electronics

Article

Fairness-Aware Dynamic Ride-Hailing Matching Based on Reinforcement Learning

Yuan Liang [1,2]

1 School of Information Engineering, Suqian University, Suqian 223800, China; liangyuan120@buaa.edu.cn
2 Guangxi Key Laboratory of Trusted Software, Guilin University of Electronic Technology, Guilin 541004, China

Abstract: The core issue in ridesharing is designing reasonable algorithms to match drivers and passengers. The ridesharing matching problem, influenced by various constraints such as weather, traffic, and supply–demand dynamics in real-world scenarios, requires optimization of multiple objectives like total platform revenue and passenger waiting time. Due to its complexity in terms of constraints and optimization goals, the ridesharing matching problem becomes a central issue in the field of mobile transportation. However, the existing research lacks exploration into the fairness of driver income, and some algorithms are not practically applicable in the industrial context. To address these shortcomings, we have developed a fairness-oriented dynamic matching algorithm for ridesharing, effectively optimizing overall platform efficiency (expected total driver income) and income fairness among drivers (entropy of weighted amortization fairness information between drivers). Firstly, we introduced a temporal dependency of matching outcomes on subsequent matches in the scenario setup and used reinforcement learning to predict these temporal dependencies, overcoming the limitation of traditional matching algorithms that rely solely on historical data and current circumstances for order allocation. Then, we implemented a series of optimization solutions, including the introduction of a time window matching model, pruning operations, and metric representation adjustments, to enhance the algorithm's adaptability and scalability for large datasets. These solutions also ensure the algorithm's efficiency. Finally, experiments conducted on real datasets demonstrate that our fairness-oriented algorithm based on reinforcement learning achieves improvements of 81.4%, 28.5%, and 79.7% over traditional algorithms in terms of fairness, platform utility, and matching efficiency, respectively.

Keywords: fairness aware; ride-hailing matching; reinforcement learning; time window matching model; bipartite matching

check for updates

Citation: Liang, Y. Fairness-Aware Dynamic Ride-Hailing Matching Based on Reinforcement Learning. *Electronics* **2024**, *13*, 775. https://doi.org/10.3390/electronics13040775

Academic Editors: Peiying Zhang, Haotong Cao and Keping Yu

Received: 22 January 2024
Revised: 8 February 2024
Accepted: 14 February 2024
Published: 16 February 2024

1. Introduction

With the rise of mobile internet and improvement in economic levels, more and more people are opting for ride-hailing services to address their transportation needs. In 2020, the daily average order volume on the Didi (https://www.didiglobal.com/ (accessed on 1 February 2024)) platform reached 500 million orders per day. Didi is a prominent mobile transportation platform offering a wide array of ride-hailing services to millions of users across Asia, Latin America, and Australia, providing services including taxi hailing, private car hailing, social ridesharing, and bike sharing. The platform utilizes sophisticated mobile technology to connect passengers with drivers, offering a convenient, fast, and safe way to navigate urban environments. By integrating various forms of transportation services into a single platform, Didi aims to optimize urban transit systems and promote a shift towards more sustainable mobility solutions. Eventually, the drivers complete the customers' requests offline. Therefore, allocating these tasks to drivers in a reasonable, scientific, and effective manner is a challenging issue for ride-hailing platforms.

Take Didi as an example. The platform initially adopted a driver bidding model for task allocation. However, this design gave drivers more autonomy, leading to a low response rate for orders going to remote areas or for short distances, failing to guarantee user benefits. To improve the order response rate, Didi launched its Express car service in 2015, changing the task allocation model from driver bidding to platform-based algorithmic dispatch. This significantly improved the order response rate. However, in this scenario, the design of the algorithm affects the matching results. Assigning "lucrative orders" based on driver ratings often leads to a "winner-takes-all" phenomenon—drivers may put in the same working hours but end up with vastly different earnings, failing to meet the fairness needs at the driver level.

We find that, based on the dispatch model, effectively matching drivers and passengers affects the platform's transaction efficiency, the quality of user experience, and the drivers' willingness to work and faces three main challenges:

(1) Difficulty in balancing multiple benefits: Common-order matching strategies often use greedy algorithms, such as nearest assignment and first come, first served. These principles single-mindedly focus on the passenger's perspective to achieve quality control and fair allocation. However, the results heavily depend on random factors such as the current spatial distribution and the timing of order arrivals, lacking effective protection mechanisms for drivers' interests. Therefore, effectively balancing the interests of drivers, passengers, and the platform is a primary core issue;
(2) Difficulty in assessing future earnings: The current dynamic matching algorithms for ride-hailing services complete a new match between incoming orders and available drivers every two seconds. Even when considering balancing drivers' earnings and prioritizing large orders for drivers with lower earnings for the day, such strategies are still based on historical experience and current situations. They overlook the driver's subsequent spatial location and whether their service period can continue to be profitable, failing to effectively assess future situations. Thus, effectively evaluating the future earnings of drivers and then balancing the order distribution to achieve fairness in driver earnings are other core issues;
(3) Difficulty in designing a fair mechanism: Due to factors such as platform scoring, service time zones, and random elements, drivers may receive drastically different earnings despite putting in the same work hours and intensity. However, the existing research lacks assessment and protection of fairness. Therefore, designing reasonable constraints and fairness protection mechanisms is also a crucial core issue.

Additionally, traditional ride-hailing matching focuses on static scenarios, which do not align with real-world situations. However, in dynamic scenarios, if every order is matched immediately upon arrival, the matching results will overly depend on random factors like the order arrival sequence, which presents certain limitations. To achieve the best possible outcomes over a larger scope and improve matching efficiency while ensuring real-time performance, we adopt a time window matching model. Therefore, we design a system where, every two seconds, incoming orders and available drivers are combined into a large bipartite graph, and we aim to maximize the sum of the weights of the matching edges in this graph. However, reasonably designing the edge weights to effectively balance indicators like driver earnings and total platform revenue is also a challenging aspect of our research.

In line with practical application needs, we assume that our scenario has three characteristics: drivers' statuses change constantly as orders arrive anytime and anywhere; drivers have a certain probability of rejecting orders; and the platform performs a match every two seconds. Considering the core aspects of our research and the predefined scenario, we address the issue of fairness-aware dynamic ride-hailing matching based on reinforcement learning (FMRL). First, we model drivers and passengers as nodes of a bipartite graph based on data uploaded by users and drivers, generating the bipartite graph through edge weight calculation and pruning algorithms. Then, we introduce reinforcement mechanisms and event window matching algorithms in the edge weight design, making it adaptive and overcoming the shortsightedness of traditional methods. Next, we generate matching results using bipartite graph matching algorithms like the KM algorithm. Finally, we verify the effectiveness of the proposed method through experiments on real datasets. In summary, the main contributions of this paper are as follows:

- This paper introduces the temporal dependence of matching results on subsequent matches by employing reinforcement learning methods to predict this dependence. FMRL overcomes the shortsightedness inherent in traditional matching algorithms;
- A series of optimization schemes, including time window matching models, pruning operations, and adjustment of indicator representations, are introduced. These enhancements significantly increase the algorithm's adaptability and scalability to large datasets;
- Through conducting experiments on ride-hailing matching algorithms, it is demonstrated that the fairness distribution algorithm based on reinforcement learning achieves substantial improvements over traditional algorithms. Specifically, enhancements of 81.4% in fairness, 28.5% in platform utility, and 79.7% in matching efficiency are observed.

Section 1 is the introduction; Section 2 reviews related work; Section 3 defines the problem and algorithm evaluation criteria; Section 4 discusses the methods; Section 5 presents the experiments; and the conclusion is arranged in Section 6.

2. Related Work

We divide the related work into two categories: ride-hailing matching algorithms and fairness mechanism design.

2.1. Ride-Hailing Matching Algorithms

In this paper, drivers are regarded as crowdsourced workers, and orders are seen as spatiotemporal crowdsourcing tasks. Therefore, the matching in ride hailing essentially belongs to task allocation in the field of spatiotemporal crowdsourcing. Didi and Uber (https://www.uber.com/ (accessed on 1 February 2024)), as typical applications of spatiotemporal crowdsourcing in the field of ride hailing, have matching algorithms whose optimization is crucial for enhancing user experience, optimizing platform revenue, and ensuring fair earnings for drivers.

Traditional ride-hailing matching algorithms often rely on linear programming methods [1], designing multiple constraints, and determining optimization objectives without considering the dynamic nature of scenarios. Linear programming can yield optimal solutions for modeled problems, but, in real, dynamic scenarios, drivers and passengers join and leave the platform at any time and place, and the platform must respond in real time to passenger requests, making it impossible to design effective algorithms for a global optimal solution. Moreover, the constraints of ride-hailing matching problems are variable, and the optimization objectives are diverse. Reasonable and complex linear programming models often have too high a complexity to be computationally effective.

Therefore, the current research often involves designing efficient approximate algorithms to solve the ride-hailing order matching problem. The most basic idea of approximate algorithms is based on real-time allocation using a greedy approach, meaning, when an order arrives, the matching occurs immediately based on the current optimal strategy for driver–passenger pairing. However, matching based solely on one-sided supply often has temporal limitations and high computational complexity. To balance real-time user needs, algorithmic approximation ratios, and computational resource consumption, fixed time window online matching methods are often used in practice [2]—performing a match every fixed period. In the modeling of the matching problem, the current research often involves designing approximate algorithms based on bipartite graph matching [3]. The main idea is to abstract drivers and passengers as nodes in a bipartite graph [4–6], design edge weights to contain multidimensional information like spatiotemporal aspects, order cancellation rates, and expected earnings, and then use bipartite graph matching algorithms like the KM algorithm to find the maximum match.

Additionally, ride-hailing matching algorithms adopt methods such as the Distance-Based Greedy Algorithm (DG), Earnings-Ratio-Based Greedy algorithm (ERG), Integer Linear Programming (ILP) [7], and Reassignment (REA) [8]. The Distance-Based Greedy Algorithm assigns tasks to the driver nearest to the order's starting point. The Earnings-Ratio-Based Greedy Algorithm ranks drivers in descending order based on a weighted amortization fairness index and orders in descending order of earnings ratio, then matches them sequentially to ensure driver fairness. Linear programming [7,9] transforms into an integer programming problem, taking passenger fairness as the optimization objective with constraints considering driver income fairness. The Reassignment algorithm [8,10] first calculates the allocation results based on order response rate and fairness and then adjusts based on these two allocation results, balancing between response rate and fairness.

However, in ride-hailing matching scenarios, the previous research [7,8,11–13] on dual-objective optimization for total platform revenue and driver income fairness has the following limitations: (1) It overlooks the impact of existing arrangements on future spatiotemporal distribution and matching situations. Many current algorithms set application scenarios that overly exceed the settings of actual scenarios, neglecting many details. Moreover, the driver's location and service status depend on the matching situation in previous time sequences. This dependency can affect the overall revenue and fairness of driver earnings from the matching results, making it difficult to verify the effectiveness of the algorithm in real scenarios. (2) Linear programming is a classic approach to solving this optimization problem. However, to balance total platform revenue and fairness among drivers' incomes, ride-hailing matching is a dual-objective optimization problem. The computational complexity of linear programming algorithms is too high, making them unsuitable for expansion to real-world, large-scale data scenarios.

2.2. Fairness Mechanism Design

In addressing fairness mechanism design issues in dynamic ride-hailing matching algorithms, the existing research suffers from a narrow perspective and lacks individual measurement indicators [14–17]. Firstly, the current studies mainly optimize algorithms from the perspective of passengers or platforms, such as maximizing platform revenue or minimizing passenger waiting time, with fewer studies considering the perspective of drivers. Secondly, many allocation algorithms aim to optimize single global indicators such as maximizing total revenue [3,18,19], minimizing travel costs [4,20], or reducing passenger waiting times [21], failing to effectively balance the interests of all parties.

Recently, Refs. [7,8,22,23] have started to explore balancing objectives like driver income during the task allocation process. As future scenarios are unpredictable, algorithms allocate orders based only on historical data and current situations. To balance the income disparity among drivers, the Earnings-Ratio-Based Greedy Algorithm tends to assign high-value orders to lower-income drivers. However, this approach overlooks factors like the order's destination and the driver's future service zones, which can significantly impact their future earnings. This shortsightedness in existing research methods also impacts fairness to some extent. Moreover, platforms currently aim to maximize total revenue through matching optimization. In this context, many orders that do not yield high profits are discarded, and many drivers in 'cold zones' are not allocated orders [24–26], failing to meet fairness requirements.

In addition, in the matching scenario of ride-hailing services, existing algorithms are mostly based on the current scene and fixed algorithms for matching [17,27,28], which cannot make reasonable use of past experience because they ignore future spatiotemporal distribution and income changes and have the shortcomings of shortsightedness and poor adaptability [29]. Therefore, a preliminary idea is to introduce reinforcement learning into the algorithm design to overcome the problem of poor algorithm adaptability. Reinforcement learning has a wide range of applications in ride-hailing scenarios, such as dynamic pricing [30] and online matching [31–33]. Dynamic pricing [30] sets different prices as actions, and the corresponding revenue is the total platform revenue generated by this pricing subsequently. Regarding online matching [32], the existing research often models drivers as intelligent agents whose state, actions, and benefits can be intuitively defined. In practical scenarios, to avoid multi-agent problems, all drivers are often modeled as the same intelligent agent. Ref. [34] integrates reinforcement learning for online matching, dividing ride-hailing matching into two parts: online planning and offline learning. Online planning defines edge weight calculation methods related to order price, location value, and pick-up distance and then completes matching through the maximum bipartite graph matching formula. Offline learning achieves value updates through temporal differential errors where the value of each grid depends on the driver's income within that grid. Ref. [18] achieved optimization based on [34], representing each location as a vector and updating the value function of the driver's location through a deep neural network. To make the value function smoother and more robust, Lipschitz regularization is adopted.

Due to the lack of research on fairness mechanisms in the field of ride-hailing matching, we expanded the scope of our fairness study to the domain of spatiotemporal crowdsourcing. In [24], the author, based on the issue of computational resource allocation, detailed the theory of fair distribution, setting dynamic programming constraints and optimization objectives such as incentive sharing, customized allocation strategies, non-deception, and optimality. Refs. [24,35] extended this to the field of dynamic computational resource allocation, achieving dynamic fair distribution through the relaxation of some constraints in fair distribution theories. The research on fairness mechanisms often considers various task arrival distributions, testing the algorithm's matching rate and fairness under different scenarios. We considered several task arrival distributions as follows: (1) adversarial order, where the order arrival is precisely the worst-case scenario for the algorithm; (2) random order, where order arrivals conform to the average real situation; (3) independent and identically distributed (i.i.d.), where dynamically arriving orders and task participants are distributed in a non-independent and identically distributed manner; (4) known i.i.d., where orders and participant distributions are non-independent and identically distributed and the algorithm is aware of these distribution characteristics in advance; (5) known adversarial distribution, where orders and participants satisfy a certain random, independent distribution, distinct from each other, and the algorithm knows their distribution characteristics in advance.

3. Problem Definition and Algorithm Evaluation Criteria

In this section, we first provide a definition of the problem; secondly, we introduce bipartite graph matching; then, we discuss the time window matching model. Following that, we offer formal definitions of the relevant entities and models within the algorithm, and, finally, we explore the rational design of algorithm evaluation metrics and provide their formal definitions.

3.1. Problem Definition

This subsection specifically describes the modeling approach for ride-hailing drivers' orders and their symbolic abstract representation methods, as well as the algorithm matching model based on bipartite graphs.

3.1.1. Order and Driver

Definition 1. *(Order) A ride-hailing order can be represented by a tuple* $< o_r, d_r, p_r, \tau_r >$*, where* o_r, d_r, p_r, τ_r *respectively represent the starting point, destination, earnings, and duration of the task.*

For each order, the starting and ending points are defined by the passenger who submits the order. Once the order is uploaded to the platform, the platform determines the earnings and duration of the task based on the distance between the starting and ending points, as well as external conditions like weather, supply and demand, and traffic. A commonly used algorithm for determining task arrival time is the estimated time of arrival [36–38]. In the research scenario of this paper, since the error in the estimated time of arrival does not significantly affect the accuracy of the algorithm, to ensure the efficiency of the algorithm, the duration of the order as a multiple of the distance between the starting and ending points is defined.

Definition 2. *(Driver) A ride-hailing driver can be represented by a tuple* $< l_w^{(t)}, u_w^{(t)}, a_w^{(t)} >$, *where* $l_w^{(t)}, u_w^{(t)}, a_w^{(t)}$ *respectively represent the driver's current location, income, and status.*

Here, $l_w^{(t)}$ indicates the driver's GPS information at time t. When the driver is in a free state, their income $u_w^{(t)}$ at time t is zero. When assigned to task r, their income $u_w^{(t)}$ is the unit time earnings brought by the order, calculated as p_r/τ_r.

Furthermore, a driver's status can be categorized into three states: offline, idle, and on duty. The range of values for $a_w^{(t)}$ is $0, 1$. When the driver is offline, $a_w^{(t)}$ is 0. When the driver is idle or on duty, $a_w^{(t)}$ is 1. Within a time window, it is assumed that the value of $a_w^{(t)}$ remains constant. A detailed explanation of the symbols used in this paper is provided in Table 1.

Table 1. Symbol description.

Symbol	Description
T	Divide a day into different time periods using a time window in hours
t	Batch processing time window in seconds
W, R	Driver, order
$W^{(t)}, R^{(t)}$	The set of all idle drivers and unmatched order in batch t
$l_w^{(t)}, u_w^{(t)}, a_w^{(t)}$	Driver's location, income, and status in batch t
o_r, d_r, p_r, τ_r	The starting point, destination, earnings, and duration of the task
$D^{(t)}$	To-be-matched bipartite graph established in batch t
$E^{(t)}$	Matchable edges in batch t
$M^{(t)}$	Actual matching edges in batch t
$\lambda_{w,r}$	The rejection rate of matching driver orders (w, r)
$\theta_{w,r}$	Matching the edge weights of driver orders (w, r)
U	Total revenue
F_w	Weighted amortized fairness
$\xi_{(t)}$	The weighted amortized fairness during that time period
F	Time benefit fairness
V	Value function, where $V(L)$ represents the value function of location l
H, S	The value functions of hexagons and quadrilaterals
S	Guidance scheme for idle drivers

In evaluating actual ride-hailing matching algorithms, many existing research approaches simulate batches of independent drivers for each time window based on the actual spatiotemporal distribution of drivers [4,21]. However, in reality, a driver's matching result in one state often affects their spatiotemporal distribution in the next phase. Therefore, existing algorithm evaluation approaches overlook the impact of algorithms on driver distribution and cannot comprehensively evaluate the effectiveness of algorithms. To make the research scenario more realistic, FMRL maintains a fixed group of drivers and simulates changes in driver distribution through actual order acceptance, repositioning, and online/offline behaviors.

3.1.2. Bipartite Matching

Definition 3. *(Bipartite Graph) The bipartite graph* $G^{(t)} =< R^{(t)}, W^{(t)}, E^{(t)} >$ *represents the matching status between drivers and orders at time* t.

Here, the sets $R^{(t)}$ and $W^{(t)}$ respectively represent the set of orders available for matching and the set of idle drivers at time t. If driver w and order r can be matched, then there exists an edge $(r, w) \in E^{(t)}$ in the bipartite graph. The weight of the edge (r, w) is denoted by $\theta_{r,w}$.

In setting the edge weights, traditional ride-hailing matching algorithms (such as the Distance-Based Greedy Algorithm or Earnings-Ratio-Based Greedy Algorithm) often set them according to the objectives they aim to optimize (e.g., distance between driver and order, earnings ratio of the order). In this paper, to design a ride-hailing matching algorithm that balances both the total efficiency of the platform and the fairness of driver income, the earnings of orders p_r and the cancellation rate $\lambda_{r,w}$ are taken into consideration. Here, the cancellation rate is predicted based on the customer's historical order behavior.

3.2. Time Window Matching Model

To simplify the problem as much as possible, the existing research has largely focused on studying the ride-hailing matching problem in static scenarios where all information about drivers and orders is known in advance. Hence, with predetermined optimization objectives and constraints, and given that all inputs are known beforehand, static matching algorithms can achieve globally optimal solutions. However, in real-world scenarios, passenger orders often require real-time responses, making traditional offline solutions impractical.

To implement dynamic matching in ride-hailing services, an intuitive solution is to immediately match new orders with available drivers as they arrive. However, once a match is made, it cannot be undone, even if a more suitable driver becomes available in the next moment or if an order arrives that is closer to or more profitable for the already matched driver. Changing the match could improve overall performance. Therefore, the immediate matching algorithm has a shortsighted flaw, leading to local optima and compromising the global optimization goal. Additionally, this order-arrival-based computation model significantly reduces algorithm efficiency when the volume of orders increases, failing to meet the need for algorithmic efficiency.

To balance the real-time needs of passengers while considering algorithm efficiency and optimization objectives, this paper adopts a time window matching model [7,8]. In designing the size of the time window based on long-term exploration in the industry, FMRL chooses every two seconds [39] as a time window in which to perform bipartite graph matching once for new orders arriving on the platform and idle drivers based on the designed algorithm strategy. The time window matching model is widely used in ride-hailing matching platforms [18,39,40].

3.3. Algorithm Evaluation Criteria

This subsection primarily explores the evaluation criteria for algorithms that balance the overall efficiency of the platform with the fairness of driver income. First, we introduce the definition of platform utility evaluation metrics; secondly, we present the definition of fairness evaluation metrics and their representation for optimization.

3.3.1. Utility Function

Definition 4. *(Utility Function) Given the set of drivers W and the set of tasks R, the total utility function of the platform is the sum of all drivers' expected earnings before time T, expressed by the following equation:*

$$U = \sum_{w \in W} \mathbb{E}[\sum_{t \in T} u_w^{(t)}] \tag{1}$$

Here, each driver's expected earnings depend on the sum of all matched order earnings during their service time as follows:

$$\mathbb{E}[\sum_{t \in T} u_w^{(t)}] = \sum_{t \in T} \sum_{(r,w) \in M^{(t)}} (1 - \lambda_{w,r}) \cdot p_r \tag{2}$$

Based on this definition, the paper uses the utility function as an evaluation metric for the algorithm in optimizing the total revenue of the platform.

3.3.2. Fairness

In terms of fairness evaluation metrics, the existing research often uses amortized fairness [7] to characterize and optimize income fairness among drivers.

Definition 5. *(Amortized Fairness) Amortized fairness is defined as the unit time earnings of drivers during their online periods.*

$$\frac{\sum_{t \in T} u_w^{(t)}}{\sum_{t \in T} a_w^{(t)}} \tag{3}$$

However, due to the differences in working hours and patterns among drivers, amortized fairness does not adequately reflect the fairness of earnings between drivers. For instance, there will inevitably be differences in earnings between drivers who primarily work during the day and those who work at night. Additionally, the city in which a driver operates and their frequent areas of activity can also impact their unit time earnings. Therefore, enforcing the same unit time earnings for drivers in each time period and area can actually harm the fairness of driver income. To offset the impact of differences in driving habits, this paper defines weighted amortized fairness.

Definition 6. *(Weighted Amortized Fairness) Weighted amortized fairness refers to the cumulative weighted earnings per unit time for drivers during their online periods.*

$$F_w = \frac{\sum_{t \in T} \frac{u_w^{(t)}}{\xi^{(t)}}}{\sum_{t \in T} a_w^{(t)}} \tag{4}$$

where $\xi^{(t)}$ represents the weighting factor used to normalize the income for different times of the day. Since drivers often assess their earnings based on daily and hourly wages in actual matching scenarios [41], the formula defines T as the interval for one day and t as the interval for two seconds, with $\xi^{(t)}$ being the average hourly wage of drivers during that time period.

As can be seen from the formula, weighted amortized fairness, building upon the concept of amortized fairness, divides each driver's hourly earnings by the average hourly wage of drivers during that time period. This calculation method aims to eliminate the impact of drivers' working habits on the fairness metric as much as possible. However, this type of fairness is established only on an individual driver basis. To evaluate the algorithm's protection of individual driver fairness from a macro perspective, temporal earnings fairness is defined.

Definition 7. *(Temporal Earnings Fairness) Given the set of drivers W and the dynamically arriving set of tasks R, the overall temporal earnings fairness of drivers is measured by the entropy change of the weighted average amortized fairness. The formula for calculating temporal earnings fairness, which is the metric for evaluating the fairness of the algorithm in this paper, is as follows:*

$$F = -\sum_{w \in W} \log\left(\frac{F_w}{\max_{w' \in W} F_{w'}}\right) \tag{5}$$

When F is larger, it indicates that the distribution of the amortized fairness metric among drivers is more dispersed, implying poorer income fairness among drivers. Therefore, the optimization objective for fairness is to minimize temporal earnings fairness.

4. Method

We adopt a fairness allocation algorithm based on reinforcement learning. First, we provide an overview of this approach, discussing its framework, implementation mechanism, and design philosophy. Next, we explore the specific implementation mechanisms, optimization ideas, and algorithmic processes of each submodule. Finally, we discuss the integration of the algorithm.

4.1. Algorithm Overview

The fairness allocation algorithm based on reinforcement learning optimizes the two following key metrics: total platform revenue and driver income fairness. In optimizing platform utility, traditional algorithms are limited by their inability to effectively forecast future arrivals, exhibiting shortsightedness compared to optimal matching results in static scenarios.

To overcome this shortsightedness, the fairness allocation algorithm based on reinforcement learning introduces a reinforcement learning mechanism in setting edge weights. It adjusts the weights according to the matching situation and the relative value of the order's starting and ending regions, allowing the matching result to consider future order arrivals and driver spatiotemporal distribution. In terms of optimizing driver income fairness, the algorithm considers this aspect during the matching process, aiming to reduce the income disparity among drivers. The algorithm operates online, dynamically considering the supply and demand of drivers and orders. Additionally, by incorporating pruning and various acceleration mechanisms, the algorithm ensures efficient matching on large datasets, and the algorithm overview is shown in Figure 1.

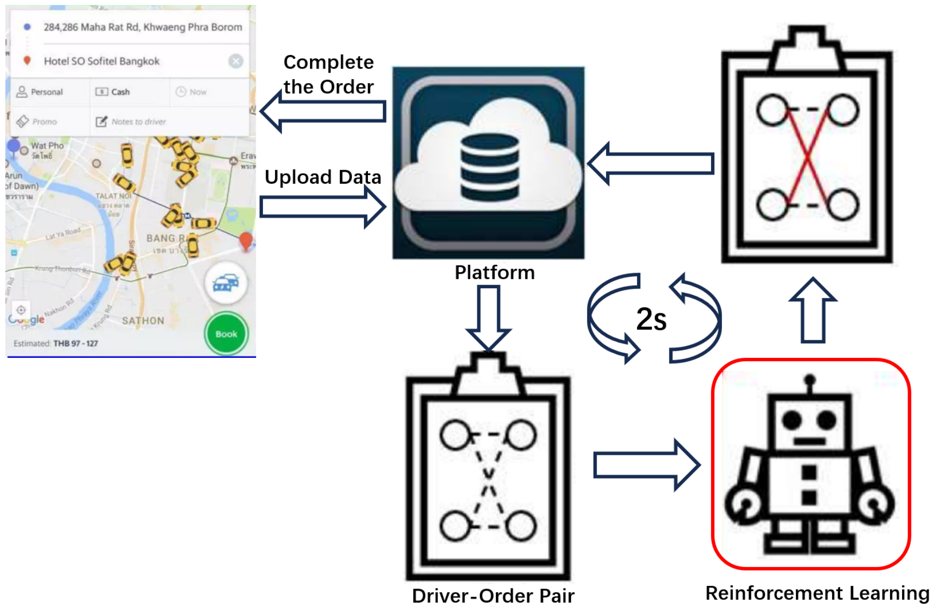

Figure 1. Algorithm overview.

The fairness allocation algorithm based on reinforcement learning consists of a value prediction module based on reinforcement learning and an efficient matching module for multiple objectives, detailed in Figure 2.

(1) Value prediction module based on reinforcement learning: This module adjusts the weights of edges in the bipartite graph to better reflect the two following optimization objectives: total platform efficiency and driver income fairness. In traditional solutions, edge weights are usually set as the earnings of individual orders to optimize platform efficiency. This module balances platform efficiency and driver income fairness by incorporating the impact of current matches on future spatiotemporal distribution and individual driver fairness metrics into the edge weight design. Here, total platform

efficiency is reflected by the income of individual orders, and driver income fairness is represented by a regional value function reflecting the current and future volume of orders in a region. The adjustment of edge weights is achieved through online reinforcement learning. Additionally, to alleviate the cold start issue in online learning, a driver guidance algorithm is designed;

(2) Efficient matching module for multiple objectives: This module performs matching on the bipartite graph after the adjustment of edge weights. The core of this module is the Fairness Augmentation Algorithm, which fundamentally implements the KM algorithm. To ensure fairness, the augmentation process includes checks for temporal earnings fairness, achieving bipartite graph matching that balances fairness and efficiency. Furthermore, to enhance algorithm efficiency, the module improves performance in large-scale, sparse graphs through special case determinations.

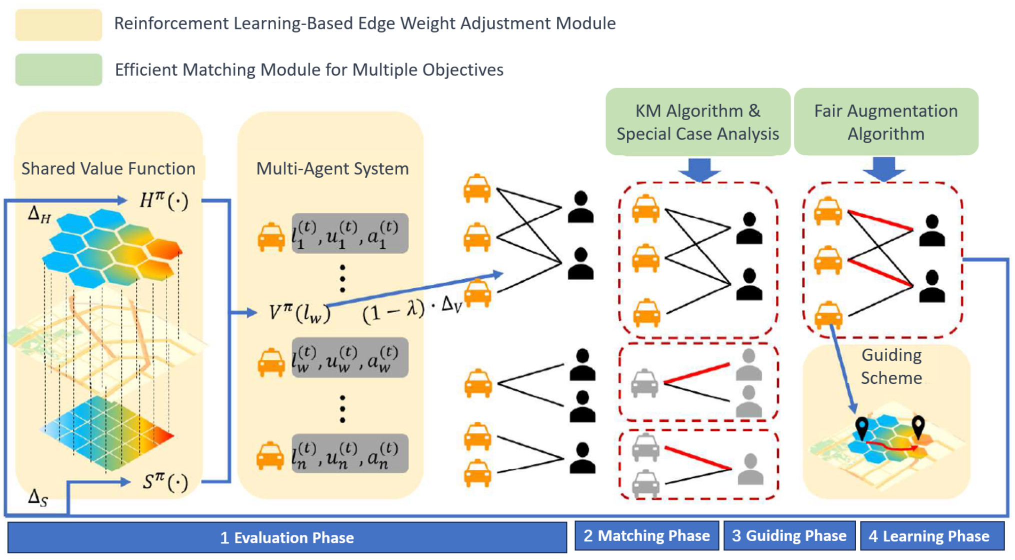
Reinforcement Learning-Based Edge Weight Adjustment Module
Efficient Matching Module for Multiple Objectives
Shared Value Function
Multi-Agent System
KM Algorithm & Special Case Analysis
Fair Augmentation Algorithm
$H^\pi(\cdot)$
Δ_H
$l_1^{(t)}, u_1^{(t)}, a_1^{(t)}$
$V^\pi(l_w)$
$(1-\lambda)\cdot\Delta_V$
$l_w^{(t)}, u_w^{(t)}, a_w^{(t)}$
$l_n^{(t)}, u_n^{(t)}, a_n^{(t)}$
$S^\pi(\cdot)$
Δ_S
Guiding Scheme
1 Evaluation Phase
2 Matching Phase
3 Guiding Phase
4 Learning Phase

Figure 2. Overview of value prediction module based on reinforcement learning.

4.2. Value Prediction Module Based on Reinforcement Learning

This subsection introduces the specific implementation mechanism of the reinforcement learning mechanism.

4.2.1. Edge Weight Adjustment Formula Based on Online Reinforcement Learning

In reinforcement learning, an agent continually interacts with the environment and updates the value function V_w^π based on the feedback to its actions from the environment. V_w^π represents the expected cumulative reward that agent w can obtain under the current state by taking the next action according to policy π. The optimal policy can be learned by constantly interacting with the environment and evaluating the feedback obtained from these interactions.

In the ride-hailing matching scenario, each driver is an agent. In each time window, the agent can perform one of the following two actions: serve the order r assigned by the platform or remain idle. The corresponding rewards for these choices are p_r and 0, respectively, representing the driver's expected income from making this choice. The state of driver w is represented as $(l_w^{(t)}, t)$, where $l_w^{(t)}$ is their location in time window t. To more accurately optimize the driver–task matching situation in each time window from both the perspective of fairness and total revenue, $V_w^\pi(l_w^{(t)}, t)$ represents the value function obtained by driver w making a choice under policy π in time window t.

As the optimization objective includes fairness, ensuring as small a difference as possible in each driver's income, some trade-offs must be made in total platform revenue. Therefore, optimizing for fairness to some extent contradicts the goal of maximizing total platform revenue. Under two conflicting optimization objectives, directly specifying policy π presents certain difficulties. Hence, we adopt an iterative approach to learning the value function with the policy, as demonstrated below.

$$V_w^{\pi}(l_w^{(t)}, t) \leftarrow V_w^{(t)}(l_w^{(t)}, t) + \beta \cdot \Delta_w (\forall w \in W) \tag{6}$$

where β represents the learning rate, and the calculation formula for Δ_w is as follows:

$$\Delta_w = \begin{cases} pr + V_w^{\pi}(d_r, t + \tau_r) - V_w^{\pi}(l_w, t), & \text{if } w \text{ gets } r \\ 0, & \text{if } w \text{ is idle} \end{cases} \tag{7}$$

The introduction of the reinforcement mechanism is aimed at better characterizing and predicting future scenarios. Therefore, whether the policy π is learned or not is not the primary concern of this paper. For convenience, after formulating the value function based on iterative updates, its superscript π is omitted. Additionally, the matching algorithm incorporating online reinforcement learning is also applicable to learning short-term dynamic data.

4.2.2. Reducing the Size of the State Space

For an agent, i.e., a driver, in a given time window, changes can be made in both spatial and service states. The spatial state is represented by N_s, and the time state by N_T. If a city is divided into square kilometers, and time is divided into the average service duration of an order, which is 20 min, the combined spatial and temporal states result in a state space of over 500,000 choices for each agent. Even if each agent works for 14 h a day—the maximum allowed by the platform—and makes one state change in each time window, the maximum number of states explored per day is only 250,000. The vastness of the state space, coupled with the limited capacity to explore it, makes it difficult for agents to effectively learn through reinforcement. Therefore, it is necessary to reduce the size of the state space. However, simple state discretization does not adequately address this issue. This paper adopts the following two methods to reduce the state space available to agents.

(1) Approximating the spatial value function

Previously, to eliminate the impact of individual driver service patterns on fairness metrics, drivers' earnings were divided by the average earnings of drivers in that hour. Compared to an hour, the average duration of each order is only twenty minutes, making the temporal factor's impact on the function's value negligible. Therefore, in calculating the value function for individual orders, only the spatial factor's impact on the function value is considered, i.e., $V_w(d_r, t) + \tau_r = V_w(d_r, t)$. Thus, the update function can be rewritten as follows:

$$\Delta_w = \begin{cases} (p_r + \gamma^{\tau_r} V_w(d_r)) - V_w(l_w), & \text{if } w \text{ gets } r \\ 0, & \text{if } w \text{ is idle} \end{cases} \tag{8}$$

where γ is the discount factor. When the duration of an order is longer, the value function of the destination has a smaller influence on the driver's decision to accept the order, effectively correcting the approximation error for long-distance requests.

(2) Information sharing among agents

In the ride-hailing matching scenario, all drivers' actions are aligned with the goal of optimizing their total earnings. Therefore, drivers in the same spatiotemporal state should have a converging assessment of the value of a certain location. Consequently, all agents adopt a shared value function, which effectively circumvents the limited exploration capabilities of individual agents and, to some extent, alleviates the challenges that a large state space poses to the convergence of the value function. By applying this strategy, the update equation can be simplified as follows:

$$V(l) \leftarrow V(l) + \beta' \cdot \sum_{w: l_w^{(t)} \in l} \Delta_w \tag{9}$$

where β' is the normalized learning rate, l is the region in which the driver is located, and the value function of region l is the sum of this expected original value function and the 'earnings' of all drivers in that region during this time window.

The contributions of the above two optimization strategies to reducing the size of the state space can be summarized as follows: (1) By applying an approximate spatial value function, the temporal factor's impact can be ignored, reducing the state space size from $N_t \cdot N_s$ to N_s. (2) Using information sharing among agents allows all drivers to collaboratively explore a single value function, thereby greatly reducing both the total number of states in the state space and the relative value of the state space exploration capability.

4.2.3. Algorithm Adaptability

In traditional ride-hailing matching applications, space is often discretized [40], limiting the portability of value function computation methods. To eliminate this impact, this paper smooths the value function over space with the following specific strategies: (1) discretizing space into two layers based on different division methods; (2) smoothing the value functions of different layers.

In this paper, the two layers of discretization are hexagonal and quadrilateral layers. The hexagonal layer is composed of hexagonal grids, while the quadrilateral layer is composed of quadrilateral grids. Based on the geometric characteristics of these two types of grids, different smoothing strategies are set. In the hexagonal layer, each grid has six symmetric sides and six completely equivalent adjacent grids. The value function is smoothed through the adjacent grid values around the central grid, making it more fitting for irregular urban layouts. In the quadrilateral layer, the latitude and longitude of the sides are identical, making it more suitable for exploring reasonable value functions in regular areas. In Figure 3, the value function of the hexagonal layer reflects the distribution of the city's main thoroughfares, and the value function of the quadrilateral layer reflects the radiation situation of the city center. Thus, by combining these two types of regional division, a location's value can be effectively and comprehensively characterized. Next, we will provide a detailed introduction to the geometric characteristics of hexagonal and quadrilateral layers.

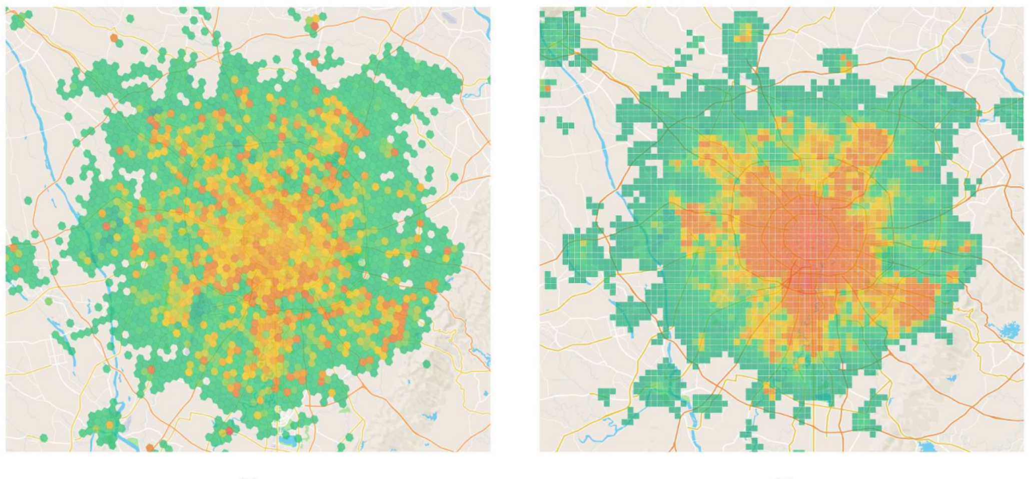

Figure 3. Value function distribution of hexagonal layer (**1**) and quadrilateral layer (**2**).

(1) Characteristics of hexagonal layers:

(a) **Uniformity and symmetry**: Due to their six-fold symmetry, hexagonal layers offer higher uniformity and symmetry compared to quadrilateral layers. This means that, in a hexagonal layer, each cell has a consistent distance to its neighboring cells, which helps reduce directional bias during spatial analysis [42];

(b) **Coverage efficiency**: Hexagons provide the smallest perimeter for the same area, which means they are more material efficient when covering large areas compared to quadrilateral layers. Therefore, in applications requiring equal area division, hexagonal layers can cover space more effectively, reducing edge effects;

(c) **Adjacency consistency**: Hexagonal layers have six direct neighbors. This consistency aids in more accurate spatial analysis, especially in simulating natural phenomena (such as fluid dynamics or population migration), providing a better reflection of actual conditions.

(2) Characteristics of quadrilateral layers:

(a) **Simplicity and compatibility**: Quadrilateral layers have a simple structure and are highly compatible with existing map projections and computer display systems, making them easy to implement and use, especially in standard GIS (Geographic Information System) software and computer graphics;

(b) **Computational efficiency**: In some computational processes, quadrilateral layers may offer higher computational efficiency due to their simple structure. For example, in raster analysis or image processing, quadrilateral layers can directly correspond to pixel layers, simplifying calculations;

(c) **Adaptability**: Quadrilateral layers can better adapt to irregular boundaries and linear structures, which is particularly useful in applications such as urban planning and architectural layout. Quadrilateral layers can be more easily resized and shaped to fit specific geographical features.

Finally, the smoothed value function is defined as follows:

$$V(l) = \frac{1}{|DIR_H| + |DIR_S|}\left(\sum_{x \in DIR_H} H(l+x) + \sum_{x \in DIR_S} S(l+x)\right) \tag{10}$$

where H and S respectively represent the value functions of the hexagonal and quadrilateral layers, and their update methods are as shown in Equation (10). DIR_H and DIR_S respectively indicate the smoothing directional offsets for the two layers, meaning that the value function of each location is the average value of the adjacent quadrilateral and hexagonal grid layers.

4.2.4. Avoiding the Cold Start Problem

Due to the characteristics of online reinforcement learning, the value function is initialized to 0 in the initial stage of online learning, causing the estimated edge weights to degenerate into initial weights (the earnings of individual orders). This problem is known as the cold start problem. The cold start issue results in the inability of the matching results to effectively incorporate predictions for future orders in the initial stage of matching.

To overcome the cold start issue, this paper guides drivers to appropriate locations in the initial stage of learning. Where a driver is guided depends on the distance and the increment of the value function. To simplify this issue, the hexagonal region value function is used as a guiding objective.

4.2.5. Algorithm Integration

The edge weight adjustment method based on reinforcement learning comprises three stages: evaluation, learning, and guidance.

The primary purpose of the evaluation stage is to analyze the current state of the environment and make reasoned estimates of the value of various grids. This stage uses historical data and real-time information (such as driver locations, passenger requests, and rejection rates) to assess the value of each area, thereby determining which areas are more attractive to drivers, as shown in Algorithm 1.

Suppose a city is divided into multiple hexagonal grids. During the evaluation stage, the algorithm identifies that a particular hexagonal grid area (Area A) experiences a significant increase in ride requests during the evening rush hour, while adjacent Area B has relatively lower demand. Based on this information, the algorithm adjusts the value score of Area A upwards, reflecting its high demand during specific times.

Algorithm 1: The evaluation stage of the value prediction module.

```
Input: Bipartite graph $G^{(t)} =< W^{(t)}, R^{(t),E^{(t)}} >$, driver rejection rate $\lambda$, hexagonal
       network layer value function $H$, quadrilateral network layer value function
       $S$.
Output: The bipartite graph after edge weight adjustment
        $G^{(t)} =< W^{(t)}, R^{(t)}, E^{(t)} >$.
$E'^{(t)} \leftarrow \emptyset$;
for $w, r, \theta_{w,r} \in E^{(t)}$ do
    Compute $V(l_w), V(d_r)$;
    $\theta_{w,r} \leftarrow (1 - \lambda_{w,r} \cdot (p_r + \gamma^{\tau_r} V(d_r) - V(l_w)))$;
    $E'^{(t)} \leftarrow E'^{(t)} \cup (w, r, \theta_{w,r})$;
end
Return the bipartite graph after edge weight adjustment $G^{(t)}$;
```

The learning stage of the value prediction module, based on actual matching results, updates the value functions of the quadrilateral and hexagonal network layers, and the learning stage updates the value functions based on the analysis of matching results, i.e., it learns and adjusts the value assessments of each area based on the actual effectiveness of matches. The goal of this stage is to ensure that the value prediction module can continuously adapt to changes in the environment, improving the accuracy of predictions and matching efficiency, as shown in Algorithm 2.

Continuing with the evaluation stage example: if the adjusted matching strategy leads to an increased rate of driver acceptance in Area B, the learning stage would update the value score for Area B based on this outcome. This indicates that increasing the attractiveness of Area B helps balance driver distribution and improve service coverage.

Algorithm 2: The learning stage of the value prediction module.

```
Input: Matching result $M^{(t)}$, order set $W^{(t)}$, hexagonal network layer value
       function $H$, quadrilateral network layer value function $S$.
Output: The updated hexagonal and quadrilateral network layer value functions
        $H, S$.
for $w \in W^{(t)}$ do
    if There is a request r that causes $(w, r) \in M^{(t)}$ then
        $\Delta_H \leftarrow p_r + \gamma^{\tau_r} H(d_r) - H(l_w)$;
        $\Delta_S \leftarrow p_r + \gamma^{\tau_r} S(d_r) - S(l_w)$;
    end
    else
        $\Delta_H \leftarrow 0$;
        $\Delta_S \leftarrow 0$;
    end
    $H \leftarrow H + \beta \cdot \Delta_H$;
    $S \leftarrow S + \beta \cdot \Delta_S$;
end
Return the updated hexagonal and quadrilateral network layer value functions
  $H, S$;
```

To avoid drivers in ‘cold zones’ waiting blindly for a long time, the guidance stage of the value prediction module directs idle drivers to locations where they are more likely to receive orders. Based on the actual value functions of each location and their proximity, the algorithm provides real-time recommendations to drivers, helping them more efficiently locate their next passenger. The guidance stage of the value prediction module is shown in Algorithm 3.

In the scenario evaluation and learning example, suppose the algorithm determines that Area C is expected to have high demand in the next few hours. The guidance stage would recommend drivers nearby to move towards Area C even if the current request volume in that area is not high. This predictive guidance helps preempt potential demand spikes, reducing passenger wait times.

Algorithm 3: The guidance stage of the value prediction module.

```
Input: Set of idle drivers $W^{(t)}$, optional guide to set of target locations $A_h$, $\xi$
       values in each batch from the beginning to the present.
Output: Guidance Scheme $\mathcal{S}$.
$\mathcal{S} \leftarrow \emptyset$;
Sort in ascending order $\frac{\sum_{i=1}^{t} u_w^{(i)} / \xi^{(i)}}{\sum_{i=1}^{t} a_w^{(i)}}$;
for $w \in W^{(t)}$ do
    $g \leftarrow \arg\max_{g' \in A_h} \frac{V(g) - V(l_w)}{dist(g', l_w)}$;
    $\mathcal{S} \leftarrow \mathcal{S} \cup (w, g)$;
end
Return guidance Scheme $\mathcal{S}$;
```

4.3. *Efficient Matching Module for Multiple Objectives*

This module specifically discusses how the matching algorithm optimizes for both total platform efficiency and driver income fairness.

For dual-objective optimization problems [43], the existing research often adopts linear programming algorithms [7] and reallocation algorithms. However, linear programming algorithms are often very inefficient due to the complexity of constraints and the diversity of optimization goals. Meanwhile, reallocation algorithms require different matching results for different optimization goals followed by adjustments and trade-offs between these results. Neither method is straightforward or efficient. This paper embeds fairness checks directly into the matching process for efficient optimization of both objectives.

The basic augmentation algorithm uses the KM algorithm [44] to maximize total platform efficiency. The core idea of the KM algorithm is to traverse each node in the graph and find an augmenting path that alternates between unmatched and matched edges. If the sum of the weights of unmatched edges is greater than that of the matched edges, the matched edges are turned into unmatched edges, and vice versa, to increase the total platform efficiency in the current match.

The core idea of the Fairness Augmentation Algorithm is to check the future earnings ratio of each driver when a replaceable augmenting path is found. If there is a significant difference in future earnings between drivers, this augmenting path is not used. However, refusing augmentation for the sake of fairness can somewhat reduce total platform revenue. Moreover, checking fairness for each augmentation adds some time, affecting the efficiency of the algorithm. To minimize the impact on total revenue and efficiency, only the fairness difference between two adjacent drivers on the augmenting path is checked. If it is within a certain range, the augmentation is accepted. Algorithm 4 describes the Fairness Augmentation Algorithm, which has a time complexity of $O(N^2 M)$, where $M = \min(|W^{(t)}|, |R^{(t)}|)^2$, $N = \max(|W^{(t)}|, |R^{(t)}|)$.

Algorithm 4: Fairness Augmentation Algorithm.

```
Input: Bipartite graph $G^{(t)} =< W^{(t)}, R^{(t)}, E^{(t)} >$, weighted amortization fairness
       $\xi$ from initial to present values in each batch.
Output: Matching Scheme $M^{(t)}$.
$M^{(t)} \leftarrow \emptyset$;
for $w \in W^{(t)}$ do
    while There is an augmenting road passing through $p$ do
        Accept ← True;
        for Each pair nearby driver $w_{pre}, w_{cur}$ do
            $r_{pre} \leftarrow$ current assigned request of $w_{pre}$;
            $r_{cur} \leftarrow$ current assigned request of $w_{cur}$;
        end
        if $\left|\frac{\sum_{i=1}^{t} \frac{u_{w_{pre}}^{(i)}}{\xi^{i}} + \frac{pr_{pre}}{\xi^{(t)}}}{\sum_{i=1}^{t} a_{w_{pre}}^{(i)} + \tau_{r_{pre}}} - \frac{\sum_{i-1}^{t} \frac{u_{w_{cur}}^{(i)}}{\xi^{(i)}} + \frac{pr_{cur}}{\xi^{(t)}}}{\sum_{i=1}^{t} a_{w_{cur}}^{(i)} + \tau_{r_{cur}}}\right| > \epsilon$ then
            Accept ← False;
            Break;
        end
        else
            Update matching $M^{(t)}$ based on augmented path $P$;
        end
    end
end
Return matching Scheme $M^{(t)}$;
```

5. Experiments

This section first introduces the specific experimental setup, including the hardware environment, datasets, and comparison algorithms. Next, a preliminary assessment of the experimental results is provided. Then, using real-world datasets, detailed analysis and interpretation of the algorithm's effectiveness are conducted. Finally, the fairness and effectiveness of the fairness-aware dynamic ride-hailing matching based on reinforcement learning (FMRL) algorithm are evaluated.

5.1. Experimental Setup

The order data for the experiment come from real data of the Didi ride-hailing platform from three major cities in Southwest, East, and Central China. Based on actual driver data, behaviors such as order rejection and logging on and off are simulated. To verify the portability of the proposed algorithm to different city data, experiments were conducted on real data from these three cities over a period of 21 days. To validate the effectiveness, fairness, and efficiency of the algorithm horizontally, four common algorithms in the ride-hailing matching scenario are used as benchmark algorithms.

The specific environment uses an Intel Xeon CPU E5-2630 v4 with 12 GB of RAM, running on Centos 7 OS, and the programming language is Python 3.8.

5.2. Comparison Algorithms

(1) Distance-Based Greedy Algorithm (DG): Based on the idea of greediness, this algorithm matches each request to the nearest driver. It is a simple and naive algorithm widely used in the industry;

(2) Earnings-Ratio-Based Greedy Algorithm (ERG): Within a time window, firstly, idle drivers are sorted in descending order based on weighted amortized fairness, and incoming platform orders are sorted in descending order of earnings ratio. Then, matching is performed according to the order of the two sorted lists, ensuring drivers with lower earnings ratios receive the more profitable orders as much as possible;

(3) Integer Linear Programming (ILP) algorithm [7]: This is the first work to use amortized fairness as an income fairness evaluation criterion. In this algorithm, the amortized fairness of drivers is taken as the optimization objective;

(4) Reassignment algorithm [8] (REA): First, two types of matching results are obtained based on total platform utility and driver fairness, then adjustments are made based on these two results, striking a balance between total platform utility and fairness. This is currently the only method that optimizes across both dimensions of total platform utility and fairness.

5.3. Algorithm Evaluation

This subsection evaluates the algorithm in three dimensions: fairness, platform utility, and efficiency. Definitions of fairness and platform utility have already been introduced in Section 3, and efficiency is assessed by the time consumed by the matching algorithm to process orders per hour. The specific parameters of the algorithm are shown in Table 2.

Table 2. Parameter settings.

Parameter	Meaning	Value
γ	Discount factor	0.9
β'	Learning rate	0.025

The experiment was conducted on real data from three major cities in Southwest, East, and Central China, labeled as cities A, B, and C. The overall experimental results are summarized in Table 3.

Table 3. The overall experimental results.

City	Algorithm	F (Work Days)	U (Work Days)	F (Weekends)	U (Weekends)
A	DG	41,767	2,242,977	31,962	2,131,996
	ERG	48,549	2,256,277	35,903	2,172,489
	ILP	44,072	2,246,077	30,573	2,122,834
	REA	44,251	2,250,118	27,865	2,134,534
	FMRL (ours)	22,656	2,656,773	13,384	2,565,060
B	DG	31,217	1,297,735	28,563	1,360,659
	ERG	38,594	1,373,478	37,464	1,452,488
	ILP	29,496	1,281,273	25,077	1,336,712
	REA	31,744	1,319,958	33,671	1,373,691
	FMRL (ours)	7420	1,479,348	4976	1,616,385
C	DG	12,897	872,693	9044	865,031
	ERG	13,704	885,558	8793	882,573
	ILP	12,377	862,588	8439	863,245
	REA	13,491	880,847	8684	881,519
	FMRL (ours)	2553	1,114,000	2392	1,109,474

Table 3 briefly describes the fairness and efficiency indicators of the algorithm on datasets from the three cities. Across the data from all three cities, our proposed algorithm showed an improvement of 45.5% to 81.4% in fairness indicators on weekdays and 52.0% to 86.7% on weekends compared to the baseline algorithms. For platform utility indicators, there was an improvement of 7.7% to 29.1% on weekdays and 11.3% to 28.5% on weekends, as shown in Table 4.

Table 4. Evaluation of improvement effect.

Improvement Rate	Weekdays	Weekends
Fairness (F)	45.7–81.4%	52.0–86.7%
Total platform utility (U)	7.7–29.1%	11.3–28.5%

Figure 4 visually present the utility function, amortized fairness, and temporal earnings fairness distribution for the five algorithms.

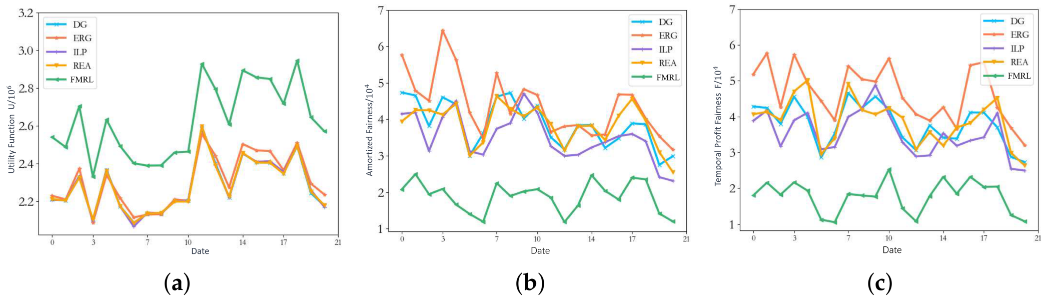

Figure 4. The results on utility function, amortization fairness, and temporal earnings fairness. (**a**) Utility; (**b**) amortization fairness; (**c**) temporal earnings fairness.

It is evident that the proposed FMRL outperforms the other four algorithms in terms of efficiency, effectively ensuring the income fairness of drivers.

In specific algorithm comparisons, compared to the Integer Linear Programming (ILP) algorithm, FMRL showed an average improvement of 68.4% in fairness and 22.1% in platform utility. Compared to the Reassignment algorithm (REA), it showed an average improvement of 69.3% in fairness and 20.1% in platform revenue. Overall, the proposed algorithm consistently outperformed the baseline algorithms on real order data from the three cities in terms of total platform revenue and fairness.

Figure 5 compares the efficiency of different algorithms in cities A, B, and C.

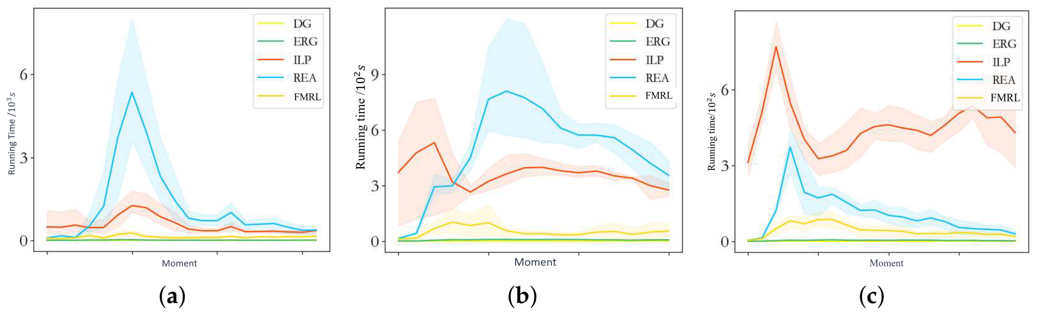

Figure 5. The efficiency of different algorithms in cities A, B, and C.(**a**) City A; (**b**) city B; (**c**) city C.

The Distance-Based Greedy Algorithm (DG) and the Earnings-Ratio-Based Greedy Algorithm (ERG), due to their simplistic algorithmic concepts, have time complexities of $O(|E|\log|E|)$ and $O(|W|\log|W|)$, respectively. As their problem-solving approach and mathematical modeling are consistent, these two algorithms have a similar runtime to our proposed algorithm, but their results are poorer in terms of fairness and efficiency metrics due to their single-objective greedy approach.

The Integer Linear Programming (ILP) algorithm and Reassignment algorithm (REA), due to their fundamentally different approaches and methods, can take up to eight times longer to run than the algorithm proposed in this paper. Additionally, changes in traffic conditions significantly impact the runtime of these two baseline algorithms. During peak hours, when orders flood in, the time consumption of these algorithms surges. Furthermore, the efficiency of the ILP algorithm is also constrained by the number of drivers, and the algorithm's efficiency does not significantly improve even when there are fewer orders.

5.4. Case Analysis

To scientifically elucidate the effectiveness of the algorithm, this subsection delves into an in-depth analysis of the algorithm's operating mechanism based on data from city A.

Figure 6 shows the adjustment mechanisms of different allocation algorithms for long-distance orders from city A's data. Figure 6(1) corresponds to the Distance-Based Greedy Algorithm, Figure 6(2) to the Earnings-Ratio-Based Greedy Algorithm, Figure 6(3) to the Integer Linear Programming algorithm, Figure 6(4) to the Reassignment algorithm, and Figure 6(5) to the Fairness Matching Reinforcement Learning (FMRL) algorithm, i.e., the FMRL algorithm proposed in this paper.

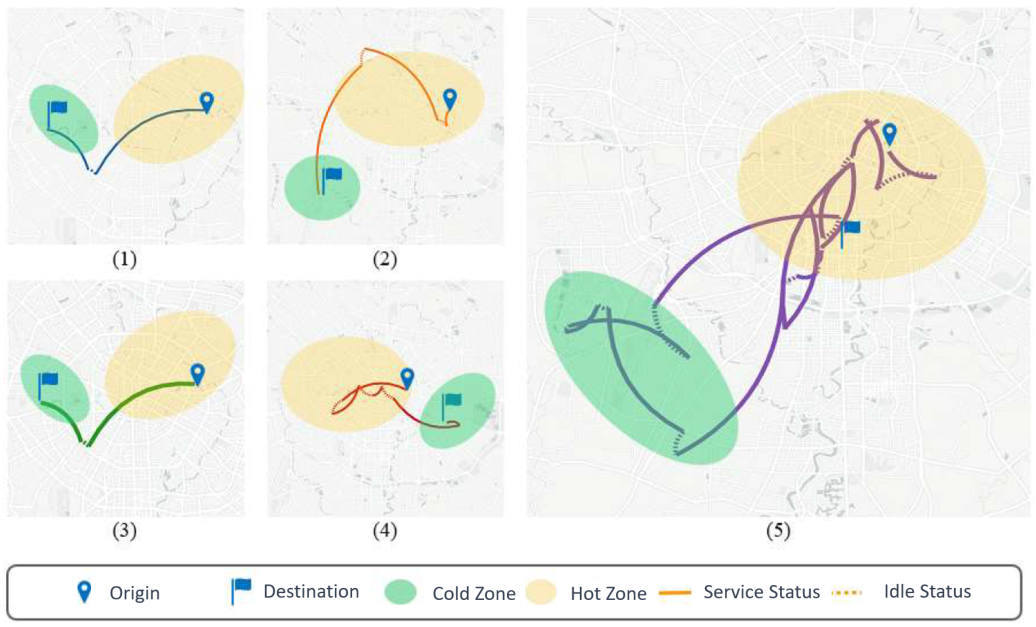

Figure 6. The trajectory of a driver under different algorithms. (**1**) Distance-Based Greedy Algorithm; (**2**) Earnings-Ratio-Based Greedy Algorithm; (**3**) Integer Linear Programming Algorithm; (**4**) Reassignment Algorithm; (**5**) FMRL Algprithm.

The Distance-Based Greedy Algorithm only matches drivers to the nearest orders, failing to effectively balance driver income and the prediction of future order arrivals. Once a match is made, even if more suitable orders arrive later, the driver cannot be matched to them. This approach does not effectively ensure driver income fairness.

The Earnings-Ratio-Based Greedy Algorithm only considers the highest earnings ratio of previous orders received by a driver when matching. Drivers serving the highest-earning orders are later matched to orders with lower earnings ratios. It does not take the distance between orders and drivers into account, resulting in lost platform revenue. It also fails to effectively assess future earnings, harming driver income fairness.

The Integer Linear Programming algorithm reduces the priority of a driver to receive orders after serving a long-distance order as their earnings ratio becomes very high.

The Reassignment algorithm strikes a balance between fairness and platform efficiency. Experimental results show that it can improve platform efficiency by increasing order matching rates. However, it still fails to consider situations where drivers in remote areas cannot receive orders; thus, it does not effectively ensure driver income fairness.

The Fairness Matching Reinforcement Learning (FMRL) algorithm takes into account the impact of current allocations on future spatiotemporal distribution and fairness. In addition, the guidance algorithm for drivers effectively avoids leaving drivers stranded in remote areas. As shown in the figure, when a driver completes several orders in the central area and their earnings ratio is high, the platform tends to assign them a long-distance order. Afterwards, to prevent drivers in remote areas from being unable to receive orders for extended periods, the platform guides them to areas where they can receive orders, helping them return to central areas.

5.5. Algorithm Fairness Evaluation

One of the core contributions of this paper is the incorporation of weighted amortized fairness. Figure 7a shows the daily changes in the number of drivers, the number of requests, and the weighted amortized fairness $\xi^{-1}(t)$ (the reciprocal of the median income per unit time of drivers).

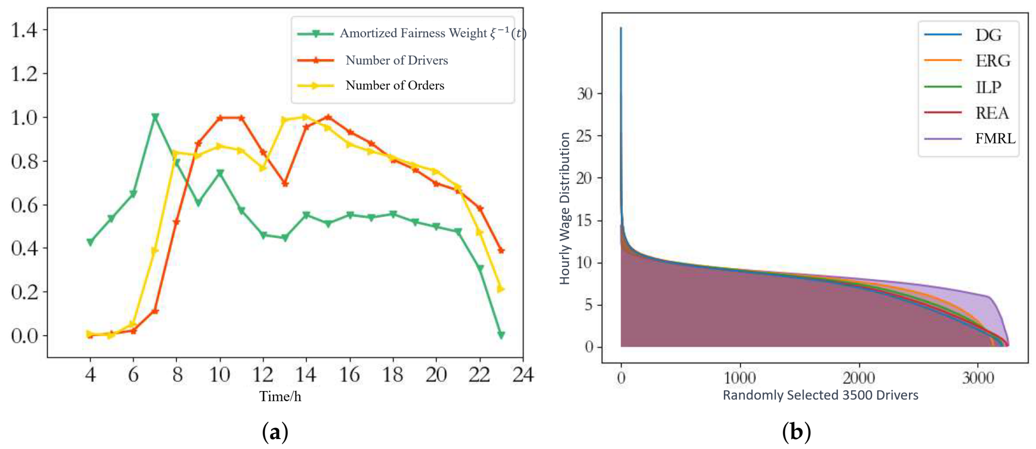

Figure 7. (**a**) The daily changes in drivers, requests, and weighted amortized fairness. (**b**) The hourly wages of 3500 randomly selected drivers.

From the number of drivers and orders, we can infer the traffic situation for each time period. There is a surge in the volume of driver orders from 6 AM to 8 AM, reaching a peak at noon and then gradually declining. The variation in the weighted amortized fairness $\xi^{-1}(t)$ generally follows this pattern. Temporally, its trend slightly precedes the volume of driver orders, thereby effectively predicting future traffic conditions.

Figure 7b shows the hourly wages of 3500 randomly selected drivers. It is evident from the graph that the FMRL algorithm effectively ensures the fairness of driver income. Compared to other algorithms, the distribution by the algorithm proposed in this paper averages the income of the 3500 drivers, reducing the number of drivers with excessively high and low incomes.

5.6. Algorithm Effectiveness Evaluation

To holistically assess the effectiveness of the algorithm, the four baseline algorithms and the proposed algorithm are evaluated on weekdays and weekends across the dimensions of amortized fairness and total platform revenue. Table 5 shows that, even though the Integer Linear Programming algorithm and the Reassignment algorithm among the benchmarks consider the fairness indicator, the FMRL algorithm still achieves an improvement of 57.9% to 62.2% in fairness indicators and 17.3% to 17.6% in total platform efficiency compared to the other four benchmark algorithms.

When evaluating the performance of the five algorithms on weekdays and weekends, the proposed algorithm showed stable performance and consistent improvement, performing even better on weekend data. In conclusion, the FMRL algorithm outperforms existing algorithms in terms of total platform efficiency, driver fairness, and algorithmic efficiency.

Table 5. Algorithm effectiveness evaluation.

City	Algorithm	F (Work Days)	U (Work Days)	F (Weekends)	U (Weekends)
A	DG	44,442	2,242,977	33,057	2,131,996
	ERG	54,270	2,248,669	38,315	2,166,980
	ILP	40,066	2,243,334	30,772	2,120,947
	REA	42,003	2,242,561	31,832	2,130,431
	FMRL (ours)	20,539	2,359,431	12,963	2,447,596

6. Conclusions and Future Work

This section first summarizes our work presented in this paper and then reflects on and anticipates future works.

6.1. Conclusions

This paper has designed a fairness-oriented dynamic matching algorithm for ride-hailing services, effectively optimizing both the total platform efficiency (expected total income of drivers) and driver income fairness (weighted amortized fairness entropy among drivers). The core contributions of this work are as follows:

(1) Considering future matching scenarios : To overcome the shortsightedness of existing research algorithms and to optimize the fairness allocation algorithm globally, future potential spatiotemporal distributions of drivers and orders are reasonably considered when making matches. Additionally, a guidance algorithm is designed to direct drivers to areas where they are likely to receive future orders, preventing the issue of drivers waiting a long time without orders;

(2) Designing a reinforcement-learning-based fair allocation algorithm: The value function at each location incorporates the matching results of previous orders, thus introducing the value function of the start and end points of orders into the edge weight. This approach allows the driver–order edge weight to reflect future earnings scenarios. Moreover, the value functions of locations are determined jointly by quadrilateral and hexagonal layers, effectively reflecting the distribution of the city center and main thoroughfares. This calculation method performs well in cities of various sizes, making the algorithm highly portable;

(3) Validation on real datasets: The algorithm's effectiveness, fairness, and efficiency were validated through experiments on real datasets. Unlike previous studies where drivers are generated independently at each moment, this experiment considered the impact of the matching algorithm on the real distribution of drivers, maintaining a fixed group of drivers and simulating their behaviors of logging on and off and accepting and rejecting orders. Thus, the data were more closely aligned with real situations, making the experimental results more credible.

6.2. Future Work

There are several issues to be explored and expanded upon in future research.

Firstly, regarding fairness calculation indicators, currently, only the income fairness of the driver community is considered. Issues like preferential treatment of certain customers by platforms such as Ele.me highlight the need for ensuring fairness among customer groups as well. In the future, the connotation and definition of passenger fairness indicators can be explored, and constraints for this type of fairness can be introduced into the algorithm.

Secondly, the existing definitions of fairness have certain limitations and are only applicable to specific groups for specific problems. Therefore, based on this, more generalized definitions and mechanisms for fairness can be explored.

Finally, simulators that mimic real traffic conditions are a current research hotspot. In the future, designing simulators that more closely resemble real situations can be explored, for instance, maintaining a fixed group of passengers, protecting current data as much as possible in the generation process, and ensuring greater autonomy. Additionally, the efficiency of simulators should be considered, requiring cross-platform integration and parallel operations.

Funding: This work was supported in part by the Guangxi Key Laboratory of Trusted Software (no. KX202037), the Project of Guangxi Science and Technology (no. GuiKeAD 20297054) and the Guangxi Natural Science Foundation Project (no. 2020GXNSFBA297108).

Data Availability Statement: We conduct experiments on a simulator developed by a major ride-hailing platform. The data comes from within the simulator, provided by the relevant cities for ride-hailing data. The simulator provides an interface for inputting algorithms, with the link being https://sumo.dlr.de/docs/index.html#simulation (accessed on 1 February 2024). Named SUMO (Simulation of Urban MObility), it is an open-source, highly configurable, multimodal traffic simulation software frequently used to simulate urban traffic flows.

Conflicts of Interest: The author declares no conflict of interest.

References

1. Charnes, A.; Cooper, W.W.; Stedry, A. Static and Dynamic Assignment Models with Multiple Objectives, and Some Remarks on Organization Design. *Manag. Sci.* **1969**, *15*, 365. [CrossRef]
2. Emek, Y.; Kutten, S.; Wattenhofer, R. Online matching: Haste makes waste! In Proceedings of the Forty-Eighth Annual ACM Symposium on Theory of Computing (STOC'16), Cambridge, MA, USA, 19–21 June 2016; pp. 333–344.
3. Bernstein, A.; Dudeja, A.; Langley, Z. A framework for dynamic matching in weighted graphs. In Proceedings of the 53rd Annual ACM SIGACT Symposium on Theory of Computing (STOC'21), Virtual, 21–25 June 2021; Khuller, S., Williams, V.V., Eds.; Association for Computing Machinery: New York, NY, USA, 2021; pp. 668–681.
4. Tong, Y.; She, J.; Ding, B.; Chen, L.; Wo, T.; Xu, K. Online Minimum Matching in Real-Time Spatial Data: Experiments and Analysis. *Proc. VLDB Endow.* **2016**, *9*, 1053–1064. [CrossRef]
5. Wang, Y.; Tong, Y.; Long, C.; Xu, P.; Xu, K.; Lv, W. Adaptive Dynamic Bipartite Graph Matching: A Reinforcement Learning Approach. In Proceedings of the 2019 IEEE 35th International Conference on Data Engineering (ICDE), Macao, China, 8–11 April 2019; pp. 1478–1489.
6. Yang, S.; Wen, J.; Eckert, S.T.; Wang, Y.; Liu, D.J.; Wu, R.; Li, R.; Zhan, X. Prioritizing genetic variants in GWAS with lasso using permutation-assisted tuning. *Bioinformatics* **2020**, *36*, 3811–3817. [CrossRef] [PubMed]
7. Sühr, T.; Biega, A.J.; Zehlike, M.; Gummadi, K.P.; Chakraborty, A. Two-Sided Fairness for Repeated Matchings in Two-Sided Markets: A Case Study of a Ride-Hailing Platform. In Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining (KDD'19), Anchorage, AK, USA, 4–8 August 2019; pp. 3082–3092.
8. Lesmana, N.S.; Zhang, X.; Bei, X. Balancing Efficiency and Fairness in On-Demand Ridesourcing. In Proceedings of the 33rd International Conference on Neural Information Processing Systems (NIPS'19), Vancouver, BC, Canada, 8–14 December 2019; pp. 5310–5320.
9. Spivey, M.Z.; Powell, W.B. The Dynamic Assignment Problem. *Transp. Sci.* **2004**, *38*, 399–419. [CrossRef]
10. Li, Y.; Yiu, M.L.; Xu, W. Oriented Online Route Recommendation for Spatial Crowdsourcing Task Workers. In *Advances in Spatial and Temporal Databases—SSTD 2015*; Springer: Cham, Switzerland, 2015; Volume 9239, pp. 137–156.
11. Wu, B.; Han, K.; Zhang, E. On the task assignment with group fairness for spatial crowdsourcing. *Inf. Process. Manag.* **2023**, *60*, 103175. [CrossRef]
12. Kumar, A.; Vorobeychik, Y.; Yeoh, W. Using Simple Incentives to Improve Two-Sided Fairness in Ridesharing Systems. In Proceedings of the 33rd International Conference on Automated Planning and Scheduling (ICAPS), Prague, Czech Republic, 8–13 July 2023; Koenig, S., Stern, R., Vallati, M., Eds.; AAAI Press: Palo Alto, CA, USA, 2023; pp. 227–235.
13. Wen, J.; Yang, S.; Wang, C.D.; Jiang, Y.; Li, R. Feature-splitting algorithms for ultrahigh dimensional quantile regression. *J. Econom.* **2023**, *in press*. [CrossRef]
14. Zhang, J.; Jiang, T.; Gao, X.; Chen, G. An Online Fairness-Aware Task Planning Approach for Spatial Crowdsourcing. *IEEE Trans. Mob. Comput.* **2024**, *23*, 150–163. [CrossRef]
15. Wang, Y.; Li, Y.; Bonchi, F.; Wang, Y. Balancing Utility and Fairness in Submodular Maximization. In Proceedings of the 27th International Conference on Extending Database Technology (EDBT), Paestum, Italy, 25–28 March 2024; pp. 1–14.
16. Guo, X.; Xu, H.; Zhuang, D.; Zheng, Y.; Zhao, J. Fairness-Enhancing Vehicle Rebalancing in the Ride-hailing System. *arXiv* **2024**, arXiv:2401.00093.
17. Jalota, D.; Solovey, K.; Tsao, M.; Zoepf, S.; Pavone, M. Balancing fairness and efficiency in traffic routing via interpolated traffic assignment. *Auton. Agents Multi-Agent Syst.* **2023**, *37*, 32. [CrossRef]
18. Tang, X.; Qin, Z.T.; Zhang, F.; Wang, Z.; Xu, Z.; Ma, Y.; Zhu, H.; Ye, J. A Deep Value-network Based Approach for Multi-Driver Order Dispatching. In Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining (KDD'19), Anchorage, AK, USA, 4–8 August 2019; pp. 1780–1790.

19. Garg, N.; Ranu, S. Route Recommendations for Idle Taxi Drivers: Find Me the Shortest Route to a Customer! In Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining (KDD'18), London, UK, 19–23 August 2018; pp. 1425–1434.
20. Wang, X.; Agatz, N.A.H.; Erera, A.L. Stable Matching for Dynamic Ride-Sharing Systems. *Transp. Sci.* **2018**, *52*, 850–867. [CrossRef]
21. Xu, P.; Shi, Y.; Cheng, H.; Dickerson, J.P.; Sankararaman, K.A.; Srinivasan, A.; Tong, Y.; Tsepenekas, L. A Unified Approach to Online Matching with Conflict-Aware Constraints. In Proceedings of the Thirty-Third AAAI Conference on Artificial Intelligence, Honolulu, HI, USA, 27 January–1 February 2019; pp. 2221–2228.
22. Song, X.; Wang, E.; Liu, W.; Liu, Y.; Dong, Y. Fairness task assignment strategy with distance constraint in Mobile CrowdSensing. *CCF Trans. Pervasive Comput. Interact.* **2023**, *5*, 184–205. [CrossRef]
23. Yang, S.; Wen, J.; Zhan, X.; Kifer, D. ET-lasso: A new efficient tuning of lasso-type regularization for high-dimensional data. In Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining (KDD'19), Anchorage, AK, USA, 4–8 August 2019; pp. 607–616.
24. Nanda, V.; Xu, P.; Sankararaman, K.A.; Dickerson, J.P.; Srinivasan, A. Balancing the Tradeoff between Profit and Fairness in Rideshare Platforms during High-Demand Hours. In Proceedings of the Thirty-Fourth AAAI Conference on Artificial Intelligence, New York, NY, USA, 7–12 February 2020; pp. 2210–2217.
25. Dickerson, J.P.; Sankararaman, K.A.; Srinivasan, A.; Xu, P. Assigning Tasks to Workers based on Historical Data: Online Task Assignment with Two-sided Arrivals. In Proceedings of the 17th International Conference on Autonomous Agents and Multiagent Systems (AAMAS), Stockholm, Sweden, 10–15 July 2018; pp. 318–326.
26. Dickerson, J.P.; Sankararaman, K.A.; Srinivasan, A.; Xu, P. Allocation Problems in Ride-sharing Platforms: Online Matching with Offline Reusable Resources. *ACM Trans. Econ. Comput.* **2021**, *9*, 13:1–13:17. [CrossRef]
27. Chen, Z.; Cheng, P.; Chen, L.; Lin, X.; Shahabi, C. Fair Task Assignment in Spatial Crowdsourcing. *Proc. VLDB Endow.* **2020**, *13*, 2479–2492. [CrossRef]
28. Chakraborty, S.; Calo, S.B.; Wen, J. Using Disentangled Learning to Train an Interpretable Deep Learning Model. U.S. Patent 2022/0198266 A1, 23 June 2022.
29. Hassan, U.U. Adaptive Task Assignment in Spatial Crowdsourcing. Ph.D. Thesis, University of Galway, Galway, Ireland, 2016.
30. Chen, H.; Jiao, Y.; Qin, Z.T.; Tang, X.; Li, H.; An, B.; Zhu, H.; Ye, J. InBEDE: Integrating Contextual Bandit with TD Learning for Joint Pricing and Dispatch of Ride-Hailing Platforms. In Proceedings of the 2019 IEEE International Conference on Data Mining (ICDM), Beijing, China, 8–11 November 2019; pp. 61–70.
31. Wang, Z.; Qin, Z.T.; Tang, X.; Ye, J.; Zhu, H. Deep Reinforcement Learning with Knowledge Transfer for Online Rides Order Dispatching. In Proceedings of the 2018 IEEE International Conference on Data Mining (ICDM), Singapore, 17–20 November 2018; pp. 617–626.
32. Wang, Y.; Wu, J.; Hua, X.; Liu, C.H.; Li, G.; Zhao, J.; Yuan, Y.; Wang, G. Air-Ground Spatial Crowdsourcing with UAV Carriers by Geometric Graph Convolutional Multi-Agent Deep Reinforcement Learning. In Proceedings of the 2023 IEEE 39th International Conference on Data Engineering (ICDE), Anaheim, CA, USA, 3–7 April 2023; pp. 1790–1802.
33. Wang, Y.; Liu, C.H.; Piao, C.; Yuan, Y.; Han, R.; Wang, G.; Tang, J. Human-Drone Collaborative Spatial Crowdsourcing by Memory-Augmented and Distributed Multi-Agent Deep Reinforcement Learning. In Proceedings of the 2022 IEEE 38th International Conference on Data Engineering (ICDE), Kuala Lumpur, Malaysia, 9–12 May 2022; pp. 459–471.
34. Lin, K.; Zhao, R.; Xu, Z.; Zhou, J. Efficient Large-Scale Fleet Management via Multi-Agent Deep Reinforcement Learning. In Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining (KDD'18), London, UK, 19–23 August 2018; pp. 1774–1783.
35. Cai, Z.; Li, C.; Wen, J.; Yang, S. Asset splitting algorithm for ultrahigh dimensional portfolio selection and its theoretical property. *J. Econom.* 2022, *in press*. [CrossRef]
36. Hong, H.; Lin, Y.; Yang, X.; Li, Z.; Fu, K.; Wang, Z.; Qie, X.; Ye, J. HetETA: Heterogeneous Information Network Embedding for Estimating Time of Arrival. In Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining (KDD'20), Virtual, 6–10 July 2020; pp. 2444–2454.
37. Fu, K.; Meng, F.; Ye, J.; Wang, Z. CompactETA: A Fast Inference System for Travel Time Prediction. In Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining (KDD'20), Virtual, 6–10 July 2020; pp. 3337–3345.
38. Yuan, H.; Li, G.; Bao, Z.; Feng, L. Effective Travel Time Estimation: When Historical Trajectories over Road Networks Matter. In Proceedings of the 2020 ACM SIGMOD International Conference on Management of Data (SIGMOD'20), Portland, OR, USA, 14–19 June 2020; pp. 2135–2149.
39. Zhang, L.; Hu, T.; Min, Y.; Wu, G.; Zhang, J.; Feng, P.; Gong, P.; Ye, J. A Taxi Order Dispatch Model based On Combinatorial Optimization. In Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD'17), Halifax, NS, Canada, 13–17 August 2017; pp. 2151–2159.
40. Xu, Z.; Li, Z.; Guan, Q.; Zhang, D.; Li, Q.; Nan, J.; Liu, C.; Bian, W.; Ye, J. Large-Scale Order Dispatch in On-Demand Ride-Hailing Platforms: A Learning and Planning Approach. In Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining (KDD'18), London, UK, 19–23 August 2018; pp. 905–913.

41. Robinson, H.C. Making a Digital Working Class: Uber Drivers in Boston, 2016–2017. Ph.D. Thesis, Massachusetts Institute of Technology, Cambridge, MA, USA, 2017; pp. 19–80.
42. Li, C.; Li, R.; Wen, J.; Yang, S.; Zhan, X. Regularized linear programming discriminant rule with folded concave penalty for ultrahigh-dimensional data. *J. Comput. Graph. Stat.* **2023**, *32*, 1074–1082. [CrossRef]
43. Chen, W.; Hu, W.; Li, F.; Li, J.; Liu, Y.; Lu, P. Combinatorial Multi-Armed Bandit with General Reward Functions. In Proceedings of the Thirtieth Annual Conference on Neural Information Processing Systems (NIPS), Barcelona, Spain, 5–10 December 2016; pp. 1651–1659.
44. Edmonds, J.; Karp, R.M. Theoretical Improvements in Algorithmic Efficiency for Network Flow Problems. *J. ACM* **1972**, *19*, 248–264. [CrossRef]