공학석사 학위논문

Adaptive Matching Time Intervals based on Reinforcement Learning for Ride-Hailing Services

승차 공유 서비스를 위한 강화학습 기반의
적응형 매칭 시간 간격 결정

2023년 2월

서울대학교 대학원

공과대학 건설환경공학부

신 용 근

Adaptive Matching Time Intervals based on Reinforcement Learning for Ride-Hailing Services

지도 교수 김 동 규

이 논문을 공학석사 학위논문으로 제출함
2022년 12월

서울대학교 대학원
공과대학 건설환경공학부
신 용 근

신용근의 공학석사 학위논문을 인준함
2023년 2월

Abstract

Ride-hailing services helped daily travel by efficiently matching passengers and drivers. These services face inefficiency in system operations due to supply and demand imbalances. A widely adopted strategy is fixed batch-based matching, which accumulates requests and idle drivers and matches them in batches. Recent studies have proposed adaptive matching time intervals to consider dynamic supply and demand patterns. However, matching failure factors such as passenger request cancellation and driver acceptance are not considered. This study aims to control adaptive matching time intervals based on reinforcement learning considering matching failure factors. To this end, we propose a two-step framework to maximize the matching success rate. First, an agent based on Deep Q-Network (DQN) determines the matching time interval, and then combinatorial optimization is performed based on the driver's acceptance probability. We conduct experiments on various supply-demand patterns based on synthetic and real datasets and compare performance with previous strategies. We confirmed that the proposed strategy reduces the proportion of expired requests and achieves the highest matching success rate. We also discussed the trade-off between fixed matching time intervals and matching success rates and interpreted agent policies. Our approach provides insight by discussing matching failure factors, which cannot be captured with performance alone.

Keyword : Ride-Hailing Service, Reinforcement Learning, Deep Q-Network (DQN), Combinatorial Optimization, Matching Failure

Student Number : 2021-21896

Table of Contents

List of Tables

List of Figures

Chapter 1. Introduction

Ride−hailing services such as Uber, Lyft, and KaKaoT (in Korea) provide a platform that allows passengers to use taxis at any time and place by sending requests for their departure and destination. Unlike the traditional taxi system, where passengers had to catch taxis on the street, these services help daily travel as intermediaries that pair passenger requests with nearby idle drivers in real−time.

However, one of the operational challenges facing these services is the imbalance between supply and demand. There may be gaps between supply and demand, such as no idle driver available at peak times when demand increases or, conversely, a large supply at a non−peak time when demand decreases (Yang et al., 2002). To this end, operators consider various strategies to maximize system revenue (e.g., matching success rates) or to improve passengers' degree of satisfaction by minimizing passenger waiting time (Qin et al., 2021a): taxi demand prediction (Tong et al., 2017; Zhao et al., 2016), Vehicle repositioning (Liu et al., 2022; Oda et al., 2018;), Order dispatching (Li et al., 2019; Xu et al., 2018), Dynamic pricing (Bimpikis et al., 2019; Chen et al., 2019), etc.

A widely adopted strategy in the Ride−hailing service is order dispatching, a method in which a centralized system searches for the driver and sends the request to the appropriate driver when it receives a passenger request. Previous studies use terms such as request dispatching or taxi dispatching, but this depends on the assigned object. This study uses order dispatching in terms of assigning passenger requests to vehicles.

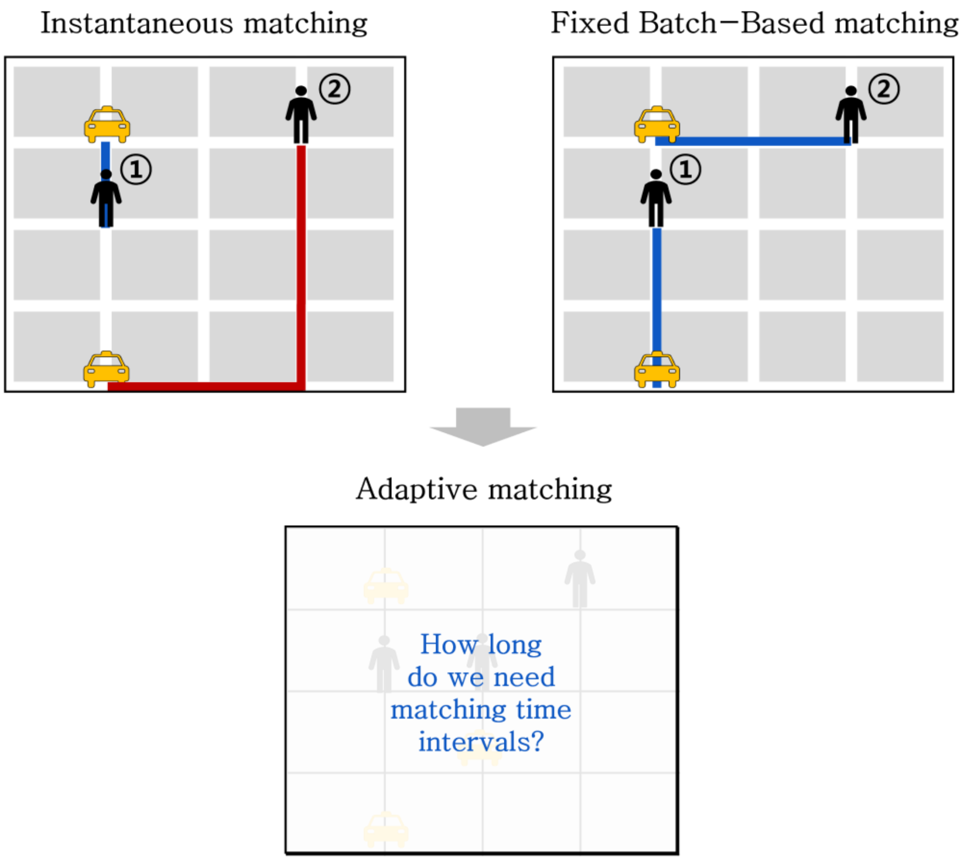

Instantaneous matching
Fixed Batch–Based matching
①
②
Adaptive matching
How long do we need matching time intervals?

Figure 1.1 Potential benefits of matching time intervals in order dispatching

Figure 1.1 shows the potential benefits of controlling matching time intervals when two idle taxi drivers and two passenger requests arrive in order. Traditional dispatch systems have adopted strategies that immediately assign passengers to nearby drivers upon request (Lee et al., n.d.) (Instantaneous matching). In this case, the first passenger will match the nearby idle driver, but the second passenger will match the relatively farther away driver. This strategy is inefficient because it does not consider the system's perspective. Considering these limitations, most studies use batch processing based on fixed time intervals (Fixed Batch–Based matching). This strategy improves overall efficiency by accumulating requests and drivers in the queue during matching time intervals and optimizing objective functions such as the passenger's total waiting time. In this case, the first passenger will have to wait a little longer, but all passengers will board the taxi within a reasonable time, reducing the total waiting time. Furthermore, to consider dynamic supply-demand patterns, there have recently been attempts to determine adaptive matching time intervals based on Deep Reinforcement Learning (Ke et al., 2022; Qin et al., 2021a; Wang et al., 2019).

The critical factor to consider in order dispatching is matching failure factors based on passenger or driver behavior patterns. **Figure 1.2** shows the matching failure factors in the ride-hailing service from the passenger's new request to the destination. During this process, passengers can cancel requests based on their behavior, and drivers can reject requests they do not prefer. For example, passengers may cancel requests if waiting time (e.g., match waiting time and pick-up waiting time) exceeds the tolerable range before or after matching. The driver may also accept or reject according to spatiotemporal characteristics such as the origin/destination context of the assigned request. Therefore, we must consider these matching failure factors when controlling the matching time interval.

These matching failure factors are important in controlling the matching time intervals. However, many studies assume that the driver always accepts requests assigned by the platform operator or do not consider passengers' cancellation.

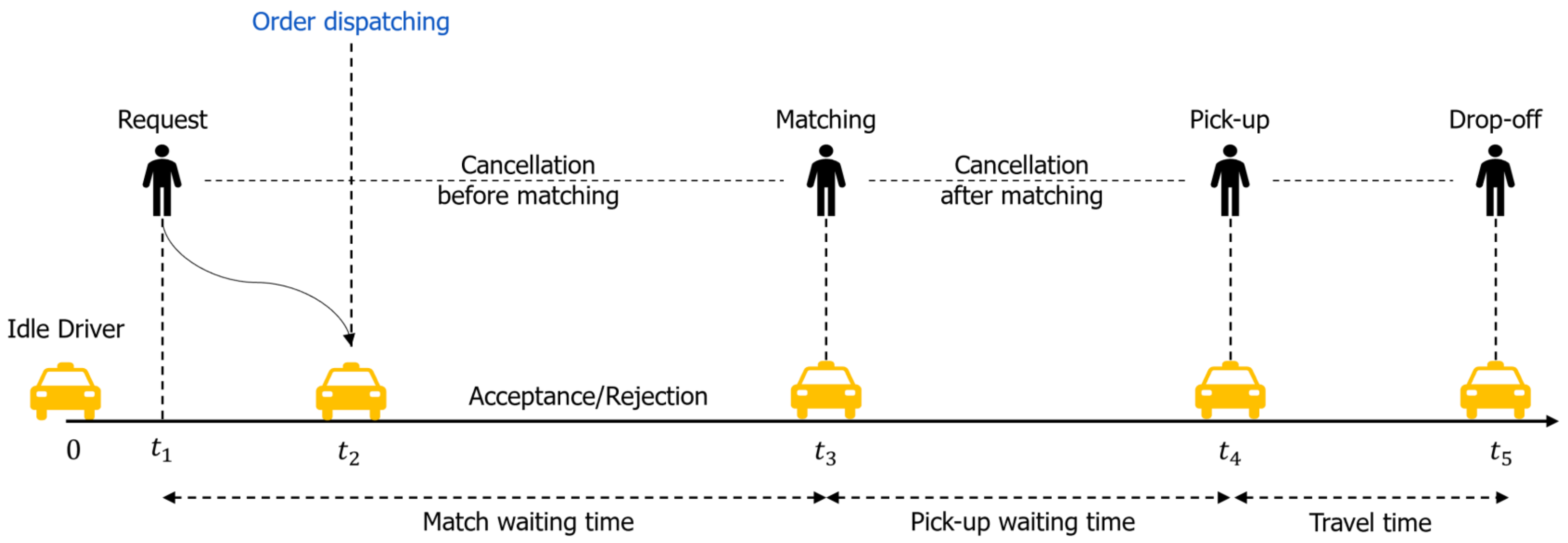


Figure 1.2 Matching failure factors between passenger requests and drivers in ride−hailing service

Considering these limitations, Zhang et al. (2019) proposed a dispatch system that estimates the driver's acceptance probability through a data−based approach and performs optimal matching based on it. This affects direct operational efficiency gains to maximize matching success rates. This approach can also improve the accuracy of the estimated acceptance probability as data accumulates in the future. We aim to extend the study of Zhang et al. (2019) by controlling adaptive matching time intervals.

This study focuses on determining adaptive matching time intervals based on reinforcement learning, considering the matching failure factors according to passengers and drivers. To this end, we construct a two−step framework to maximize the matching success rate. 1) The reinforcement learning agent determines the matching time intervals, and 2) performs combinatorial optimization based on the driver's acceptance probability at the current matching time interval.

In the first step, the platform operator is considered an agent, and the dynamics of order dispatching are modeled as the Markov decision process (MDP). The agent decides whether to match or hold each matching time interval. You can control the time interval based on the state variables that the agent observes. Once the agent has decided to match, proceed to the second step, and perform combinatorial optimization in the current queue to assign the driver a passenger request. This is formulated to maximize the average driver's acceptance probability.

The key contributions of this study are as follows: (a) proposal of a two−step framework based on reinforcement learning and combinatorial optimization; (b) Development of calibrated simulators using real datasets and validation strategies with various experiments; (c) Providing analytical insight by visualizing optimal policies of the agent.

The remainder of this paper is organized as follows. First, we review previous studies to control matching time intervals or how to match passengers and drivers. And we formulate the problem of this study and describe reinforcement learning algorithms to solve them. The overall simulation framework and its components are also described in detail. The next section describes the data and experimental settings. Then, a detailed analysis of the results is provided. Lastly, findings are discussed along with brief concluding remarks and notes on future research plans.

Chapter 2. Literature Review

Previous studies can be classified into two categories: when to match in the temporal context and how to pair drivers and passengers in the current time interval. They focus on one category, and the other adopts the existing method. In this section, the former was referred to as online matching and order dispatching as commonly used terms.

Online matching is formulated as an online bipartite matching problem. It is related to the optimal matching of bipartite graphs in situations where one or two groups arrive online (Karp et al., 1990; Mehta et al., 2007). It is used in various fields, such as crowdsourcing (Tong et al., 2016) between work and workers, advertising exposure strategies in search engines, and passenger requests and drivers (this study).

Previous studies have tried to improve performance by dynamically controlling spatio-temporal variables such as matching time intervals or radius (Özkan and Ward, 2020; Yan et al., 2020; Yang et al., 2020).

Özkan and Ward (2020) developed a dynamic matching strategy to maximize the cumulative matching rate. Based on the continuous linear program (CLP), they proposed a matching policy that describes passenger and driver arrival rates and waiting time features. Yang et al. (2020) jointly optimized two spatio-temporal variables, matching time interval and matching radius, and measured system performance such as matching rate, passenger waiting time, and pick-up waiting time. They also compared and validated the performance according to various supply–demand scenarios. Yan et al. (2020) proposed joint pricing and matching optimization benefits. They confirmed that this strategy could alleviate the price volatility problem using the dynamic pricing strategy as a single model.

An approach based on these parameters requires unrealistic assumptions for each parameter (Qin et al., 2021a), and it is difficult to reflect the stochastic characteristics of the real–world environment. It is impossible to mathematically formulate the dynamics of supply–demand patterns in a complex ride–hailing service market (Ke et al., 2022).

Recently, researchers have attempted to determine adaptive matching time intervals (Ke et al., 2022; Qin et al., 2021b; Wang et al., 2019). Reinforcement learning is a sequential decision that affects the current action. This is a powerful way to learn and experience optimal policies (Sutton and Bart 1998).

Wang et al. (2019) modeled the system from the perspective of a platform operator. They configure the currently unmatched bipartite graph as a state and decide whether to hold or match with action. The reward is the sum of the weights on the graph and is determined by the total revenue of each driver in the actual experiment. Ke et al. (2020) constructed a multi–agent framework based on individual passenger requests. They used Deep Q–Networks (DQN), an Actor–Critic model based on deep neural networks, and designed reward functions focusing on passenger waiting time. Qin et al. (2021) determine adaptive matching time intervals from the perspective of a single system. They also focused on passenger waiting time and improved the reward sparsity problem, which occurs only when the agent decides the action through the decision of the reward function.

Previous studies on order dispatching focus on how to pair drivers and passengers in the current time step.

As mentioned in the previous section, the traditional dispatch system has instantaneous matching or fixed batch−based matching strategy for passenger requests. In the latter case, it is formulated as a linear assignment problem (LAP), which optimizes the objective function, such as the pick−up waiting time, for passenger requests and drivers. This approach focuses only on the distance between passengers and drivers, so the overall matching success rate is not directly considered.

Zhang et al. (2017) formulated a combinatorial optimization algorithm based on the driver's acceptance probability to maximize the matching success rate. Estimating the driver's acceptance probability is modeled as a binary classification problem, and classification models such as logistic regression and gradient−boosted decision tree (GBDT) are available. Based on this, they formulate it as a combinatorial optimization problem to maximize the average acceptance probability in the current time step. They derive an approximation solution using the hill climbing algorithm.

As a result, it shows better results from the perspective of matching success rate and passenger waiting time than the traditional two strategies described above. Since passenger waiting time is also reflected in the driver's acceptance probability model, it can be confirmed that performance is improved from this point of view.

Xu et al. (2018) are divided into learning and planning steps, and in the learning step, each driver is modeled as an agent, and the state−value function is learned in advance based on historical data. The planning step updates the weights of the bipartite graphs to be currently allocated based on the value functions learned in the previous step. Finally, the final allocation is achieved through combinatorial optimization that maximizes the sum of weights. Li et al. (2019) used multi−agent reinforcement learning to maximize accumulated driver income (ADI). They spread the decision−making of individual drivers globally based on mean field approximation.

In addition, there have been applied studies that have either relocated vehicles based on demand prediction models (Liu et al., 2022b), or developed models that can be generalized at other times and in other cities based on transfer learning (Wang et al., 2018).

In the above studies on online matching and order dispatching, the former lacks consideration of the matching failure factors to be considered in ride−hailing services, such as passenger request cancellation, driver acceptance, and rejection. In the latter case, most studies focus on the performance of the current time step so that it may be suboptimal from a future perspective. Reinforcement learning−based studies consider these limitations, but the factors of matching failure are not considered. By considering each limit, we maximize the matching success rate by determining the adaptive matching time interval based on reinforcement learning by considering the matching failure factors.

Chapter 3. Methodology

3.1. Problem Statement

This section describes the background knowledge of this study and the combinatorial optimization problem for controlling adaptive matching time intervals. The proposed method in this work is a two-stage framework consisting of determining matching time intervals and solving combinatorial optimizations. Figure 3.1 describes how the proposed method works when passenger requests and idle driver information are sent to the platform.

The horizontal axis is a queue monitored by the platform, where passengers' requests and idle drivers come online. The reinforcement learning agent corresponding to the operator observes the queue and decides whether to dispatch. Once the agent decides to dispatch, it performs combinatorial optimization between passenger requests and idle drivers within the current matching time interval. Here, the matching failure factors mentioned in Figure 1.2 may cause the passenger to cancel the request. Depending on the driver's acceptance, it may re-enter the queue or eventually expire. Expired requests occur when the number of re-enters to the queue reaches the threshold (discussed in Section 4.1). We implement these processes repeatedly, and the agent learns optimal policies to maximize the matching success rate. That is, the matching time interval is controlled adaptively.

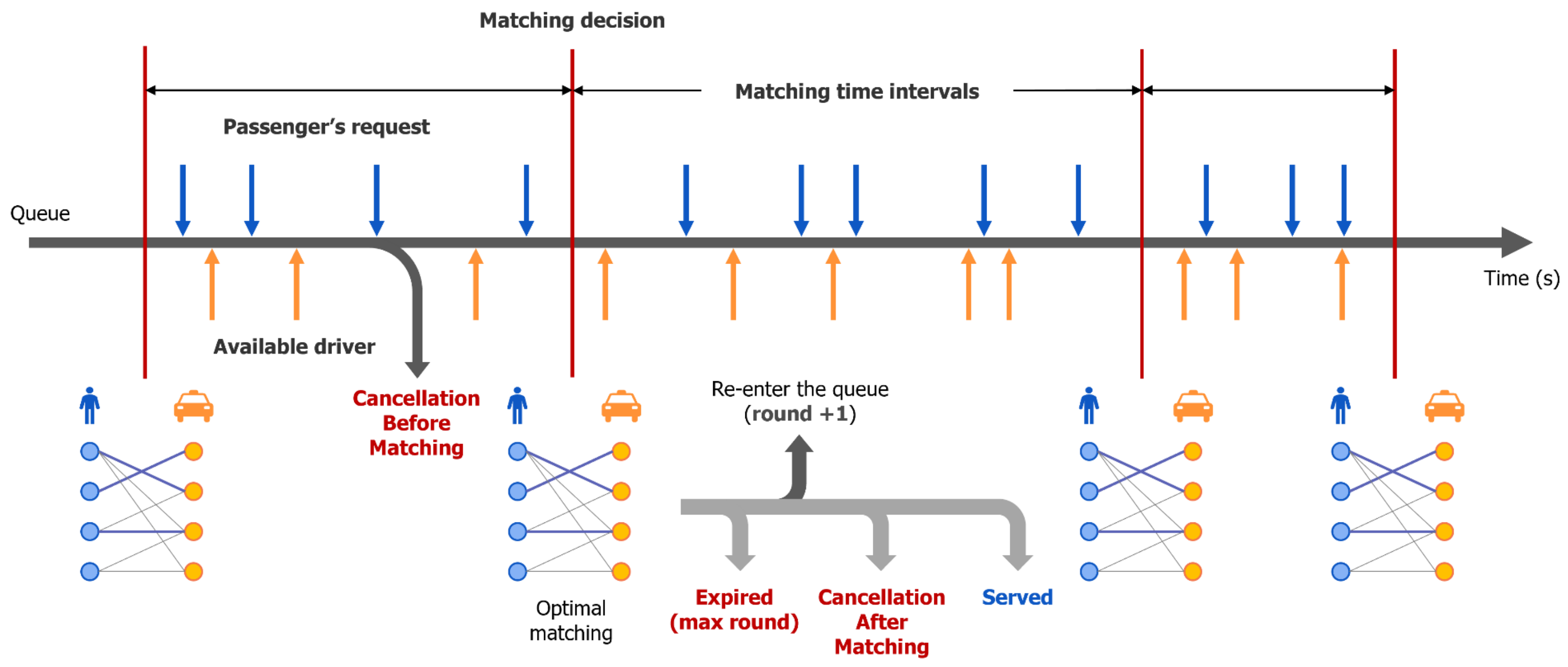


Figure 3.1 The process of order dispatching based on the proposed strategy (Source: Own elaboration based on Qin et al., 2020)

3.2. MDP formulation

In this section, we model sequential decision problems that control matching time intervals as a Markov Decision Process (MDP) (Puterman 2014), which includes an agent, a set of states and actions, state transitions, and rewards.

The agent is considered a platform operator (e.g., single agent setting), and each time step $t \in \{1,2,\dots,T\}$ repeatedly performs a two-step framework. The first step is to decide whether to keep or match drivers with passenger requests accumulated in the queue at a given time step. The second step is to perform combinatorial optimization if the agent decides to match the previous step.

At every time step t, the agent observes the states associated with the number of idle drivers and requests from passengers in the queue. The set of states is represented by $S(t) = \{N_P(t-1), N_D(t-1), \lambda_p(t), \lambda_D(t)\}$. $N_P(t-1)$ and $N_D(t-1)$ are the number of passenger requests and idle drivers present in the queue at the previous time step $t-1$, respectively, and $\lambda_p(t)$ and $\lambda_D(t)$ are the number of new passenger requests and idle drivers in the queue at the current time step.

The set of actions taken by the agent is represented by $A(t) = \{0,1\}$. $A(t) = 0$ means to suspend the matching decision and move to the next time step $t+1$, and $A(t) = 1$ moves to the second step to perform combinatorial optimization.

The dynamics of the environment as it moves from time step t to the next time step $t+1$ are represented by state transition probability $P(S(t+1) \,|\, S(t), A(t))$. This means the probability that the agent and the environment interact to update from state $S(t)$ of time step t to state $S(t+1)$ of next time step $t+1$ by action $A(t)$.

The first is that after the agent has made a matching decision $A(t) = 1$, the final pick−up may be completed, or passengers may cancel the request without waiting for the matched driver. The second is that the agent may hold the decision $A(t) = 0$ and remain in the queue, or passengers may cancel the request due to a longer wait before matching. If the last attempt was made to match, but the driver refused or did not attempt to match, re−enter the queue. New drivers and requests are also queued every time step. Detailed dynamics are discussed in the next section.

The reward given when an agent takes action is expressed as $R(S(t), A(t))$ and is set to the number of requests finally served in the current time step. Therefore, the agent receives rewards after making matching decisions and otherwise receives zero rewards. This is related to the objective function for combinatorial optimization in the second stage leading to the maximum matching success rate. Based on the reward function, the agent learns the optimal policy π to maximize cumulative rewards. This is expressed as **Equation (3.1).**

$$V_\pi\big(S(t)\big) = E_\pi\left[\sum_{t=0}^{\infty} \gamma^t \, R\big(S(t), A(t)\big)\right] \tag{3.1}$$

$V_\pi\big(S(t)\big)$ is the expected value of the cumulative reward, and γ represents the discount factor. This reflects the weight between the long−term and the current reward.

3.3. Simulation Framework

This section describes the ride-hailing simulator and system architecture that implements the dynamics mentioned in **Figure 3.1**. The simulator induces the agent to learn the optimal policy by interacting with the environment that emulates the ride-hailing dynamics based on taxi data.

Figure 3.2 shows the environment, reinforcement learning agent, and three main modules. The environment initializes the spatial map information, the pool to store the passenger's request and driver, and the time. It then repeats the following steps during each time interval: Generate new passenger requests and idle driver information, store them in the pool, and extract candidates for matching with queues. Then apply the action according to the policy of the reinforcement learning agent (dispatch or skip). In the former case, combinatorial optimization is performed by the Order dispatcher module, and in the latter, it re-enters the queue. Finally, update passenger requests and driver status and provide the agent with updated status and rewards. The environment repeats during consecutive time steps and considers it an episode. When one episode ends, the simulator is reinitialized.

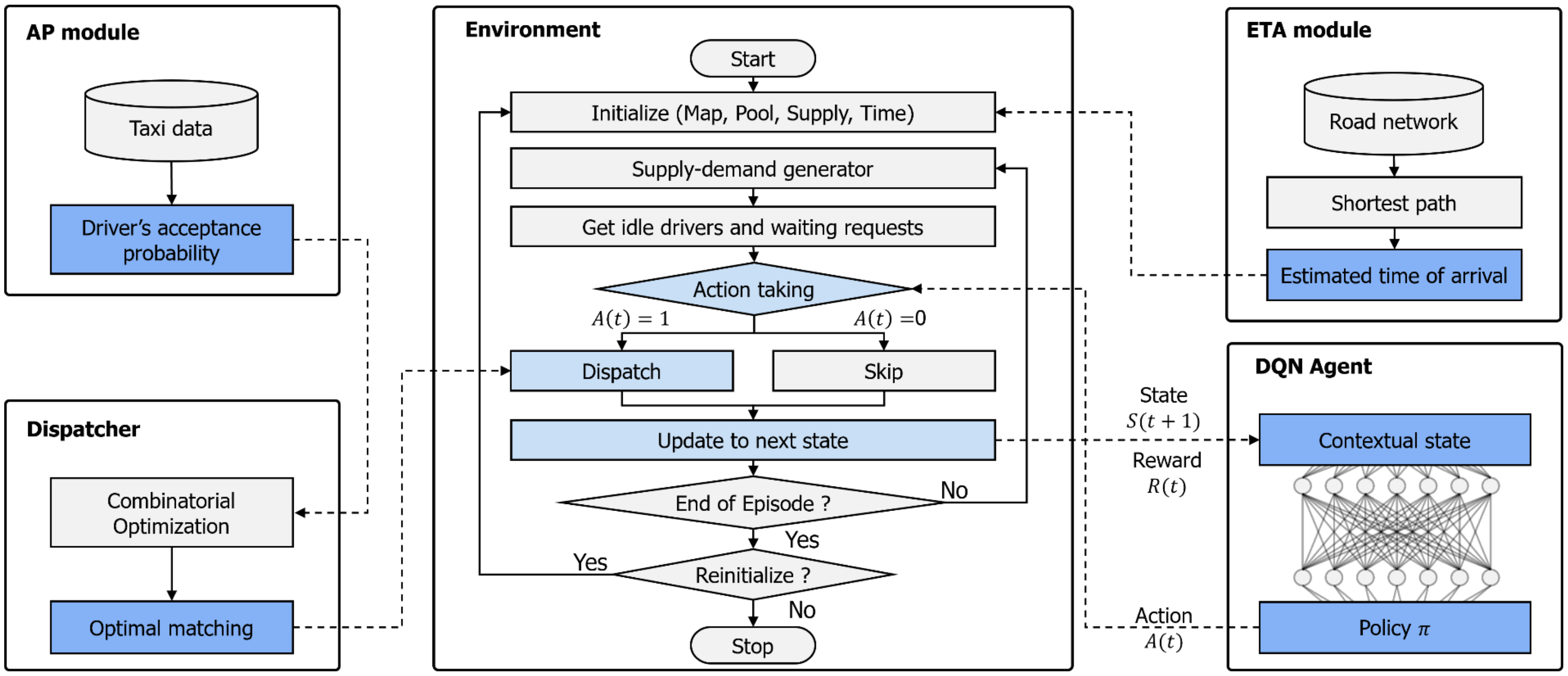


Figure 3.2 The ride−hailing simulator framework

The simulator contains three main modules. ETA (Estimated Time of Arrival) module maps location information based on a road network and estimated arrival times. Order dispatcher module performs combinatorial optimization based on the driver's acceptance probability. AP (Acceptance Probability) module provides an estimated driver's acceptance probability based on historical data.

ETA module contains road network and shortest path information. We acquired Singapore's road network of 23,805 nodes and 35,649 edges from OpenStreetMap (OpenStreetMap, n.d.). Passenger requests and driver GPS information are mapped to the nearest node, and the shortest path between nodes is calculated based on the Dijkstra algorithm (Dijkstra 1958). Through this, we can provide the environment with the expected pick-up waiting time and the travel time to the destination.

Order dispatcher module is based on the method proposed by Zhang et al. (2017). They formulated it as a combinatorial optimization problem to maximize the driver's acceptance probability, which affects the direct matching success rate (Zhang et al., 2017). At each matching time interval, the objective function and constraints are formulated as shown in **Equation (3.2)**, given the requests of N passengers $(i = 1,2,\dots,N)$ and M idle drivers $(j = 1,2,\dots,M)$.

$$\begin{aligned}
&Maximize\ \frac{1}{N}\left[\sum_{i=1}^{N}\left(1-\prod_{j=1}^{M}(1-p_{ij})^{a_{ij}}\right)\right] \\
&s.t.\quad \sum_{i=1}^{n} a_{ij} \le 1,\ \ \forall j \\
&\qquad\ a_{ij} \in \{0,1\} \\
&\qquad\ 0 \le i \le N,\ 0 \le j \le M
\end{aligned} \tag{3.2}$$

p_{ij} represents the driver's acceptance probability. A_{ij} is a dummy variable to optimize and is defined as follows:

$$a_{ij} = \begin{cases} 1, & Passenger\ i\ is\ dispatched\ to\ driver\ j, \\ 0, & Passenger\ i\ is\ not\ dispatched\ to\ driver\ j \end{cases} \tag{3.3}$$

The order dispatcher module performs combinatorial optimization by **Equation (3.2)** whenever the agent dispatches. It provides an approximate solution using the Hill-Climbing method, a heuristic algorithm.

AP module provides the driver's acceptance probability p_{ij} mentioned in **Equation (3.2)**. The driver's acceptance probability is a binary classification problem of whether the driver has accepted a particular request and can be estimated based on historical data (Zhang et al., 2017). We used logistic regression (Hosmer et al., 2013), which is widely adopted for these problems. The acceptance probability p_{ij} of driver j for passenger i is defined as shown in **Equation (3.4)**.

$$p_{ij} = p(y = 1 | p_i, d_j) = \frac{1}{\exp(-w^T x_{ij})} \tag{3.4}$$

The logistic regression is expressed as a sigmoid function between 0 and 1. It is derived from the logit transformation of the target in linear regression, where x_{ij} is the feature vector representing passenger i and driver j. We obtained a model with approximately 60% accuracy based on pick-up distance and temporal context. The distribution of acceptance probabilities varies with temporal characteristics and decreases with longer pick-up distances. In the future, we can improve accuracy and provide sophisticated acceptance probabilities based on more feature vectors and accumulated data.

Passenger requests and drivers may correspond to one of four and three statuses at each matching time interval. **TABLE 3.1** lists the types and descriptions of each status. Requests and drivers first entered the queue are "WAIT" and "IDLE," respectively. Depending on the interaction of the environment, the state changes depending on the several events described in **Figure 3.1**.

Passengers can cancel before matching (CBM) according to the match waiting time or cancel after matching (CAM) according to the pick-up waiting time. These passengers leave the queue. Regardless of passenger cancellation, there may be expired requests (EXPIRED) due to drivers not continuing to accept the request. This can be set as a threshold for the number of attempts according to the platform operating rules, and this is limited to four in this study. If they accept the assigned request, drivers will switch to the "PICKUP" status. If the request is canceled, the driver will switch to the "IDLE" status. If not, the driver will switch to the "HIRED" status and leave the queue.

TABLE 3.1 Description of passenger requests and driver status in the environmental simulator

Status	Description
1. Passenger	
SERVED	Completed request
WAIT	Waiting for matching in the queue
CBM	Cancellation before matching
CAM	Cancellation after matching
EXPIRED	Expired due to no driver accepting regardless of passenger cancellation
2. Driver	
IDLE	Waiting for matching in the queue
PICKUP	Picking up matched passengers
HIRED	Traveling to destination with passengers

3.4. Deep Q-Networks (DQN)

This section describes Deep Q-Networks (DQN) (Mnih et al., 2015), a reinforcement learning algorithm for optimal policies that determine matching time intervals.

Q-learning (Watkins and Dayan, 1992) is the base model of DQN. It stores the Q-value according to state and action in the Q-table and updates this table repeatedly while interacting with the environment. In this way, the Q-function that returns the current Q-value is approximated. These mechanisms are formulated as **Equation (3.5)** based on the MDP modeled in the previous section.

$$\begin{aligned} Q(s(t), a(t) \leftarrow Q\big(s(t), a(t)\big) \\ + \alpha\big[R(t) + \gamma \max \mathrm{Q}\big(\mathrm{s(t+1), a(t+1)}\big) \\ - \mathrm{Q(s(t), a(t)}\big] \end{aligned} \tag{3.5}$$

α represents the learning rate for each episode, and γ is the discount factor. The Q-value updates it according to the current state and action and the maximum Q-value that can be received in the next state.

DQN improves the problem of storing too large a table size as dimensions increases in q-learning. To approximate the Q-function, it uses neural networks to optimize the loss function based on the difference between the predicted value of the Q-network and the target Q-value. The loss function is given in **Equation (3.6)**, where y represents the target Q-value.

$$L(\theta) = E[(Q(s(t), a(t)|\theta) - y)^2] \tag{3.6}$$

Additionally, researchers structured the predicted values and target Q−values into separate networks, improving the problem of non−convergence due to fluctuations in target Q−values. The estimation parameter of **Equation (3.7)** and **Equation (3.6)** are set differently. In the terminal state, the current compensation is received.

$$y = \begin{cases} R(t) \\ R(t) + \gamma \max\left(Q(s(t+1), a(t+1)|\theta^-)\right) \end{cases} \tag{3.7}$$

DQN increases efficiency through deep neural networks, and experience replay algorithms using memory buffers are added. It accumulates data in memory buffers instead of using it directly in neural networks and learns with minibatch through random sampling. This approach improves instability due to nonlinear functions and solves sample correlation problems.

Chapter 4. Results

4.1. Data Description

We experimented using the real dataset provided by MVL, which operates the TADA service (TADA, n.d.) in Singapore. The dataset was collected over two months, from 2020-11-01 to 2020-12-31, and consists of three tables: The passenger's request information, the driver's GPS record, and the list of requests sent to the driver.

The passenger's request information includes the request ID, date, origin and destination latitude and longitude, and the final status identifier (completion, cancellation, expiration, etc.). The driver's GPS records included the driver's ID, date and time, vehicle latitude and longitude, and status identifiers (during idle or hired) and were recorded every 10 minutes. Finally, a list of requests sent to the driver also called ping information, is an N × M dimension table for M vehicles and N requests. Through this, we combined the request and driver GPS tables to extract information about the matching failure factors. And we analyzed the distribution of cancellations before and after matching according to the match waiting time and pick-up waiting time of passengers. It was also used to estimate the driver's acceptance probability based on the ping information.

The distribution of passenger requests and drivers extracted from the dataset is listed in **TABLE 4.1**. There were about 8,000 vehicles and about 700,000 requests. According to the status code classified based on the final status identifier, the requests of the finally serviced passengers were 32.7%. Among the requests that failed to match, the expired request was the highest (51.9%), and the cancellation before matching was about 8%. Expired requests are expected to be the highest because the dataset only includes matching TADA drivers. That is, passengers who have sent a request but have used other services (i.e., competitors) are marked as expired, and the driver's acceptance is not recorded. So, we calibrated the simulator so that the distribution of expired requests was somewhat mitigated.

TABLE 4.1 Distribution of historical data on passenger requests and drivers

Category	Count	Ratio (%)
Number of vehicles	8,122	100
Number of requests	729,992	100
Served request	239,024	32.7
Failed request	490,968	67.3
Cancellation before matching	57,908	7.9
Cancellation after matching	32,414	4.4
Expired	378,948	51.9
Rejected	21,698	3

Figure 4.1 shows the distribution of passenger requests aggregated into hours of the day. Total requests and served requests account for about 30% with similar patterns. The hour of the day appears as four peak times. It includes the morning peak (travel to work) and the afternoon peak (travel to home) that appear in the transportation system. The other two peak times occur during the

day and late at night. It can be seen as leisure or business activities in the former case. In the latter case, it can be seen as the demand of passengers who want to return home quickly or the available public transportation time has ended.

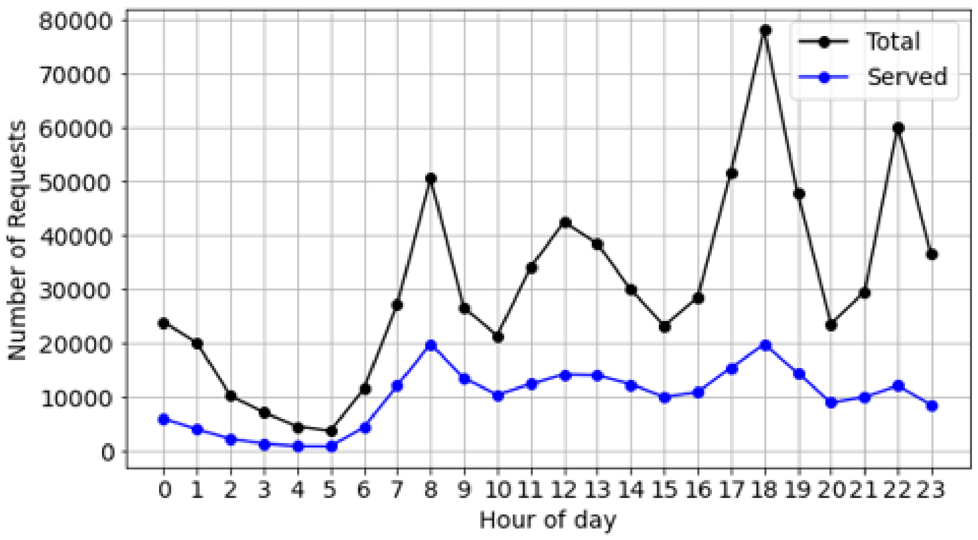


Figure 4.1 Distribution of total and served requests

4.2. Experimental Setup

The area of Singapore is 719.9km^2, slightly larger than Seoul (605.2km^2), the capital of Korea. In this work, we limited the spatial range observed by agents because we focus on controlling matching time intervals from a temporal perspective. Therefore, the target area is the 10km × 10km grid (**Figure 4.2**) in Singapore's Northeast region, where demand is relatively high. Previous work has confirmed that the more spatial units are subdivided using smaller grids, the better the performance and the significant increase in training time (Qin et al., 2021). we used one grid but can subdivide the spatial units or expand the action dimension from a spatial perspective in the future.

The time range is between 8 am and 9 am on weekdays and is provided to the environment by sampling 1,000 requests and drivers per episode. Each matching time interval is set to 1s, and the simulation starts at 8 am, lasts for 10 min (i.e., 600-time steps), and is reinitialized. The average vehicle speed is set to 20 km/h. And the maximum number of allocatable times for each request is limited to 5. During this time, the agent interacts with the environment and learns the optimal policies.

Baseline algorithms use the previous dispatch system, Instantaneous (Instant.) and Fixed Batch-Based (FBB) matching, as mentioned in **Figure 1.1**. Instant applies the action at all matching time intervals, the latter at fixed time intervals. The fixed time interval is set to the optimal value with the highest matching success rate through repeated experiments.

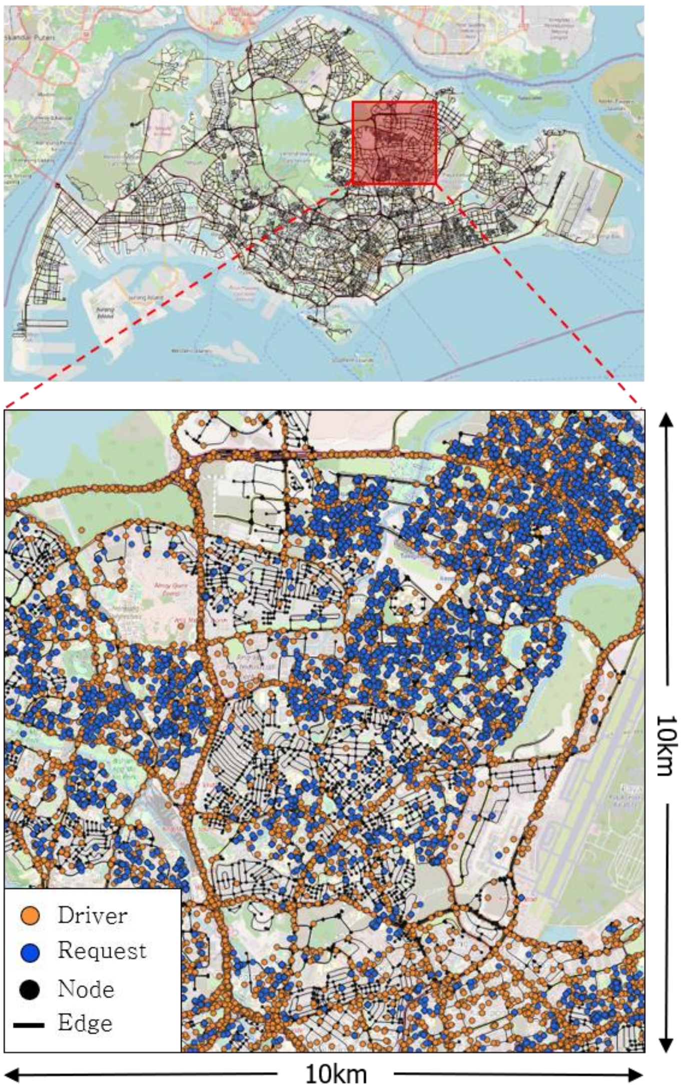


Figure 4.2 Spatial distribution of requests and drivers in the target area of Singapore

We also conducted experiments based on the spatial distribution and density variation of supply and demand based on the synthetic dataset. The environment is set to 100−time steps for a 4 km × 4 km grid. The generative patterns of demand and supply are uniform in the temporal context, and experiments are conducted on the uniform and Gaussian distributions in the spatial context. In the gaussian distribution, the mean and standard deviation of driver and passenger locations is (2 km, 2 km) and (0.6 km, 0.6 km), respectively. The arrival density (person/s) is also set from 0.5 to 2.

We assume that the maximum time passengers can tolerate cancellations before and after matching follows a gaussian distribution. Cancellations before matching have a mean and standard deviation of 25s and 12s, respectively, and cancellations after matching are 600s and 120s. Passengers cancel the request when this threshold is exceeded.

The proposed methods and baselines are evaluated regarding matching success rate ($= served\ requests\ /\ total\ requests$) and matching failure factors (CBM, CAM, EXPIRED). The passenger waiting time is measured for the final served request. Approximation networks used in DQN consist of two fully connected layers of 256 and 256 neurons. All experiments are measured as the average value after 50 experiments to ensure the robustness of the results.

4.3. Experiments on synthetic datasets

We observe the effects of changes in fixed matching time intervals before applying the proposed method. **Figure 4.3** shows the difference in the matching success rate according to the matching time interval. The experiment was conducted with passenger requests and drivers in uniform distribution and arrival density of 1.0 (person/s).

The matching time interval varies from 1s to 18s at 2s intervals. Smoothed curves increase performance as fixed matching time intervals increase and then decrease (8−10s) at the highest matching success rate. This represents the potential benefits of matching time intervals mentioned in Figure 1.1, and too long matching time intervals can instead reduce performance. These results suggest that an appropriate matching time interval is required in specific scenarios.

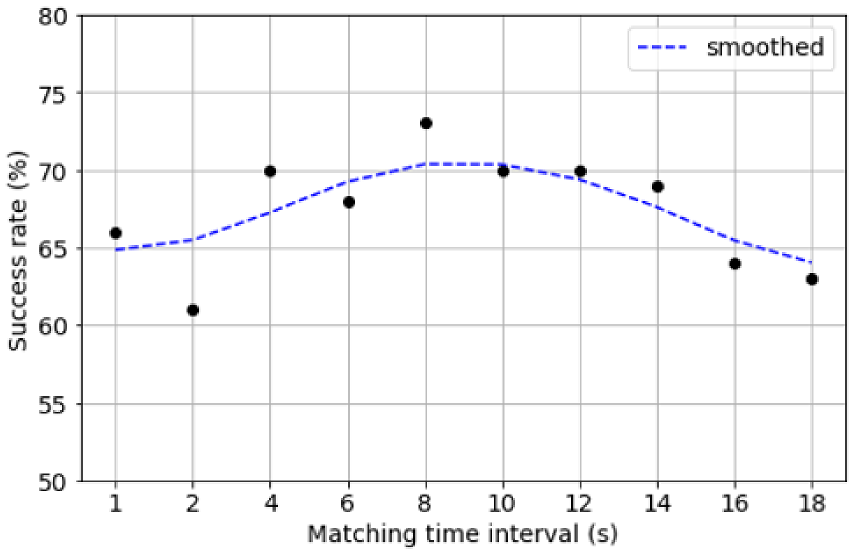


Figure 4.3 The difference in matching success rate according to matching time interval

Figure 4.4 shows the detailed matching failure factors according to the matching time interval. As the matching time interval increases, passengers' cancellation before matching (CBM) increases and expired requests (EXPIRED) decrease. We confirm the trade−off between these two factors, and the matching success rate is maximized at the matching time interval of 8−10s, which is the crossing point (**Figure 4.3**). This can be understood in the same context as previous studies (Qin et al., 2021) regarding passenger waiting time. They discussed a trade−off in the relationship between match waiting time and pick−up waiting time and noted that this tends to be observed only in specific demand−supply ratios.

Cancellation after matching (CAM) is not controlled by the proposed method but can be derived by other factors. For example, if the matching time interval is short, it is likely to be assigned to drivers at relatively long distances. This increases the pick−up waiting time, which may cause the request to be canceled.

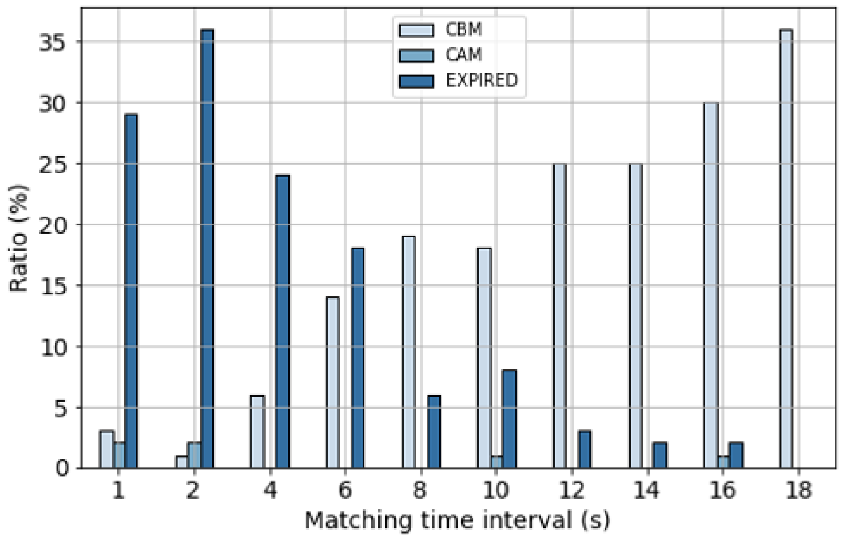
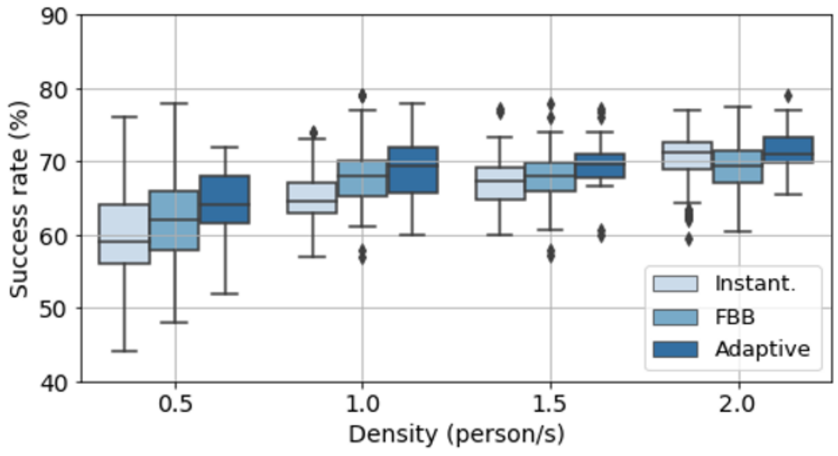


Figure 4.4 The trade−off between matching failure factors according to the matching time interval

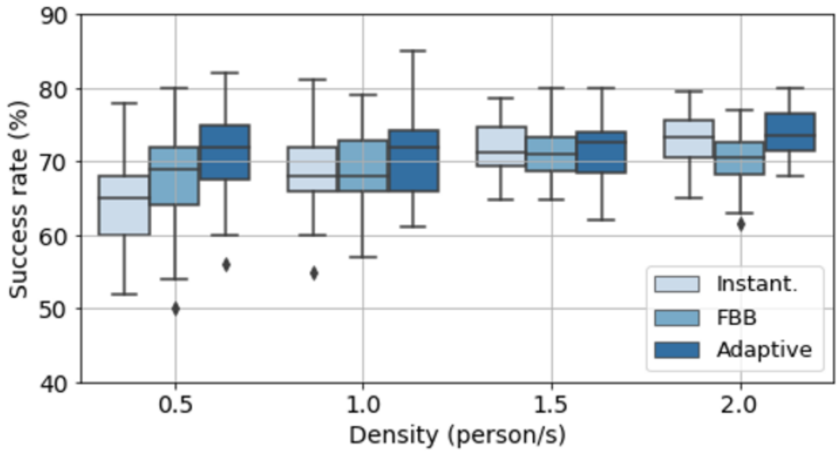
Figure 4.5 shows the matching success rate for three methods for the synthetic dataset. The spatial distribution of demand and supply follows the uniform (**Figure 4.5a**) and gaussian (**Figure 4.5b**) distribution, with the arrival density of passengers and drivers increasing from 0.5 (person/s) to 2.0 (person/s). Through this, we compared the performance of the strategies according to the spatiotemporal distribution.

Adaptive matching based on the DQN agent shows the highest matching success rate in most scenarios. When the demand and supply generation patterns are uniform (**Figure 4.5a**), as the density increases, each method does not show the advantages of other strategies over the immediately matching strategy. FBB even recorded lower performance than Instant at a density of 2.0 (person/s). This is because as the density increases, the number of valid pairs between the driver and the passenger increases, and thus the advantage of controlling the matching time interval decreases.

When the demand and supply generation pattern is gaussian (**Figure 4.5b**), the overall matching success rate increases, but the performance difference between strategies decreases. At the largest density of 2.0 (person/s), FFB has the lowest performance. This is because Gaussian has a higher effect even at the same density.



(a)



(b)

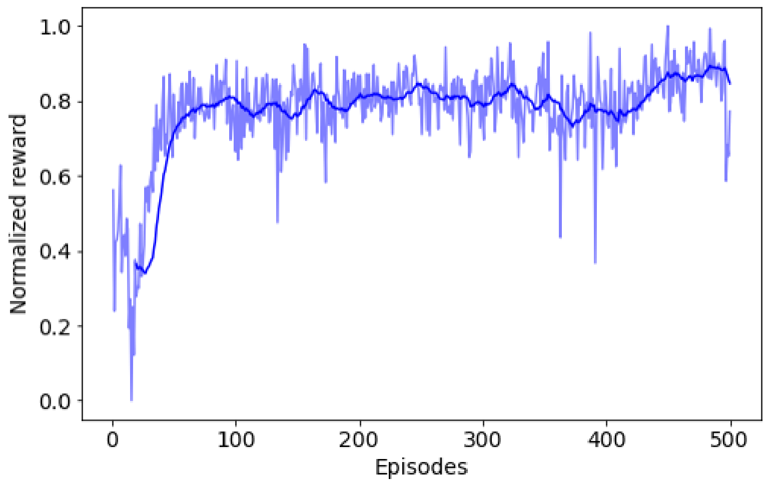
Figure 4.5 Comparison of model performance for uniform (a) and gaussian (b) distribution according to density variation

4.4. Experiments on real datasets

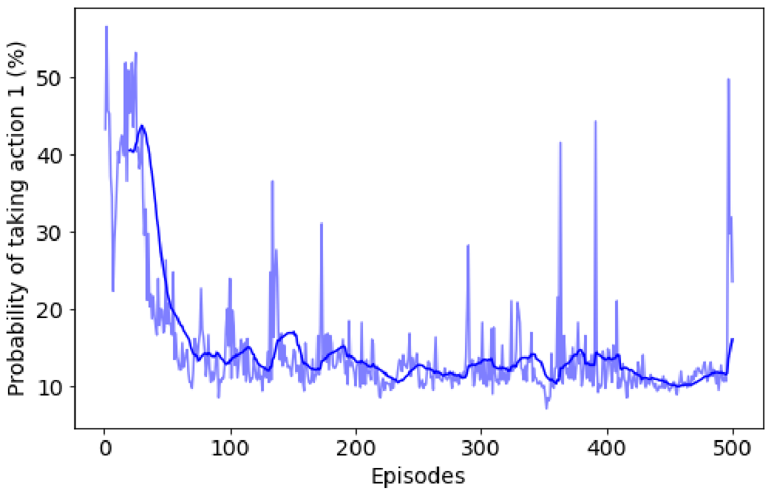
We evaluated the performance of the proposed method based on a simulator calibrated with real datasets. DQN agent interacts with the environment and learns optimal policies based on epsilon greedy exploration. The hyper-parameters used for learning were set to 0.9, 0.01, and 200 for the start, end, and decay weights, respectively. The learning rate for approximation of the state-value function is set to 0.001, and the batch size for experience replay is set to 64.

Figure 4.6 shows the convergence curve for the normalized reward (**Figure 4.6a**) and probability of taking action 1 (**Figure 4.6b**) while the DQN agent is learning. Through the convergence curve, we can confirm that the agent learns optimal policies while maximizing cumulative rewards. Total action probability refers to the proportion of the matching action of the agent in all matching time intervals. For example, suppose each matching time interval is 1 second and interacts with the environment for 100-time steps. In that case, it can be understood that 50% action probability is an average of 2s, which determines the matching time interval.

Therefore, DQN agent initially randomly takes action to about 0.5 and then decreases the proportion of total action probabilities to about 0.1. And when the agent reaches a certain probability, it is shown that it tries to optimize while maintaining the matching time interval. It can be understood that the agent is learning the policy to converge with the reward by controlling the matching time interval.



(a)



(b)

Figure 4.6 Convergence curve (a) and probability of taking action 1 by DQN agent (b) while learning process

TABLE 4.2 lists the results comparing the average performance metrics for the three methods, including the proposed method. And **Figure 4.7** shows multiple experimental results for matching failure factors. The adaptive matching strategy showed the highest performance (60.1%) compared to other strategies, followed by FBB (55.3%) and Instant. (50.6%). Interestingly, the distribution of matching failure factors appears differently in the three strategies.

The instantaneous matching (Instant) strategy shows the lowest cancellation before matching rate (CBM) compared to other strategies (1.6%), but the expired request (EXPIRED) rate is the highest (39.5%). It can be understood because of repeated allocation to drivers with low acceptance probabilities while idle drivers are not accumulated in the queue.

The fixed batch-based matching (FBB) strategy reduces the expired request (EXPIRED) rate (16.5%) by combinatorial optimization based on a fixed matching time interval. However, the cancellation before matching (CBM) rate increases with match waiting time (13.8%). As the matching requests increase, the cancellation after matching (CAM) rate for the pick-up waiting time also increases (14.4%). Although pick-up distance is an influential variable in the driver's acceptance probability, cancellation increases with the gap between passengers and drivers.

The proposed adaptive matching strategy method showed the highest matching success rate. The expired request (EXPIRED) rate was the lowest (12.5%). The objective function of the two-step framework induces as many requests as possible to be matched with the driver. The matching success rate is maximized from a long-term perspective through the balance between matching failure factors. Although the cancellation before matching (CBM) rate was higher than other strategies, the expired request rate was the lowest, and the cancellation after matching (CAM) rate was lower than FBB.

TABLE 4.2 Comparison of average performances of different strategies

Average performances	Instant.	FBB	Adaptive
Success rate(%)	50.6	55.3	60.1
CBM (%)	1.6	13.8	12.4
CAM (%)	8.3	14.4	13.7
EXPIRED(%)	39.5	16.5	12.5
Match waiting time(s)	1.1	10.4	8.5
Pick−up waiting time(s)	175.2	196.0	224.2
Total waiting time(s)	176.3	206.4	232.7

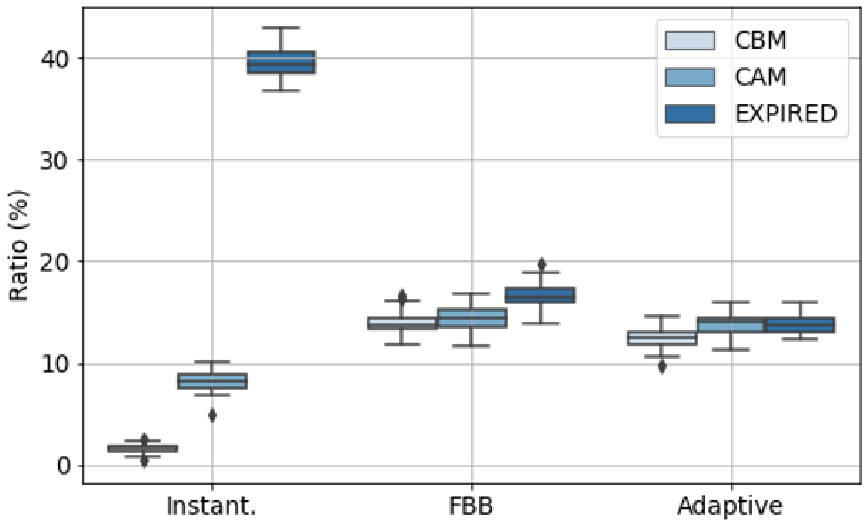
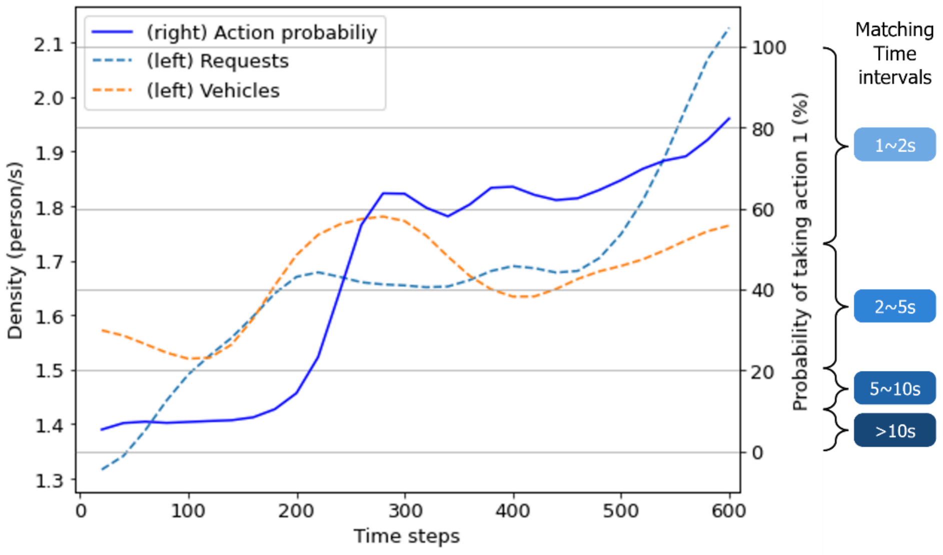


Figure 4.7 Comparison of matching failure factors between different strategies

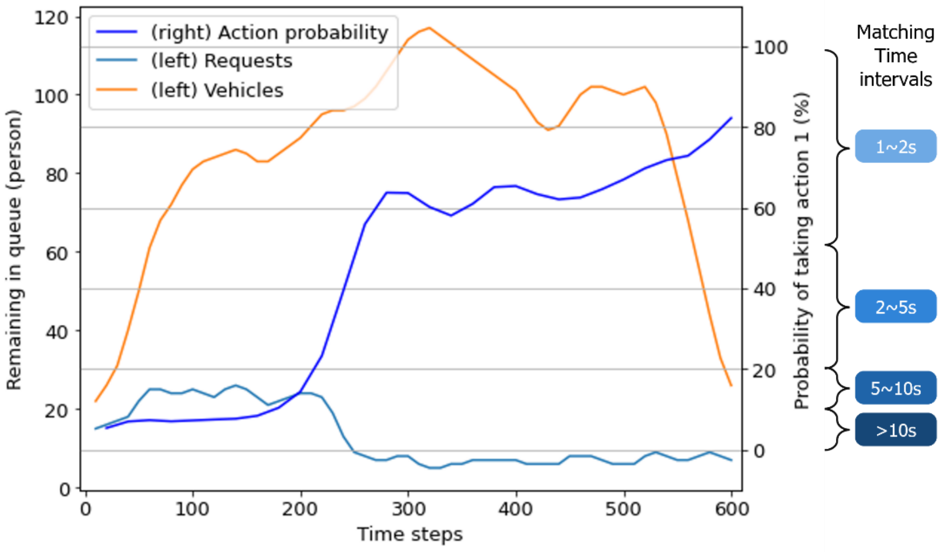
We interpreted how matching actions unfold under the learned policy based on the method motivated by Qin et al. (2021). **Figure 4.8** shows the action probability curves of DQN agents according to the (**Figure 4.8a**) arrival density and (**Figure 4.8b**) queue of passengers and drivers at each time step. In sampled datasets, the arrival density for passenger requests increases, and drivers appear relatively uniform. Passenger requests leave the queue due to success or failure, and the idle drivers remain relatively long.

The action probability curve can be divided into three stages. The first is the initial stage (0–200s), with fewer drivers and requests accumulated in the queue. The agent maintains the matching time interval of 5 to 10 s because the arrival density is relatively low. The second is the intermediate stage (200–400s). As the arrival density of the request increases to 1.7 (person/s) and the remaining idle drivers are accumulated, the agent reduces the matching time interval to 2 to 5s. Later, like instantaneous matching, the matching time interval is reduced by 1 to 2s and maintained with arrival density. The third is the final step (400-600s), maintaining instantaneous matching and reducing the interval further by increasing the arrival density of requests.

From this, we can intuitively confirm that DQN agents adaptively control matching time intervals according to generation patterns and queue. It suggests that the proposed method can increase the matching success rate by using comparison strategies in a timely manner.



(a)



(b)

Figure 4.8 Observation of agent matching actions according to learned policies with (a) density variation and (b) queue (motivated by Qin et al., 2021)

We analyzed the performance for each time step range in detail. **Figure 4.9** shows the matching success rate between strategies in each time step range. The overall matching success rate was high in the order of Adaptive, FBB, and Instant (**TABLE 4.2**), but there are differences in each time step range. **TABLE 4.3** lists details of the matching failure factors. The values in the table refer to the percentage of requests made in each time step range. We discuss different perspectives depending on the time-step range.

The first is the range [0-200s] with potential benefits of matching time intervals, where Adaptive has the highest matching success rate. Adaptive has a lower percentage of expired requests (EXPIRED) compared to FBB, but fewer cancellations before matching (CBM) increase. This suggests that the proposed method can achieve better results than trade-offs due to fixed matching time intervals. In the second range (200-400s], the FBB achieves slightly higher performance. The fixed matching time interval (FBB) appears to be more stable because Adaptive needs to transform the matching time interval. However, performance differences are not noticeable. The third range (400-600s] is in which instantaneous matching is required. Therefore, the advantage of the matching time interval is not noticeable, so Instant outperforms FBB.

We discussed the advantages and disadvantages of each strategy by separating the aggregated results into each time step range. This allows us to understand under which conditions the proposed method can have a high matching success rate.

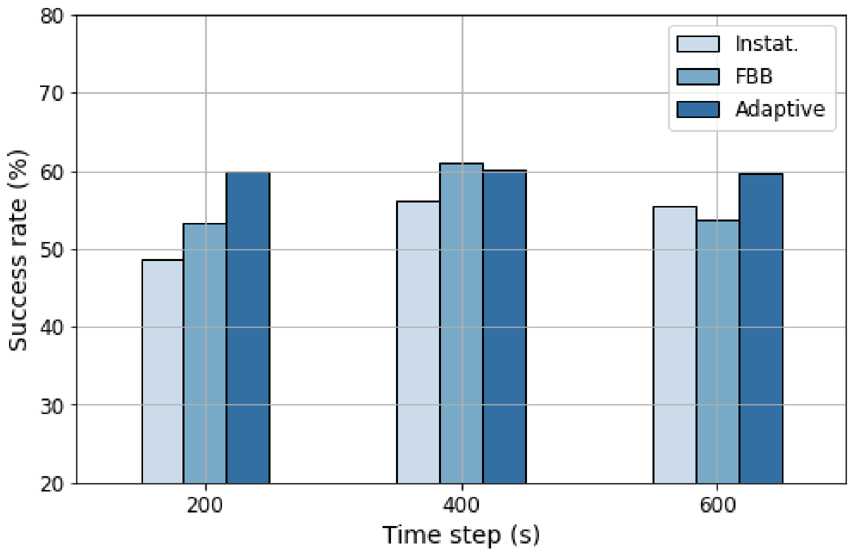


Figure 4.9 Success rate between strategies according to time step

TABLE 4.3 Comparison of results between our and other strategies according to time step

Time step	Method	Served	CBM	CAM	Expired	Total
[0,200]	Instant.	49%	2%	5%	45%	300
	FBB	53%	19%	10%	18%	
	Adaptive	60%	24%	10%	6%	
(200,400]	Instant.	56%	3%	3%	38%	331
	FBB	61%	14%	7%	18%	
	Adaptive	60%	4%	4%	33%	
(400,600]	Instant.	56%	2%	4%	38%	369
	FBB	54%	14%	12%	20%	
	Adaptive	60%	2%	6%	32%	

Chapter 5. Conclusion

In ride−hailing services, the supply−demand imbalance problem causes inefficiency in terms of system operation. To address this, recent studies are interested in determining matching time intervals based on reinforcement learning in the temporal context. However, matching failure factors according to passengers and drivers are not considered. Passengers can cancel their requests before and after matching as the waiting time increases, and drivers can accept requests based on their preferences. These have a significant impact on the order dispatching system.

This study aimed to determine adaptive matching time intervals based on reinforcement learning, considering the matching failure factors. To this end, we propose a two−step framework to maximize the matching success rate. The reinforcement learning agent determines the matching time interval in the first step. And then, combinatorial optimization is performed based on the estimated driver acceptance probability within the current matching time interval.

The agent interacts with simulators developed based on real datasets and learned optimal policies using Deep Q−Network. We compared and evaluated the performance of the existing and proposed strategies through experiments based on synthetic datasets and real datasets. Finally, we discuss how agent controls matching time intervals by providing demand and supply samples to the environment and observing action probability based on learned policies.

This approach allows us to consider integrally the control of matching time intervals in the temporal context and the matching failure factors that have been overlooked in previous studies. Therefore, we were able to discuss the performance as well as the detailed matching failure factors. In the last section, we provided insights through a visual analysis of agent policies that are difficult to understand intuitively.

We first conducted experiments on the spatial distribution and density variation of demand and supply generation patterns based on the synthetic datasets in the grid network. The results showed that the proposed method had the highest matching success rate in most experimental settings. We also discussed the inevitability of instantaneous matching as the arrival density increases.

Next, we observed the distribution of matching success rates and matching failure factors on the results based on real datasets. Each strategy showed a different distribution of cancellations or expired requests. However, we confirm that the proposed method can control these matching failure factors and maximize the matching success rate.

Finally, we visualized and analyzed the agent's policy patterns. We observe that the agent initially increases the matching time interval and then decreases as the arrival density of the request increases and the remaining idle drivers are accumulated. We have identified the benefits of adaptively controlling matching time intervals according to supply−demand patterns.

In this study, we limited the target area. Our experiments' short time steps make it difficult to discuss macroscopic supply−demand patterns. Therefore, we will expand our spatio−temporal context and scope in the future. We will then design the experiment with quantitative supply-demand indicators and discuss generalized performance. For spatial ranges, we can control the matching radius (Yang et al., 2020) or extend to a framework for learning multi-dimensional action policies based on grids in Singapore. We will also be able to expand the dimension of the state variable and improve the matching efficiency based on the taxi demand prediction model.

Bibliography

Bimpikis, K., Candogan, O., Saban, D., 2019. Spatial pricing in ride−sharing networks. Oper Res 67, 744–769. https://doi.org/10.1287/opre.2018.1800

Chen, H., Jiao, Y., Qin, Z., Tang, X., Li, H., An, B., Zhu, H., Ye, J., 2019. InBEDE: Integrating contextual bandit with td learning for joint pricing and dispatch of ride−hailing platforms, in: Proceedings − IEEE International Conference on Data Mining, ICDM. Institute of Electrical and Electronics Engineers Inc., pp. 61–70. https://doi.org/10.1109/ICDM.2019.00016

Karp, R.M., Vazirani, U. v, Vazirani, V. v, 1990. An Optimal Algorithm for On−line Bipartite Matching.

Ke, J., Xiao, F., Yang, H., Ye, J., 2022. Learning to Delay in Ride−Sourcing Systems: A Multi−Agent Deep Reinforcement Learning Framework. IEEE Trans Knowl Data Eng 34, 2280–2292. https://doi.org/10.1109/TKDE.2020.3006084

Lee, D.−H., Wang, H., Cheu, R.L., Teo, S.H., n.d. Taxi Dispatch System Based on Current Demands and Real−Time Traffic Conditions.

Li, M., Yang, Y., Wang, C., Qin, Z., Gong, Z., Wu, G., Jiao, Y., Wang, J., Ye, J., 2019. Efficient ridesharing order dispatching with mean field multi−agent reinforcement learning, in: The Web Conference 2019 − Proceedings of the World Wide Web Conference, WWW 2019. Association for Computing Machinery, Inc, pp. 983–994. https://doi.org/10.1145/3308558.3313433

Liu, Z., Li, J., Wu, K., 2022a. Context−Aware Taxi Dispatching at

City−Scale Using Deep Reinforcement Learning. IEEE Transactions on Intelligent Transportation Systems 23, 1996–2009. https://doi.org/10.1109/TITS.2020.3030252

Liu, Z., Li, J., Wu, K., 2022b. Context−Aware Taxi Dispatching at City−Scale Using Deep Reinforcement Learning. IEEE Transactions on Intelligent Transportation Systems 23, 1996–2009. https://doi.org/10.1109/TITS.2020.3030252

Mehta, A., Saberi, A., Vazirani, U., Vazirani, V., 2007. AdWords and generalized online matching. Journal of the ACM 54. https://doi.org/10.1145/1284320.1284321

Mnih, V., Kavukcuoglu, K., Silver, D., Rusu, A.A., Veness, J., Bellemare, M.G., Graves, A., Riedmiller, M., Fidjeland, A.K., Ostrovski, G., Petersen, S., Beattie, C., Sadik, A., Antonoglou, I., King, H., Kumaran, D., Wierstra, D., Legg, S., Hassabis, D., 2015. Human−level control through deep reinforcement learning. Nature 518, 529–533. https://doi.org/10.1038/nature14236

Özkan, E., Ward, A.R., 2020. Dynamic matching for real−time ride sharing. Stochastic Systems 10, 29–70. https://doi.org/10.1287/stsy.2019.0037

Qin, G., Luo, Q., Yin, Y., Sun, J., Ye, J., 2021a. Optimizing matching time intervals for ride−hailing services using reinforcement learning. Transp Res Part C Emerg Technol 129. https://doi.org/10.1016/j.trc.2021.103239

Qin, G., Luo, Q., Yin, Y., Sun, J., Ye, J., 2021b. Optimizing matching time intervals for ride−hailing services using reinforcement learning. Transp Res Part C Emerg Technol 129. https://doi.org/10.1016/j.trc.2021.103239

Qin, Z., Tang, X., Jiao, Y., Zhang, F., Xu, Z., Zhu, H., Ye, J., 2020. Ride−hailing order dispatching at DiDi via reinforcement learning.

Interfaces (Providence) 50, 272–286. https://doi.org/10.1287/INTE.2020.1047

Tong, Y., Chen, Y., Zhou, Z., Chen, L., Wang, J., Yang, Q., Ye, J., Lv, W., 2017. The Simpler the Better: A unified approach to predicting original taxi demands based on large−scale online platforms, in: Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining. Association for Computing Machinery, pp. 1653–1662. https://doi.org/10.1145/3097983.3098018

Tong, Y., She, J., Ding, B., Wang, L., Chen, L., 2016. Online mobile Micro−Task Allocation in spatial crowdsourcing, in: 2016 IEEE 32nd International Conference on Data Engineering, ICDE 2016. Institute of Electrical and Electronics Engineers Inc., pp. 49–60. https://doi.org/10.1109/ICDE.2016.7498228

Wang, Y., Tong, Y., Long, C., Xu, P., Xu, K., Lv, W., 2019. Adaptive dynamic bipartite graph matching: A reinforcement learning approach, in: Proceedings − International Conference on Data Engineering. IEEE Computer Society, pp. 1478–1489. https://doi.org/10.1109/ICDE.2019.00133

Wang, Z., Qin, Z., Tang, X., Ye, J., Zhu, H., 2018. Deep Reinforcement Learning with Knowledge Transfer for Online Rides Order Dispatching, in: Proceedings − IEEE International Conference on Data Mining, ICDM. Institute of Electrical and Electronics Engineers Inc., pp. 617–626. https://doi.org/10.1109/ICDM.2018.00077

Watkins, C.J.C.H., Dayan, P., 1992. Q−Learning.

Xu, Z., Li, Z., Guan, Q., Zhang, D., Li, Q., Nan, J., Liu, C., Bian, W., Ye, J., 2018. Large−scale order dispatch in on−demand ride−hailing platforms: A learning and planning approach, in: Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining. Association for Computing Machinery, pp. 905–913. https://doi.org/10.1145/3219819.3219824

Yan, C., Zhu, H., Korolko, N., Woodard, D., 2020. Dynamic pricing and matching in ride-hailing platforms. Naval Research Logistics 67, 705–724. https://doi.org/10.1002/nav.21872

Yang, H., Qin, X., Ke, J., Ye, J., 2020. Optimizing matching time interval and matching radius in on-demand ride-sourcing markets. Transportation Research Part B: Methodological 131, 84–105. https://doi.org/10.1016/j.trb.2019.11.005

Yang, H., Wong, S.C., Wong, K.I., n.d. Demand-supply equilibrium of taxi services in a network under competition and regulation.

Zhang, L., Hu, T., Min, Y., Wu, G., Zhang, J., Feng, P., Gong, P., Ye, J., 2017. A taxi order dispatch model based on combinatorial optimization, in: Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining. Association for Computing Machinery, pp. 2151–2159. https://doi.org/10.1145/3097983.3098138

Zhao, K., Khryashchev, D., Freire, J., Silva, C., Vo, H., 2016. Predicting taxi demand at high spatial resolution: Approaching the limit of predictability, in: Proceedings - 2016 IEEE International Conference on Big Data, Big Data 2016. Institute of Electrical and Electronics Engineers Inc., pp. 833–842. https://doi.org/10.1109/BigData.2016.7840676

Oda, Takuma, and Carlee Joe-Wong. "MOVI: A model-free approach to dynamic fleet management." IEEE INFOCOM 2018-IEEE Conference on Computer Communications. IEEE, 2018.

Sutton, Richard S., and Andrew G. Barto. "Introduction to reinforcement learning." (1998): 551283.

Puterman, Martin L. Markov decision processes: discrete stochastic dynamic programming. John Wiley & Sons, 2014.

Dijkstra, Edsger W. "A note on two problems in connexion with graphs." Edsger Wybe Dijkstra: His Life, Work, and Legacy. 2022. 287-290.

Hosmer Jr, David W., Stanley Lemeshow, and Rodney X. Sturdivant. Applied logistic regression. Vol. 398. John Wiley & Sons, 2013.

OpenStreetMap, n.d. OpenStreetMap [WWW Document]. URL https://www.openstreetmap.org/ (accessed 5.1.22).

TADA, n.d. TADA[WWW Document]. URL https://tada.global/ (accessed 5.1.22).

Abstract

승차 공유 서비스들은 승객과 운전자들을 효율적으로 연결함으로써 일상 생활의 이동에 많은 도움을 주고 있다. 이러한 서비스들은 수요와 공급의 불균형 문제로 인해 시스템 운영 측면에서 비효율적인 상황에 직면한다. 이를 위해 일정한 매칭 시간 간격 동안 승객의 요청과 공차 통행 중인 운전자들을 모아 일괄적으로 매칭하는 전략을 주로 사용한다. 최근에는 수요와 공급의 동적 패턴을 효과적으로 반영하기 위한 적응형 매칭 시간 간격에 대한 연구들이 있었으나, 승객의 요청 취소와 운전자 거부와 같은 매칭 실패 요인들은 간과되었다. 본 연구의 목표는 매칭 실패 요인이 존재하는 상황에서 강화학습 기반의 적응형 매칭 시간 간격을 통해 매칭 성공률을 최대화하는 것이다. 연구 방법은 2단계 프레임워크로 구성된다. 먼저 DQN (Deep Q-Network) 기반의 강화학습 에이전트는 각 매칭 시간 간격마다 배차 행동(Dispatch action)을 결정하며, 이후에는 운전자의 수락확률을 기반으로 한 조합최적화가 수행된다. 실제 데이터셋을 기반으로 한 실험을 통해 이전 전략들과 성능을 비교하고 매칭 실패 요인들에 대한 분석을 수행한다. 실험 결과, 제안된 방법은 대부분의 실험에서 가장 높은 매칭 성공률을 보였다. 구체적으로는 운전자의 미 수락에 의한 만료 요청의 비율을 감소시키며, 승객의 요청 취소 비율을 효율적으로 제어하는 것을 확인했다. 또한 학습된 에이전트의 정책 해석과 집계된 결과의 세분화를 기반으로 추가 분석이 수행되었다. 이러한 접근 방식은 매칭 성공률과 세부적인 매칭 실패 요인들에 대한 논의를 통해 기존 연구에서 간과되었던 통찰력을 제공한다.

Keyword : Ride-Hailing Service, Reinforcement Learning, Deep Q-Network (DQN), Combinatorial Optimization, Matching Failure
Student Number : 2021-21896