, while after the iteration 17 both scenarios exceeds this value. The scenario with $p = 8$ and $q = 10$ converges to a value around 0.1 for the average distance while the one with $p = 4$ and $q = 5$ converges to a value around 0.15, since in the scenario with $p = 8$ and $q = 10$ more options are offered to the drivers in more iterations and that increases the chance of more mistaken estimates (Figure 5.3 (C)). For the actual objective value, both scenarios converges to a value around -3 after the 16th iteration with the scenario with $p = 4$ and $q = 5$ converges faster (Figure 5.3 (B)).

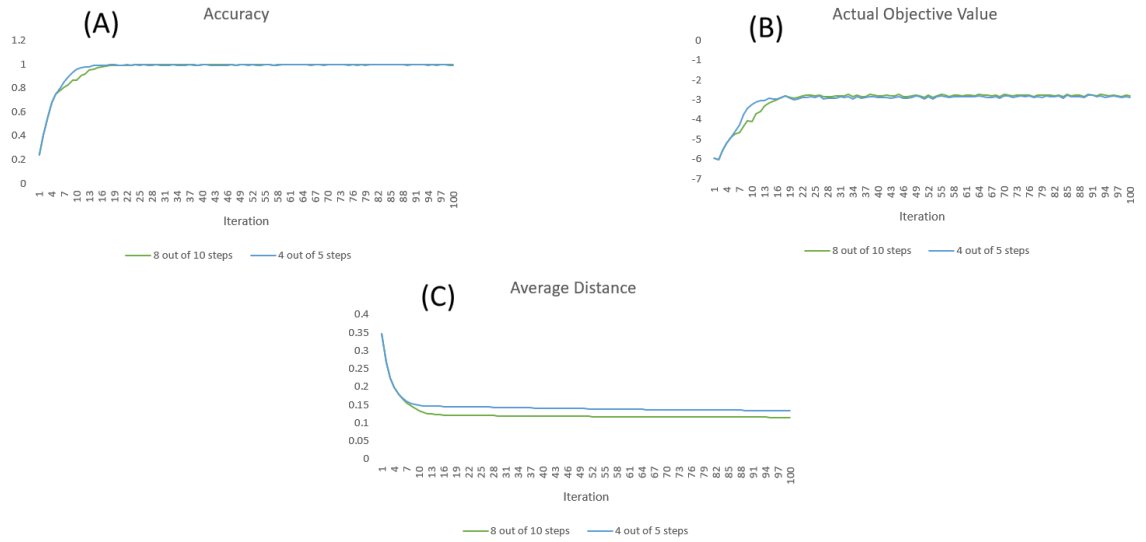

Figure 5.3 The effect of the number of steps in adaptive strategy

5.1.4 The effect of minimum menu size (1 or 2) in adaptive strategy

Two strategies, with accuracy 0.7 and $\theta = 6$ are compared, while in one strategy the minimum menu size is 1 and in the other is 2. As shown in Figure 5.4, changing the minimum menu size from 1 to 2 leads to no difference for all performance measures.

5.1.5 The effect of no-choice in fixed strategy

Figure 5.5 explores the impact of providing the drivers with the no-choice option while having $\theta = 6$. In terms of accuracy and actual objective values, the scenario without the no-choice option converges to higher values. The scenario without the

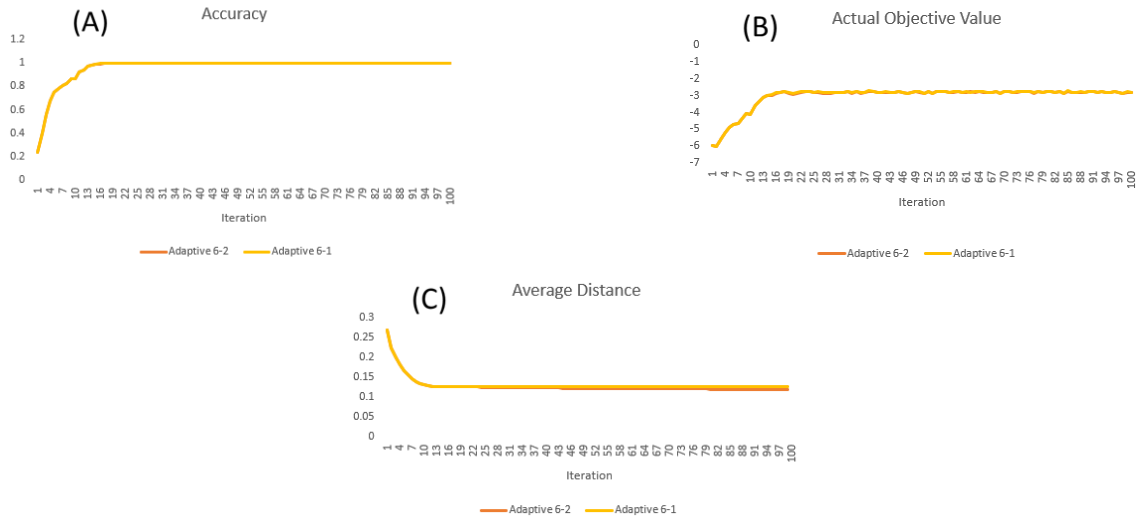

Figure 5.4 The effect of minimum menu size (1 or 2) in adaptive strategy

no-choice option exceeds 95% accuracy while the one with the no-choice option exceeds 0.6. For the actual objective value, the scenario without the no-choice option converges to a value around -3 and the one with this option converges to a value around -4. Further, the scenario with the no-choice option underperforms the other in terms of average distance and converges to a value around 0.2 while the scenario without the no-choice option converges to a value close to 0.15.

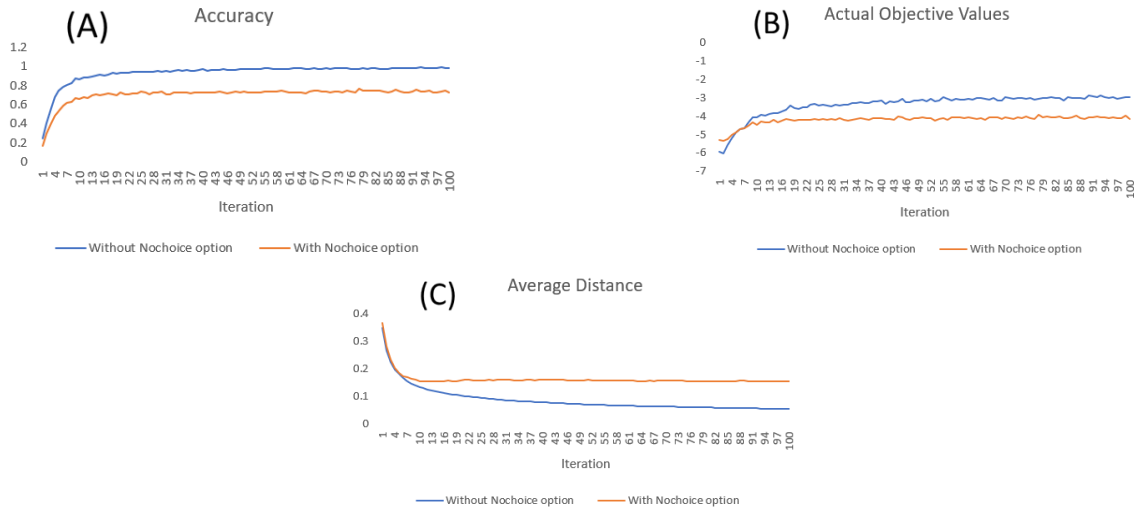

Figure 5.5 The effect of no-choice in fixed strategy

5.1.6 The effect of adaptive versus fixed strategy

The effect of FM versus AM strategy is discussed by comparing AM strategy starting from $\theta = 6$ and decreasing to $\theta = 2$ with FM strategies having $\theta = 2$, $\theta = 3$, and $\theta = 6$ in three cases.

As shown in Figure 5.6 (A) and Figure 5.7 (A), one can observe that for the FM strategies with $\theta = 2$ and $\theta = 3$ the accuracy exceeds 98% around the iteration 7 while AM strategy reaches this value around the 13th iteration. That is, FM strategies with $\theta = 2$ and $\theta = 3$ converge to higher accuracy values faster than AM strategy. On the other hand, FM strategy with $\theta = 6$ exceeds 95% accuracy around the 45th iteration (Figure 5.8 (A)). Further, FM strategy with $\theta = 6$ provides closer estimated locations to the actual locations than all other scenarios as shown in Figure 5.8 (C). However, AM strategy leads to closer estimates than FM strategies with $\theta = 2$ and $\theta = 3$ (see Figure 5.6 (C) and Figure 5.7 (C)). Finally, AM strategy reaches higher values (around -3) and outperforms all other scenarios in terms of actual objective function (See Figure 5.6 (B), Figure 5.7 (B), And Figure 5.8 (B)).

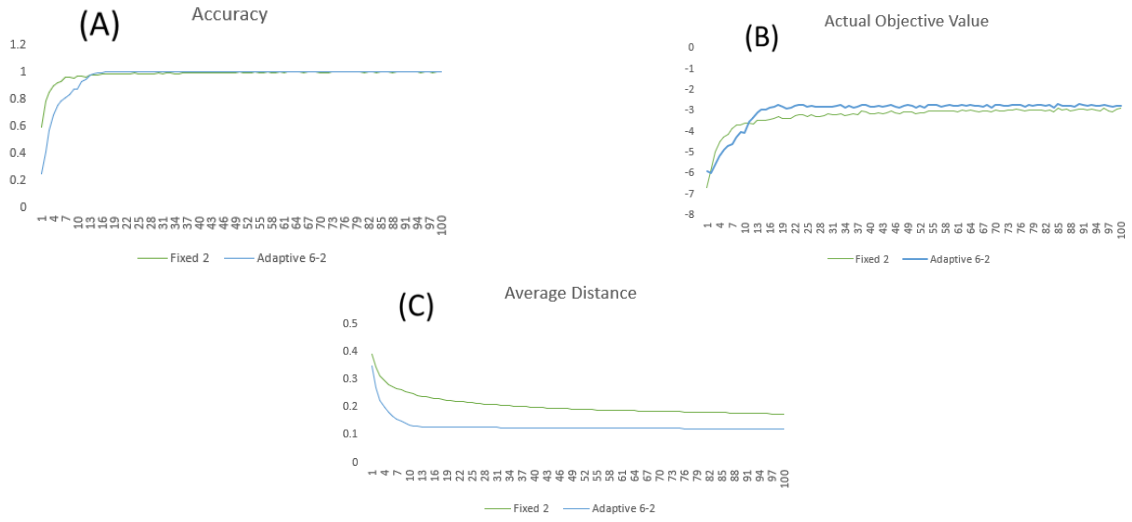

Figure 5.6 The effect of adaptive versus fixed strategy case one

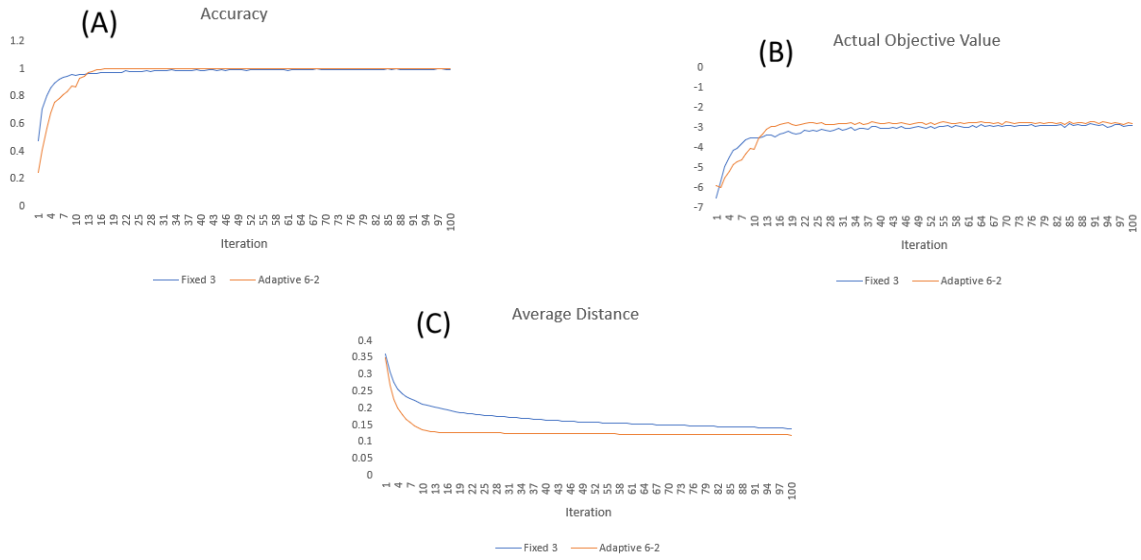

Figure 5.7 The effect of adaptive versus fixed strategy case two

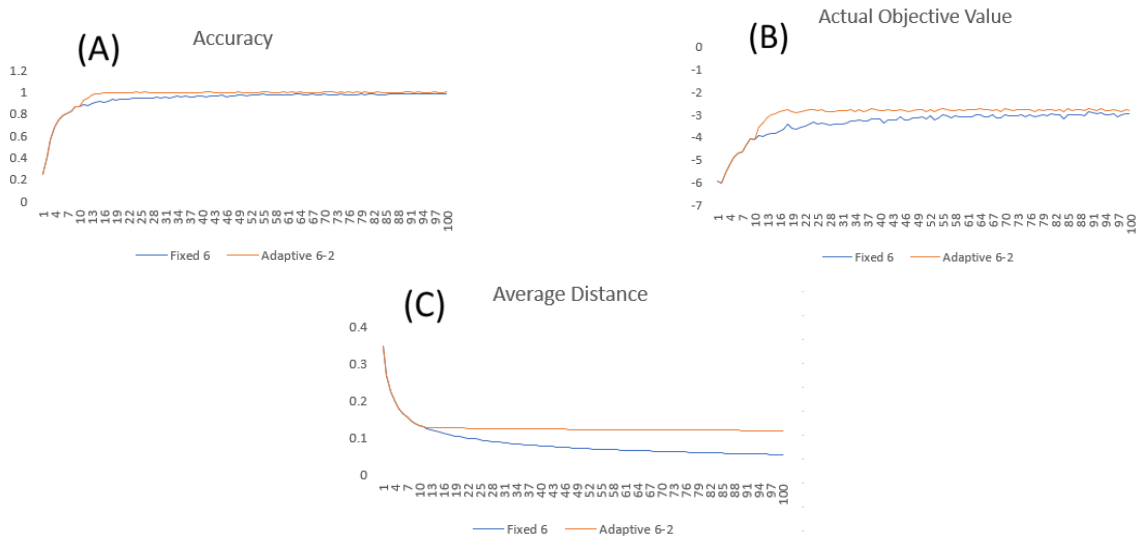

Figure 5.8 The effect of adaptive versus fixed strategy case three

6. Conclusion

In peer-to-peer logistics platforms where the agents are iteratively provided a menu of requests, the agents' preference information can be gradually collected to learn their utility functions and predict their future request preferences. However, there exists a trade-off between the set of requests that profits the platform and the ones that aid the learning process. Therefore, in this paper, we investigate the effect of different menu sizes on the learning process and the platform's profit.

As an illustrative example, we considered a ride-sharing setting where each driver's utility for a customer is the distance between the driver's actual location and the customer's pickup/drop-off origin. However, our methodology can be generalized to other peer-to-peer logistics platforms and other utility functions. Our experiments compared the performance of different menu size strategies as a function of varying factors. Among the experimented comparison cases, the most significant performance difference is observed in the comparison against the adaptive menu size strategy compared to the fixed menu size strategy. In the adaptive menu size strategy, by offering more options in the earlier iterations, the chance of having false estimated preferences and updating the drivers' estimated locations increases. Therefore, the adaptive strategy leads to reach more accurate estimates for the driver's location. Further, the adaptive method benefits the platform by providing fewer options as we become more certain about the drivers' actual locations.

BIBLIOGRAPHY

Agatz, N., Erera, A., Savelsbergh, M., & Wang, X. (2012). Optimization for dynamic ride-sharing: A review. *European Journal of Operational Research*, *223*(2), 295–303.

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.

Ausseil, R., Pazour, J. A., & Ulmer, M. W. Supplier menus for dynamic matching in peer-to-peer transportation platforms.

Bailey, J. P., Karabulut, E., & Pazour, J. Predicting spatial preferences by learning utility functions from partial ordinal information via classification. *working paper*.

Horner, H., Pazour, J., & Mitchell, J. (2021). Optimizing driver menus under stochastic selection behavior for ridesharing and crowdsourced delivery. *Transportation research*.

Mofidi, S. S. & Pazour, J. A. (2019). When is it beneficial to provide freelance suppliers with choice? a hierarchical approach for peer-to-peer logistics platforms. *Transportation Research Part B: Methodological*, *126*, 1–23.

Najmi, A., Rey, D., & Rashidi, T. H. (2017). Novel dynamic formulations for real-time ride-sharing systems. *Transportation research part E: logistics and transportation review*, *108*, 122–140.

Nourinejad, M. & Roorda, M. J. (2016). Agent based model for dynamic ridesharing. *Transportation Research Part C: Emerging Technologies*, *64*, 117–132.

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.