Timing the Match: A Deep Reinforcement Learning Approach for Ride-Hailing and Ride-Pooling Services

Yiman Bao, Jie Gao*, Jinke He, Frans A. Oliehoek, Oded Cats

Abstract-Efficient timing in ride-matching is crucial for improving the performance of ride-hailing and ride-pooling services, as it determines the number of drivers and passengers considered in each matching process. Traditional batched matching methods often use fixed time intervals to accumulate ride requests before assigning matches. While this approach increases the number of available drivers and passengers for matching, it fails to adapt to real-time supply-demand fluctuations, often leading to longer passenger wait times and driver idle periods. To address this limitation, we propose an adaptive ride-matching strategy using deep reinforcement learning (RL) to dynamically determine when to perform matches based on real-time system conditions. Unlike fixed-interval approaches, our method continuously evaluates system states and executes matching at moments that minimize total passenger wait time. Additionally, we incorporate a potentialbased reward shaping (PBRS) mechanism to mitigate sparse rewards, accelerating RL training and improving decision quality. Extensive empirical evaluations using a realistic simulator trained on real-world data demonstrate that our approach outperforms fixed-interval matching strategies, significantly reducing passenger waiting times and detour delays, thereby enhancing the overall efficiency of ride-hailing and ride-pooling systems.

Index Terms—Deep reinforcement learning, ride-hailing, ride-pooling, matching optimization, matching time, proximal policy optimization.

I. INTRODUCTION

R Ide-hailing services, such as Uber¹ and Lyft ², have become integral to urban transportation, offering passengers flexible, on-demand mobility while optimizing vehicle dispatch to alleviate congestion and reduce emissions [1, 2]. As an extension of ride-hailing, ride-pooling enables multiple passengers with similar routes to share a single vehicle, improving vehicle utilization and lowering travel costs, contributing further to sustainable urban mobility [3, 4]. However, as these services expand, ensuring efficient matching in the face of dynamic supply and demand remains a fundamental challenge.

Yiman Bao, Delft University of Technology, Delft, The Netherlands, the department of Transport & Planning (e-mail: baoyiman5@gmail.com).

Jie Gao, corresponding author, Delft University of Technology, Delft, The Netherlands, the department of Transport & Planning (e-mail: J.Gao-1@tudelft.nl).

Jinke He, Delft University of Technology, Delft, The Netherlands, the department of Intelligent Systems (e-mail: J.He-4@tudelft.nl).

Frans A. Oliehoek, Delft University of Technology, Delft, The Netherlands, the department of Intelligent Systems (e-mail: F.A.Oliehoek@tudelft.nl).

Oded Cats, Delft University of Technology, Delft, The Netherlands, the department of Transport & Planning (e-mail: o.cats@tudelft.nl).

To improve matching efficiency, ride-hailing platforms typically employ two primary strategies. One approach is first dispatch, where requests are assigned as they arrive based on available drivers. While this minimizes response time, it often leads to suboptimal matches, as better driver-passenger pairings could emerge if requests were aggregated over time. To address this, platforms such as Uber adopt batched matching, where ride requests and available drivers are accumulated over a fixed time interval before matches are assigned [5]. By increasing the pool of potential matches, this method improves assignment quality and enhances ride-pooling efficiency. However, fixed-interval batching does not account for real-time fluctuations in supply and demand. Performing matches at predetermined intervals may not coincide with the optimal matching moment, where additional waiting no longer improves the quality of matches. As a result, fixed batching strategies can lead to unnecessary passenger delays or missed opportunities for more efficient assignments.

The dynamic and stochastic nature of ride-hailing environments makes it challenging to determine optimal matching timing using fixed-rule strategies. Instead, an adaptive approach that learns to identify these moments based on realtime system conditions is needed. Reinforcement Learning (RL), a framework for sequential decision-making under uncertainty, is well-suited for this problem. By modeling ridematching as a Markov Decision Process (MDP) [6], RL can continuously evaluate system states and dynamically adjust matching timing to balance efficiency and responsiveness. In this work, we propose an RL-based adaptive matching strategy that determines when to execute matches based on realtime system conditions. Unlike fixed-interval batching, our approach continuously monitors supply-demand variations and optimizes timing to minimize passenger wait times. To address sparse reward challenges and accelerate learning, we introduce a potential-based reward shaping (PBRS) mechanism that improves RL training efficiency. Furthermore, we develop a realistic ride-matching simulator trained on real-world data, enabling comprehensive evaluation of our approach against existing first dispatch and batched matching strategies. The main contributions of this work are as follows:

 We formulate the matching process of ride-hailing and ride-pooling as a Partially Observed MDP (POMDP). At each time step, the match-maker determines whether the current moment is optimal for matching and decides whether to execute the matching. Once a match is con-

¹https://www.uber.com/nl/en/

²https://www.lyft.com/

firmed, the match-maker performs the optimal matching between idle drivers and unserved passengers in the accumulated pool.

- 2) We design a RL framework based on the formulated POMDP, including actions, states, and rewards. To address the issue of sparse rewards, we introduce Potential Based Reward Shaping (PBRS), which proofs that rewards are appropriately allocated to each action.
- 3) We develop an efficient and realistic simulator to model the matching process of ride-hailing and ride-pooling. This simulator, based on real-world data, includes functionalities such as data generation and spatial matching. We conduct a series of training and validation experiments using this simulator.
- 4) We employ Proximal Policy Optimization (PPO) as the RL method and investigate the impact of applying PBRS on the training outcomes. Additionally, we explore how the decision-making effectiveness of the match-maker evolves across different training phases.
- 5) We compare the strategies trained by RL with the first dispatch strategy and batched matching strategies at different time intervals. Our approach demonstrates superior performance compared to the latter two. Furthermore, we analyze the adaptability of our approach, revealing that it flexibly makes different action decisions in response to varying supply and demand conditions.

The remainder of the paper is structured as follows: Section 2 reviews the related literature and summarizes the research gap. Section 3 introduces the proposed approach for optimizing the timing for batched matching. Section 4 introduces the designed simulator. Section 5 conducts extensive numerical experiments on a real-world data. Finally, conclusions are given in Section 6.

II. LITERATURE REVIEW

A. Matching in Ride-hailing Systems

In ride-hailing systems, matching strategies are designed to efficiently allocate available drivers to passenger requests, ensuring timely pickups and maximizing system-wide efficiency. The matching process is typically formulated as an optimization problem, where assignments are made to maximize an objective function such as minimizing passenger wait times or maximizing driver utilization. Yan et al. [7] modeled this problem as a bipartite graph, where nodes represent drivers and passengers, and edges indicate potential matches, weighted by their quality. The optimization objective is to maximize the total weighted matchings. In some studies, the ride-hailing matching process is also modeled as a game among passengers, drivers, and the platform, where each party seeks a matching outcome that maximizes its own benefit. Li et al. [8] formulate this problem as a Stackelberg game, in which the ride-hailing platform acts as the leader, setting prices to ensure that none of the three parties can achieve a more profitable allocation, thereby determining an outcome that maximizes their respective profits.

Beyond static optimization formulations, dynamic modeling approaches have been proposed to account for real-time

decision-making in ride-hailing systems. Beojone et al. [9] developed a mixed continuous-discrete time Markov Chain model to predict driver earnings and mobility patterns, enabling more informed dispatching. Similarly, Zhang et al. [10] formulated idle driver routing as a Markov Decision Process (MDP), improving vehicle utilization by optimizing movement strategies before new requests arrive. These studies highlight the importance of incorporating dynamic system states into ride-matching models, providing valuable insights for real-time decision-making.

Another important factor influencing ride-hailing efficiency is driver heterogeneity, which affects individual responses to ride requests. Do et al. [11] analyzed factors leading to driver request rejections, while Liu et al. [12] applied inverse reinforcement learning to detect anomalous driver behavior and improve matching success rates. Additionally, Shi et al. [13] observed that part-time drivers exhibit lower patience levels than full-time drivers, making them more sensitive to wait times. To address this, they proposed a priority-based matching policy (PMP) that prioritizes part-time drivers to reduce abandonment rates. With the growing adoption of electric vehicles (EVs), ride-hailing matching must consider EVspecific constraints such as range limits and charging delays. Li et al. [14] proposed a stochastic optimization framework that integrates vehicle repositioning and matching. The framework proactively guides idle EVs based on demand forecasts and optimizes matching to minimize passenger wait times and charging-related costs. Furthermore, some studies have explored the integration of driver rebalancing with matching strategies. Guo et al. [15] introduced the Matching-Integrated Vehicle Rebalancing (MIVR) model, which simultaneously considers passenger-driver assignments and vehicle repositioning to enhance fleet efficiency.

In addition to spatial assignment strategies, matching quality is influenced by both the matching radius and timing. Yang et al. [16] identified these two parameters as key determinants of system performance. A shorter matching radius reduces pickup distances but may lower the overall matching rate, whereas a longer matching interval allows for the accumulation of more unserved passengers and idle drivers, potentially improving assignment quality at the cost of increased passenger wait times. To optimize these trade-offs, various approaches have been proposed. Chen et al. [17] developed a Deep Learning-based Matching Radius Decision (DL-MRD) model that predicts key system performance metrics across different matching radii, enabling the ride-hailing platform to select an optimal radius based on real-time supply and demand conditions. To account for the temporal impact on matching outcomes, Shi et al. [18] consider how current matching and dispatch decisions influence future ride-hailing supply and demand, leading to variations in vehicle value across different regions and time periods. They introduce a vehicle value function to quantify the spatiotemporal value of vehicles in each region and incorporate it into the design of order-matching and idle vehicle dispatch algorithms.

B. Matching in Ride-pooling Systems

Ride-pooling introduces additional complexity compared to ride-hailing, as it requires not only the assignment of drivers to passengers but also the coordination of passengers with similar routes while optimizing pick-up and drop-off sequences to minimize detours. Agatz et al. [19] categorized ride-sharing into several types, including single-driver, single-rider; single-driver, multiple-riders; multiple-drivers, single-rider; and multiple-drivers, multiple-riders. Among these, ride-hailing corresponds to the single-driver, single-rider case, whereas ride-pooling aligns with the single-driver, multiple-rider model, introducing additional constraints related to route compatibility, shared occupancy optimization, and dynamic reassignments.

A variety of optimization-based approaches have been developed to address ride-pooling matching. Alonso-Mora et al. [20] proposed an anytime optimal algorithm that balances fleet size, vehicle capacity, passenger wait times, trip detours, and operational costs. Their method begins with a greedy allocation and iteratively refines the solution through constrained optimization, ensuring computational scalability while converging to an optimal allocation. Li et al. [21] focused on scenarios where each driver serves at most two passengers, introducing a rolling horizon approach with financial incentives to encourage ride-pooling adoption. Their model, formulated as a multi-stage integer programming problem, seeks to maximize system-wide profitability. Chen et al. [22] modeled ride-pooling services using a layered OD graph, decomposing the matching process into three subproblems: vehicle dispatching, order pooling sequence, and routing optimization. A minimum-cost network flow model was employed to optimize the order matching sequence.

Given the computational challenges of real-time ridepooling matching, various studies have explored ways to improve algorithmic efficiency. Simonetto et al. [23] formulated the ride-pooling problem as a linear assignment problem between vehicles and passenger requests, employing a federated optimization framework to enhance computational speed. Meshkani et al. [24] introduced a decentralized ridepooling model based on vehicle-to-infrastructure (V2I) and infrastructure-to-infrastructure (I2I) communication, achieving a 25.53-fold improvement in computational efficiency over conventional methods.

Some studies have sought to jointly optimize ride-hailing and ride-pooling operations by developing unified matching models. Qin et al. [25] proposed a multi-objective integer linear programming framework that simultaneously considers three service modes: ride-pooling (RP), non-ride-pooling (NP), and a "bundled" hybrid model. Their two-stage Kuhn-Munkres (2-KM) algorithm iteratively refines passenger-vehicle assignments, ensuring an adaptive balance between shared and private rides. Similarly, Zhou et al. [26] introduced a decision-making framework for ride-hailing platforms that optimizes passenger-vehicle matching while considering passengers' monetary and travel experience trade-offs.

Matching timing plays a crucial role in determining the effectiveness of ride-pooling. A longer matching time window

allows for accumulating a larger set of potential passengers with compatible routes, reducing overall detour distances and improving efficiency. However, longer wait times may negatively impact user experience. The challenge, therefore, lies in dynamically determining the optimal moment to execute matching decisions rather than relying on predefined fixedinterval batching methods. Guo et al. [27] introduced a realtime ride-pooling framework with dynamic time windows and expectation-based migration, where ride requests can be strategically deferred to future time windows based on historical demand patterns. Their multi-strategy graph search heuristic improves scalability, enabling the system to handle large-scale ride-pooling scenarios. Zhao et al. [28] proposed an embedding-based online matching framework that adapts to network structure changes. This approach constructs a dynamic heterogeneous ride-pooling network incorporating various node attributes and driver-passenger connections. By continuously updating representations to capture network evolution, it efficiently identifies and ranks candidate passengers for real-time matching. Furthermore, Yang et al. [29] proposed a proactive vehicle dispatch strategy that forecasts future supply and demand using a Poisson distribution. The strategy estimates potential distance savings from future orders and solves a bipartite matching problem to assign passengers to partially occupied or idle vehicles or defer them to the next matching round. However, most traditional operations research approaches rely on static demand forecasts and do not incorporate real-time adaptive decision-making, limiting their effectiveness in highly dynamic environments.

Given the high uncertainty in real-time ride-pooling demand, existing approaches that rely on predefined matching intervals or offline demand estimation lack the flexibility to adapt to evolving conditions. This study addresses these limitations by developing a reinforcement learning (RL)-based approach that optimizes the timing of ride-pooling matches, enabling real-time adjustments to supply-demand variations. By leveraging RL, the proposed method learns when to execute ride-matching decisions based on system conditions rather than relying on fixed rules or heuristic estimations. This allows the system to continuously adapt to changing demand patterns, improving efficiency while maintaining acceptable passenger wait times.

C. RL for Ride-hailing and Ride-pooling Systems

In real-world ride-hailing and ride-pooling systems, supply and demand conditions are highly dynamic and continuously evolving, posing challenges for effective ride-matching. Reinforcement learning (RL) has emerged as a promising approach for addressing these complexities due to its adaptive decision-making capabilities. By continuously learning from changing market conditions, RL can optimize decision policies while adapting to uncertainties in real-time demand fluctuations. Additionally, its scalability and transferability enable rapid deployment across different urban environments, ensuring sustained operational efficiency.

Several studies have explored RL applications in ridehailing and ride-pooling to address various optimization challenges. Qiao et al. [30] proposed a multi-agent RL-based algorithm (ERPM) for intelligent ride-hailing demand prediction. ERPM mitigates convergence issues in traditional RL models, which arise from the high dimensionality of gridbased demand forecasting, by leveraging the Actor-Critic strategy to optimize ride-hailing dispatch decisions. Similarly, Zhang et al. [31] developed NondBREM, an offline deep reinforcement learning framework for large-scale ride-hailing order dispatch. To avoid costly and unsafe interactions with the environment, it learns solely from historical data. The framework incorporates a Nondeterministic Batch-Constrained Q-learning (NondBCQ) module to reduce extrapolation errors and a Random Ensemble Mixture (REM) module to enhance generalization and robustness. For ride-pooling, Hu et al. [32] proposed the Localized Bipartite Match Graph Attention Q-Learning (BMG-Q) framework, specifically designed for ride-pooling order dispatch. It achieves approximately 10\% higher cumulative rewards than baseline reinforcement learning models while reducing overestimation bias by over 50% . Additionally, BMG-Q demonstrates robustness to task and fleet size variations, making it an effective, scalable, and resilient framework for ride-pooling operations.

A closely related study by Ke et al. [33] applied multiagent RL to determine whether passengers should delay matching to improve their pairing outcome. Their approach treats each passenger as an independent agent, benefiting from multiple reward signals per time step due to the continuous arrival of ride requests, thereby mitigating sparse reward issues. However, their method optimizes individual waiting decisions rather than system-wide efficiency, potentially leading to locally optimal rather than globally optimal solutions. Additionally, their study does not consider ridepooling scenarios, limiting its applicability to shared mobility services with more complex passenger-routing constraints. In contrast, this study models the ride-matching system as a single learning agent, optimizing matching timing across both ride-hailing and ride-pooling. Unlike fixed-time matching strategies, which operate on predefined intervals regardless of real-time conditions, our approach dynamically determines when to execute matching based on observed supply-demand fluctuations. Additionally, rather than optimizing individual passenger decisions, as seen in Ke et al. [33], our method seeks a globally optimal matching policy that enhances overall system efficiency. Furthermore, while most RL-based studies focus exclusively on ride-hailing, our framework extends to ride-pooling, capturing the added complexity of shared trips and optimizing both driver-passenger and passenger-passenger assignments. The positioning of our work relative to existing research is summarized in Table I.

Table I RESEARCH COMPARISON

Researchers	Ride-Hailing	Ride-Pooling	Spatial Matching	Temporal Matching	RL
Yan et al.[7]	✓		√		
Zhang et al.[10]	✓		✓		
Yang et al.[16]	✓		✓	✓	
Shi et al.[18]	✓		✓	✓	
Agatz et al.[19]		✓	✓		
Alonso-Mora et al.[20]		✓	✓		
Qin et al.[25]	✓	✓	✓		
Yang et al.[29]		✓	✓	✓	
Qiao et al.[30]	✓				✓
Hu et al.[32]		✓			✓
et al.[33]	✓		✓	✓	✓
Our Research	✓	✓	✓	✓	✓

III. OPTIMIZING THE TIMING OF BATCHED MATCHING

This section introduces the methodology developed to optimize the timing for batched matching in ride-hailing and ride-pooling systems. First, we provide a general description of the problem. Second, we model the problem as a sequential decison-making problem under the reinforcement learning (RL) framework. Lastly, we describe how to solve this optimization problem with the popular Proximal Policy Optimization (PPO) algorithm [34].

A. Batched matching

Batched matching is an optimization problem that assigns drivers to rider requests collected within a batching window. The goal is to maximize or minimize certain objective functions—such as social, economic, or service quality metrics—relative to a set of feasible assignments. As illustrated

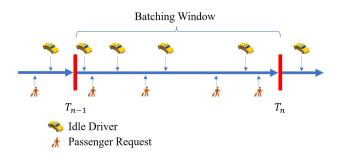


Figure 1. Comparison of Rewards Before and After PBRS

in Figure 1, rider requests are collected during a pre-defined batching window (e.g., T_n in Fig. 1). At the end of this window, a batched matching algorithm computes the assignments and sends the results to both drivers and riders. Any unmatched rider requests from this batch are carried over and resolved in subsequent batching windows.

Formally, we divide the time of a day into a set of time windows $\{T_0,...,T_n,..\}$ where $T_n=[t_{n-1},t_n]$ and $t_n-t_{n-1}=\Delta T \ \forall n$. Instead of executing the matching algorithm at every predefined t_n , we aim to dynamically determine the optimal timing for conducting the matching. That is, find the

best t_n throughout the day to execute the matching algorithm. The objective is to minimize the total waiting time in the system by dynamically adjusting the batching window based on system conditions.

B. Deciding when to match as an RL task

The problem of determining when to match can be naturally formalized as a sequential decision-making problem and thus an RL task. By formulating the problem as a Markov Decision Process [35], we enable the training of an adaptive strategy for determining the optimal matching moments with RL methods.

1) Finite-horizon Markov Decision Process

The process of determining when to match in ride-hailing and ride-pooling services can be formulated as a finite-horizon Markov Decision Process (MDP), which is usually described as a 5-tuple (S, A, P, R, T):

- S: The state space represents the set of all possible system states. Each state $s \in S$ describes the system status of ride-hailing or ride-pooling at a specific time step.
- A: The action space A represents the decisions available to the match-maker at each time step: whether or not to execute a matching operation in the ride-hailing or ridepooling system. Each action a ∈ A directly impacts the system's state transition.
- P: The transition function defines the transition probability P(s'|s, a) from a state s ∈ S to another state s' ∈ S after taking action a ∈ A. This reflects the dynamics of passengers and drivers.
- R: The reward function $R(s_t, a_t, s_{t+1})$ defines the immediate reward obtained by the match-maker when executing action a_t in state s_t at time step t, subsequently transitioning to the next state s_{t+1} . This signal reflects the waiting time of passengers in a given state, capturing the efficiency of the ride-hailing and ride-pooling systems.
- T: The horizon represents the total number of time steps in the finite-horizon MDP.

In a finite-horizon MDP, the match-maker interacts with the environment by observing its state $s_t \in \mathcal{S}$ and selecting an action $a_t \in \mathcal{A}$ at each time step $t = 0, \dots, T-1$. The objective is to maximize the expected cumulative reward (total return) G over the finite horizon:

$$G = \mathbb{E}\left[\sum_{t=0}^{T-1} R(s_t, a_t, s_{t+1})\right] \tag{1}$$

Formulating the problem of when to match as a finite-horizon MDP captures the sequential and dynamic nature of this problem, allowing for adaptive strategies to handle real-time supply-demand fluctuations. This framework provides a foundation for applying RL algorithms to learn efficient solutions. The key components of MDP are given as follows.

2) Action

The action of the match-maker is represented by a binary variable $a_t \in \{0,1\}$, where $a_t = 0$ represents skip matching, accumulating more passengers and drivers for future matches. $a_t = 1$ indicates conduct the matching, triggering the spatial matching algorithm to assign drivers to passengers.

3) State

Determining the optimal timing for matching in ride-hailing and ride-pooling services is a complex decision problem, which has been formulated as a finite-horizon MDP in the previous section, where the match-maker has complete knowledge of the system state at any time step. However, in practice, real-world ride-hailing and ride-pooling systems rarely provide full visibility due to challenges such as data delays, uncertainty about future events, and the complexity of system dynamics.

Given these challenges, the problem is better modeled as a Partially Observable Markov Decision Process (POMDP) [36], where the match-maker must make decisions based on partial observations of the system state. While we leave the exploration of more sophisticated state designs and the use of recurrent neural networks (RNNs) [37, 38, 39] for future work, we simplify the problem by treating it as an MDP. This is achieved by selecting a set of relevant state variables to define the state space, enabling computationally efficient training while retaining the key elements necessary for effective decision-making.

In this study, the state at each time step t is represented as:

$$s_t = \left[T_t, \Delta T_t, N_p(t), \overline{W}_p(t), W_{\text{max}}(t), N_d(t) \right], \quad (2)$$

where each variable denotes a relevant statistic:

- T_t: Current system time, providing context for time-ofday variations.
- ΔT_t : Time elapsed since the last matching operation.
- N_p(t): Number of unmatched passenger requests in the system.
- ullet $W_p(t)$: Average waiting time of unmatched passenger requests.
- $W_{\rm max}(t)$: Maximum waiting time among unmatched passenger requests.
- $N_d(t)$: Number of available drivers in the system.

4) Reward

a) Natural Reward Design

The reward function in this RL framework aims to minimize passenger waiting time. In ride-hailing systems, total waiting time consists of matching waiting time (time until a driver is assigned) and driver arrival waiting time (time until the driver arrives). In ride-pooling, an additional detour delay due to shared routes is included.

For ride-hailing, the reward $R^H(s_t, a_t, s_{t+1})$ at state s_t , after executing action a_t and transitioning to state s_{t+1} , is defined as:

$$R^{H}(s_{t}, a_{t}, s_{t+1}) = -(\phi R_{m}(s_{t}, a_{t}, s_{t+1}) + R_{w}(s_{t}, a_{t}, s_{t+1}))$$
(3)

where $\phi \in [0,\infty)$ is the coefficient reflecting passengers' relative tolerance for waiting to be picked up compared to the waiting to be matched. $R_m(s_t,a_t,s_{t+1})$ represents the incremental waiting time for all unmatched passengers at the current state s_t , after taking action a_t and transitioning to state s_{t+1} , which is formulated as $R_m(s_t,a_t,s_{t+1}) = \Delta t N_p^{unmatch}(s_t,a_t)$. Here Δt represents the length of the time step, $N_p^{unmatch}(s_t,a_t)$ represents the number of unmatched orders at state s_t after taking action a_t . $R_w(s_t,a_t,s_{t+1})$

represents the waiting time for matched passengers to be picked up by drivers at the current state s_t , after taking action a_t and transitioning to state s_{t+1} , which is formulated as:

$$R_w(s_t, a_t, s_{t+1}) = \begin{cases} 0, & a_t = 0\\ f_{pick}(s_t), & a_t = 1 \end{cases}$$
 (4)

where $f_{pick}(s_t)$ represents the function to calculate the sum of the waiting time for matched passengers to be picked up by drivers at the current state s_t , assuming a matching decision is made. The specific calculation method of $f_{pick}(s_t)$ will be detailed in the Matching Algorithm section.

For ride-pooling, the reward function $R^P(s_t, a_t, s_{t+1})$ additionally includes detour delay $R_d(s_t, a_t, s_{t+1})$:

$$R^{P}(s_{t}, a_{t}, s_{t+1}) = -(\phi R_{m}(s_{t}, a_{t}, s_{t+1}) + \tau R_{d}(s_{t}, a_{t}, s_{t+1}) + R_{w}(s_{t}, a_{t}, s_{t+1}))$$
(5)

where τ is a another weighting coefficient reflecting passengers' higher tolerance for detour delay compared to the time spent waiting to be matched. $R_d(s_t, a_t, s_{t+1})$ represents the detour delay caused by ride-pooling for all matched passengers at state s_t , after taking action a_t and transitioning to state s_{t+1} , which is formulated as:

$$R_d(s_t, a_t, s_{t+1}) = \begin{cases} 0, & a_t = 0\\ f_{detour}(s_t), & a_t = 1 \end{cases}$$
 (6)

where $f_{detour}(s_t)$ represents the function to calculate the sum of the detour delay for matched passengers at the current state s_t , assuming a matching decision is made. The specific calculation method of $f_{detour}(s_t)$ will also be detailed in the Matching Algorithm section.

b) Reward Sparsity and Potential-based Reward Shaping Although the reward functions R^H and R^P for ridehailing and ride-pooling are well-defined and theoretically suitable for learning an adaptive match-maker policy to determine matching decisions, they suffer from the challenge of sparse rewards. The sparse reward problem arises because $R_w(s_t, a_t, s_{t+1})$ and $R_d(s_t, a_t, s_{t+1})$ yield non-zero feedback only when $a_t = 1$, providing no information when $a_t = 0$. As a result, the match-maker receives limited guidance during training, making it difficult to learn an effective policy.

To address this issue, we employ *Potential-Based Reward Shaping* (PBRS) [40], a method designed to refine the reward structure without altering the optimal policy. In the following, we describe how the PBRS is applied to mitigate the sparse reward problem and improve learning efficiency.

PBRS modifies the reward function of RL with a potential function $\Phi: \mathcal{S} \to \mathbb{R}$ intending to provide a denser learning signal for faster convergence. The modified reward function is defined as follows in finite-horizon MDPs:

$$R'(s_t, a_t, s_{t+1}) = \begin{cases} R(s_t, a_t, s_{t+1}) + \Phi(s_{t+1}) - \Phi(s_t) & t \neq T - 1, \\ R(s_{T-1}, a_{T-1}, s_T) - \Phi(s_{T-1}) + \Phi(s_0), & t = T - 1. \end{cases}$$
(7

where $\Phi(s_t)$ reflects the desirability of state s_t . Importantly, as shown in Equation (7), the modified reward at the final

step T-1 differs from that of other steps. This adjustment is made to compensate for the discrepancy introduced by PBRS, ensuring that the expected total return G' under the modified reward function R' remains identical to the expected original total return G for all policies (for detailed calculations, see Equation (1)). As a result, PBRS preserves the optimal policy. The proof is provided below:

$$G' = \mathbb{E}\left[\sum_{t=0}^{T-1} R'(s_t, a_t, s_{t+1})\right]$$

$$= \mathbb{E}\left[\sum_{t=0}^{T-2} \left(R(s_t, a_t, s_{t+1}) + \Phi(s_{t+1}) - \Phi(s_t)\right)\right]$$

$$= \mathbb{E}\left[\sum_{t=0}^{T-2} R(s_t, a_t, s_{t+1}) - \Phi(s_0)\right]$$

$$+ \Phi(s_{T-1}) + R'(s_{T-1}, a_{T-1}, s_T)$$

$$= \mathbb{E}\left[\sum_{t=0}^{T-2} R(s_t, a_t, s_{t+1}) + R(s_{T-1}, a_{T-1}, s_T)\right]$$

$$= \mathbb{E}\left[\sum_{t=0}^{T-1} R(s_t, a_t, s_{t+1})\right]$$

$$= G$$
(8)

The proof above shows that incorporating a potential function into the reward structure does not alter the expected total return for any policy. Consequently, a policy that is optimal under the original reward function remains optimal under the modified reward function. Moreover, prior studies have demonstrated that, when the potential function is appropriately selected—often with the aid of domain knowledge—PBRS can significantly accelerate learning without detriment to the agent's asymptotic performance, particularly in environments characterized by sparse rewards, as in our case [40, 41, 42, 43]. Below, we describe the design of our potential function.

Our objective is to construct a potential function that prevents the rewards from being concentrated solely on the matching step, as is the case with the original rewards. Instead, the modified rewards should be distributed more uniformly across all actions in the sequence. This uniform distribution enables the match-maker to better discern advantageous states, thereby accelerating the learning process. In addressing the problem of determining when to match in ride-hailing and ride-pooling systems, the potential function must accurately reflect the desirability of the current state with respect to the optimization objective. Given that our ultimate goal is to minimize passenger waiting time, which is closely related to the matching outcome, we can hypothesize a matching action at the current time step and evaluate the result of this hypothetical match. By constructing a potential function based on the quality of this hypothetical matching result, we can better guide the match-maker in optimizing its strategy. The quality of matching results in ride-hailing and ride-pooling can be directly evaluated using the previously mentioned $f_{pick}(s_t)$ and $f_{detour}(s_t)$: the former calculates the time passengers wait for a matched driver to arrive, while the latter measures the detour delay for passengers in ride-pooling.

Below are our designs of potential functions in the cases of ride-hailing and ride-pooling:

$$\Phi^h(s_t) = -f_{pick}(s_t) \tag{9}$$

$$\Phi^p(s_t) = -(f_{pick}(s_t) + f_{detour}(s_t)) \tag{10}$$

The functions $f_{pick}(s_t)$ and $f_{detour}(s_t)$ are originally used to calculate R_w and R_d , respectively, and are only evaluated when $a_t=1$. However, by introducing them into the potential function, they can now be evaluated for all actions, regardless of whether matching occurs $(a_t=1)$ or not $(a_t=0)$. This design allows the potential function to provide a consistent evaluation of the current state, offering insights into the hypothetical impact of performing a matching operation.

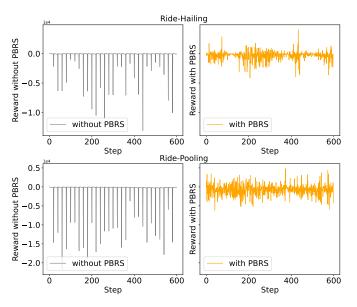


Figure 2. Comparison of Rewards without and with PBRS

Figure 2 examines the distribution of rewards without (Equations 3, 5 and with PBRS (Equations 7, 9, 10). The gray line (without-PBRS) exhibits sparse rewards with large negative fluctuations, which can hinder training convergence. In contrast, the orange line (with-PBRS) shows smoother and more frequent rewards, effectively mitigating the sparse reward issue and providing consistent feedback.

This design maintains the overall learning signal while offering the agent more frequent feedback, encouraging better decisions during waiting and matching. Consequently, PBRS enhances the training efficiency and performance of the ridehailing and ride-pooling systems.

C. Learning Algorithm

This study employs the Proximal Policy Optimization (PPO) algorithm [34] to determine the matching timing in ride-hailing and ride-pooling services. PPO is widely recognized for its stability, computational efficiency, and robust performance in dynamic environments like ride-hailing. Its clipped objective function ensures stable learning by preventing overly large policy updates, making it particularly suitable for environments

with fluctuating supply and demand. Additionally, PPO's scalability and adaptability enable it to perform effectively across a wide range of scenarios, optimizing decision-making in largescale simulations.

PPO improves upon traditional policy gradient methods such as REINFORCE [44] and A2C [45] by introducing a surrogate objective function that constrains updates to the policy. Specifically, PPO uses the following clipped objective to optimize the policy:

$$L^{\text{CLIP}}(\theta) = \mathbb{E}_t \left[\min \left(r_t(\theta) \hat{A}_t, \text{clip}(r_t(\theta), 1 - \epsilon, 1 + \epsilon) \hat{A}_t \right) \right]$$
(11)

Where $r_t(\theta) = \frac{\pi_{\theta}(a_t|s_t)}{\pi_{\theta_{\text{old}}}(a_t|s_t)}$ is the probability ratio between the new policy π_{θ} and the old policy $\pi_{\theta_{\text{old}}}$; \hat{A}_t is the estimated advantage function, which measures the relative value of action a_t in state s_t ; ϵ is a hyperparameter (typically 0.1 or 0.2) that controls the range of permissible updates to the policy.

The clipping mechanism ensures that the probability ratio $r_t(\theta)$ remains within a safe range $[1 - \epsilon, 1 + \epsilon]$, preventing large updates that could destabilize learning.

In addition to the policy update, PPO uses a learned critic [46] to estimate state values and advantages. The total loss function combines the policy loss L^{CLIP} , the value loss for the critic, and an entropy term to encourage exploration:

$$L(\theta) = L^{\text{CLIP}}(\theta) - c_1 L^{\text{VALUE}}(\theta) + c_2 L^{\text{ENTROPY}}(\theta)$$
 (12)

Where $L^{\text{VALUE}}(\theta) = \left(V_{\theta}(s_t) - V_t^{\text{target}}\right)^2$ minimizes the difference between the estimated and target state values; $L^{\text{ENTROPY}}(\theta)$ encourages policy entropy, promoting diverse action selection; c_1 and c_2 are coefficients controlling the relative importance of value loss and entropy.

In this study, PPO trains an agent (match-maker) to decide when to match based on passenger demand, driver availability, and system delays. The neural network architecture is based on an actor-critic framework with multi-layer perceptrons (MLPs). Both the actor and critic networks consist of three hidden layers with 64 units per layer, activated by the Tanh function. The actor network outputs action probabilities, while the critic estimates state values. Actions are sampled from a Bernoulli distribution.

The agent's performance is tracked to monitor key metrics, such as passenger delays. By determining the matching timing, this framework demonstrates significant improvements in system efficiency, effectively addressing the challenges of dynamic ride-hailing and ride-pooling environments.

The basic flow of the PPO algorithm implemented in this study is shown in the pseudocode below. And details of the training parameters are provided in the Appendix.

This pseudocode summarizes the PPO algorithm's structure, highlighting policy optimization, value estimation, and gradient updates, which allow the agent to effectively learn strategies for optimizing matching intervals.

IV. SIMULATOR DESIGN

To validate and evaluate the proposed approach, we develop a simulator to model the dynamics of ride-hailing and ridepooling services. The simulator utilizes a real public dataset

Algorithm 1 PPO Algorithm

```
1: for episode = 1 to E do
          Reset the environment and get initial state s_0
 2:
          Initialize episode return Return = 0 and done =
 3:
     False
          for t = 0 to T - 1 do
 4:
                Sample action a_t from policy \pi_{\theta}(a_t|s_t)
 5:
                Execute action a_t in environment
 6:
 7:
                Observe reward r_t, next state s_{t+1}, and done status
     d_t
                Store (s_t, a_t, r_t, s_{t+1}, d_t) in trajectory buffer
 8:
                if done then
 9:
                     break
10:
                end if
11:
          end for
12:
          Compute discounted returns R_t and advantages \hat{A}_t
13:
     using GAE
          for k = 1 to K do
14:
                Sample minibatch from trajectory buffer
15:
                Compute importance ratio r_t(\theta) = \frac{\pi_{\theta}(a_t|s_t)}{\pi_{\theta_{\text{old}}}(a_t|s_t)}
16:
                Compute clipped surrogate objective:
17:
                L^{\text{CLIP}}(\theta) = \mathbb{E}_t \left[ \min \left( r_t(\theta) \hat{A}_t, \text{clip}(r_t(\theta), 1 \pm \epsilon) \hat{A}_t \right) \right]
18:
               Compute value function loss: L^{\text{VALUE}}(\theta) = \left(V_{\theta}(s_t) - V_t^{\text{target}}\right)^2 Compute entropy bonus:
19:
20:
21:
                L^{\text{ENTROPY}}(\theta) = \mathbb{E}_t \left[ -\pi_{\theta}(a_t|s_t) \log \pi_{\theta}(a_t|s_t) \right]
22:
                Compute total loss:
23:
                L(\theta) = L^{\text{CLIP}}(\theta) - c_1 L^{\text{VALUE}}(\theta) + c_2 L^{\text{ENTROPY}}(\theta)
24.
                Update policy and value function using gradient
25:
     ascent on L(\theta)
          end for
26:
27: end for
```

of taxi trips in Manhattan, New York City [47], to generate realistic passenger orders and driver distributions, capturing both spatial and temporal fluctuations in supply and demand. Details on the generation passenger and driver data process are provided in Subsection A. Also, the simulator incorporates matching algorithms for both ride-hailing and ride-pooling settings, as described in Subsection B.

To ensure a balance between realism and computational efficiency, several assumptions are made. Vehicles are assumed to travel at a constant speed of 40 km/h to simplify travel time calculations. Passengers cancel their orders if unmatched within five minutes, and drivers wait at pick-up points for up to 10 minutes before repositioning. Additionally, pooling is limited to two passenger orders per vehicle.

A. Passenger and Driver Data Generation

The simulator leverages historical for-hire vehicle (FHV) trip records provided by the New York City government [47], specifically from January to March 2022. These records include request times, pickup and drop-off details, and predefined origin-destination zones. The zones are classified according to the structure already provided in the dataset, reflecting the geographic divisions used in the original records. To model

the stochastic nature of ride requests and driver availability, a Poisson distribution is employed, capturing temporal and spatial variations in demand and supply. Each zone is assigned a unique Poisson distribution with a calibrated mean rate (λ) calculated as the average number of requests per time period, derived from the historical data. This approach enables realistic event generation that reflects time- and location-specific demand patterns.

Passenger destinations are represented using a transition probability matrix constructed from historical trip data. This matrix encodes the likelihood of travel between geographical zones, capturing observed patterns such as commuting routes and peak demand areas to ensure realistic traffic flow simulation. Each entry in the matrix represents the proportion of trips originating from a specific zone and ending in another zone. This is calculated as the ratio of trips between the two zones to the total number of trips originating from the same zone, providing a detailed and data-driven representation of passenger movement within the city.

The generated dataset comprises passenger request and cancellation times, pickup points, destinations, driver locations, and availability, creating a dynamic, second-by-second simulation environment. As depicted in Figure 3, the simulated passenger request data and idle driver data closely matches real-world daily variations in ride demand and supply. This setup supports the reinforcement learning framework by providing varied conditions for training and evaluation, with continuous decision points driven by fluctuating demand and supply. Parameter adjustments allow for diverse scenario testing, ensuring the algorithm's robustness and adaptability to real-world conditions.

B. Matching Algorithms

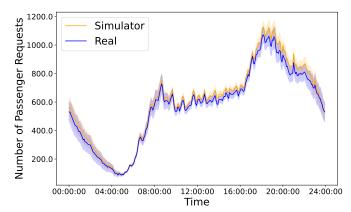
This subsection describes the matching algorithms designed for ride-hailing and ride-pooling services. For ride-pooling, the matching process consists of two stages: first, orders with similar origins and destinations are paired through passenger-passenger matching; second, the paired passengers are matched with a driver. For ride-hailing, the passenger-driver matching process is similar to ride-pooling, except it involves a single passenger being matched with a driver instead of a passenger pair.

1) Passenger-Passenger Matching (ride-pooling)

In the simulation, the passenger-passenger matching algorithm matches passenger orders with the goal of minimizing the detour distance. To quantify the impact of detour on travel efficiency, we define detour delay rate, which is formulated as:

$$DDR = \frac{Distance without detour}{Distance after detour}$$

where "Distance without detour" represents the direct travel distance for a single passenger, and "Distance after detour" is the actual distance when accommodating multiple passengers. Higher DDR values correspond to more efficient pairings with minimal detour effects, while lower values indicate significant disruptions.



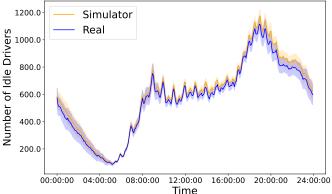


Figure 3. Comparison of Passenger Request Data and Idle Driver Data Generated by the Simulator and Real Data

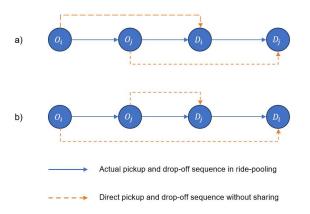


Figure 4. Possible Pickup and Drop-off Sequences for Two Passenger Orders

However, when calculating the DDR values for two matched passenger orders i and j, their DDR values are usually different. Additionally, different pickup and drop-off sequences may result in varying DDR values. As shown in Figure 4, the pickup and drop-off sequences can be categorized into two cases, labeled as (a) and (b) in the figure. Here, O_i and D_i represent the origin and destination of passenger i, respectively.

In case (a), both orders i and j experience detours due to the ride-pooling route. For order i, the distance after the detour

is increased due to the detour through O_j . Similarly, for order j, the distance after the detour is increased due to the detour through O_i . Therefore, the DDR values for orders i and j under pickup and drop-off sequence (a), denoted as $DDR^a(i)$ and $DDR^a(j)$, can be computed as follows:

$$DDR^{a}(i) = \frac{d(O_{i}, D_{i})}{d(O_{i}, O_{i}, D_{i})}$$
(13)

$$DDR^{a}(j) = \frac{d(O_{j}, D_{j})}{d(O_{j}, D_{i}, D_{j})}$$
(14)

Where $d(\cdot)$ represents the shortest path distance in a road network when traveling through a specified sequence of locations. To evaluate the matching quality between orders i and j in case (a), we define $DDR^a(i,j)$ as the smaller value between $DDR^a(i)$ and $DDR^a(j)$. This represents the DDR of the passenger in the pair who is more significantly affected by the detour. The corresponding formula is as follows:

$$DDR^{a}(i,j) = \min(\frac{d(O_{j}, D_{j})}{d(O_{j}, D_{i}, D_{j})}, \frac{d(O_{j}, D_{j})}{d(O_{j}, D_{i}, D_{j})})$$
(15)

In case (b), the situation is more straightforward. As shown in Figure 4, only order i experiences a detour, while order j is directly delivered from its origin to its destination. Consequently, the DDR value for order j, $DDR^b(j)$, is equal to 1. To evaluate the matching quality in case (b), $DDR^b(i,j)$ is determined by the DDR value of order i, $DDR^b(i)$. The corresponding formula is as follows:

$$DDR^{b}(i,j) = \frac{d(O_{i}, D_{i})}{d(O_{i}, O_{j}, D_{j}, D_{i})}$$
(16)

Therefore, by calculating the DDR values for the two pickup and drop-off sequences, the optimal sequence can be selected, which corresponds to the sequence with the larger DDR value. The DDR value of the optimal sequence is then used to assess the matching quality between orders i and j, denoted as DDR(i,j). The formula for computing DDR(i,j) is as follows:

$$DDR(i, j) = \max \begin{pmatrix} \min \left(\frac{d(O_i, D_i)}{d(O_i, O_j, D_i)}, \frac{d(O_j, D_j)}{d(O_j, D_i, D_j)} \right), \\ \frac{d(O_i, D_i)}{d(O_i, O_j, D_j, D_i)} \end{pmatrix},$$
(17)

Once the pickup and drop-off sequence between orders i and j has been determined, the detour distance for order i, Δd_i , caused by the ride-pooling can also be calculated. The corresponding formulas are as follows:

$$\Delta d_i = d(O_i, \dots, D_i \mid \sigma^*) - d(O_i, D_i) \tag{18}$$

where $d(O_i, D_i)$ is the shortest path distance for passenger i if served independently; $d(O_i, \ldots, D_i \mid \sigma^*)$ is the actual travel distance from O_i to D_i under the optimal pickup and drop-off sequence σ^* .

So, the passenger-passenger matching process comprises three steps:

- a. Retrieve Passenger Orders: At each time step, the algorithm retrieves all active passenger orders, including information such as origin, destination, and request time.
- b. Evaluate Matches: By iterating over all active passenger orders, all feasible passenger-passenger pairs are enumerated. For each feasible pair, the matching quality and the resulting detour for each order after matching are calculated based on equations (17) and (18).
- c. Optimize Passenger-passenger Pairing: To handle cases where orders can be matched with multiple others, we optimize pairing as an integer linear optimization problem, represented by a graph $G=(P_{s_t},E_{s_t})$, where P_{s_t} is the set of passenger orders at state s_t , and E_{s_t} represents the set of feasible passenger pairs. Each edge $e \in E_{s_t}$ has a weight $\omega(e)$, which reflects the matching utility (based on the Detour Delay Rate, DDR). The decision variable x(e) is binary, where x(e)=1 indicates that the pair corresponding to edge e is selected.

The optimization problem is formulated as:

$$\max \sum_{e \in E_{s_t}} \omega(e) \cdot x(e), \tag{19}$$

$$\sum_{e \in \delta(p)} x(e) \le 1, \qquad \forall p \in P_{s_t}, \qquad (20)$$

$$x(e) \in \{0, 1\}, \qquad \forall e \in E_{s_t}. \tag{21}$$

where, $\delta(p)$ denotes the set of edges connected to passenger order p. The first constraint ensures that each passenger can only be matched with one other passenger. Solving this optimization problem yields the set of optimal passenger-passenger pairs PP_{s_t} at state s_t , which maximizes the total DDR.

Based on the optimal matching results, the sum of the detour delay for matched passengers at the current state s_t can be calculated:

$$f_{detour}(s_t) = \sum_{i \in PP_{s_t}} \Delta d_i / v$$
 (22)

Where v represents the average vehicle speed.

The passenger-passenger matches are then treated as single entities in the Passenger-Driver Match phase, allowing for seamless integration into the overall ride-hailing system.

2) Passenger-Driver Matching

This phase finalizes the matching process for regular ridehailing services and serves as the second stage for ridepooling, where passengers (or passenger pairs in ride-pooling) are assigned to the nearest available driver to minimize pickup waiting time. The driver-passenger matching problem is modeled as an Integer Linear Programming (ILP) problem, formulated as:

$$\min \sum_{i \in D_{s_t}} \sum_{j \in PP_{s_t}} T_{ij} \cdot x_{ij}, \tag{23}$$

$$\sum_{i \in D_{s_t}} x_{ij} \le 1,\tag{24}$$

$$\sum_{j \in PP_{s_*}} x_{ij} \le 1,\tag{25}$$

$$x_{ij} \in \{0, 1\}, \qquad \forall i \in D_{s_t}, \ \forall j \in PP_{s_t}, \quad (26)$$

where: D_{s_t} is the set of available drivers at state s_t . PP_{s_t} is the set of unmatched passenger orders (ride-hailing) or passenger pairs (ride-pooling) at s_t . T_{ij} is the pickup time between driver $i \in D_{s_t}$ and passenger (or passenger pair) $j \in PP_{s_t}$, estimated as $T_{ij} = \frac{d_{ij}}{v}$, where d_{ij} is the distance between driver i and passenger (or passenger pair) j. x_{ij} : A binary decision variable, where $x_{ij} = 1$ if driver i is assigned to passenger (or passenger pair) j, and $x_{ij} = 0$ otherwise. Constraints (37) and (38) indicate that each passenger (or passenger pair) can be matched to at most one driver. And each driver can serve at most one passenger (or passenger pair). The optimal result of this optimization problem, x_{ij}^* , is used to calculate $f_{\text{pick}}(s_t)$, which represents the total pickup waiting time at state s_t . Mathematically, we have:

$$f_{\text{pick}}(s_t) = \min \sum_{i \in D_{s_t}} \sum_{j \in PP_{s_t}} T_{ij} \cdot x_{ij}^*,$$

V. EXPERIMENTS

In this section, extensive experiments and sensitivity analyses are conducted to evaluate the performance of the proposed methods

A. Experimental Settings, Baselines and Evaluation Metrics

The match-maker was trained and evaluated using the simulator we designed. Each *episode* represents 10 minutes of service, consisting of 600 discrete time steps (one second per time step). During each episode, the match-maker interacted with the simulator to decide when to perform matching operations. Its performance was assessed across multiple episodes to ensure robustness and generalizability under diverse conditions

Passenger orders and driver data for each episode were generated using the method described in the section IV-A. This method is based on historical FHV trip records from Manhattan, New York City, spanning from January to March 2023 [47]. The dataset reflects the fluctuations in supply and demand, as well as the geographical distribution, throughout the day for both ride-hailing and ride-pooling systems, as shown in Figure 5.

To evaluate the effectiveness of our approach, we compare it against the **batched matching** strategy with fixed time intervals, a strategy currently implemented by ride-hailing platforms such as Uber, serves as the baseline for our study. In the ride-hailing simulator, the optimal batched matching interval is fixed at 15 seconds, while in the ride-pooling simulator, it is set to 20 seconds. These intervals were selected

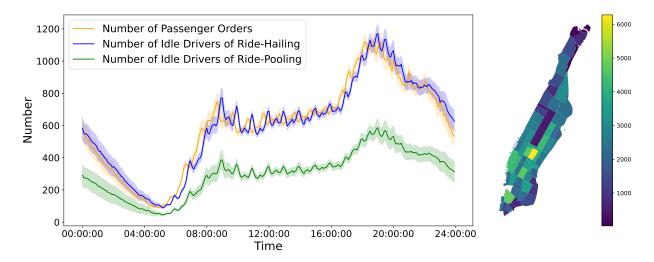


Figure 5. Daily Supply-Demand Fluctuations and Geographical Distribution in Ride-Hailing and Ride-Pooling Systems

based on testing within the same simulator to ensure they provide a representative evaluation across various performance metrics. In the subsequent part of experiments, the **first dispatch** strategy will also be compared with the proposed approach. It is important to note that, since the simulation runs in discrete time steps, the first dispatch strategy implies that matching occurs at every time step within the simulator.

To evaluate the performance of our approach, we designed the following evaluation metrics:

- Average Pickup Time: The average pick-up waiting time per passenger after being matched.
- Average Matching Time: The average waiting time from request to successful matching per passenger.
- Average Detour Delay(only for ride-pooling): The average delay time per passenger due to detours.
- Average Total Waiting Time: The overall average waiting time per passenger.
- **Total Pickup Time**: The total time spent waiting for drivers to pick up passengers.
- Total Matching Time: The total time spent waiting for matching to occur across all passengers.
- **Total Detour Delay**(only for ride-pooling): The total delay caused by detours in ride-pooling.
- **Total Waiting Time**: The cumulative waiting time of all passengers, measuring overall waiting.

The experiments are designed to answer the following questions:

- Section V-B How does the proposed RL approach converge during training for ride-hailing and ride-pooling services, and what is the impact of Potential-Based Reward Shaping on learning efficiency and performance? And how does the proposed RL strategy evolve during training?
- Section V-C What are its performance advantages compared to first dispatch and batched matching strategies (including baseline) in ride-hailing and ride-pooling services?
- Section V-D What is the *adaptability* of the proposed ap-

proach in response to real-time supply-demand variations in ride-hailing and ride-pooling systems?

B. Training Performance

This experiment evaluates the training performance of the RL strategy for Ride-Hailing and Ride-Pooling services during peak traffic hours (8:30–8:40 a.m.) in Manhattan. Each service mode was trained over more than 4,800 episodes (2,880,000 steps), reflecting a highly dynamic supply-demand environment. Passenger requests and driver locations were generated using a Poisson distribution, with parameters fitted and estimated based on the historical data mentioned before. Variability across episodes was introduced by probabilistically sampling different passenger and driver datasets. It is important to note that the datasets of passengers and drivers used in each episode for the experiments in this and subsequent sections are independently generated and unique. The same episodes are not reused across experiments.

To evaluate the impact of Potential-Based Reward Shaping (PBRS), two reward designs—one with PBRS and one with-out—were compared. Each training experiment was repeated five times with different random seeds to ensure reliability, generalizability, and reproducibility. Results were averaged, and confidence intervals were calculated to account for randomness and ensure statistical reliability. The baseline strategy was tested over 1,000 random episodes under identical simulation settings for fair comparisons.

Figure 6 illustrates the Total Return progression during training for the traditional ride-hailing service with and without PBRS. The gray curve (without PBRS) shows that, in the initial few hundred episodes, the Total Return does not improve with training. Instead, it decreases rapidly before stabilizing at a low level in the later stages. Upon analyzing the actions taken by the agent, it was observed that the issue of sparse rewards prevented the agent from consistently receiving $R_w(s_t,a_t)$ signals. As a result, in the later stages of training, the agent repeatedly chose to wait instead of initiating matching actions. This behavior caused passenger

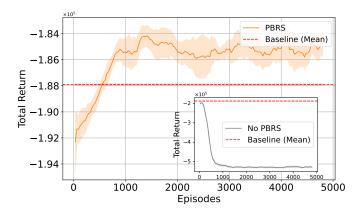


Figure 6. Training Curve of Traditional Ride-Hailing Service

waiting times to increase continuously, further lowering the overall system performance. In contrast, the orange curve, representing the PBRS-enhanced strategy, demonstrates faster convergence and significantly higher Total Return, stabilizing around -186,000 after approximately 600 episodes. PBRS mitigates the sparse reward issue by providing more frequent feedback, enabling the agent to make better decisions and improve learning efficiency. The confidence intervals for the PBRS curve narrow as training progresses, reflecting reduced variability and improved stability in the strategy.

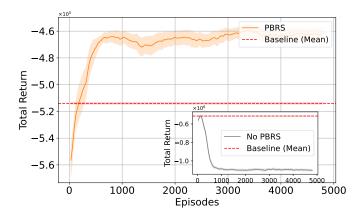


Figure 7. Training Curve of Ride-Pooling Service

The same trend can also be found in the Ride-Pooling service, as shown in Figure 7. The gray curve, representing the non-PBRS strategy, exhibits a sharp decline in Total Return during the initial episodes, followed by stabilization at a very low level, far below the baseline of -520,000. In contrast, the orange curve, representing the PBRS-enhanced strategy, shows a rapid improvement in Total Return during the first 500 episodes. It stabilizes at approximately -460,000 after 1,000 episodes, well above the baseline level. The frequent feedback provided by PBRS helps the agent learn more efficiently, improving its ability to handle complex multi-passenger scenarios. Early in training, the PBRS curve displays wider confidence intervals, reflecting variability due to exploration. However, as training progresses, the confidence intervals narrow, indicating that the strategy becomes more stable and consistent.

Figure 6 and Figure 7 show that PBRS improves the learning process in both ride-hailing and ride-pooling environments, achieving Total Return levels consistently above the baseline in both cases. The ride-pooling curve stabilizes with narrower confidence intervals, reflecting more consistent performance in the learned strategy. In contrast, traditional ride-hailing shows more fluctuations after convergence, which could be due to the simpler matching conditions inherent in this service mode. Additionally, the rapid initial increase in ride-pooling Total Return highlights its effectiveness in optimizing multipassenger matching during the early stages of training.

To further analyze the evolution of the strategy during training, we selected five training checkpoints on the PBRS-enhanced curve (at episode 50, 500, 1000, 2000, and 4800), representing different learning stages from initial exploration to convergence. The strategies at these checkpoints, along with the baseline strategy, were tested over 1,000 episodes in the same simulation environment to ensure fair comparison. Key performance metrics were recorded to evaluate the changes and improvements in the RL strategy across different training stages. The results are shown in Figures 8 and 9, with system evaluation metrics detailed in Appendix Figures 12 and 13.

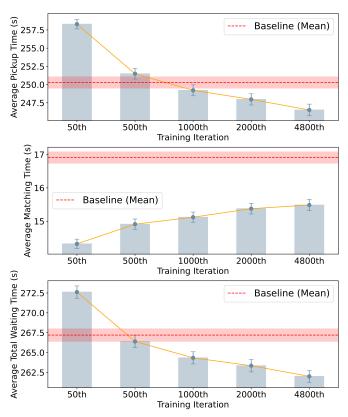


Figure 8. Comparison of Metrics Across Training Episodes and Baseline Strategy for Ride-Hailing Services

Figure 8 demonstrates that as training progresses, the Average Pickup Time decreases from 258 seconds at the early stages to 247 seconds by the 4800th episode. Although the reduction in Pickup Time per passenger is modest, it results in significant system-wide improvements, highlighting the effectiveness of the trained strategy in optimizing Pickup Time.

In contrast, the baseline strategy shows limited capacity to reduce Pickup Time.

The Average Matching Time increases slightly from 14.4 seconds to 15.7 seconds during training. This suggests that the trained strategy prioritizes pickup optimization, leading to minor trade-offs in Matching Time. Despite this increase, the Matching Time remains lower than the baseline strategy.

The Average Total Waiting Time per Passenger decreases from 270 seconds to 262 seconds, which is lower than the baseline strategy's 267 seconds. This indicates that the trained strategy effectively balances the trade-off between Pickup Time and Matching Time, ultimately reducing passenger waiting times across the entire ride-hailing matching process.

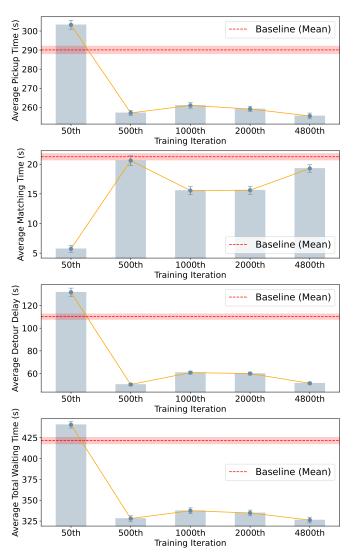


Figure 9. Comparison of Metrics Across Training Episodes and Baseline Strategy for Ride-Pooling Services

In ride-pooling, the trained strategy must balance Pickup Time, Detour Delay, and Matching Time to further reduce Total Waiting Time. The results in Figure 9 provide a clear visualization of this process.

The Average Pickup Time decreases from approximately 310 seconds at episode 50 to 250 seconds by episode 4800, stabilizing around this value, while the baseline remains higher

at 290 seconds, reflecting improved efficiency in reducing post-matching waiting time. The Detour-related metric shows a similar substantial improvement. Average Detour Delay decreases from 125 seconds to 55 seconds, compared to the baseline's 110 seconds.

The Average Matching Time increases during training, peaking at 20 seconds before stabilizing around 18 seconds by episode 4800, outperforming the baseline's 22 seconds. This trend highlights the trained strategy's ability to handle complex ride-pooling scenarios, albeit with slightly longer matching times due to the increased complexity of decision-making, compared to ride-hailing.

With Pickup Time, Detour Delay, and Matching Time all lower than the baseline, the Average Total Waiting Time naturally shows improvement. The Average Total Waiting Time decreases from 440 seconds to 330 seconds by episode 4800, while the baseline remains at 420 seconds.

In summary, the PBRS-enhanced RL strategy, for both ride-hailing and ride-pooling, achieves better matching results by slightly extending the Matching Time, thereby effectively reducing the Total Waiting Time. Compared to the baseline strategy with fixed matching intervals, the PBRS-enhanced RL strategy not only delivers superior matching result but also achieves shorter Matching Time, resulting in a lower Total Waiting Time overall.

C. Performance comparison between different matching strategies

We compare the performance of the proposed approach with batched matching and first dispatch strategies. For batched matching, we adjust the matching interval to examine its impact on performance, using 5, 15, 30, and 60 seconds for ride-hailing, and 10, 20, 40, and 80 seconds for ride-pooling, as shown in Tables II and III, respectively.

The comparative results presented in Table II highlight the effectiveness of the proposed approach in minimizing Average Total Waiting Time, achieving the lowest value of 262.482 seconds among all strategies evaluated. This performance demonstrates an improvement over both the first dispatch strategy and batched matching configurations with fixed intervals. While the average pickup time (247.176 seconds) for the proposed approach is slightly higher than that of the 30-second and 60-second batched strategies, this is Compensated by a significant reduction in average matching time, which is reduced to 15.306 seconds. These findings indicate that the proposed approach achieves a better trade-off between pickup time and matching time, ultimately leading to superior performance in reducing total waiting time.

Similarly, Table III confirms the advantages of the proposed approach in ride-pooling. While the 20-second fixed interval strategy performs best among the batched matching strategies, the proposed approach achieves a lower average total waiting time of 337.810 seconds. Additionally, it outperforms all other strategies in Average Pickup Time, Average Matching Time, and Average Detour Delay, further demonstrating its effectiveness in optimizing system performance across multiple metrics.

Table II
PERFORMANCE COMPARISON OF DIFFERENT MATCHING STRATEGIES IN RIDE-HAILING

Strategies	Average Pickup Time (s)	Average Matching Time (s)	Average Detour Delay (s)	Average Total Waiting Time (s)
First Dispatch	300.661 ± 1.851	1.704±0.366	-	302.365±1.619
5s (Fixed Time Interval)	276.801 ± 1.821	6.807 ± 0.331	-	283.608 ± 1.564
15s (Fixed Time Interval)	251.992 ± 1.601	16.691 ± 0.317	-	268.683 ± 1.322
30s (Fixed Time Interval)	245.571 ± 1.574	31.122 ± 0.308	-	276.693 ± 1.211
60s (Fixed Time Interval)	235.422 ± 1.411	62.533 ± 0.310	-	297.955 ± 1.198
Our approach	247.176 ± 1.629	15.306 ± 0.318	-	$262.482{\pm}1.452$

Table III
PERFORMANCE COMPARISON OF DIFFERENT MATCHING STRATEGIES IN RIDE-POOLING

Strategies	Average Pickup Time (s)	Average Matching Time (s)	Average Detour Delay (s)	Average Total Waiting Time (s)
First Dispatch	416.293 ± 1.367	2.011 ± 0.568	190.514 ± 2.011	608.818±2.099
10s (Fixed Time Interval)	355.919 ± 1.311	12.054 ± 0.455	145.355 ± 1.919	513.328 ± 1.876
20s (Fixed Time Interval)	291.899 ± 1.285	22.038 ± 0.373	113.098 ± 1.755	427.035 ± 1.670
40s (Fixed Time Interval)	280.719 ± 1.233	47.178 ± 0.371	102.827 ± 1.571	430.724 ± 1.322
80s (Fixed Time Interval)	265.113 ± 1.301	92.519 ± 0.398	75.422 ± 1.249	433.054 ± 1.317
Our approach	257.586 ± 0.934	19.512 ± 0.433	60.712 ± 0.918	337.810 ± 1.249

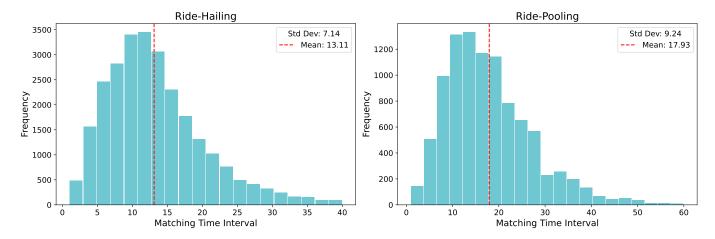


Figure 10. Distribution of Dynamic Time Intervals

D. Adaptability

The experiment is designed to test the adaptability of the RL strategy by analyzing how it adjusts matching intervals dynamically in response to fluctuating supply-demand conditions for ride-haling and ride-pooling services.

Figure 10 shows the distribution of dynamic matching intervals for ride-hailing and ride-pooling systems. In ride-hailing, intervals are mostly concentrated between 10–20 seconds but occasionally extend to 40 seconds. For ride-pooling, intervals are more dispersed and often exceed 30 seconds to accumulate sufficient requests for effective pooling, highlighting the complexity of coordinating multiple passengers.

Figure 11 provides further insights into how matching intervals respond to supply-demand conditions. In ride-hailing (left chart), intervals extend as passenger orders increase, particularly when orders exceed 30. Conversely, intervals shorten when driver supply is limited, prioritizing reduced waiting times. This indicates a preference for longer intervals in balanced supply-demand conditions and shorter ones under tight supply. In ride-pooling (right chart), intervals are generally longer, especially when passenger orders exceed 50, as the system delays matching to maximize pooling efficiency.

However, under imbalanced conditions, shorter intervals are used to address supply-demand gaps promptly.

These observations demonstrate that the proposed approach effectively adjusts matching intervals based on real-time supply and demand conditions. It extends intervals to optimize matching decisions when the system is balanced and uses shorter intervals to handle imbalances and minimize waiting times.

VI. CONCLUSION AND FUTURE WORK

The objective of this study is to leverage reinforcement learning techniques, specifically a dynamic strategy based on the Proximal Policy Optimization (PPO) algorithm, to determine the optimal matching timing in ride-hailing and ride-pooling services. The study introduces a dynamic optimization framework that continuously guides the match-maker on when to perform matching to reduce passenger waiting time. To address sparse rewards, Potential-Based Reward Shaping is employed, accelerating convergence to optimal strategies. This study also addresses the multi-passenger matching complexity in ride-pooling by implementing an algorithm that reduces detour delay, improving passenger satisfaction.

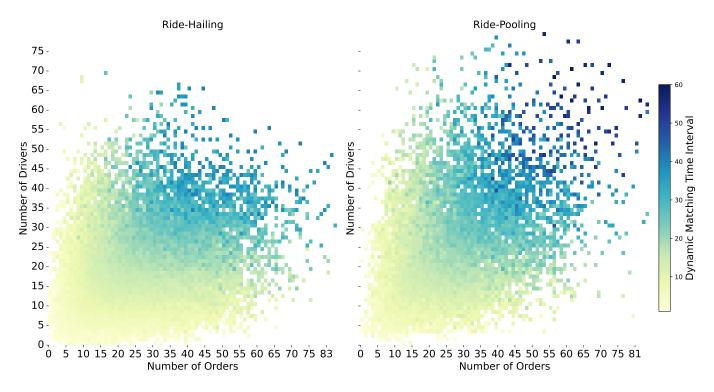


Figure 11. Matching Interval Dynamics with Order and Driver Counts

Through comprehensive simulation experiments, this study systematically validates the effectiveness of the PPO-based dynamic matching strategy in ride-hailing systems. The experimental results reveal three significant advantages: (1) a substantial reduction in passenger waiting time, (2) accelerated learning efficiency through PBRS implementation, and (3) enhanced pick-up and detour management capabilities in both ride-hailing and ride-pooling scenarios. Notably, the proposed strategy demonstrates remarkable robustness under varying supply-demand conditions, consistently outperforming conventional fixed-interval matching approaches. These findings not only confirm the superior flexibility and operational efficiency of the PPO-based method compared to traditional solutions but also provide valuable insights for promoting sustainable development in intelligent transportation systems.

The findings suggest that the developed reinforcement learning framework can significantly enhance urban mobility services by dynamically perform matching at the optimal timing in response to real-time fluctuations in supply and demand, thereby reducing waiting times and detours while improving user satisfaction. This study presents a practical application for intelligent optimization in current platforms like Uber and Lyft, proposing a pathway to improved service efficiency and environmental sustainability.

Future work could incorporate real-time traffic data, simulate hybrid environments, and extend ride-pooling to multiple passengers, better reflecting real-world conditions. Exploring alternative reinforcement learning algorithms and enhancing state representation through graph-based structures or advanced feature engineering may further refine decision-making and scalability, advancing intelligent optimization in urban

mobility.

APPENDIX A PARAMETERS OF PPO

The Proximal Policy Optimization (PPO) algorithm in this project is controlled by a variety of parameters, which allow for flexible configuration and tuning to optimize performance. Below is a description of each parameter and its role in the PPO implementation.

- **exp_name**: The name of this experiment, which defaults to the name of the script.
- **seed**: Sets the random seed for the experiment, ensuring reproducibility. (**Default: 10**)
- torch_deterministic: If enabled, sets torch.backends.cudnn.deterministic = False for deterministic operations in PyTorch. (Default: True)
- **cuda**: If enabled, CUDA (GPU) will be used by default to accelerate computation. (**Default: True**)
- **track**: Enables tracking of the experiment using Weights and Biases. (**Default: True**)
- wandb_project_name: Sets the project name in Weights and Biases. (Default: "cleanRL")
- wandb_entity: Specifies the entity (team) name for the Weights and Biases project. (Default: "baoyiman")
- capture_video: If enabled, videos of the agent's performance are saved to the videos folder. (Default: False)
- env_id: Specifies the environment ID, used to load the appropriate environment (Ride_hailing).
- learning_rate: The learning rate for the optimizer, set at 2.5e-4.

- num_envs: Defines the number of parallel game environments. (Default: 4)
- num_steps: The number of steps taken in each environment per policy rollout. (**Default: 120**)
- anneal_lr: Enables learning rate annealing for both policy and value networks. (Default: True)
- gamma: The discount factor γ for future rewards. (Default: 1)
- gae_lambda: Lambda for Generalized Advantage Estimation (GAE). (Default: 0.95)
- num_minibatches: Defines the number of mini-batches per update. (Default: 8)
- **update_epochs**: Specifies the number of epochs (K) to update the policy. (**Default: 4**)
- norm_adv: Toggles advantage normalization for the PPO update. (Default: True)
- **clip_coef**: Sets the surrogate clipping coefficient to prevent large policy updates. (**Default: 0.2**)
- **clip_vloss**: Enables a clipped loss function for the value function, as specified in the PPO paper. (**Default: True**)
- ent_coef: The coefficient for the entropy term to encourage exploration. (Default: 0.01)
- vf_coef: The coefficient for the value function term. (Default: 0.5)
- max_grad_norm: Sets the maximum gradient norm for gradient clipping. (Default: 1)
- target_kl: The target threshold for the Kullback-Leibler (KL) divergence. If exceeded, the policy update halts. (Default: None)
- batch_size: The batch size, computed at runtime.
- minibatch_size: The mini-batch size, computed at runtime
- **num_iterations**: The number of training iterations, calculated at runtime.

These parameters enable control over the PPO algorithm's behavior, including exploration, stability, and computational efficiency, thereby facilitating effective training of the agent in the specified environment.

APPENDIX B COMPARISON OF SYSTEM STRATEGY PERFORMANCE METRICS

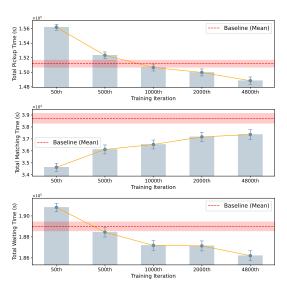


Figure 12. Comparison of System Waiting Time Across Training Episodes and Baseline Strategy for Ride-Hailing Services

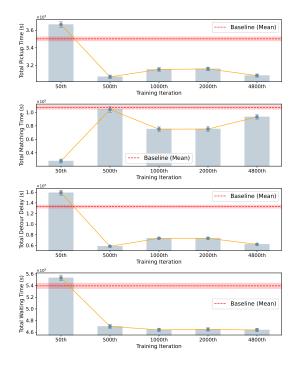


Figure 13. Comparison of System Waiting Time Across Training Episodes and Baseline Strategy for Ride-Pooling Services

REFERENCES

- [1] A. E. Brown and W. LaValle, "Hailing a change: comparing taxi and ridehail service quality in los angeles," *Transportation*, vol. 48, pp. 1007–1031, 2020.
- [2] K. Wei, V. Vaze, and A. Jacquillat, "Transit planning optimization under ride-hailing competition and traffic congestion," *Transp. Sci.*, vol. 56, pp. 725–749, 2021.

- [3] S. Shaheen, "Shared mobility: The potential of ride hailing and pooling," pp. 55–76, 2018.
- [4] J. Ke, H. Yang, and Z. Zheng, "On ride-pooling and traffic congestion," *Transportation Research Part B-methodological*, vol. 142, pp. 213–231, 2020.
- [5] Uber. (2023) How batch matching works. Accessed: October 5, 2024. [Online]. Available: https://www.uber. com/us/en/marketplace/matching/
- [6] M. L. Puterman, Markov decision processes: discrete stochastic dynamic programming. John Wiley & Sons, 2014.
- [7] C. Yan, H. Zhu, N. Korolko, and D. Woodard, "Dynamic pricing and matching in ride-hailing platforms," *Naval Research Logistics (NRL)*, vol. 67, no. 8, pp. 705–724, 2020.
- [8] R. Li, Y. Liu, X. Liu, and Y. M. Nie, "Allocation problem in cross-platform ride-hail integration," *Transportation Research Part B: Methodological*, vol. 188, p. 103056, 2024.
- [9] C. V. Beojone and N. Geroliminis, "Relocation incentives for ride-sourcing drivers with path-oriented revenue forecasting based on a markov chain model," *Transportation Research Part C: Emerging Technologies*, vol. 157, p. 104375, 2023.
- [10] K. Zhang, A. Mittal, S. Djavadian, R. Twumasi-Boakye, and Y. M. Nie, "Ride-hail vehicle routing (river) as a congestion game," *Transportation Research Part B: Methodological*, vol. 177, p. 102819, 2023.
- [11] M. Do, W. Byun, D. K. Shin, and H. Jin, "Factors influencing matching of ride-hailing service using machine learning method," *Sustainability*, vol. 11, no. 20, 2019.
- [12] S. Liu, Z. Wang, Y. Zhang, and H. Yang, "Anomalous ride-hailing driver detection with deep transfer inverse reinforcement learning," *Transportation Research Part C: Emerging Technologies*, vol. 159, p. 104466, 2024.
- [13] J. Shi, X. Li, Y. Aneja, and X. Li, "Ride-matching for the ride-hailing platform with heterogeneous drivers," *Transport Policy*, vol. 136, pp. 169–192, 2023.
- [14] X. Li, H. Normandin-Taillon, C. Wang, and X. Huang, "Bm-rcwtsg: An integrated matching framework for electric vehicle ride-hailing services under stochastic guidance," Sustainable Cities and Society, vol. 108, p. 105485, 2024.
- [15] X. Guo, N. S. Caros, and J. Zhao, "Robust matching-integrated vehicle rebalancing in ride-hailing system with uncertain demand," *Transportation Research Part B: Methodological*, vol. 150, pp. 161–189, 2021.
- [16] H. Yang, X. Qin, J. Ke, and J. Ye, "Optimizing matching time interval and matching radius in on-demand ride-sourcing markets," *Transportation Research Part B: Methodological*, vol. 131, pp. 84–105, 2020.
- [17] T. Chen, Z. Shen, S. Feng, L. Yang, and J. Ke, "Dynamic matching radius decision model for on-demand ride services: A deep multi-task learning approach," *Transportation Research Part E: Logistics and Transportation Review*, vol. 193, p. 103822, 2025.
- [18] B. Shi, Y. Xia, S. Xu, and Y. Luo, "A vehicle value based ride-hailing order matching and dispatching algorithm,"

- Engineering Applications of Artificial Intelligence, vol. 132, p. 107954, 2024.
- [19] N. Agatz, A. Erera, M. Savelsbergh, and X. Wang, "Optimization for dynamic ride-sharing: A review," *European Journal of Operational Research*, vol. 223, no. 2, pp. 295–303, 2012.
- [20] J. Alonso-Mora, S. Samaranayake, A. Wallar, E. Frazzoli, and D. Rus, "On-demand high-capacity ride-sharing via dynamic trip-vehicle assignment," *Proceedings of the National Academy of Sciences*, vol. 114, no. 3, pp. 462– 467, 2017.
- [21] Y. Li and Y. Liu, "Optimizing flexible one-to-two matching in ride-hailing systems with boundedly rational users," *Transportation Research Part E: Logistics and Transportation Review*, vol. 150, p. 102329, 2021.
- [22] X. Chen and X. Di, "A network equilibrium model for integrated shared mobility services with ride-pooling," *Transportation Research Part C: Emerging Technologies*, vol. 167, p. 104837, 2024.
- [23] A. Simonetto, J. Monteil, and C. Gambella, "Real-time city-scale ridesharing via linear assignment problems," *Transportation Research Part C: Emerging Technologies*, vol. 101, pp. 208–232, 2019.
- [24] S. M. Meshkani and B. Farooq, "Centralized and decentralized algorithms for two-to-one matching problem in ridehailing systems," *EURO Journal on Transportation and Logistics*, vol. 12, p. 100106, 2023.
- [25] X. Qin, H. Yang, Y. Wu, and H. Zhu, "Multi-party ride-matching problem in the ride-hailing market with bundled option services," *Transportation Research Part* C: Emerging Technologies, vol. 131, p. 103287, 2021.
- [26] Z. Zhou, C. Roncoli, and C. Sipetas, "Optimal matching for coexisting ride-hailing and ridesharing services considering pricing fairness and user choices," *Transportation Research Part C: Emerging Technologies*, vol. 156, p. 104326, 2023.
- [27] Y. Guo, Y. Zhang, and Y. Boulaksil, "Real-time ride-sharing framework with dynamic timeframe and anticipation-based migration," *European Journal of Operational Research*, vol. 288, no. 3, pp. 810–828, 2021.
- [28] Y. Zhao, L. Tang, Y. Liang, Z. He, and J. Ma, "Optimizing matching for on-demand ride-pooling with stochastic day-to-day dynamics," *ACM Transactions on Knowledge Discovery from Data*.
- [29] C. Yang, X. Wang, Y. Feng, L. Hu, X. Yang, and Z. He, "A prediction-based forward-looking vehicle dispatching strategy for dynamic ride-pooling," *IEEE Transactions on Intelligent Transportation Systems*, 2024.
- [30] S. Qiao, N. Han, J. Huang, Y. Peng, H. Cai, X. Qin, and Z. Lei, "An three-in-one on-demand ride-hailing prediction model based on multi-agent reinforcement learning," *Applied Soft Computing*, vol. 149, p. 110965, 2023.
- [31] H. Zhang, G. Wang, X. Wang, Z. Zhou, C. Zhang, Z. Dong, and Y. Wang, "Nondbrem: nondeterministic offline reinforcement learning for large-scale order dispatching," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 38, no. 1, 2024, pp. 401–409.

- [32] Y. Hu, S. Feng, and S. Li, "Bmg-q: Localized bipartite match graph attention q-learning for ride-pooling order dispatch," *arXiv* preprint arXiv:2501.13448, 2025.
- [33] J. Ke, F. Xiao, H. Yang, and J. Ye, "Learning to delay in ride-sourcing systems: A multi-agent deep reinforcement learning framework," *IEEE Transactions on Knowledge* and Data Engineering, vol. 34, no. 5, pp. 2280–2292, 2020.
- [34] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, "Proximal policy optimization algorithms," *arXiv preprint arXiv:1707.06347v2*, 2017.
- [35] M. L. Puterman, *Markov Decision Processes: Discrete Stochastic Dynamic Programming*, 1st ed. New York, NY, USA: John Wiley & Sons, Inc., 1994.
- [36] L. P. Kaelbling, M. L. Littman, and A. R. Cassandra, "Planning and acting in partially observable stochastic domains," *Artificial Intelligence*, vol. 101, no. 1-2, pp. 99–134, 1998.
- [37] D. E. Rumelhart, G. E. Hinton, and R. J. Williams, "Learning representations by back-propagating errors," *nature*, vol. 323, no. 6088, pp. 533–536, 1986.
- [38] S. Hochreiter and J. Schmidhuber, "Long Short-Term Memory," *Neural Computation*, vol. 9, no. 8, pp. 1735–1780, 1997.
- [39] M. Hausknecht and P. Stone, "Deep Recurrent Q-Learning for Partially Observable MDPs," *arXiv preprint arXiv:1507.06527*, 2017.
- [40] A. Y. Ng, D. Harada, and S. Russell, "Policy invariance under reward transformations: Theory and application to reward shaping," in *Icml*, vol. 99, 1999, pp. 278–287.
- [41] S. Devlin and D. Kudenko, "Theoretical considerations of potential-based reward shaping for multi-agent systems," in *Tenth International Conference on Autonomous Agents* and Multi-Agent Systems. ACM, 2011, pp. 225–232.
- [42] S. Ibrahim, M. Mostafa, A. Jnadi, H. Salloum, and P. Osinenko, "Comprehensive overview of reward engineering and shaping in advancing reinforcement learning applications," *IEEE Access*, 2024.
- [43] M. İ. Bal, "Reward shaping for efficient exploration and acceleration of learning in reinforcement learning," Master's thesis, Middle East Technical University, 2022.
- [44] R. J. Williams, "Simple statistical gradient-following algorithms for connectionist reinforcement learning," *Machine learning*, vol. 8, pp. 229–256, 1992.
- [45] V. Mnih, "Asynchronous methods for deep reinforcement learning," *arXiv preprint arXiv:1602.01783*, 2016.
- [46] R. S. Sutton, D. McAllester, S. Singh, and Y. Mansour, "Policy gradient methods for reinforcement learning with function approximation," *Advances in neural information* processing systems, vol. 12, 1999.
- [47] N. Y. C. Taxi and L. C. (TLC), "Nyc tlc trip record data," https://www.nyc.gov/site/tlc/about/tlc-trip-record-data. page, accessed: 2024-12-09.

Yiman Bao

PLACE PHOTO HERE

Jie Gao

Jinke He