Transportation Research Part C 187 (2026) 105644

Contents lists available at ScienceDirect

Transportation Research Part C

journal homepage: www.elsevier.com/locate/trc

A timely match for ride-hailing and ride-pooling services using a deep reinforcement learning approach

Yiman Bao [a], Jie Gao [a,*], Jinke He [b], Frans A. Oliehoek [b], Oded Cats [a]

[a] Delft University of Technology, Department of Transport & Planning, Delft, 2628 CN, The Netherlands
[b] Delft University of Technology, Department of Intelligent Systems, Delft, 2628 CD, The Netherlands

ARTICLE INFO

Keywords:
Deep reinforcement learning
Ride-hailing
Ride-pooling
Matching timing
Reward shaping
Proximal policy optimization

ABSTRACT

Efficient matching in ride-hailing and ride-pooling services depends not only on how matches are constructed, but also on when the platform triggers a matching operation. Many systems use batched matching with a fixed time interval to accumulate requests before matching, which increases the candidate set but cannot adapt to real time supply-demand fluctuations and may induce unnecessary waiting. This paper proposes a reinforcement learning approach that learns when to trigger matching based on current system conditions. We formulate the timing problem as a finite-horizon Markov decision process and train the policy using the Proximal Policy Optimization algorithm. To address sparse and delayed feedback, we introduce a finite-horizon, potential-based reward shaping scheme that preserves the optimal policy while densifying the learning signal; the same framework applies to both ride-hailing and ride-pooling, where detour delay is incorporated into the reward for pooling. Using a data-driven simulator calibrated on NYC trip records, the learned policy adapts matching timing decisions to the current state of waiting requests and available drivers and outperforms fixed-interval, rule-based dynamic, and first-dispatch baselines. It reduces total waiting time by 3.1% in ride-hailing and 20.1% in ride-pooling, and detour delay by 36.1% in pooling, while maintaining short matching times.

1. Introduction

Ride-hailing services, such as Uber[1] and Lyft,[2] have become an integral part of urban transportation, offering passengers flexible, on-demand mobility while optimizing vehicle dispatch to alleviate congestion and reduce emissions (Brown and LaValle, 2020; Wei et al., 2021). 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 (Shaheen, 2018; Ke et al., 2020b). However, as these services expand, the matching process becomes increasingly complex due to dynamic and uncertain supply and demand conditions.

To improve matching efficiency, ride-hailing platforms typically employ two primary strategies. One approach is first dispatch, a real-time matching strategy where ride requests are immediately assigned to available drivers upon arrival. This method prioritizes instant allocation, ensuring that each request is processed as soon as both the request and a suitable driver are available, without delay. While this minimizes response time, it often leads to suboptimal matches, as better driver-passenger pairings could emerge if

* Corresponding author.
E-mail address: J.Gao-1@tudelft.nl (J. Gao).

[1] https://www.uber.com/nl/en/
[2] https://www.lyft.com/

https://doi.org/10.1016/j.trc.2026.105644
Received 25 September 2025; Received in revised form 26 January 2026; Accepted 15 March 2026
Available online 27 March 2026

requests were bundled 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 (Uber, 2023). By increasing the pool of potential matches, using this strategy 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-interval batching strategies can lead to unnecessary passenger delays or missed opportunities for more efficient assignments. These limitations point to a key missing decision: how long the platform should wait before executing the next matching operation.

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 real-time system conditions is needed. Reinforcement Learning (RL), a framework for sequential decision-making under uncertainty, is well-suited for this problem. By modeling ride-matching as a Markov Decision Process (MDP), 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 timing strategy that determines when to initiate matches based on real-time system conditions. Unlike fixed-interval batching, our approach continuously monitors supply-demand variations and optimizes timing to minimize passenger wait times, which includes both the time to be assigned a driver and the time until the driver arrives. Moreover, in the MDP formulation, the reward associated with preceding waiting actions can only be observed after a matching decision has been executed, which inherently leads to sparse rewards. 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. In this formulation, the RL agent operates at the level of matching timing and treats the underlying ride-hailing and ride-pooling assignment algorithm as a modular component; the same MDP and learning scheme can, in principle, be combined with more advanced rolling-horizon or insertion-based pooling algorithms. The main contributions of this work are as follows:

- We formulate the matching timing decision as a Markov Decision Process (MDP), where the key decision is when to execute a matching operation. This differs from existing MDP-based dispatch methods, which often rely on fixed-interval or event-driven matching. In our formulation, the match-maker observes the current system state, such as the number of idle drivers, unserved passengers, and other related information, and decides whether to perform matching at that moment. When matching is executed, a fixed matching algorithm is applied to the current pool of requests and idle drivers. This enables the system to adaptively control the timing of matching based on real-time demand and supply conditions.
- We design an RL framework based on the formulated MDP, including a sparse reward structure reflecting the system-level impact of each timing decision. To facilitate learning under this sparse feedback, we apply Potential-Based Reward Shaping (PBRS), adapting it to our matching in ride-hailing and ride-pooling settings. We demonstrate its practical value in improving learning efficiency and policy quality in this domain-specific context, where conventional reward signals are delayed and sparse.
- We develop an efficient and realistic simulator to model the matching process of ride-hailing and ride-pooling. Built on real-world data, the simulator includes functionalities such as request generation and spatial matching, and is used to support the training and evaluation of our RL-based approach. We employ Proximal Policy Optimization (PPO) and investigate the impact of applying Potential-Based Reward Shaping (PBRS) on the training process. We also examine how the match-maker's decision-making evolves across different training phases, providing insight into how it learns to navigate trade-offs and gradually converges to an optimal strategy.
- We compare the RL-trained strategies with the first-dispatch strategy, batched matching strategies using various time intervals and the rule-based dynamic matching strategy. Our approach consistently achieves better performance across key metrics, including passenger waiting time and detour delay. We also analyze the adaptability of the learned policy, showing that it adjusts its matching decisions in response to fluctuations in supply and demand.

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 in ride-hailing and ride-pooling systems. Section 4 introduces the designed simulator based on real-world datasets. Section 5 conducts extensive numerical experiments. Finally, we conclude this study and outline our future work in Section 6.

2. Literature review

2.1. Matching in ride-hailing systems

In ride-hailing systems, the central task is to allocate available drivers to passenger requests in a timely and efficient manner. Early research commonly formulates this task as a static optimization problem, such as bipartite graph matching, where the objective is to maximize the weighted sum of feasible assignments (Yan et al., 2020). While such approaches capture the quality of matches, they assume static conditions and do not reflect the highly dynamic nature of real operations.

To address temporal variability, more recent studies employ dynamic models. For example, Beojone and Geroliminis (2023) developed a mixed continuous–discrete Markov Chain to capture driver mobility and earnings, while Zhang et al. (2023) formulated idle driver routing as a Markov Decision Process (MDP) to improve utilization by anticipating future requests. These studies highlight the importance of incorporating system dynamics into dispatch strategies.

Another research strand investigates heterogeneity in driver behavior. Empirical studies show that drivers differ in patience and acceptance decisions: some may reject requests due to personal preferences or expected earnings (Do et al., 2019), while others exhibit anomalous patterns detectable via inverse reinforcement learning (Liu et al., 2024a). Part-time drivers, in particular, tend to abandon requests more readily, prompting policies that prioritize them to reduce cancellations (Shi et al., 2023). Moreover, integrated frameworks such as the Matching-Integrated Vehicle Rebalancing (MIVR) model jointly optimize matching and repositioning to enhance fleet efficiency (Guo et al., 2021a).

In addition to spatial assignment, matching outcomes are strongly influenced by operational parameters such as radius and timing. A shorter radius reduces pick-up distance but limits the feasible pool of requests, whereas a longer batching interval increases match opportunities but prolongs waiting (Yang et al., 2020). Attempts to balance these trade-offs include block-based partitioning of service areas (Feng et al., 2022). However, most of these strategies remain rule-based and cannot flexibly adapt to real-time demand fluctuations, which motivates more adaptive solutions. Along this line, Chen et al. (2025) develop a deep multi-task learning framework for broadcasting-mode services that predicts system performance under candidate matching radii and selects, in real time, the radius maximizing a composite objective, yielding sizable gains over fixed or heuristic radii.

2.2. Matching in ride-pooling systems

Ride-pooling extends the ride-hailing problem by allowing multiple passengers with overlapping routes to share a vehicle. This adds complexity, as it requires not only driver–passenger assignment but also passenger–passenger compatibility and sequencing of pick-ups and drop-offs. Agatz et al. (2012) categorize such services into single-driver–single-passenger, single-driver–multi-passenger, and more complex variants, with ride-pooling corresponding to the multi-passenger case.

Optimization-based methods dominate early research. For instance, Alonso-Mora et al. (2017) proposed an anytime optimal algorithm that balances fleet capacity, waiting, and detours through iterative constrained optimization. Li and Liu (2021) introduced a rolling-horizon model for small-capacity pooling, integrating incentives to encourage adoption. Similarly, Kucharski and Cats (2020) embedded passenger utilities into the objective to generate mutually attractive shared rides. These approaches improve assignment quality but face computational challenges at scale. To enhance computational efficiency, several works focus on algorithmic acceleration. Simonetto et al. (2019) reformulated pooling as a linear assignment problem, applying federated optimization to improve scalability. Meshkani and Farooq (2023) proposed a decentralized vehicle-to-infrastructure architecture, achieving significant gains in computational speed. Beyond assignment efficiency, service quality constraints such as detour and waiting time have been recognized as fundamental design elements in ride-pooling systems. Ouyang et al. (2021) analyzed reservation-based carpooling under explicit detour and waiting time restrictions, providing theoretical insights into the trade-offs between pooling efficiency and passenger inconvenience. Such results highlight that pooling performance is tightly coupled with how temporal and spatial flexibility is constrained.

More recent contributions seek to unify ride-hailing and pooling operations and to support more flexible pooling configurations. For example, Qin et al. (2021b) proposed a multi-objective integer programming framework for joint optimization across pooled, non-pooled, and hybrid services, while Zhou et al. (2023) introduced a decision framework balancing monetary cost and user experience. Liu et al. (2024b) investigated dynamic ridesharing by jointly optimizing order dispatching and vehicle repositioning, demonstrating the importance of coordinating short-term matching decisions with long-term supply–demand balance. In addition, recent studies extended ride-pooling into high occupancy transit-like attractive shared rides (Kucharski and Cats, 2024) and on-demand feeder services with more than two riders (Fan et al., 2025), illustrating the growing interest in multi-passenger and adaptive pooling mechanisms.

Despite these advances, the timing of matching remains a critical determinant of efficiency. Longer batching windows accumulate more compatible passengers and reduce detour distances, but at the cost of extended waiting times. Conversely, shorter windows reduce delays but sacrifice pooling opportunities. Dynamic time-window approaches (Guo et al., 2021b) partially address this issue, yet still rely heavily on historical patterns and lack robustness to real-time uncertainty. This indicates a gap for methods that can learn adaptive timing policies directly from evolving system conditions.

2.3. Reinforcement learning for ride-hailing and ride-pooling

Reinforcement learning (RL) provides a natural framework for sequential decision-making in dynamic ride-hailing and ride-pooling environments. By interacting with the system and updating policies over time, RL can adapt to fluctuating demand and supply while maintaining scalability across different urban contexts.

Several applications have demonstrated its potential. Qiao et al. (2023) proposed a multi-agent RL method (ERPM) for demand prediction and dispatching, while Mao et al. (2020) designed a deep RL framework for real-time taxi reallocation, achieving near-optimal performance under stochastic dynamics. Other work has considered passenger-side decisions: Ke et al. (2020a) modeled whether passengers should delay matching, using multi-agent RL to exploit continuous arrivals and mitigate sparse rewards. While effective at the individual level, such approaches may yield locally optimal solutions and generally neglect ride-pooling. Complementing these directions, Liu et al. (2025) learn an on-demand, adaptable matching radius via RL—applicable to both instant and batch modes—demonstrating improvements in revenue and waiting-time outcomes on real data. Similarly, Gao et al. (2025) explicitly formulate the matching radius as a decision variable and propose a deep reinforcement learning framework with a dual-replay-buffer mechanism to dynamically adjust the matching radius in response to spatiotemporal supply–demand variations. Their results show that adaptive radius control can outperform fixed-radius baselines in terms of matching rate, pick-up distance, and driver utilization.

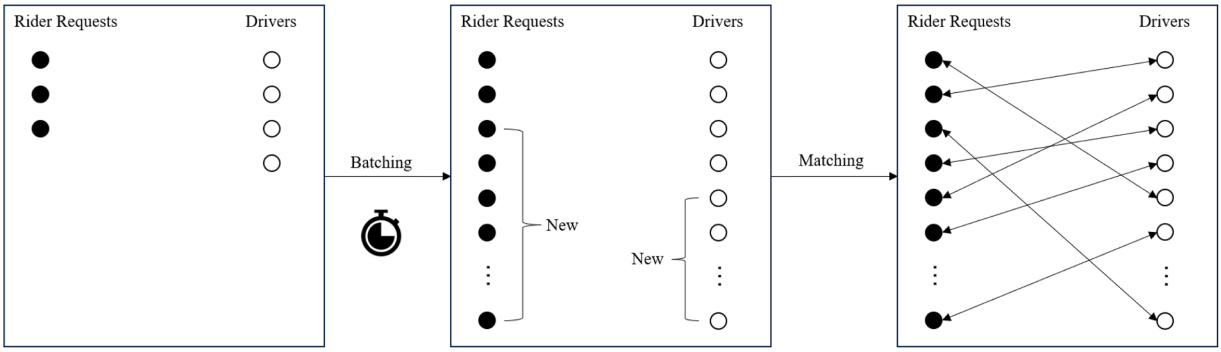


Fig. 1. Batched Matching Demonstration. rider requests and available drivers are collected within a batching window. At the end of the window, the platform executes a matching algorithm. Unmatched requests are carried over to the next window.

A closely related contribution is the work of Qin et al. (2021a), who explicitly examined the problem of optimizing matching time intervals in ride-hailing using policy gradient methods. Their study showed that adaptively delaying matching can significantly improve system efficiency compared to fixed-interval batching, and they were among the first to highlight the sparse reward issue in this setting. However, their reward shaping design modifies the total return, thereby changing the optimal policy in theory. This limitation motivates the need for shaping techniques that provide denser learning signals while preserving policy optimality.

Overall, existing RL-based studies focus primarily on ride-hailing and dispatching, with limited exploration of ride-pooling and, critically, the timing of matches. Moreover, many strategies optimize individual rather than system-wide outcomes, or adopt reward designs that undermine theoretical guarantees. These gaps motivate the present study, which formulates the timing of both ride-hailing and ride-pooling as a single-agent RL problem. By optimizing the initiation of matching decisions under real-time conditions, and by applying a potential-based reward shaping (PBRS) method that provably preserves the optimal policy, our approach achieves globally efficient outcomes that outperform fixed-interval or heuristic-based strategies.

3. 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 decision-making problem under the reinforcement learning (RL) framework. Lastly, we describe how to solve this optimization problem.

3.1. Batched matching

Batched matching assigns available drivers to rider requests collected within a batching window, aiming to optimize a platform objective (e.g., maximize served requests, minimize pickup/waiting time, or a weighted service metric) subject to feasibility constraints such as capacity, and pickup distance/time limits. As illustrated in Fig. 1, rider requests and available drivers are accumulated during a pre-defined window. At the window's end, the platform solves the matching problem and dispatches the assignments; any unmatched requests are carried over to the next window and resolved later.

Formally, we divide the day into a sequence of time windows $\{T_0, T_1, \dots, T_n, \dots\}$, where each window is defined as $T_n = [t_{n-1}, t_n)$ and has fixed length $\Delta T = t_n - t_{n-1}$. Rather than executing the matching algorithm at every boundary t_n, the platform can *dynamically* decide when to trigger matching based on current system conditions, such as request numbers, driver availability, and their distributions. The goal is to minimize total system-wide waiting time by deciding when to match in response to real-time demand and supply dynamics. In the following, we model it as a sequential decision making problem.

3.2. MDP modeling

The problem of deciding when to execute a matching operation in ride-hailing and ride-pooling services can be naturally formulated as a finite-horizon Markov Decision Process (MDP) (Bellman, 1957; Puterman, 1994). This is because the system evolves sequentially over discrete time steps under uncertainty: the platform observes the current supply-demand status, makes a decision on whether to trigger matching, and then receives feedback in the form of passenger waiting times or detour delays. Such sequential decision-making under stochastic dynamics is well aligned with the MDP framework.

Formally, a finite-horizon MDP is defined as a 5-tuple $(\mathcal{S}, \mathcal{A}, P, R, T)$, where $\mathcal{S}$ denotes the state space, $\mathcal{A}$ the action space, P the transition dynamics, R the reward function, and T the horizon length. At each time step $t = 0, \dots, T-1$, the match-maker observes the current state $s_t \in \mathcal{S}$, chooses an action $a_t \in \mathcal{A}$, and receives an immediate reward $R(s_t, a_t, s_{t+1})$ after transitioning to the next state s_{t+1}. The optimization objective is to maximize the expected cumulative reward (total return) over the horizon:

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

This formulation captures the sequential and dynamic nature of the matching timing problem, providing a principled foundation for applying reinforcement learning methods. The specific designs of the state, action, and reward components are detailed in the following subsections.

State: The state is defined as the information available to the platform at each decision epoch. At time t, the system state is defined as

$$s_t = \left(\tau_t,\ OD_t,\ D_t,\ \overline{W}_p(t),\ W_{\max}(t)\right),$$

where τ_t denotes the current time of day. We represent the service region using a finite set of discrete GPS points $\mathcal{L}$. $OD_t = \{OD_t(o,d) \mid o,d \in \mathcal{L}\}$ represents the origin–destination (OD) demand profile of unmatched passenger requests, with each element $OD_t(o,d)$ denoting the number of unmatched requests whose pickup and drop-off locations are at spatial points o and d, respectively, $D_t = \{D_t(\ell) \mid \ell \in \mathcal{L}\}$ denotes the spatial distribution of available drivers over the service region, where $D_t(\ell)$ represents the number of idle drivers located at point ℓ, and $\overline{W}_p(t)$ and $W_{\max}(t)$ denote the mean and maximum waiting times of unmatched passengers, respectively. Vehicles that are already serving passengers (including vehicles with passengers on board and remaining capacity) are not included in $D_t(\ell)$ and are not considered candidates for matching. Modeling rolling insertions would require extending the simulator and enriching the state with additional vehicle status information (e.g., remaining capacity and route commitments), which is a potential subject for future work.

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 matching algorithm to assign drivers to passengers. Importantly, the match-maker does not choose individual driver–passenger assignments, passenger pairing, routing, or pricing; these internal decisions are handled by the fixed matching procedure whenever $a_t = 1$. Thus, the MDP is above the matching layer: it learns when to call the matching algorithm, not how the matching algorithm is designed.

Reward: 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:

$$R^H(s_t, a_t, s_{t+1}) = -(\phi R_m(s_t, a_t, s_{t+1}) + R_w(s_t, a_t, s_{t+1})) \tag{2}$$

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:

$$R_w(s_t, a_t, s_{t+1}) = \begin{cases} 0, & a_t = 0 \\ f_{pick}(s_t), & a_t = 1 \end{cases} \tag{3}$$

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 $R_d(s_t, a_t, s_{t+1})$:

$$R^P(s_t, a_t, s_{t+1}) = -\left(\phi R_m(s_t, a_t, s_{t+1}) + \tau R_d(s_t, a_t, s_{t+1}) + R_w(s_t, a_t, s_{t+1})\right) \tag{4}$$

where τ is 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:

$$R_d(s_t, a_t, s_{t+1}) = \begin{cases} 0, & a_t = 0 \\ f_{detour}(s_t), & a_t = 1 \end{cases} \tag{5}$$

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. Because $R_m(s_t, a_t, s_{t+1}) = \Delta t N_p^{unmatch}(s_t, a_t)$ keeps increasing whenever requests remain unmatched, repeatedly choosing $a_t = 0$ (never triggering matching) leads to poor returns due to the continuous accumulation of system-wide waiting costs. The timing policy therefore faces a trade-off: postponing matching may improve assignment quality by leveraging a larger candidate set (e.g., reducing pickup and detour costs), but it simultaneously increases accumulated passenger waiting cost. The optimal policy balances these effects to minimize total system cost.

3.3. Reward sparsity and potential-based reward shaping

Although the reward functions R^H and R^P for ride-hailing 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

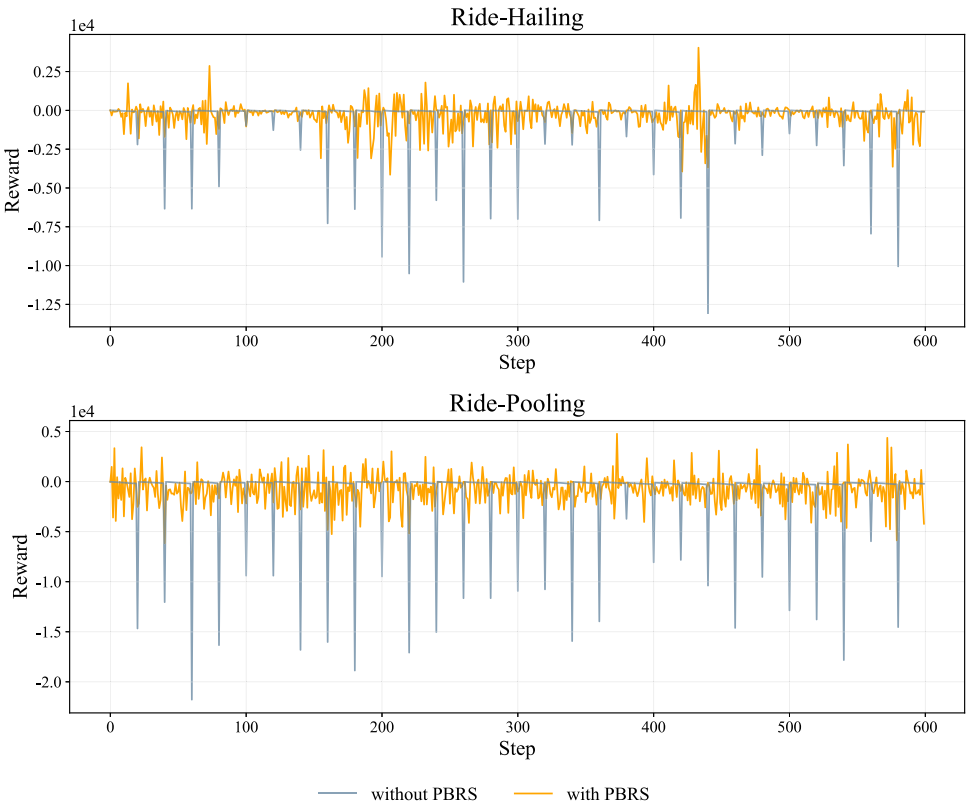


Fig. 2. Comparison of rewards without and with PBRS.

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 $a_t = 0$, while the magnitude of $R_m(s_t, a_t, s_{t+1})$ is relatively small compared to the other two rewards. As a result, the match-maker receives limited guidance during training, making it difficult to learn an effective policy. This phenomenon is manifested, as shown by the gray polyline in Fig. 2.

We address reward sparsity by *designing a problem-specific shaping scheme* based on potential-based reward shaping (PBRS) (Ng et al., 1999). Specifically, PBRS modifies the reward function of RL with a potential function $\Phi : S \to \mathbb{R}$ intending to provide a denser learning signal for faster convergence. The modified reward function is defined as follows in finite-horizon MDPs:

$$R'(s_t, a_t, s_{t+1}) = \begin{cases} R(s_t, a_t, s_{t+1}) + \Phi(s_{t+1}) - \Phi(s_t) & t \neq T-1, \\ R(s_{T-1}, a_{T-1}, s_T) - \Phi(s_{T-1}) + \Phi(s_0), & t = T-1. \end{cases} \tag{6}$$

where $\Phi(s_t)$ reflects the desirability of state s_t. After applying the shaped reward, the total return G' becomes:

$$G' = \mathbb{E}\left[\sum_{t=0}^{T-1} R'(s_t, a_t, s_{t+1})\right]. \tag{7}$$

Importantly, as shown in Eq. (6), 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 (as shown in Eq. (1)) for all policies (for detailed calculations, see Proposition (1)). As a result, PBRS preserves the optimal policy. The proof is provided below.

Proposition 1 (PBRS).

Let $T \in \mathbb{N}$ and define R' by Eq. (6). For any policy π and any trajectory $(s_0, a_0, \ldots, s_T)$ generated by π,

$$\sum_{t=0}^{T-1} R'(s_t, a_t, s_{t+1}) = \sum_{t=0}^{T-1} R(s_t, a_t, s_{t+1}).$$

Consequently, the expected episodic return under R' equals that under R, and all optimal policies are preserved.

Proof. By Eq. (6),

$$\sum_{t=0}^{T-1} R'(s_t, a_t, s_{t+1}) = \sum_{t=0}^{T-2} \big(R(s_t, a_t, s_{t+1}) + \Phi(s_{t+1}) - \Phi(s_t)\big) + R(s_{T-1}, a_{T-1}, s_T) - \Phi(s_{T-1}) + \Phi(s_0)$$

$$= \sum_{t=0}^{T-2} R(s_t, a_t, s_{t+1}) + \underbrace{\Big(\sum_{t=0}^{T-2} \Phi(s_{t+1}) - \sum_{t=0}^{T-2} \Phi(s_t) \Big)}_{= \Phi(s_{T-1}) - \Phi(s_0)} + R(s_{T-1}, a_{T-1}, s_T) - \Phi(s_{T-1}) + \Phi(s_0)$$

$$= \sum_{t=0}^{T-1} R(s_t, a_t, s_{t+1}).$$

Taking expectations under any policy π yields $G' = G$. □

The proposition above implies that shaping with Eq. (6) does not change the expected episodic return under any policy; hence optimal policies are preserved. Prior work further shows that, with an appropriate potential, PBRS can markedly accelerate learning in sparse-reward problems (Ng et al., 1999; Devlin and Kudenko, 2011; Ibrahim et al., 2024; Bal, 2022). Based on this, we design a problem-specific potential that redistributes feedback across both actions.

Let $f_{\text{pick}}(s_t)$ and $f_{\text{detour}}(s_t)$ denote the aggregate pickup time and aggregate detour used in R_w and R_d. In the base reward, these terms are evaluated only when $a_t = 1$; within the *potential*, they are evaluated at every step ($a_t \in \{0, 1\}$), yielding a measure of the current state aligned with the objective (minimizing waiting and, for pooling, detour), thus accurately reflecting the overall system potential at each time step. We define

$$\Phi^h(s_t) = -f_{\text{pick}}(s_t), \tag{8}$$

$$\Phi^p(s_t) = -\big(f_{\text{pick}}(s_t) + f_{\text{detour}}(s_t)\big). \tag{9}$$

Substituted into Eq. (6), the shaping increment becomes $-\big(f(s_{t+1}) - f(s_t)\big)$ with $f = f_{\text{pick}}$ (ride-hailing) or $f_{\text{pick}}+f_{\text{detour}}$ (ride-pooling), providing informative signal even when $a_t = 0$ and thus alleviating sparsity without altering the optimal policy.

Fig. 2 examines the distribution of rewards without (Eqs. 2, 4) and with PBRS (Eqs. 6, 8, 9) for ride-hailing and ride-pooling services. 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 ride-hailing and ride-pooling systems.

3.4. Learning algorithm

To learn the matching timing policy in both ride-hailing and ride-pooling settings, we adopt the Proximal Policy Optimization (PPO) algorithm (Schulman et al., 2017). PPO offers a favorable balance between policy improvement stability and sample efficiency, making it well suited for dynamic environments characterized by fluctuating supply and demand. In our case, the decision is binary, whether to trigger a matching operation at each decision point, and PPO enables the policy to adapt to complex, state-dependent conditions that affect system efficiency.

PPO optimizes a surrogate objective based on clipped policy ratios to ensure conservative policy updates. Let π_θ denote the current policy and $\pi_{\theta_{\text{old}}}$ the policy before the update. The clipped objective is defined as:

$$L^{\text{CLIP}}(\theta) = \mathbb{E}_t\big[\min\big(r_t(\theta)\hat{A}_t,\ \text{clip}\big(r_t(\theta), 1-\epsilon, 1+\epsilon\big)\hat{A}_t\big)\big], \tag{10}$$

where $r_t(\theta) = \frac{\pi_\theta(a_t|s_t)}{\pi_{\theta_{\text{old}}}(a_t|s_t)}$ is the probability ratio, $\hat{A}_t$ is the advantage estimate at time t, and ϵ is a clipping threshold that restricts large policy updates. This formulation stabilizes training while maintaining sufficient exploration.

To further improve learning, we incorporate a learned state-value function $V_\phi(s_t)$ to reduce variance in the advantage estimates. The overall loss function combines the clipped policy loss, value loss, and an entropy bonus to encourage exploration:

$$L(\theta) = L^{\text{CLIP}}(\theta) - c_1 L^{\text{VALUE}}(\theta) + c_2 L^{\text{ENTROPY}}(\theta), \tag{11}$$

where $L^{\text{VALUE}}(\theta) = \big(V_\phi(s_t) - V_t^{\text{target}}\big)^2$ is the squared error between the predicted and target state values, and L^{ENTROPY} is the entropy of the policy distribution. The coefficients c_1 and c_2 control the contribution of the value and entropy terms, respectively.

Pseudocode 1 and Fig. 3 summarize the PPO algorithm's structure. The policy is implemented using an actor–critic architecture with separate networks for the actor and critic. Each network consists of three fully connected layers with 64 hidden units per layer and Tanh activations. The actor outputs the Bernoulli probability of triggering a match, while the critic estimates the value of the current state. Training is performed using batches of trajectories collected from parallel simulations, where each episode proceeds in discrete decision steps. When the policy chooses to trigger ($a_t = 1$), the simulator executes the matching algorithm and updates the state accordingly; otherwise, the system evolves passively, reflecting new arrivals and departures. The reward at each step is shaped using the potential-based formulation, which provides dense feedback aligned with the platform's operational objectives.

4. Simulator design

To validate and evaluate the proposed approach, we develop a simulator to model the dynamics of ride-hailing and ride-pooling services. The simulator trains the agent through the reinforcement learning framework and the PPO algorithm described above, and

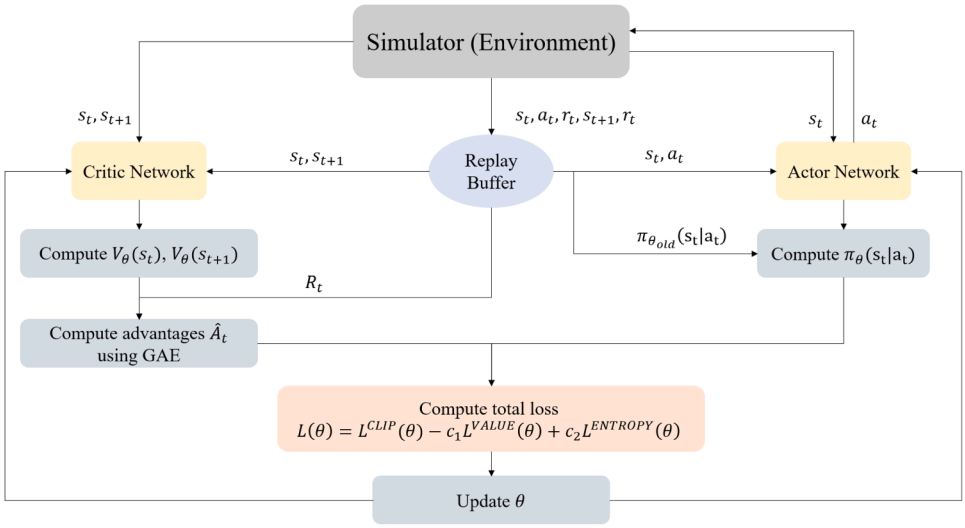


Fig. 3. Algorithm structure of PPO.

Algorithm 1 PPO algorithm.

for episode = 1 to E **do**
  Reset the environment and get initial state s_0
  Initialize episode return $Return = 0$ and $done = False$
  for $t = 0$ to $T - 1$ **do**
    Sample action a_t from policy $\pi_\theta(a_t|s_t)$
    Execute action a_t in environment
    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
    if $done$ **then**
      break
    end if
  end for
  Compute discounted returns R_t and advantages $\hat{A}_t$ using GAE
  for $k = 1$ to K **do**
    Sample minibatch from trajectory buffer
    Compute importance ratio $r_t(\theta) = \frac{\pi_\theta(a_t|s_t)}{\pi_{\theta_{\text{old}}}(a_t|s_t)}$
    Compute clipped surrogate objective:
    $L^{\text{CLIP}}(\theta) = \mathbb{E}_t\left[\min\left(r_t(\theta)\hat{A}_t, \text{clip}(r_t(\theta), 1 \pm \epsilon)\hat{A}_t\right)\right]$
    Compute value function loss:
    $L^{\text{VALUE}}(\theta) = \left(V_\theta(s_t) - V_t^{\text{target}}\right)^2$
    Compute entropy bonus:
    $L^{\text{ENTROPY}}(\theta) = \mathbb{E}_t\left[-\pi_\theta(a_t|s_t)\log\pi_\theta(a_t|s_t)\right]$
    Compute total loss:
    $L(\theta) = L^{\text{CLIP}}(\theta) - c_1 L^{\text{VALUE}}(\theta) + c_2 L^{\text{ENTROPY}}(\theta)$
    Update policy and value function using gradient ascent on $L(\theta)$
  end for
end for

the overall workflow is illustrated in Fig. 4. The simulator utilizes a real public dataset of taxi trips in Manhattan, New York City (Taxi and Commission, 2024), 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 min before repositioning. Additionally, pooling is limited to two passenger orders per vehicle.

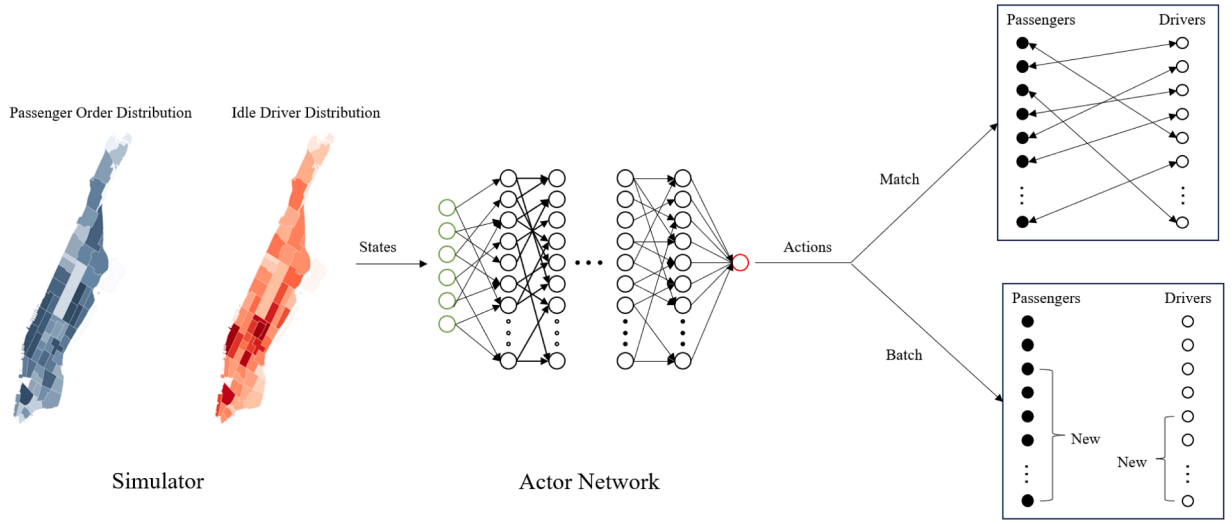


Fig. 4. Interaction between simulator and actor network.

4.1. Passenger and driver data generation

The simulator leverages historical for-hire vehicle (FHV) trip records provided by the New York City government (Taxi and Commission, 2024), 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. The simulated passenger request data and idle driver data closely match 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.

4.2. 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. Please note that these matching procedures serve as the fixed matching module throughout this study. Our reinforcement learning framework operates on top of this module by deciding when to execute a matching, and is agnostic to the internal structure of the pooling algorithm. In particular, the pooling implementation adopts a batch-based two-stage abstraction with at most two passengers per vehicle. This is a simplification compared with rolling or insertion-based pooling in operational systems which is needed to provide a tractable and consistent setting for analyzing the impact of matching-timing decisions.

4.2.1. 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:

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

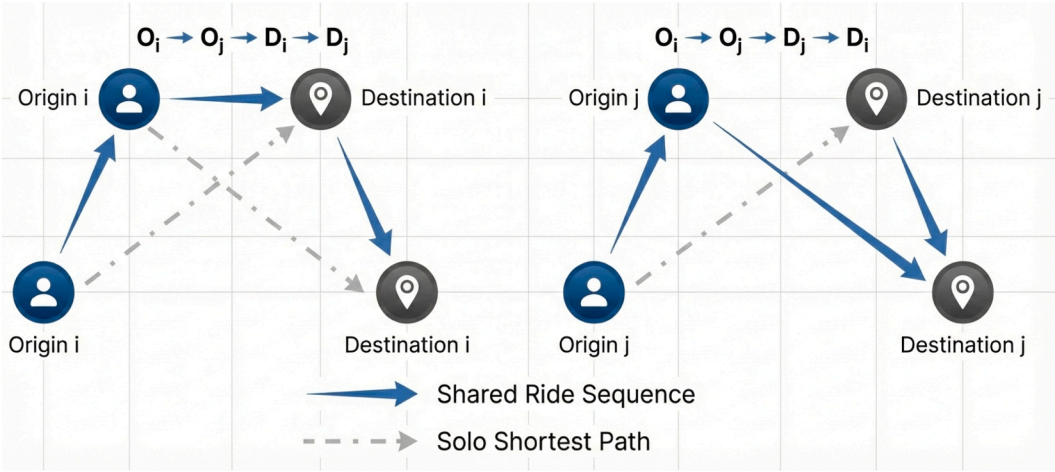


Fig. 5. Possible pickup and drop-off sequences for two passenger orders.

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.

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 Fig. 5, 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:

$$DDR^a(i) = \frac{d(O_i, D_i)}{d(O_i, O_j, D_i)} \tag{12}$$

$$DDR^a(j) = \frac{d(O_j, D_j)}{d(O_j, D_i, D_j)} \tag{13}$$

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:

$$DDR^a(i,j) = \min\left(\frac{d(O_j, D_j)}{d(O_j, D_i, D_j)}, \frac{d(O_j, D_j)}{d(O_j, D_i, D_j)}\right) \tag{14}$$

In case (b), the situation is more straightforward. As shown in Fig. 5, 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:

$$DDR^b(i,j) = \frac{d(O_i, D_i)}{d(O_i, O_j, D_j, D_i)} \tag{15}$$

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:

$$\mathrm{DDR}(i,j) = \max\left(\min\left(\frac{d(O_i, D_i)}{d(O_i, O_j, D_i)}, \frac{d(O_j, D_j)}{d(O_j, D_i, D_j)}\right), \frac{d(O_i, D_i)}{d(O_i, O_j, D_j, D_i)}\right) \tag{16}$$

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:

$$\Delta d_i = d(O_i, \dots, D_i \mid \sigma^*) - d(O_i, D_i) \tag{17}$$

where $d(O_i, D_i)$ is the shortest path distance for passenger i if served independently; $d(O_i, \dots, D_i \mid \sigma^*)$ is the actual travel distance from O_i to D_i under the optimal pickup and drop-off sequence σ^*. Therefore, 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 Eqs. (16) and (17).
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.

   Let $\delta(p)$ denote the set of edges incident to passenger order p in the graph, i.e., all feasible passenger–passenger pairs involving p. The matching problem is formulated as:

$$\max \sum_{e \in E_{s_t}} \omega(e) \cdot x(e), \tag{18}$$

$$\sum_{e \in \delta(p)} x(e) \le 1, \qquad \forall p \in P_{s_t}, \tag{19}$$

$$x(e) \in \{0, 1\}, \qquad \forall e \in E_{s_t}. \tag{20}$$

   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:

$$f_{detour}(s_t) = \sum_{i \in PP_{s_t}} \Delta d_i / v \tag{21}$$

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. We note that this batch-based passenger–passenger pairing mechanism differs from rolling or insertion-based pooling approaches used in many operational systems. Here it serves as a simplified, yet flexible, abstraction on top of which we study the effect of matching timing decisions.

4.2.2. Passenger-driver matching

This phase finalizes the matching process for regular ride-hailing services and serves as the second stage for ride-pooling, 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:

$$\min \sum_{i \in D_{s_t}} \sum_{j \in PP_{s_t}} T_{ij} \cdot x_{ij}, \tag{22}$$

$$\sum_{i \in D_{s_t}} x_{ij} \le 1, \tag{23}$$

$$\sum_{j \in PP_{s_t}} x_{ij} \le 1, \tag{24}$$

$$x_{ij} \in \{0, 1\}, \qquad \forall i \in D_{s_t},\ \forall j \in PP_{s_t}, \tag{25}$$

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:

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

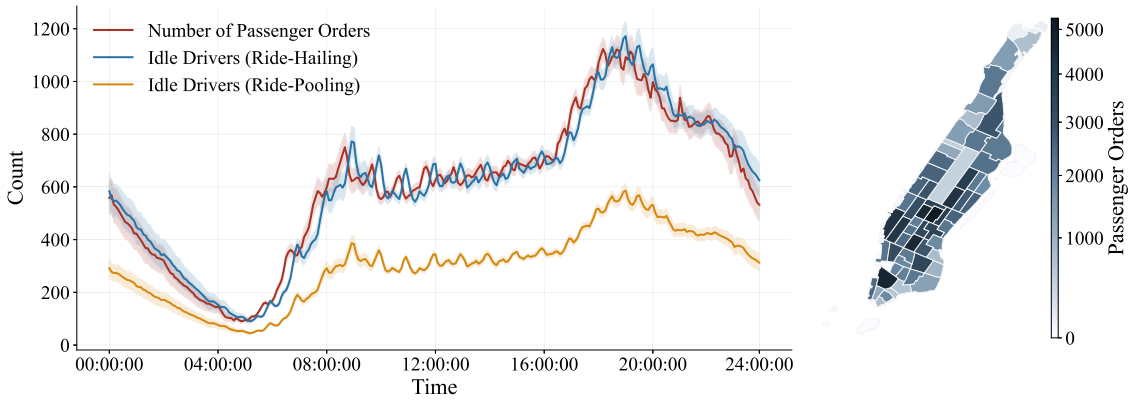


Fig. 6. Daily supply-demand fluctuations and geographical distribution in ride-hailing and ride-pooling systems.

5. Experiments

In the following, we describe the design and results of a series of experiments and sensitivity analyses which are conducted to evaluate the performance of the proposed methods.

5.1. Experimental settings, baselines and evaluation metrics

The match-maker was trained and evaluated using the simulator we designed. Each *episode* represents 10 min 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.

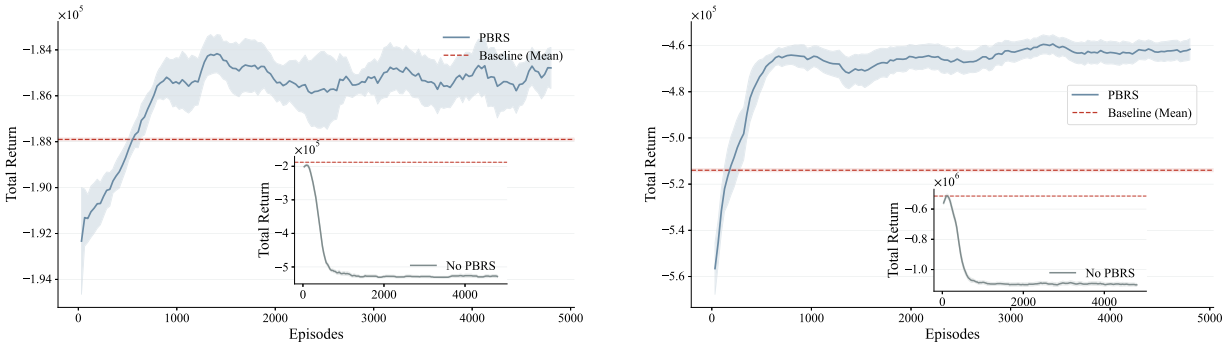
Passenger orders and driver data for each episode were generated using the method described in the Section 4.1. This method is based on historical FHV trip records from Manhattan, New York City, spanning from January to March 2022 (Taxi and Commission, 2024). The dataset reflects the fluctuations in supply and demand, as well as the geographical distribution, throughout the day for both ride-hailing and ride-pooling systems, as shown in Fig. 6. In all experiments, matching is performed over the entire service region rather than within a restricted matching radius. Given the relatively compact spatial extent of Manhattan, this setting allows the solver to obtain high-quality matching results without incurring excessive computational overhead.

To evaluate the effectiveness of our approach, we compare it against several batched matching strategies, which defer matching decisions to accumulate a pool of passengers and drivers before executing assignments. These include the widely adopted *fixed-interval batched matching* strategy, where matches are performed at uniform time intervals, as currently implemented by ride-hailing platforms such as Uber. In our simulation, the optimal fixed matching interval is set to 15 s for ride-hailing and 20 s for ride-pooling, determined through empirical testing to ensure fair evaluation across performance metrics. In addition, we introduce a *queue-length–triggered dynamic batching* strategy, where matching is triggered once the number of unmatched passengers reaches a predefined threshold (e.g., 15). This rule-based approach reflects a more adaptive matching mechanism that responds to real-time system demand, while remaining simple and interpretable. For completeness, we also compare against the *first dispatch* strategy, in which matching occurs immediately upon each new request. Since our simulation operates in discrete time steps, this strategy corresponds to executing matching at every time step.

Importantly, in order to isolate the effect of *when* matching is executed, all compared strategies are evaluated under identical simulation settings and use the same matching algorithms whenever a matching operation is executed. Specifically, at each decision epoch the match-maker chooses between waiting and executing a matching action (match vs. wait in our MDP formulation). When the match action is selected, the simulator applies the same matching procedure for every method. Importantly, the RL state representation is used only to inform our learned matching-time policy (the match vs. wait decision); it does not change candidate construction, grouping, or the matching algorithm used by the benchmark strategies once matching is executed. Therefore, performance differences reflect differences in matching time decisions, rather than differences in the matching algorithms. Rolling-horizon or insertion-based ride-pooling relaxes the batched matching abstraction and targets a different problem setting, and is thus outside the scope of this controlled timing study.

To evaluate the performance of our approach, we examine 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 passenger delay time incurred due to detours for picking up or dropping off additional ride-pooling passengers.
- **Average Total Waiting Time**: The overall average waiting time per passenger.



(a) Ride-Hailing Service (b) Ride-Pooling Service

Fig. 7. Training curves of (a) Ride-hailing and (b) Ride-pooling services.

The simulator and training algorithms developed in this study were implemented using Python 3.12. All codes are openly accessible at the following repository.[3] The training of the RL agents was conducted on the TU Delft supercomputer *Delft Blue*, whose detailed system specifications are available online.[4] Subsequent testing of the RL strategies was performed on a Lenovo Legion Y9000P IRX9 laptop equipped with an Intel Core i9-14900HX processor, 32 GB of RAM, and an NVIDIA GeForce RTX 4070 GPU.

5.2. 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 without—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 (based on standard error) were calculated to account for randomness and ensure statistical reliability. The baseline strategy, which corresponds to the optimal fixed-interval batched matching strategy under this scenario, was tested over 1,000 random episodes under identical simulation settings to ensure fair comparisons.

Fig. 7(a) 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 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.

The same trend can also be found in the Ride-Pooling service, as shown in Fig. 7(b). 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.

Fig. 7(a) and (b) 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 to this service mode. Additionally, the rapid initial

[3] https://github.com/baoyiman/A-Timely-Match-A-Deep-Reinforcement-Learning-Approach-for-Ride-Hailing-and-Ride-Pooling-Services.git
[4] https://www.tudelft.nl/dhpc/system

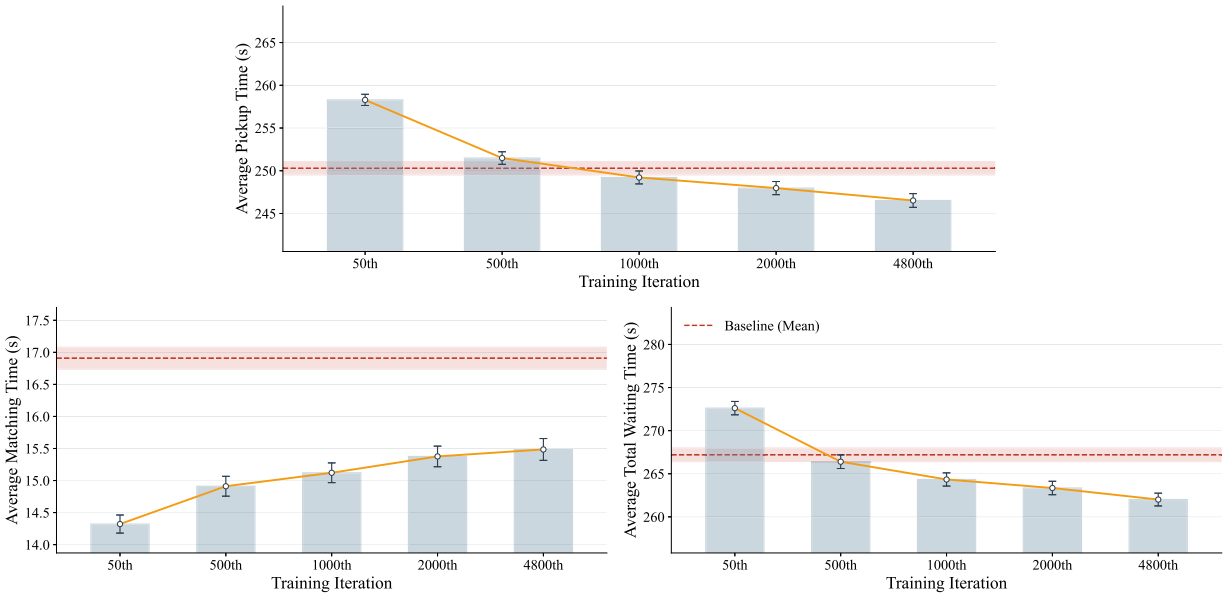


Fig. 8. Comparison of metrics across training episodes and baseline strategy for **Ride-Hailing** services.

increase in ride-pooling Total Return highlights its effectiveness in optimizing multi-passenger 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 and their standard error were recorded to evaluate the changes and improvements in the RL strategy across different training stages. The results are shown in Figs. 8 and 9.

Fig. 8 demonstrates that as training progresses, the Average Pickup Time decreases from 258 s at the early stages to 247 s 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 s to 15.7 s 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 s to 262 s, which is lower than the baseline strategy's 267 s. 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.

In ride-pooling, the trained strategy must balance Pickup Time, Detour Delay, and Matching Time to further reduce Total Waiting Time. The results in Fig. 9 provide a clear visualization of this process. The Average Pickup Time decreases from approximately 310 s at episode 50 to 250 s by episode 4800, stabilizing around this value, while the baseline remains higher at 290 s, reflecting improved efficiency in reducing post-matching waiting time. The Detour-related metric shows a similar substantial improvement. Average Detour Delay decreases from 125 s to 55 s, compared to the baseline's 110 s.

The Average Matching Time increases during training, peaking at 20 s before stabilizing around 18 s by episode 4800, outperforming the baseline's 22 s. 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 s to 330 s by episode 4800, while the baseline remains at 420 s.

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, Our results show that the PBRS-enhanced RL strategy reduces the Matching Time by 8.8% in ride-hailing and 11.9% in ride-pooling , while simultaneously reducing the Total Waiting Time by 3.1% in ride-hailing and 20.1% in ride-pooling. This demonstrates that 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.

5.3. Performance comparison between different matching strategies

We compare the performance of the proposed approach with fixed-interval batched matching, queue-length–triggered dynamic batching strategy, which we refer to as dynamic in the table, and first dispatch strategies. For fixed-interval batched matching, we

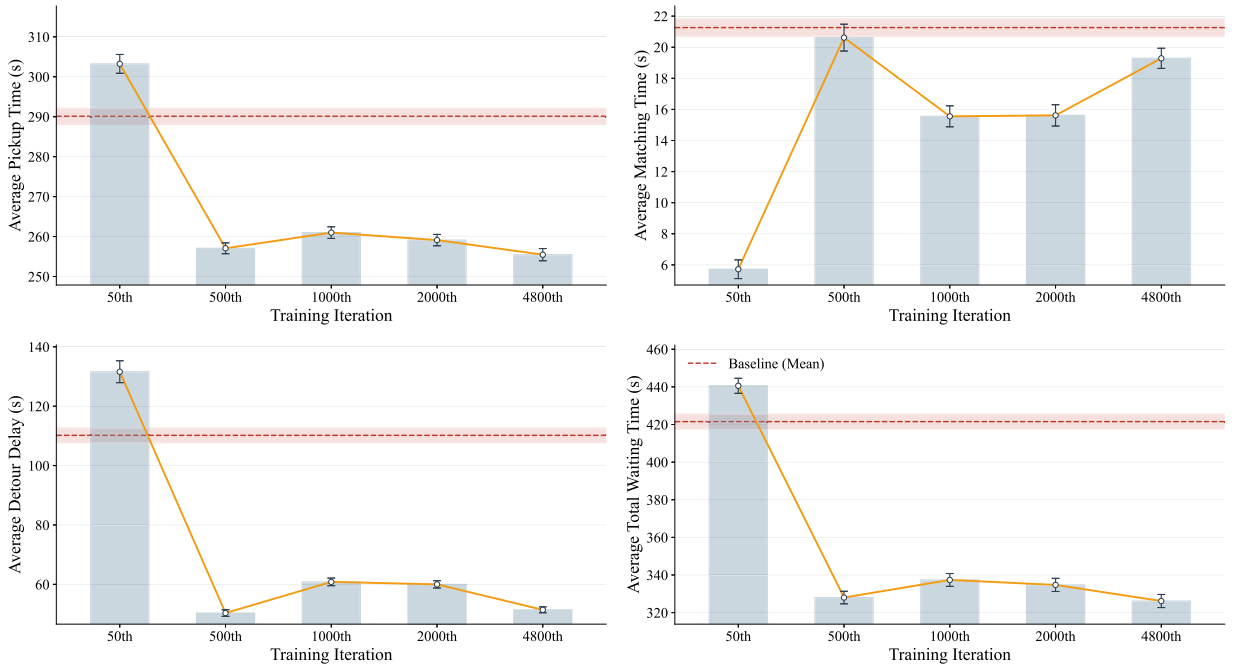


Fig. 9. Comparison of metrics across training episodes and baseline strategy for **Ride-Pooling** services.

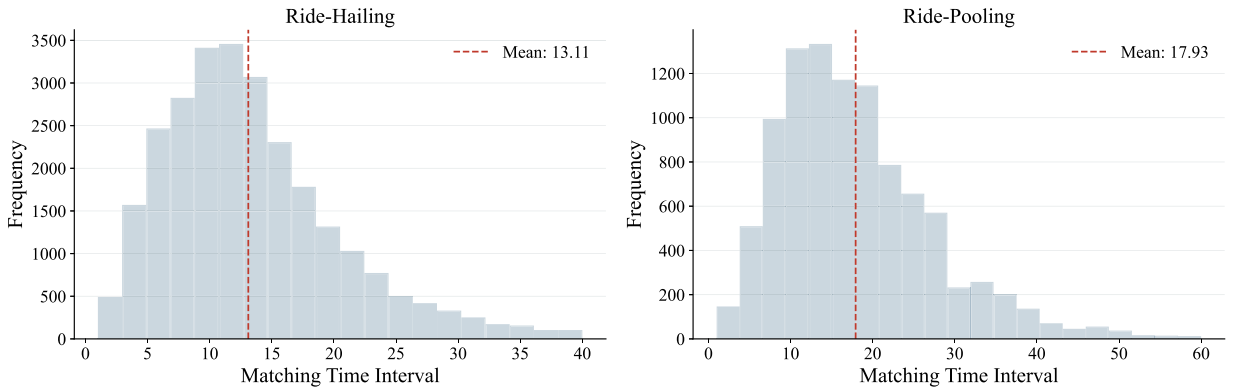


Fig. 10. Distribution of dynamic time intervals.

adjust the matching interval to examine its impact on performance, using 5, 15, 30, and 60 s for ride-hailing, and 10, 20, 40, and 80 s for ride-pooling, as shown in Tables 1 and 2, respectively. And for the queue-length–triggered dynamic batching strategy, the matching process is triggered once the number of accumulated passenger orders reaches predefined thresholds of 20, 40, or 60 in both ride-hailing and ride-pooling scenarios. The values in both tables are averaged over repeated experiments, with standard errors included.

The results in Table 1 demonstrate the effectiveness of the proposed approach in minimizing average total waiting time in the ride-hailing setting. It achieves the lowest value of 262.5 ± 1.5 s across all strategies, outperforming the best fixed-interval strategy (15 s: 268.7 ± 1.3 s) and the best dynamic threshold-based strategy (threshold $Q = 40$: 269.7 ± 1.4 s). Although the average pickup time of the proposed method (247.2 ± 1.6 s) is slightly longer than that of the 30 s and 60 s fixed-interval baselines, this is offset by a significant reduction in average matching time (15.3 ± 0.3 s). These results indicate that the proposed method achieves a more effective trade-off between matching delay and pickup efficiency, leading to superior overall performance.

Table 2 shows consistent results in the ride-pooling context. The proposed method yields the lowest average total waiting time of 337.8 ± 1.2 s, outperforming the best fixed-interval baseline (20 s: 427.0 ± 1.7 s) and the best dynamic strategy (threshold $Q = 40$: 419.8 ± 1.7 s) by substantial margins. It also achieves the lowest values for average pickup time, matching time, and detour delay. This consistent dominance across multiple performance metrics highlights the method's effectiveness in managing the additional complexity introduced by pooling.

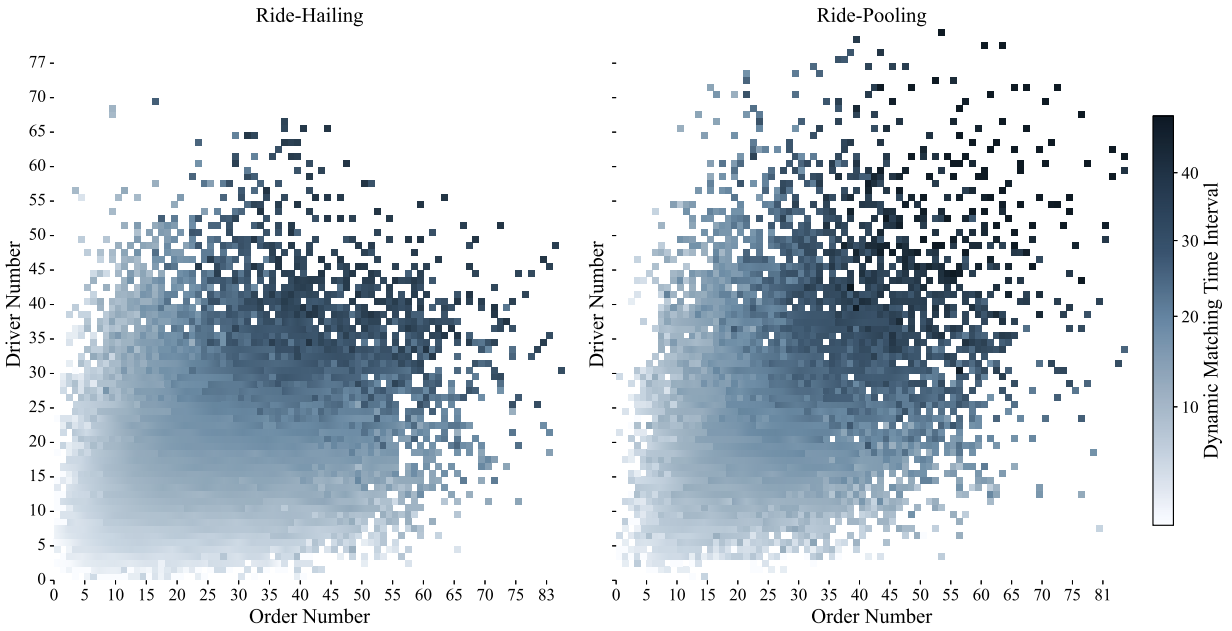


Fig. 11. Matching interval dynamics with order and driver counts.

Table 1
Ride-hailing: comparison of matching strategies. "Fixed" executes batching every Δ seconds. "Dynamic" triggers a batch when the number of unmatched passengers reaches Q. Bold indicates the best performance in each column. Mean ± standard error over 20 runs.

Strategy	Avg. Pickup Time (s)	Avg. Matching Time (s)	Avg. Total Waiting Time (s)
First Dispatch	300.7±1.9	1.7±0.4	302.4±1.6
Fixed (Δ = 5 s)	276.8±1.8	6.8±0.3	283.6±1.6
Fixed (Δ = 15 s)	252.0±1.6	16.7±0.3	268.7±1.3
Fixed (Δ = 30 s)	245.6±1.6	31.1±0.3	276.7±1.2
Fixed (Δ = 60 s)	235.4±1.4	62.5±0.3	298.0±1.2
Dynamic (Q = 20)	272.1±2.1	9.8±0.4	281.9±1.6
Dynamic (Q = 40)	255.1±2.1	14.6±0.4	269.7±1.4
Dynamic (Q = 60)	230.3±2.2	50.4±0.4	280.7±1.4
Our approach	247.2±1.6	15.3±0.3	**262.5±1.5**

Table 2
Ride-pooling: comparison of matching strategies. "Fixed" executes batching every Δ seconds. "Dynamic" triggers a batch when the number of unmatched passengers reaches Q. Bold indicates the best performance in each column. Mean ± standard error over 20 runs.

Strategy	Avg. Pickup Time (s)	Avg. Matching Time (s)	Avg. Detour Delay (s)	Avg. Total Waiting Time (s)
First Dispatch	416.3±1.4	2.0±0.6	190.6±2.0	608.8±2.1
Fixed (Δ = 10 s)	355.9±1.3	12.1±0.5	145.4±1.9	513.3±1.9
Fixed (Δ = 20 s)	291.9±1.3	22.0±0.4	113.1±1.8	427.0±1.7
Fixed (Δ = 40 s)	280.7±1.2	47.2±0.4	102.8±1.6	430.7±1.3
Fixed (Δ = 80 s)	265.1±1.3	92.5±0.4	75.4±1.2	433.1±1.3
Dynamic (Q = 20)	362.6±1.4	9.9±0.6	169.5±2.1	541.9±2.0
Dynamic (Q = 40)	290.3±1.4	18.8±0.6	110.7±2.0	419.8±1.7
Dynamic (Q = 60)	275.7±1.4	80.9±0.7	66.8±2.1	423.3±2.2
Our approach	**257.6±0.9**	19.5±0.4	**60.7±0.9**	**337.8±1.2**

5.4. 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.

Fig. 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 s but occasionally extend to 40 s. For ride-pooling, intervals are more dispersed and often exceed 30 s to accumulate sufficient requests for effective pooling, highlighting the complexity of coordinating multiple passengers.

Fig. 11 illustrates the relationship between matching intervals and real-time system states, specifically the number of passenger requests and available drivers. In ride-hailing (left panel), the RL policy tends to extend intervals when the number of passenger orders increases, particularly beyond 30, suggesting a strategy to increase matching opportunities. Conversely, when driver availability is low, the system shortens intervals to reduce passenger wait times. This behavior indicates that the policy favors longer intervals in near-balanced conditions and shorter intervals under supply constraints. In ride-pooling (right panel), longer intervals are frequently observed when passenger demand exceeds 50, reflecting the RL policy's preference to delay matching in order to maximize pooling efficiency. However, in scenarios with pronounced supply–demand imbalances, shorter intervals are used to prioritize service responsiveness over pooling optimization. These results confirm that the RL-based matching strategy exhibits context-aware adaptability. It dynamically adjusts the timing of matching based on real-time demand and supply, favoring longer intervals to enhance matching and pooling quality under stable conditions, and switching to shorter intervals to improve responsiveness when the system is under stress.

6. Conclusion and future work

This study presents a reinforcement learning framework that dynamically determines the matching timing, i.e., when to perform the ride-matching operation, in ride-hailing and ride-pooling systems. Unlike widely used fixed-interval or rule-based approaches in practice, our method treats this timing decision as a sequential decision problem, where the system learns to adapt to fluctuating supply and demand by observing compact system states and selecting optimal trigger points for executing matching. To address the sparse and delayed feedback inherent in matching timing decisions, we introduce a finite-horizon, potential-based reward shaping method that densifies the learning signal without altering the optimal policy. This ensures both faster convergence and theoretical soundness. The same formulation seamlessly extends to ride-pooling by incorporating detour costs into the reward structure, while keeping the underlying matching procedure fixed. Through extensive simulation experiments using NYC for-hire vehicle data, we show that the learned timing policies consistently outperform fixed-interval, rule-based dynamic, and first-dispatch strategies. The approach achieves up to a 20.1% reduction in total waiting time for ride-pooling and a 3.1% reduction for ride-hailing, while also significantly lowering detour delays and maintaining short matching times. The learned policy adapts matching timing contextually, delaying the execution of matching under balanced conditions and acting more promptly when the system is under pressure, demonstrating strong responsiveness to operational dynamics.

However, this study has several limitations that suggest several directions for future research. First, the experiments are conducted in a simulated environment, which may not fully capture the complexities of real-world scenarios, such as traffic congestion or unpredictable driver behavior, incorporating real-time traffic and operating in hybrid settings with dynamically evolving network conditions may further improve both matching quality and realism. Second, the current ride-pooling module is limited to two-passenger pooling, and extending the matching algorithm to support larger pool sizes such as three or more passengers based on the vehicle capacity remains an important next step. Third, our current implementation performs region-wide matching without an explicit spatial scope constraint. This choice is intentional: it allows a controlled investigation of the effect of when to execute batched matching. In larger or geographically dispersed markets, however, spatial constraints such as a matching radius are often required for scalability and may interact with the timing decision. A natural extension is therefore to jointly design spatial matching scope and matching timing policies within a unified optimization or learning framework. Finally, the simulator adopts a batch-based pooling abstraction in which compatible requests are grouped before assignment, whereas operational ride-pooling systems often rely on rolling-horizon or insertion-based mechanisms that update routes and accept new passengers while vehicles are in service. Integrating such rolling-horizon or insertion-based pooling mechanisms into the proposed timing framework, together with exploring alternative RL algorithms and richer state representations, would broaden its applicability.

CRediT authorship contribution statement

Yiman Bao: Writing – original draft, Visualization, Validation, Methodology, Formal analysis, Conceptualization; **Jie Gao:** Writing – review & editing, Supervision, Methodology, Formal analysis, Conceptualization; **Jinke He:** Writing – review & editing, Supervision, Methodology, Formal analysis, Conceptualization; **Frans A. Oliehoek:** Writing – review & editing, Supervision; **Oded Cats:** Writing – review & editing, Supervision, Conceptualization.

Data availability

Data will be made available on request.

References

Agatz, N., Erera, A., Savelsbergh, M., Wang, X., 2012. Optimization for dynamic ride-sharing: a review. Eur. J. Oper. Res. 223 (2), 295–303. https://doi.org/10.1016/j.ejor.2012.05.028

Alonso-Mora, J., Samaranayake, S., Wallar, A., Frazzoli, E., Rus, D., 2017. On-demand high-capacity ride-sharing via dynamic trip-vehicle assignment. Proc. Natl. Acad. Sci. 114 (3), 462–467. https://doi.org/10.1073/pnas.1611675114

Bal, M.İ., 2022. Reward Shaping for Efficient Exploration and Acceleration of Learning in Reinforcement Learning. Master's thesis. Middle East Technical University.

Bellman, R., 1957. A Markovian decision process. J. Math. Mech. 6(5), 679–684 .

Beojone, C.V., Geroliminis, N., 2023. Relocation incentives for ride-sourcing drivers with path-oriented revenue forecasting based on a Markov chain model. Transp. Res. Part C Emerging Technol. 157, 104375. https://doi.org/10.1016/j.trc.2023.104375

Brown, A.E., LaValle, W., 2020. Hailing a change: comparing taxi and ridehail service quality in los angeles. Transportation 48, 1007–1031. https://doi.org/10.1007/s11116-020-10086-z
Chen, T., Shen, Z., Feng, S., Yang, L., Ke, J., 2025. Dynamic matching radius decision model for on-demand ride services: a deep multi-task learning approach. Transp. Res. Part E Logist. Transp. Rev. 193, 103822.
Devlin, S., Kudenko, D., 2011. Theoretical considerations of potential-based reward shaping for multi-agent systems. In: Tenth International Conference on Autonomous Agents and Multi-Agent Systems. ACM, pp. 225–232.
Do, M., Byun, W., Shin, D.K., Jin, H., 2019. Factors influencing matching of ride-hailing service using machine learning method. Sustainability 11 (20). https://doi.org/10.3390/su11205615
Fan, W., Yan, X., Sun, Z., Yang, X., 2025. Novel operational algorithms for ride-pooling as on-demand feeder services. Transp. Res. Part C Emerging Technol. 178, 105181.
Feng, S., Ke, J., Xiao, F., Yang, H., 2022. Approximating a ride-sourcing system with block matching. Transp. Res. Part C Emerging Technol. 145, 103920. https://doi.org/10.1016/j.trc.2022.103920
Gao, J., Cheng, R., Wu, Y., Zhao, H., Mai, W., Cats, O., 2025. Optimizing matching radius for ride-hailing systems with dual-replay-buffer deep reinforcement learning. Comput. Ind. Eng. 208, 111296.
Guo, X., Caros, N.S., Zhao, J., 2021a. Robust matching-integrated vehicle rebalancing in ride-hailing system with uncertain demand. Transp. Res. Part B Methodol. 150, 161–189. https://doi.org/10.1016/j.trb.2021.05.015
Guo, Y., Zhang, Y., Boulaksil, Y., 2021b. Real-time ride-sharing framework with dynamic timeframe and anticipation-based migration. Eur. J. Oper. Res. 288 (3), 810–828. https://doi.org/10.1016/j.ejor.2020.06.038
Ibrahim, S., Mostafa, M., Jnadi, A., Salloum, H., Osinenko, P., 2024. Comprehensive overview of reward engineering and shaping in advancing reinforcement learning applications. IEEE Access 12, 175473–175500.
Ke, J., Xiao, F., Yang, H., Ye, J., 2020a. Learning to delay in ride-sourcing systems: a multi-agent deep reinforcement learning framework. IEEE Trans. Knowl. Data Eng. 34 (5), 2280–2292.
Ke, J., Yang, H., Zheng, Z., 2020b. On ride-pooling and traffic congestion. Transp. Res. Part B-Methodol. 142, 213–231. https://doi.org/10.1016/j.trb.2020.10.003
Kucharski, R., Cats, O., 2020. Exact matching of attractive shared rides (exMAS) for system-wide strategic evaluations. Transp. Res. Part B Methodol. 139, 285–310. https://doi.org/10.1016/j.trb.2020.06.006
Kucharski, R., Cats, O., 2024. Hyper pooling private trips into high occupancy transit like attractive shared rides. npj Sustainable Mobility Transp. 1 (1), 6.
Li, Y., Liu, Y., 2021. Optimizing flexible one-to-two matching in ride-hailing systems with boundedly rational users. Transp. Res. Part E Logist. Transp. Rev. 150, 102329.
Liu, S., Wang, Z., Zhang, Y., Yang, H., 2024a. Anomalous ride-hailing driver detection with deep transfer inverse reinforcement learning. Transp. Res. Part C Emerging Technol. 159, 104466. https://doi.org/10.1016/j.trc.2023.104466
Liu, Y., Feng, S., Bao, Y., Yang, H., 2025. Learning the on-demand adaptable matching range with a reinforcement learning. Transp. Res. Part C Emerging Technol. 172, 105018.
Liu, Z., Ouyang, G., Zhang, B., Du, B., Chen, C., Wu, K., 2024b. Joint order dispatching and vehicle repositioning for dynamic ridesharing. IEEE Trans. Mobile Comput. 24(4), 2628–2643.
Mao, C., Liu, Y., Shen, Z.-J.M., 2020. Dispatch of autonomous vehicles for taxi services: A deep reinforcement learning approach. Transp. Res. Part C Emerging Technol. 115, 102626. https://doi.org/10.1016/j.trc.2020.102626
Meshkani, S.M., Farooq, B., 2023. Centralized and decentralized algorithms for two-to-one matching problem in ridehailing systems. EURO J. Transp. Logist. 12, 100106. https://doi.org/10.1016/j.ejtl.2023.100106
Ng, A.Y., Harada, D., Russell, S., 1999. Policy invariance under reward transformations: theory and application to reward shaping. In: ICML. Vol. 99, pp. 278–287.
Ouyang, Y., Yang, H., Daganzo, C.F., 2021. Performance of reservation-based carpooling services under detour and waiting time restrictions. Transp. Res. Part B Methodol. 150, 370–385.
Puterman, M.L., 1994. Markov Decision Processes: Discrete Stochastic Dynamic Programming. John Wiley & Sons, Inc., New York, NY, USA. 1st edition.
Qiao, S., Han, N., Huang, J., Peng, Y., Cai, H., Qin, X., Lei, Z., 2023. An three-in-one on-demand ride-hailing prediction model based on multi-agent reinforcement learning. Appl. Soft Comput. 149, 110965. https://doi.org/10.1016/j.asoc.2023.110965
Qin, G., Luo, Q., Yin, Y., Sun, J., Ye, J., 2021a. Optimizing matching time intervals for ride-hailing services using reinforcement learning. Transp. Res. Part C Emerging Technol. 129, 103239. https://doi.org/10.1016/j.trc.2021.103239
Qin, X., Yang, H., Wu, Y., Zhu, H., 2021b. Multi-party ride-matching problem in the ride-hailing market with bundled option services. Transp. Res. Part C Emerging Technol. 131, 103287. https://doi.org/10.1016/j.trc.2021.103287
Schulman, J., Wolski, F., Dhariwal, P., Radford, A., Klimov, O., 2017. Proximal policy optimization algorithms. arXiv:1707.06347v2 .
Shaheen, S., 2018. Shared mobility: the potential of ride hailing and pooling , 55–76. https://doi.org/10.5822/978-1-61091-906-7_3
Shi, J., Li, X., Aneja, Y.P., Li, X., 2023. Ride-matching for the ride-hailing platform with heterogeneous drivers. Transp. Policy 136, 169–192. https://doi.org/10.1016/j.tranpol.2023.04.001
Simonetto, A., Monteil, J., Gambella, C., 2019. Real-time city-scale ridesharing via linear assignment problems. Transp. Res. Part C Emerging Technol. 101, 208–232. https://doi.org/10.1016/j.trc.2019.01.019
Taxi, N. Y.C., Commission, L., 2024. Tlc trip record data. Accessed: 2025-04-20. https://www.nyc.gov/site/tlc/about/tlc-trip-record-data.page.
Uber, 2023. How batch matching works. Accessed: October 5, 2024. https://www.uber.com/us/en/marketplace/matching/.
Wei, K., Vaze, V., Jacquillat, A., 2021. Transit planning optimization under ride-hailing competition and traffic congestion. Transp. Sci. 56, 725–749. https://doi.org/10.1287/TRSC.2021.1068
Yan, C., Zhu, H., Korolko, N., Woodard, D., 2020. Dynamic pricing and matching in ride-hailing platforms. Nav. Res. Logist. 67 (8), 705–724.
Yang, H., Qin, X., Ke, J., Ye, J., 2020. Optimizing matching time interval and matching radius in on-demand ride-sourcing markets. Transp. Res. Part B Methodol. 131, 84–105. https://doi.org/10.1016/j.trb.2019.11.005
Zhang, K., Mittal, A., Djavadian, S., Twumasi-Boakye, R., Nie, Y.M., 2023. Ride-hail vehicle routing (river) as a congestion game. Transp. Res. Part B Methodol. 177, 102819. https://doi.org/10.1016/j.trb.2023.102819
Zhou, Z., Roncoli, C., Sipetas, C., 2023. Optimal matching for coexisting ride-hailing and ridesharing services considering pricing fairness and user choices. Transp. Res. Part C Emerging Technol. 156, 104326. https://doi.org/10.1016/j.trc.2023.104326