Optimizing Matching Time Intervals in Ride-Hailing and Ride-Pooling Services Using Reinforcement Learning

Master Thesis

Yiman Bao (5691648)

Delft University of Technology

December 10, 2024

Optimizing Matching Time Intervals in Ride-Hailing and Ride-Pooling Services Using Reinforcement Learning

Yiman Bao

December 10, 2024

Summary

This study explores a novel approach to address the challenges of matching strategies in ride-hailing and ride-pooling services by leveraging reinforcement learning (RL) to dynamically adjust matching time intervals. Ride-hailing and ride-pooling services have become vital components of urban mobility systems, offering efficient and cost-effective transportation solutions. However, traditional strategies like real-time matching and fixed-interval matching often fail to adapt to fluctuating supply-demand conditions, leading to issues such as increased passenger wait times or inefficient driver utilization. This thesis proposes a dynamic optimization framework based on the Proximal Policy Optimization (PPO) algorithm to tackle these challenges.

The research begins by framing the problem as a sequential decision-making task within a reinforcement learning (RL) framework. This formulation is particularly suited for dynamic environments like ride-hailing and ride-pooling systems, where decisions made at one time step influence future states and outcomes. To model this complexity, a Markov Decision Process (MDP) is employed. The MDP framework allows the system to capture critical system states such as passenger demand, driver supply, and the elapsed time since the last matching operation. By representing these dynamics explicitly, the MDP formulation provides a solid foundation for designing adaptive matching strategies.

One of the significant challenges in this dynamic decision-making context is the sparse reward problem. In scenarios where meaningful rewards occur infrequently—such as reducing total passenger waiting times after a match—agents often struggle to learn effectively due to the lack of regular feedback. To address this, a Potential-Based Reward Shaping (PBRS) mechanism is integrated into the framework. This approach introduces intermediate reward signals that guide the agent toward better policies during training. By shaping the rewards using a carefully designed potential function, PBRS accelerates convergence, enabling the RL agent to identify high-quality solutions more efficiently. This innovation is crucial for improving the feasibility and performance of RL in large-scale dynamic systems.

The RL-based methodology involves the design of an agent capable of making adaptive matching decisions in response to real-time supply-demand conditions. The agent' s state representation is designed to be both compact and informative, focusing on metrics such as the current time, passenger demand, and driver availability. These metrics are chosen to ensure that the model remains computationally efficient while capturing essential information for decision-making. The compact state representation reduces the complexity of the input space, allowing the RL model to scale effectively to larger systems without compromising its ability to make informed decisions.

In addition to the state representation, the action space is carefully designed to allow the agent to dynamically adjust matching intervals. At each time step, the agent can choose between performing a matching operation or waiting to accumulate more passengers and drivers before matching. This flexibility enables the RL framework to balance short-term goals, such as minimizing immediate wait times, with long-term objectives, such as improving overall system efficiency. By adjusting matching intervals dynamically, the agent ensures that the system remains responsive to fluctuating conditions while optimizing resource allocation.

To evaluate the effectiveness of the proposed RL framework, a high-fidelity simulation environment is developed to emulate the real-world dynamics of ride-hailing and ride-pooling systems. This simulator incorporates key elements such as fluctuating demand patterns, spatial constraints, and realistic passenger and driver behaviors. For instance, passenger orders and driver locations are generated based on calibrated Poisson distributions to mimic urban transportation conditions. The simulation also accounts for spatial constraints, such as driver-passenger proximity and traffic patterns, ensuring that the RL agent operates in a realistic and challenging environment.

The simulator supports a wide range of experimental scenarios, enabling a thorough evaluation of the RL-based matching strategy. By replicating dynamic supply-demand fluctuations, the simulation tests the adaptability and robustness of the RL framework under diverse conditions. This comprehensive evaluation demonstrates the framework' s ability to consistently outperform traditional matching strategies, offering a practical solution for optimizing shared mobility systems in dynamic urban environments.

Extensive experiments were conducted to evaluate the performance of the proposed RL-based matching strategy against traditional fixed-interval and real-time matching approaches. The results consistently demonstrate that the RL strategy significantly reduces passenger waiting times across var-

ious scenarios. By dynamically adjusting matching intervals, the RL framework is able to optimize the timing of passenger-driver assignments, ensuring that matches occur at the most opportune moments. This adaptability allows the system to respond effectively to fluctuating demand, maintaining high levels of service efficiency even during periods of imbalance.

For ride-hailing services, one of the key advantages of the RL-based approach is its ability to address supply-demand imbalances. Traditional strategies like fixed-interval matching are rigid and cannot adapt to changes in real-time conditions, often leading to either excessive driver idling during low-demand periods or increased passenger wait times during peak hours. In contrast, the RL framework dynamically extends matching intervals during low-demand periods to aggregate more requests, thereby improving resource utilization. Similarly, during high-demand periods, the framework shortens matching intervals to process accumulated passenger requests promptly, ensuring a smoother passenger experience and reducing overall system congestion.

In ride-pooling scenarios, the RL strategy showcases its robustness by effectively balancing multi-passenger needs while minimizing detour delays. Unlike ride-hailing, ride-pooling involves additional complexity due to the need to coordinate multiple passengers with overlapping routes. Traditional real-time approaches often struggle to find optimal matches, leading to excessive detours or unbalanced resource allocation. The RL framework, however, dynamically adjusts matching intervals based on real-time demand patterns and supply availability. This allows the system to group passengers more efficiently, reducing both the average detour time and the total travel cost for passengers. As a result, the RL-based strategy improves overall service quality and passenger satisfaction in ride-pooling operations.

The comparative analysis further highlights the flexibility and adaptability of the RL-based approach across different service modes and demand scenarios. Under scenarios of severe supply-demand imbalance, such as a driver-to-passenger ratio of 0.5:1, the RL framework demonstrates remarkable resilience by prioritizing passengers with longer waiting times and dynamically reallocating available drivers to high-demand regions. Conversely, in oversupply scenarios, the framework extends matching intervals to maximize vehicle utilization, minimizing driver idling and optimizing operational efficiency. These capabilities underline the robustness of the RL strategy in handling a wide range of real-world challenges.

Moreover, the experiments reveal that the RL-based strategy consistently outperforms both fixed-interval and real-time matching approaches across key performance metrics, including average pickup time, average total waiting time, and detour delays. This superiority is particularly evident in scenarios with rapidly fluctuating demand, where the RL framework' s ability to adapt to real-time conditions provides a significant advantage. The RL strategy not only improves passenger experience by reducing waiting times but also enhances system-level efficiency by optimizing resource allocation, making it a highly effective solution for modern urban mobility systems.

In summary, the experimental results validate the effectiveness and versatility of the RL-based matching strategy. Its ability to dynamically adapt to diverse conditions and service modes highlights its potential for widespread application in ride-hailing and ride-pooling services. By addressing the limitations of traditional approaches, the proposed RL framework provides a robust and scalable solution for optimizing shared mobility systems in dynamic urban environments.

This thesis makes several key contributions. It introduces a dynamic matching optimization framework using RL to enhance service efficiency and passenger satisfaction. The incorporation of PBRS addresses the challenges of sparse rewards, accelerating learning and improving the quality of derived policies. The research also advances multi-passenger matching strategies in ride-pooling environments, reducing detour delays and optimizing vehicle utilization. Finally, the development of a realistic simulation environment enables robust evaluation and provides a foundation for future research.

The findings of this study emphasize the potential of RL-driven dynamic matching strategies to enhance shared mobility systems. By reducing passenger wait times, improving vehicle utilization, and adapting to fluctuating demand, the proposed framework offers a promising solution for modern urban transportation challenges. Future work could explore more sophisticated state representations or incorporate real-time information through advanced machine learning models such as Recurrent Neural Networks (RNNs), paving the way for even greater adaptability and scalability in shared mobility services.

Acknowledgments

As this journey of my master' s study draws to a close, I would like to express my heartfelt gratitude to everyone who has supported and guided me along the way. Without their invaluable assistance, this thesis would not have been possible.

First and foremost, I would like to thank the members of my thesis committee for their guidance and support. Prof. Cats provided insightful and constructive feedback on the design of my simulator, greatly enhancing its realism. Prof. Oliehoek offered valuable suggestions from his expertise in reinforcement learning, which helped refine my training approach. I am especially grateful to Dr. Gao and Jinke. Dr. Gao has been a constant source of encouragement and direction, guiding me through every stage from topic selection and research design to thesis writing, and introducing me to the world of reinforcement learning. Her rigorous academic standards and sharp insights have been a tremendous inspiration on my academic journey. Jinke provided invaluable technical advice throughout the project. Whenever I encountered technical challenges, he was always there to offer timely assistance, significantly enhancing my technical skills in the process.

I would also like to extend my deep appreciation to my family. Their unwavering support has been the foundation that has allowed me to pursue my studies and goals. No matter where I was, their encouragement and care were a constant source of warmth, enabling me to devote myself fully to my work. Thank you for your understanding and sacrifices, which have made it possible for me to pursue my dreams.

To my girlfriend, Qu Na, my deepest gratitude. She has perhaps borne the greatest burden from my commitment to this thesis, as I was often preoccupied with my research—even during our conversations. Yet, during times when I hit roadblocks, she was always there to offer encouragement and stand by my side. No matter the challenges I faced, she supported me unconditionally, sharing in both my joys and frustrations. Her patience and understanding have been invaluable, and I am incredibly grateful for her presence in my life.

Lastly, I want to thank my friends who have been by my side throughout this journey. Your companionship, support, and laughter have filled my life with countless moments of joy, making this path far less lonely. Thank you for your encouragement and friendship, which have given me the strength to reach the finish line.

Contents

1 Introduction and Research Questions **7**
- 1.1 Introduction . . . 7
- 1.2 Research Question . . . 10

2 Literature Review **10**
- 2.1 Existing Work on Matching in Ride-hailing . . . 10
- 2.2 Research Gap . . . 13

3 Methodology **14**
- 3.1 Problem Description . . . 14
- 3.2 Reinforcement Learning Framework . . . 17
  - 3.2.1 Modeling the Problem . . . 17
  - 3.2.2 Action Space . . . 19
  - 3.2.3 State design . . . 19
  - 3.2.4 Reward Design . . . 20
- 3.3 Simulator . . . 24
  - 3.3.1 Assumptions of the Simulator . . . 25
  - 3.3.2 Passenger and Driver Data Generation . . . 25
  - 3.3.3 Spatial Matching Algorithm . . . 27
  - 3.3.4 Verification Of the Simulator . . . 31
- 3.4 The Proximal Policy Optimization (PPO) Algorithm . . . 34

4 Result **35**
- 4.1 Experimental Setup and Metrics . . . 35
- 4.2 Training Performance . . . 38
- 4.3 Comparison of Strategies . . . 41
- 4.4 Ride-Hailing vs. Ride-Pooling . . . 47

5 Conclusion **50**
- 5.1 Research Contributions . . . 50
- 5.2 Key Findings and Implications . . . 51
- 5.3 Limitation and Future Work . . . 52

References **54**

Appendix **57**
- A Ride-Hailing Simulator Working Flow . . . 57
- B Ride-Pooling Simulator Working Flow . . . 58
- C Developing Environment . . . 59
- D Parameters of PPO . . . 61
- E Map of Pick-up and Drop-off Points . . . 63
- F Distribution of Passenger Orders in the Experiment . . . 64
- G Comparison of System Strategy Performance Metrics . . . 65
- H Metrics of Ride-Hailing vs. Ride-Pooling . . . 67

List of Figures

1 Impact of Batched Matching in Ride-Hailing Services . . . 15
2 Impact of Batched Matching in Ride-Pooling Services . . . 16
3 Components of Passenger Waiting Time in Ride-Hailing and Ride-Pooling Services . . . 21
4 State transition diagram of matching process . . . 23
5 Comparison of Rewards Before and After PBRS . . . 24
6 All possible pickup and drop-off sequences for two passenger orders . . . 28
7 Visualization of Passenger-Driver Match results . . . 32
8 Visualization of Passenger-Passenger Match results . . . 33
9 Fixed Time Interval Strategy in the Ride-Hailing Simulator . . . 34
10 Fixed Time Interval Strategy in the Ride-Pooling Simulator . . . 34
11 Training curve of Traditional Ride-Hailing Service . . . 39
12 Training Curve of Ride-Pooling Service . . . 40
13 Comparison of Average Total Waiting Time with Standard Error Across Different Strategies in Ride-Hailing and Ride-Pooling . . . 42
14 Comparison of Metrics Across Training Episodes and Baseline Strategy for Ride-Hailing Services . . . 43
15 Comparison of Metrics Across Training Episodes and Baseline Strategy for Ride-Pooling Services . . . 44
16 Distribution of Dynamic Time Intervals . . . 46
17 Matching Interval Dynamics with Order and Driver Counts . . . 46
18 Average Total Waiting Time in 8 Scenarios . . . 50
19 Map of Pick-up and Drop-off Points . . . 63
20 Distribution of Passenger Orders in the Experiment . . . 64
21 Comparison of System Waiting Times Across Training Episodes and Baseline Strategy for Ride-Hailing Services . . . 65
22 Comparison of System Waiting Times Across Training Episodes and Baseline Strategy for Ride-Pooling Services . . . 66

List of Tables

1 Research Comparison . . . 13
2 Comparison of Existing Literature and Contributions of This Study . . . 14
3 Experimental Scenario Design . . . 48
4 Comparison of Metrics Across Different Ride-Hailing Scenarios . . . 49
5 All Metrics of Ride-Hailing vs. Ride-Pooling . . . 67

1 Introduction and Research Questions

1.1 Introduction

In recent years, ride-hailing services, such as Uber and Lyft, have become a core component of urban transportation systems worldwide. Compared to traditional taxi services, ride-hailing platforms offer greater convenience and flexibility to passengers. With mobile applications, users can request a vehicle anytime and anywhere, and track the driver's location and estimated arrival time, significantly reducing wait times [1]. This seamless user experience has substantially increased reliance on this mode of service. In addition to enhancing individual travel convenience, ride-hailing services also play a crucial role in urban traffic management. By optimizing vehicle dispatch and real-time supply-demand matching, they effectively alleviate traffic congestion [2]. Compared to conventional taxi services, ride-hailing platforms intelligently match nearby idle vehicles with passengers, reducing unnecessary driving distances and minimizing idle time on the road [3]. Moreover, ride-hailing services contribute to environmental protection. By improving vehicle utilization—especially during peak hours—these services maximize the efficiency of each trip, thereby reducing carbon emissions [4].

In recent years, ride-pooling has emerged as a significant development within the ride-hailing industry, gaining increasing attention for its potential benefits. Unlike traditional single-passenger rides, ride-pooling enables multiple passengers to share a single vehicle journey when their origins and destinations are relatively close. This model retains the convenience of ride-hailing while further enhancing vehicle utilization and cost efficiency by combining multiple trips into one [5]. One of the main advantages of ride-pooling is its potential to significantly reduce travel costs for users. By grouping multiple passengers in the same vehicle and allowing them to share portions of their journey, ride-hailing platforms can offer lower fares [6]. Furthermore, ride-pooling decreases the total driving distance for each vehicle, reduces the frequency of empty rides, and helps to minimize passenger wait times as well as road congestion [7]. From a traffic management and environmental perspective, ride-pooling is even more beneficial than traditional ride-hailing. By lowering the number of vehicles on urban roads, ride-pooling helps alleviate traffic congestion and reduces carbon emissions, especially during peak periods when many passengers are traveling simultaneously [8]. These advantages make ride-pooling an essential strategy in the pursuit of sustainable urban transportation, as it improves vehicle efficiency without significantly extending passengers' travel time.

Despite the notable success of ride-hailing and ride-pooling in enhancing urban mobility, sustaining these services over the long term has become a critical challenge, especially as urban populations and transportation demands continue to grow. For ride-hailing and ride-pooling models to remain competitive in the future, they must consistently deliver increasingly convenient and efficient service experiences. In this regard, optimizing matching strategies—efficiently pairing passengers with available drivers—is essential for supporting the long-term growth and sustainability of these services.

In traditional ride-hailing services, passenger wait times and driver idle times are two major factors that affect both service experience and operational efficiency. If passengers wait too long, user satisfaction will drop, which can negatively impact the platform's market share. On the other hand, excessive driver idle times increase operational costs, reduce driver earnings, and may even affect driver engagement levels [9]. Therefore, designing an efficient matching strategy that minimizes both passenger wait times and driver idle times is crucial for the platform' s sustainable growth. For ride-pooling services, the complexity of matching strategies is further elevated, as they not only need to match passengers with drivers, but also optimize carpooling among multiple passengers. In this case, an effective matching strategy must consider passengers' real-time locations and destinations while minimizing detours and trip delays to maximize vehicle utilization [10]. During peak hours, ride-pooling services can reduce the number of vehicles on the road by matching several passengers to the same vehicle, thereby easing traffic congestion. However, the actual ride-pooling experience is often constrained by the efficiency of the matching algorithms. If these algorithms fail to optimize pooling effectively, passengers may experience extended detours or longer wait times, which ultimately reduces user satisfaction [11].

Thus, optimizing matching strategies is crucial for enhancing the overall efficiency of both ride-hailing and ride-pooling services, reducing resource waste, and improving user satisfaction. As fluctuations in urban transportation demand become more pronounced, imbalances between supply and demand are common. Platforms must be able to respond quickly during peak demand periods by intelligently dispatching vehicles, while efficiently managing driver resources during off-peak times to

avoid idle vehicles. Addressing these issues requires more flexible and intelligent matching strategies [12]. In the long run, the sustainable development of ride-hailing and ride-pooling depends on the continuous optimization of matching strategies. By designing algorithms that can dynamically respond to complex supply and demand fluctuations, platforms can not only improve operational efficiency but also provide users with more convenient and reliable travel experiences. This win-win situation will enable ride-hailing and ride-pooling services to continue growing and innovating in the future market.

However, most current matching strategies focus primarily on optimizing spatial distance to improve efficiency, such as minimizing the distance between passengers and drivers to make matching decisions [13]. These strategies provide certain advantages in reducing passenger wait times and minimizing driver idle times, particularly in areas with a high density of vehicles. Spatial matching can quickly pair nearby drivers with passengers, thus improving service response times. However, these approaches often overlook the temporal dimension, specifically the optimization of matching time intervals. In reality, the supply-demand relationship between passengers and drivers is not static, and both passenger requests and driver availability can fluctuate significantly over short periods of time. If matching decisions are based solely on real-time spatial information, optimal matching opportunities may be missed. For example, by accumulating more passenger requests and available drivers over a short period, the system could perform a batch matching later, leading to a more optimal overall outcome [12]. Ignoring the temporal dimension in matching can lead to suboptimal decisions, as it fails to fully leverage dynamic supply-demand information, ultimately decreasing overall service efficiency.

Some existing matching strategies, such as those used by Uber, attempt to address the challenges of real-time matching by adopting fixed time interval matching, which is called batched matching. In this approach, the system accumulates a certain number of passenger requests and available drivers over fixed intervals, then performs batch matching [14]. This batch matching approach improves overall system dispatch capabilities to some extent, particularly during peak hours, effectively reducing long passenger wait times and high driver idle rates. However, fixed interval matching lacks the necessary flexibility to adapt to real-time changes in supply and demand. Fixed time windows may result in unnecessary waiting during periods of low demand, while during periods of high demand, a long time interval could cause the system to miss optimal matching opportunities [15]. For instance, during peak hours, when passenger requests and driver availability fluctuate rapidly, fixed interval matching could cause the system to miss the best matching opportunities, leading to longer passenger wait times and increased driver idle rates. Therefore, fixed interval matching strategies are often inadequate when dealing with complex fluctuations in supply and demand.

In summary, whether relying too heavily on real-time spatial matching or using fixed interval matching, current matching strategies exhibit significant shortcomings. To further improve the overall efficiency of ride-hailing and ride-pooling systems, it is essential to introduce more flexible matching strategies that optimize both spatial and temporal dimensions, thereby achieving truly dynamic matching.

Given the limitations of existing matching strategies, particularly the emphasis on spatial optimization while neglecting dynamic adjustments to time intervals and the lack of flexibility in fixed interval strategies, adopting more intelligent optimization methods is essential. Reinforcement Learning (RL), a method that learns optimal strategies through continuous interaction with the environment, can autonomously discover the best matching decisions in complex and dynamic supply-demand environments [16]. Unlike traditional fixed or real-time matching strategies, RL can leverage historical and real-time data to progressively learn and improve decision-making processes. The system does not need predefined matching rules; instead, it gradually learns how to flexibly adjust matching strategies under various supply and demand conditions, thereby optimizing overall system performance [17]. This adaptability is particularly suited to dynamic demand adjustments in ride-hailing and ride-pooling scenarios, where supply and demand fluctuate significantly across both time and space. Traditional static or semi-static matching methods struggle to cope with such complexity. Specifically, RL can effectively address the challenge of optimizing matching time intervals. By observing the outcomes of matching decisions at different moments, the RL model can gradually learn when to execute matches and when to delay them, allowing actions to be taken at the most optimal time. More importantly, the RL model can automatically adjust matching intervals based on system conditions, overcoming the inflexibility of fixed interval strategies.

Although reinforcement learning can learn optimal matching strategies in dynamic environments, its application also faces challenges, particularly in highly complex state spaces. The state space in

ride-hailing and ride-pooling environments is highly dimensional and complex due to factors such as the geographic distribution of passengers and drivers, as well as the temporal dynamics of passenger demand [18]. Deep Reinforcement Learning (DRL), which combines the strengths of deep learning and reinforcement learning, can effectively solve decision-making problems in high-dimensional state spaces [19]. By incorporating deep neural networks, DRL can handle large amounts of input data and can learn complex nonlinear relationships, enabling the system to make better matching decisions in complex scenarios. In the context of ride-hailing and ride-pooling, DRL can solve the pain points of optimizing matching strategies in response to dynamic supply and demand changes. It can learn how to adjust matching strategies in real-time under conditions of high variability and uncertainty, ensuring the system continues to operate efficiently even in complex supply-demand conditions. Compared to traditional reinforcement learning, DRL' s deep neural networks allow it to model complex spatiotemporal patterns, making it better suited for handling the intricate passenger pooling combinations in ride-pooling and for optimizing matching time intervals. Therefore, DRL becomes the ideal solution for optimizing matching strategies in ride-hailing and ride-pooling systems. It can autonomously learn optimal spatiotemporal matching strategies while coping with the dynamic fluctuations and uncertainties of the environment, making it particularly effective for optimizing large-scale, complex transportation systems and improving overall service quality and resource utilization.

In conclusion, under the current dynamic supply-demand environment, traditional matching strategies, especially those using fixed time intervals, exhibit clear limitations and struggle to effectively address the complexities caused by fluctuations in supply and demand. To tackle these challenges, this paper proposes a dynamic matching time interval optimization method based on *Deep Reinforcement Learning (DRL)*, aiming to further improve the efficiency of *ride-hailing* and *ride-pooling* services. By using the DRL model, the system can adaptively adjust matching time intervals, optimize the matching of passengers and drivers, reduce passenger wait times, increase vehicle utilization, and minimize detour delays during ride-pooling.

This study introduces several innovations in optimizing the matching strategies of *ride-hailing* and *ride-pooling* services, addressing the inherent issues in traditional methods and enhancing overall system efficiency. The main innovations of this paper are as follows:

1. **Dynamic Optimization of Matching Time Intervals**: Traditional matching strategies typically rely on fixed time intervals, lacking the flexibility to adapt to real-time fluctuations in supply and demand. This study develops a dynamic optimization strategy for matching time intervals based on reinforcement learning (RL), allowing the system to adjust in real-time according to supply and demand changes. This dynamic optimization enables the system to balance matching efficiency and passenger wait times under varying conditions, significantly improving overall system performance.

2. **Introducing Potential-Based Reward Shaping (PBRS)**: Traditional RL methods tend to perform poorly in sparse reward environments. This study introduces a reward mechanism using potential-based reward shaping (PBRS), which significantly accelerates model convergence and improves learning efficiency. With PBRS, the model can quickly learn optimal strategies, effectively reducing total wait times and enhancing system response speed.

3. **Addressing the Complexity of Multi-Passenger Matching**: In the context of *ride-pooling*, the complexity of matching multiple passengers is significantly higher. This study proposes an efficient algorithmic framework that takes into account the similarity of passenger origin and destination points, optimizing the matching process and reducing detour times. During periods of high demand, the system can dynamically match passengers, effectively minimizing detour delays and enhancing overall passenger satisfaction.

4. **Development and Validation of an Efficient Simulator**: To validate the proposed strategy, this study designs and develops an efficient simulator capable of accurately simulating real-world supply-demand fluctuations and dynamically adjusting matching strategies. The simulator can generate realistic passenger order and driver distribution data, providing a testing platform for researchers to evaluate and optimize different matching methods before actual implementation.

Through these innovations, this paper presents a more flexible and adaptive optimization method for matching strategies, improving system efficiency, reducing passenger wait times, and providing a solid theoretical and practical foundation for future smart mobility services.

1.2 Research Question

The rapid growth of ride-hailing and ride-pooling services has significantly improved urban mobility. However, ensuring the long-term efficiency and sustainability of these services remains a challenge, especially as urban populations increase and transportation demands fluctuate. As identified in the introduction, optimizing the matching time intervals is crucial to addressing these issues, particularly when reinforced by advanced reinforcement learning techniques. This study seeks to explore how reinforcement learning can dynamically adjust matching intervals to reduce passenger wait times and improve overall service efficiency.

The primary objective of this study is summarized in the following main research question:

> How can reinforcement learning techniques be used to dynamically optimize matching time intervals in both ride-hailing and ride-pooling services to reduce passenger wait times?

To address the main question comprehensively, this study is further guided by the following sub-research questions:

- How can reinforcement learning adjust matching strategies to respond effectively to changing supply and demand in a dynamic environment?
- What methods can be applied to manage detour delays in ride-pooling services, minimizing additional travel time for passengers?
- Compared to ride-hailing services, how much additional delay does ride-pooling introduce when serving the same market demand?

By answering these questions, this study aims to develop a deeper understanding of how reinforcement learning can enhance the operational efficiency and user experience in ride-hailing and ride-pooling services. This exploration of dynamic optimization seeks to provide both theoretical insights and practical solutions for the sustainable development of urban transportation.

2 Literature Review

2.1 Existing Work on Matching in Ride-hailing

Ride-hailing services have quickly become essential components of urban public transportation systems in the digital age. Wang et al.[20] found that in Chicago, ride-hailing usage is significantly higher than traditional taxis—six times more on weekdays and eleven times more on weekends. Additionally, ride-hailing services have a wider coverage area compared to traditional taxis, Olayode et al.[21] analyzed the effects of ride-hailing on public road transportation, revealing both positive and negative impacts. On the positive side, ride-hailing improves environmental sustainability, reduces traffic congestion, and enhances accessibility. However, it also increases competition with public transportation, reduces funding for public transit, and may negatively affect driver working conditions. ride-pooling, an extension of ride-hailing services, refers to a mode of transportation where multiple passengers share the same vehicle for trips to different destinations. This mode maximizes vehicle utilization by optimizing routes and matching passenger demand, thereby reducing the cost of individual rides, alleviating traffic congestion, and mitigating environmental pollution. Alisoltani et al.[22] found that in large urban transportation networks, especially during periods of high demand density, dynamic ride-pooling systems can significantly improve traffic conditions and alleviate urban road congestion, particularly during peak hours. Ride-sharing compensates for the additional travel distance required to operate these services by distributing trips among multiple passengers. In conclution, it can be seen that ride-hailing and ride-pooling service have a huge impact on today's urban public transportation system, and their efficient operation will also bring greater convenience to people.

While ride-hailing and ride-pooling services have brought significant benefits to urban transportation, their efficiency largely depends on the effectiveness of matching algorithms. Most research has focused on optimizing spatial matching—how to allocate vehicles to passengers based on location—while less attention has been given to optimizing matching time intervals, which are essential for

balancing passenger wait times and vehicle utilization. However, insights from spatial matching algorithms can aid in developing a simulation framework that includes spatial matching to explore the optimization of matching time intervals.

The matching algorithms used in ride-hailing services fundamentally is the issue of supply-demand matching, with the ultimate goal of dynamically balancing both sides to minimize passenger wait times and reduce idle driving for drivers. Feng et al.[23] proposed a block-matching strategy, where the service area is divided into several blocks, and demand is matched within each block simultaneously. They modeled this block-matching ride-hailing system using the M/M/c queuing model. Similarly, Guo et al.[24] introduced the matching-integrated vehicle rebalancing (MIVR) model, which aims to improve rebalancing decisions by accounting for demand uncertainty. This model integrates driver-customer matching into vehicle rebalancing strategies at an aggregate level, with the objective of minimizing a generalized cost that includes total vehicle miles traveled (VMT) and the number of unfulfilled requests. Dong et al.[25] proposed a mixed network equilibrium model to capture the interplay between freelance drivers' self-directed movements and the centralized repositioning of contracted drivers to satisfy the balance of supply and demand in ride-hailing.

The matching algorithms for ride-pooling services are more complex than for standard ride-hailing, as they must consider not only driver-passenger matching but also the optimization of routes for multiple passengers with different destinations. Meshkani et al.[26] proposed a decentralized heuristic algorithm based on vehicle-to-infrastructure (V2I) communication, which, when applied to Toronto's road network, increased service rate by 24% and improved speed by 25.53 times compared to IBM's algorithm[27]. Long et al.[28] considered travel time uncertainty and explored the impact of variable costs and travelers' values of time (VOT) on ride-pooling cost savings. Wang et al.[29] developed a mathematical model to predict matching probabilities and expected ride distances, integrating seeker and taker states into a system of nonlinear equations to calculate these values dynamically. Meshkani[30] introduced GMOMatch, a graph-based many-to-one ride-matching algorithm that efficiently handles traffic congestion with high service quality. Guo et al.[31] proposed a real-time framework using a dynamic timeframe and heuristic graph search methods to enhance ride-pooling efficiency. Alonso-Mora et al.[10] presented a scalable model for real-time ride-sharing, starting with a greedy assignment and improving routes through constrained optimization to quickly converge to optimal solutions.

Some more complex models simultaneously consider both normal ride-hailing and ride-pooling services. Qin et al.[32] developed a multi-objective integer linear programming model with three modes: ride-pooling (RP), non-ride-pooling (NP), and a "bundled" option combining both. They used a two-stage Kuhn-Munkres (2-KM) algorithm to solve the passenger-vehicle matching problem through iterative matching. Beojone et al.[33] integrated both services by predicting the number of requests drivers might encounter and optimizing driver repositioning to match passengers with closer drivers.

The above literature mainly focuses on optimizing spatial matching in ride-hailing and ride-pooling services, aiming to achieve the optimal solution at a given point in time. These studies primarily address how to allocate vehicles to passengers based on their current locations, or how to reposition drivers closer to passenger requests through various dispatch strategies. However, they do not take into account the temporal dimension, where the movement of drivers and passengers over time could lead to better results. This gap in research, particularly in considering matching time intervals, opens up an opportunity to explore how dynamic adjustments in time intervals could impact the efficiency of both ride-hailing and ride-pooling services. Yang et al.[34] found that by adjusting the matching time intervals, ride-hailing platforms can accumulate more waiting passengers and available drivers within a certain period, enabling more optimal matches. A moderate increase in the matching time interval can reduce the expected pick-up distance, improve vehicle utilization, and enhance passenger satisfaction. The study also modeled how to adjust matching time intervals under different supply and demand conditions to optimize system performance. The results indicated that shorter matching time intervals are more beneficial when demand exceeds supply, while extending the matching time interval when supply exceeds demand can effectively reduce pick-up distances and improve matching rates. In practice, Uber also implements a similar approach through its Batched Matching system, where requests from multiple passengers are accumulated within a fixed time interval before matching them to available drivers. This allows Uber to optimize the allocation of drivers by considering more requests in a given area, leading to better matches and improved vehicle utilization. By delaying matching for a short period, Uber' s batched system reduces the need for immediate assignments, allowing for

the pooling of passengers with similar routes in ride-pooling scenarios, and minimizing overall detour times. The fixed interval used in batched matching helps manage fluctuations in supply and demand, ensuring that during peak times more passengers can be grouped together for more efficient rides. This strategy is particularly effective in busy urban environments where both passenger demand and driver availability change rapidly[14].

Matching at fixed time intervals has indeed been shown to have advantages over real-time matching, as it allows for more optimized pairings by accumulating available drivers and passengers. However, the fixed interval strategy can be overly rigid, requiring extensive historical data to support its effectiveness. Moreover, it lacks flexibility and adaptability, making it less transferable to new environments or changing conditions. However, reinforcement learning (RL), with its adaptability and dynamic decision-making capabilities, has become an ideal tool for optimizing matching time intervals. RL continuously learns from changes in supply and demand, dynamically adjusting the matching time intervals—shortening the interval during peak demand periods and extending it during off-peak times. This approach enhances vehicle utilization and reduces passenger wait times. Additionally, RL can handle unforeseen situations in uncertain environments, optimizing the overall system performance. Its scalability and transferability make it capable of quickly adapting to new scenarios. Several studies have already applied RL to ride-hailing and ride-pooling services. For example, Qiao et al.[35] proposed a three-in-one multi-agent reinforcement learning-based online algorithm called ERPM, designed for intelligent ride-hailing demand prediction. ERPM addresses the challenge of convergence in traditional RL models, caused by the high dimensionality of input and output data when service areas are divided into grids. Using the Actor–Critic strategy, it optimizes on-demand ride-hailing dispatch actions, efficiently predicting demand in grid areas. Similarly, Mao et al.[36] developed a novel model-free deep reinforcement learning framework to solve the taxi dispatching problem. This framework reallocates vehicles in transportation networks when there is a spatial or temporal imbalance between demand and supply. The algorithm converges to within 4% of the theoretical upper bound, regardless of whether system dynamics are deterministic or stochastic. Additionally, when considering user priorities, the learned strategy effectively balances fairness, cancellation rates, and service levels, resulting in superior dispatch policies.

Despite the focus of the aforementioned studies on developing better dispatching strategies using RL, they also demonstrate that RL is well-suited for highly dynamic ride-hailing and ride-pooling environments. RL' s adaptability to changing conditions makes it an effective approach for determining the optimal timing for matching drivers with passengers. By continuously learning from real-time supply and demand fluctuations, RL-based systems can dynamically adjust their matching intervals, leading to better system performance and user satisfaction. This not only improves vehicle utilization and reduces passenger wait times but also ensures that the system can respond efficiently to unexpected changes in traffic conditions or demand surges.Moreover, the ability of RL to optimize timing for matching actions in both ride-hailing and ride-pooling services is crucial, as it balances operational efficiency with the need to provide timely service. As a result, RL can offer significant improvements over static or pre-defined matching strategies, providing a flexible solution that can evolve with the complexities of urban transportation networks. In light of these advantages, further exploration of RL's potential in optimizing matching time intervals specifically remains an open and promising area of research, particularly for ride-pooling, where the challenge lies not only in matching passengers with drivers but also in coordinating multiple passengers with different destinations.

Qin et al.[37] have explored the use of reinforcement learning (RL) to optimize the matching time intervals in conventional ride-hailing. However, when attempting to reproduce their findings, this study identified issues in their reward design. Specifically, Qin et al. acknowledged the problem of reward sparsity and attempted to address it through reward shaping. Nevertheless, their reward shaping approach—particularly in terms of the reward component related to the time all matched passengers wait to be picked up by drivers—does not decrease with delayed matching as expected but rather continues to increase. This flawed design logic leads to alterations in the total reward after reward shaping, which does not align with real-world dynamics.

Table 1: Research Comparison

Researchers	Ride-Hailing	Ride-Pooling	Spatial Matching	Temporal Matching	RL
Feng et al.[23]	✓		✓		
Guo et al.[24]	✓		✓		
Meshkani et al.[26]		✓	✓		
Long et al.[28]		✓	✓		
Wang et al.[29]		✓	✓		
Guo et al.[31]		✓	✓		
Alonso-Mora et al.[10]		✓	✓		
Qin et al.[32]	✓	✓	✓		
Beojone et al.[33]	✓	✓	✓		
Yang et al.[34]	✓			✓	
Uber [14]	✓			✓	
Qiao et al.[35]	✓				✓
Mao et al.[36]	✓				✓
Qin et al.[37]	✓			✓	✓
This Research	✓	✓	✓	✓	✓

2.2 Research Gap

While extensive research has been conducted on optimizing spatial matching in ride-hailing and ride-pooling services, a critical dimension remains overlooked—the temporal optimization of matching. Existing studies predominantly focus on static strategies, such as fixed-interval or real-time matching, which fail to adapt to dynamic supply-demand fluctuations. This rigidity often results in suboptimal performance, manifesting as increased passenger wait times and degraded system efficiency.

Moreover, while reinforcement learning (RL) has demonstrated its potential in dynamic decision-making for ride-hailing, its application to the temporal aspect of matching remains largely unexplored. Current RL-based studies primarily address spatial optimization or vehicle dispatch, overlooking the significant benefits that dynamic adjustments to matching intervals could bring to system adaptability and performance.

The challenge becomes even more pronounced in ride-pooling scenarios, where the complexity of coordinating multiple passengers with different origins and destinations introduces additional inefficiencies. Existing approaches inadequately address these challenges, often leading to longer detours and reduced passenger satisfaction

To address these pressing gaps, this research introduces a novel framework with the following key innovations:

1. **Dynamic Temporal Optimization**: Moving beyond static fixed-interval or real-time matching, this study proposes a reinforcement learning-based approach to dynamically optimize matching time intervals, enabling the system to adapt to fluctuating supply-demand conditions.

2. **RL-Driven Adaptive Strategies**: Harnessing reinforcement learning, the proposed method autonomously adjusts matching decisions in real-time, enhancing overall efficiency and responsiveness.

3. **Advanced Multi-Passenger Coordination**: Specifically targeting ride-pooling, the framework dynamically optimizes multi-passenger matching strategies, minimizing detour delays while balancing efficiency and service quality.

4. **High-Fidelity Simulation Environment**: A spatio-temporal simulator is developed to rigorously evaluate the proposed methods, ensuring their applicability in realistic urban transportation scenarios.

5. **Potential-Based Reward Shaping**: To address the sparse reward problem, a novel reward-shaping mechanism accelerates learning convergence and improves the quality of the derived policies.

Research Dimension	Existing Literature	Contribution of This Study
Optimization of Matching Time Intervals	Fixed or real-time matching strategies, lacks flexibility	Dynamic optimization using reinforcement learning
Application of Reinforcement Learning	Spatial matching or dispatching focus, neglects time dimension	Adjusts matching time intervals dynamically
Ride-Pooling Matching	Complex multi-passenger matching, insufficient dynamic optimization	Improves service quality with dynamic optimization strategies
Simulator	Static simulators used, struggles with spatio-temporal variations	Develops a high-efficiency spatio-temporal simulator
Reward Design	Sparse reward problem, challenge remains in current research	Potential-based reward shaping resolves the issue

Table 2: Comparison of Existing Literature and Contributions of This Study

By bridging these critical gaps, this research establishes a foundation for more adaptive, efficient, and robust matching algorithms. The proposed innovations offer substantial contributions to the field of ride-hailing and ride-pooling, addressing both spatial and temporal challenges in real-world dynamic environments.

3 Methodology

This chapter introduces the methods and model design used in this study. First, it provides a detailed description of the batched matching problem in ride-hailing and ride-pooling services, analyzing the impact of different matching strategies on system efficiency and passenger experience. Then, the chapter presents a reinforcement learning-based solution, specifically focusing on how the Proximal Policy Optimization (PPO) algorithm is used to dynamically optimize the matching time window, thereby improving matching efficiency in complex supply-demand environments. Additionally, this chapter thoroughly explains the construction of the simulation environment, including the generation of passenger and driver data, the design of the state and action spaces, and the definition of the reward function, ensuring that the model can operate effectively in realistic scenarios.

3.1 Problem Description

Batched matching is an important strategy for improving the efficiency of matching in ride-hailing and ride-pooling services. In these services, the system must quickly find suitable drivers after a passenger places an order to reduce waiting times and maximize vehicle utilization. However, in dynamic and fluctuating supply-demand environments, real-time matching often struggles to achieve optimal results, especially when there is an imbalance between passenger orders and available drivers. To address this challenge, batched matching accumulates multiple passenger orders and driver resources within a fixed time window and then performs centralized matching at the end of that window to achieve overall optimization.

The key to batched matching lies in delaying the matching decision, allowing the system to make more comprehensive choices based on a larger set of information. Unlike real-time matching, which often reacts to individual orders and available drivers, batched matching reduces the impact of short-term fluctuations in supply and demand. For example, when the system receives an order and there is only one available driver nearby, real-time matching would instantly assign the driver, even if they may not be the best choice. Batched matching, on the other hand, accumulates multiple orders and driver resources and waits until the time window closes to optimize the overall matching, reducing randomness and suboptimal decisions.

In ride-hailing services, batched matching significantly reduces idle driving and improves overall passenger satisfaction. By accumulating more orders within a fixed time window, the system can match drivers with passengers located closer to them, thereby reducing passenger wait times and improving matching efficiency. At the same time, batched matching optimizes the system's overall dispatch strategy, avoiding unnecessary idle driving and increasing vehicle utilization. As shown in the Figure 1, the left illustration presents the results of real-time matching at time t, where the average pick-up distance is 4. Due to the limited matching options, passengers have longer pick-up distances. In contrast, the right illustration shows the results when matching is delayed until time t+n. By accumulating more passenger requests, the system can optimize matches, reducing the average pick-up distance to 2. This process significantly reduces idle driving and improves matching efficiency, making batched matching a more stable and efficient solution for ride-hailing services in environments with fluctuating supply and demand.

In ride-pooling services, batched matching becomes even more critical. The system not only needs to match passengers with drivers but also has to match passengers with other passengers for ride-pooling.

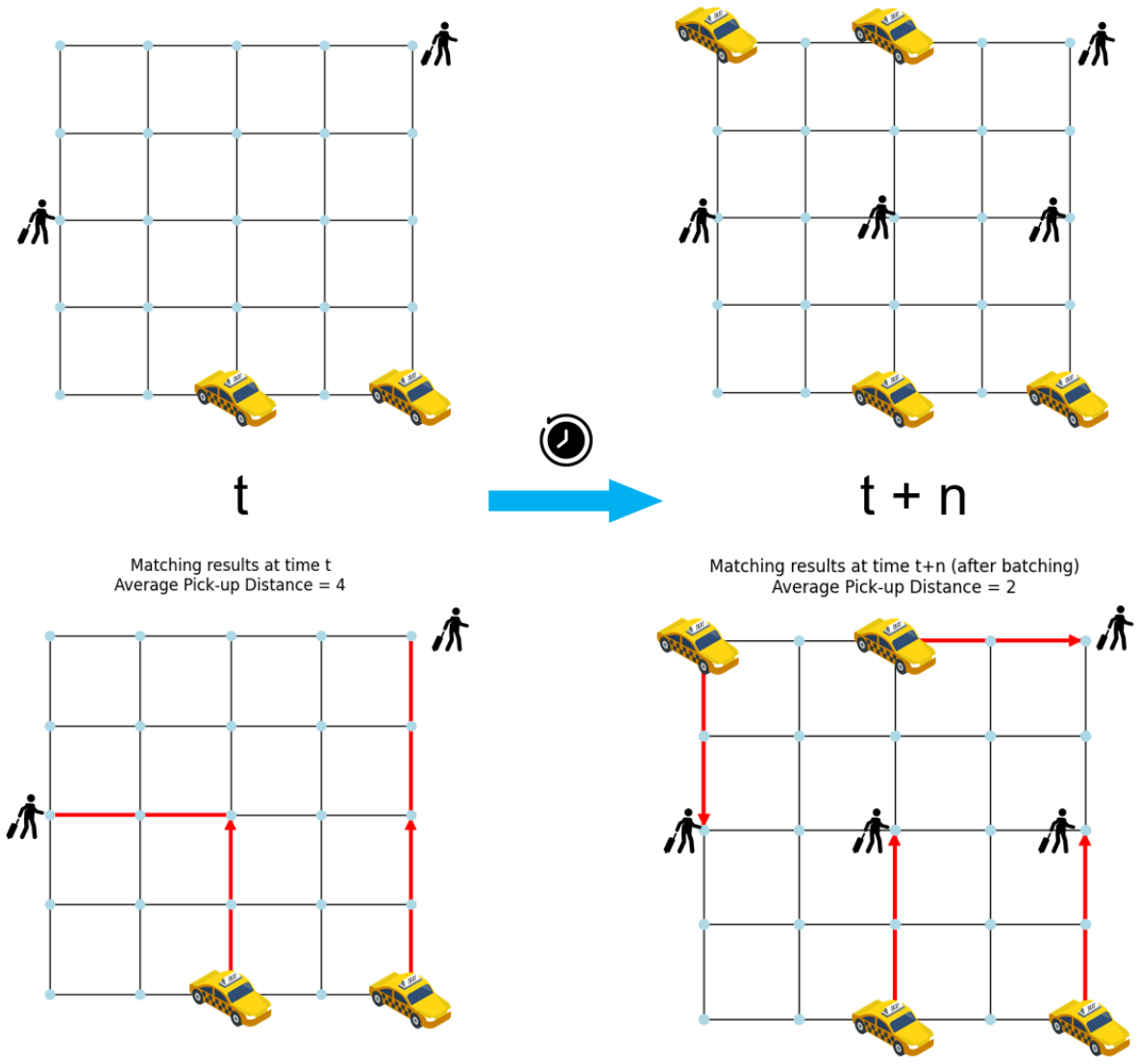

Figure 1: Impact of Batched Matching in Ride-Hailing Services

Under the batched matching framework, the system can accumulate multiple ride-pooling requests within a time window and match passengers based on similarities in their starting points, destinations, and travel routes. This strategy effectively reduces detour delays caused by ride-pooling and improves the overall efficiency of shared trips. As shown in the Figure 2, the left illustration presents the results of real-time matching at time t. At this point, due to limited matching options, the average detour distance is 2.5, leading to longer travel routes and higher detour costs. The right illustration, however, shows the results when matching is delayed until time t+n, allowing the system to accumulate more passenger requests and optimize the matching combinations to reduce detour distances. Here, the average detour distance is reduced to 0.5, significantly improving matching efficiency and reducing delays. This illustration clearly demonstrates the advantages of batched matching in enhancing the efficiency of ride-pooling services.

One of the core advantages of batched matching is its ability to smooth short-term fluctuations in the system, especially under rapidly changing supply-demand conditions. Real-time matching can be easily affected by short-term demand spikes. For instance, during peak hours or after large events, the system may receive a large number of passenger orders in a short period, and real-time matching might suffer from reduced efficiency due to the inability to handle the sudden influx of orders simultaneously.

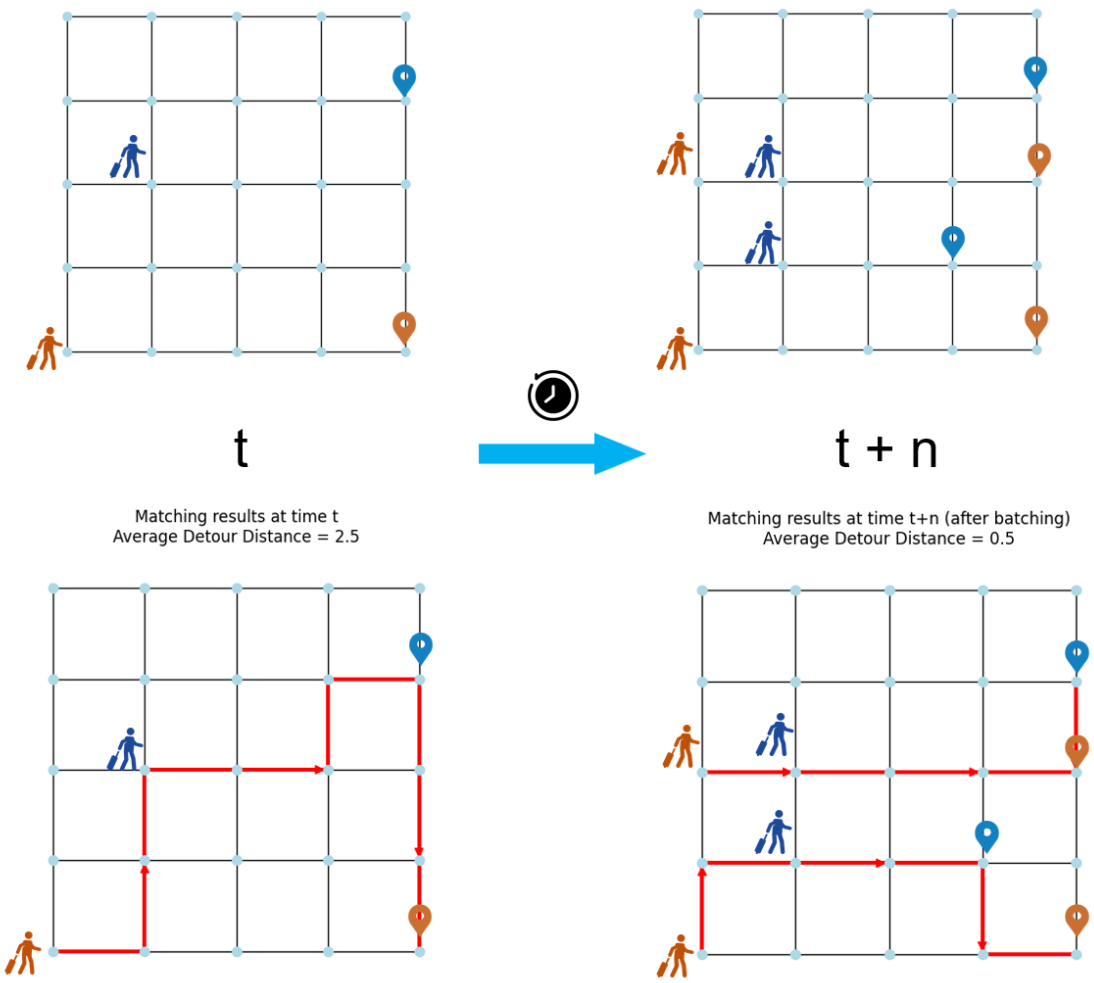

Figure 2: Impact of Batched Matching in Ride-Pooling Services

Batched matching improves the overall matching efficiency by accumulating these orders within a time window and processing them centrally, preventing excessive waiting times caused by order backlogs.

At the same time, the length of the time window in batched matching is a key factor that influences system performance. A window that is too short may lead to insufficient orders being accumulated, reducing the effectiveness of batched matching. On the other hand, a window that is too long increases passenger waiting times and negatively affects user experience. Therefore, determining the optimal window length is an important challenge in batched matching design. Dynamically adjusting the window length according to different demand scenarios and supply conditions can further enhance the system's flexibility and matching efficiency.

In summary, by accumulating more passenger orders and driver information within a fixed time window, batched matching allows for more optimal matching decisions in ride-hailing and ride-pooling services. It can reduce passenger waiting times, increase vehicle utilization, and optimize the overall service quality of shared rides through efficient pooling strategies. Batched matching presents a unique advantage in balancing real-time responsiveness with matching precision, particularly in environments with fluctuating supply and demand, offering a more stable and efficient solution for ride-hailing systems.

While batched matching presents clear advantages in improving the efficiency of ride-hailing and ride-pooling services, several challenges arise when applying this strategy in dynamic and fluctuating environments. The key difficulty lies in determining the optimal matching time window, which directly impacts both passenger satisfaction and vehicle utilization.

One of the main challenges in dynamic environments is balancing supply and demand. In real-

world ride-hailing systems, the supply of available drivers and the demand for rides are rarely in perfect balance. These conditions fluctuate throughout the day, with high demand during peak hours and lower demand during off-peak times. During peak demand periods, if the time window is too short, the system may not accumulate enough orders to optimize matching, resulting in less efficient matches and longer detour times in ride-pooling scenarios. Conversely, if the time window is too long, passengers will experience excessive waiting times, negatively affecting the user experience.

In addition to determining the optimal time window length, spatial distribution imbalances of passengers and drivers introduce further complexity. In high-density urban areas, such as city centers, a short batched matching window might still result in high matching efficiency due to the large volume of available drivers and passengers in close proximity. However, in suburban or less densely populated areas, longer matching windows might be necessary to accumulate sufficient orders and available drivers to create an efficient match. The challenge lies in dynamically adjusting the matching strategy to account for these varying conditions across different geographic zones.

Another critical challenge, especially in ride-pooling services, is managing the detour delays caused by ride-pooling. While batched matching is designed to reduce these delays by accumulating ride requests with similar origins and destinations, it is essential to ensure that the detour costs remain acceptable for all passengers involved. Matching passengers with similar routes within the optimal time window is difficult, particularly when dealing with high demand variability and differing passenger preferences.

Lastly, implementing an efficient batched matching system requires high computational efficiency. Processing large volumes of orders and driver information within the time window, while simultaneously optimizing matches across a broad geographic area, demands significant computational power. The system must be capable of handling this complexity in real time, ensuring that the optimization process does not delay the matching decisions themselves.

In summary, the core challenge in optimizing batched matching lies in dynamically adjusting the matching time window to account for fluctuating demand and supply conditions, while simultaneously minimizing passenger wait times, managing detour costs in shared rides, and ensuring computational efficiency. Addressing these challenges is critical to realizing the full potential of batched matching in improving the overall performance of ride-hailing systems. This motivates the need for advanced optimization techniques, such as reinforcement learning, which can automatically learn and adapt to these dynamic conditions.

3.2 Reinforcement Learning Framework

This subsection introduces the reinforcement learning framework developed to optimize matching intervals in ride-hailing and ride-pooling services. The framework leverages reinforcement learning to dynamically adjust matching time intervals in response to fluctuating supply-demand conditions. The problem is first modeled as a finite Markov Decision Process (MDP), followed by a detailed design of the state and action spaces, capturing critical spatiotemporal features of the system such as current time, passenger request volume, average waiting time, and driver availability. To address the challenge of sparse rewards, the framework incorporates a potential function, providing more frequent reward signals during training and accelerating convergence. Through continuous interaction with the dynamic environment, the deep reinforcement learning agent autonomously learns optimal matching strategies, effectively minimizing passenger wait times and improving service efficiency and passenger satisfaction under varying demand scenarios.

3.2.1 Modeling the Problem

The process of determining the matching timing in ride-hailing and ride-pooling services can be formulated as a finite Markov Decision Process (MDP), which provides a mathematical framework for sequential decision-making under uncertainty. The finite MDP is defined by a tuple $(\mathcal{S}, \mathcal{A}, P, R, \gamma, T)$, where:

- $\mathcal{S}$: The state space, representing the set of all possible system states. Each state $s \in \mathcal{S}$ describes the conditions or environment of the system at a specific time.
- $\mathcal{A}$: The action space, representing the set of all actions available to the agent. Each action $a \in \mathcal{A}$ influences the transition of the system state.

- $P(s'|s,a)$: The state transition probability, which defines the probability of transitioning to a new state $s' \in \mathcal{S}$ from the current state $s \in \mathcal{S}$ after taking action $a \in \mathcal{A}$.
- $R(s,a)$: The reward function, specifying the immediate reward received when the agent takes action a in state s. This reflects the desirability of a particular action in a given state.
- $\gamma \in [0,1]$: The discount factor, used to balance the importance of future rewards relative to immediate rewards. A value closer to 1 gives more weight to future rewards (set to 1 in this study).
- t: The time step, indicating the current point in the decision-making process. Typically, $t = 0, 1, 2, \ldots, T$, where T is the total duration of the process.
- T: The horizon, representing the total number of time steps in the decision-making process for a finite MDP.
- s_t: The current state at time t, representing the observed conditions of the system at this specific time step.
- a_t: The current action at time t, chosen by the agent based on the observed state s_t.
- s_{t+1} or s': The next state, representing the state to which the system transitions after the agent takes action a_t in state s_t.

In this context, the agent represents the decision-making entity responsible for dynamically determining whether and when to perform matching operations. The environment comprises the ride-hailing or ride-pooling system, which evolves based on the agent's actions and stochastic supply-demand conditions.

The agent' s objective is to maximize the cumulative discounted reward:

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

The process of determining the matching timing can be mapped to the finite MDP components as follows:

- The **state space** $\mathcal{S}$ encodes the key attributes of the system that influence matching decisions, such as the number of unmatched passengers, the number of available drivers, and passenger waiting times.
- The **action space** $\mathcal{A}$ defines the available decisions, such as whether to perform matching or wait for additional passengers and drivers to accumulate.
- The **transition dynamics** $P(s'|s,a)$ capture the stochastic nature of passenger arrivals, driver availability, and the impact of matching operations on system state.
- The **reward function** $R(s,a)$ reflects the trade-offs between minimizing passenger wait times and reducing detour delays (for ride-pooling).

Formulating the problem as an MDP provides several benefits:

- It captures the sequential nature of decision-making, where each action influences not only the immediate outcome but also the future system state.
- It allows for dynamic and adaptive strategies that respond to real-time fluctuations in supply and demand, rather than relying on static or pre-defined rules.
- It provides a solid mathematical foundation for leveraging reinforcement learning algorithms to solve the problem efficiently.

In summary, process of determining the matching timing can be effectively framed as a finite MDP, enabling the use of reinforcement learning to develop adaptive and dynamic strategies. The following sections will detail the specific design of the state space, action space, and reward function within this MDP framework.

3.2.2 Action Space

The action space in this model is designed to be simple yet effective, consisting of a binary variable that controls the matching process. At each time step, the agent can take one of two possible actions: either perform a matching operation between the available passenger orders and drivers, or not.

Formally, the action a_t at time step t is defined as:

$$a_t \in \{0, 1\} \tag{2}$$

Where:

- $a_t = 1$: The agent performs matching at the current time step. This triggers the spatial matching algorithm, which allocates available drivers to passengers based on their current locations.
- $a_t = 0$: The agent skips matching at the current time step, allowing more passengers and drivers to accumulate, potentially leading to more efficient matches in the future.

The rationale behind this binary action space is to give the agent the flexibility to decide whether immediate matching is beneficial or if delaying the matching process could result in better overall system performance. This decision-making allows the agent to optimize the trade-off between passenger wait times and vehicle utilization. By learning from different states of the system, the agent can dynamically adjust its actions to ensure optimal matching throughout the service area.

In summary, this simple action space enables the agent to control the timing of matches, which is a critical factor in improving the efficiency of the ride-hailing and ride-pooling systems.

3.2.3 State design

As mentioned above, the goal of this study is to optimally determine when to perform matching to maximize efficiency in ride-hailing and ride-pooling services. Ideally, this decision-making problem can be formulated as a finite MDP, where the agent has complete information about the environment's state and can make decisions based on this full knowledge.

In real-world ride-hailing and ride-pooling systems, the following information is typically available:

- Current time and system clock (T_t).
- Passenger-related data, such as the number of active orders, average waiting time, maximum waiting time and their geographic distribution.
- Driver-related data, such as the number of available drivers ($N_d(t)$) and their geographic distribution.
- Historical data on past matching decisions and system performance.
- Real-time geographic information, such as traffic conditions, distance between passengers and drivers, and demand patterns across regions.

Theoretically, an MDP assumes that the future state depends only on the current state and the action taken, implying that the state contains all relevant information about the system dynamics. However, in practice, the information listed above may not fully satisfy this requirement. For instance:

- Real-time geographic information and traffic data may be incomplete or delayed.
- The system lacks visibility into future passenger requests or driver availability.
- The decision-making process may depend on unobserved variables, such as individual passenger behavior or market dynamics.
- It is difficult to represent complex passenger orders and drivers' geographical distribution using low-dimensional vectors.

These limitations mean that the problem inherently becomes a Partially Observable Markov Decision Process (POMDP), where the agent does not have full visibility of the true system state.

While a complete state representation might include all available information, utilizing such a comprehensive state is often infeasible due to practical constraints, such as computational resources, real-time data accessibility, and the high dimensionality of the input space. In this study, a simplified state representation is adopted, focusing on critical observable elements that significantly impact matching decisions. This simplification allows the framework to remain computationally efficient while retaining decision-making effectiveness.

The state at each time step t is represented as:

$$s_t = \left[T_t, \Delta T_t, N_p(t), \overline{W}_p(t), W_{\max}(t), N_d(t)\right], \tag{3}$$

where:

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

While this simplified state representation balances computational efficiency and decision-making accuracy, it does not fully capture all system dynamics or leverage the available information. Future work should consider two possible directions to enhance the state representation:

1. Incorporate additional information, such as driver geographic distribution, traffic conditions, or historical patterns, into the state representation to improve decision-making accuracy.
2. Leverage Recurrent Neural Networks (RNNs) or similar architectures to process sequential data, allowing the agent to infer unobserved system dynamics and handle partial observability more effectively.

By addressing these limitations, future research could further enhance the adaptability and robustness of reinforcement learning-based matching strategies in real-world scenarios.

3.2.4 Reward Design

Natural Reward Design

The reward function in this study is designed to optimize the agent' s decision-making in selecting the ideal matching time in ride-hailing and ride-pooling service. The reward design should accurately reflect the length of the passengers' waiting time, which consists of the components illustrated in Figure 3. In ride-hailing services, the total waiting time is divided into two parts: the waiting time to be matched, which is the time from when a passenger places an order until a driver is successfully assigned, and the waiting time to be picked up, which is the time from when a driver is assigned until they arrive at the passenger's location. In ride-pooling services, the total waiting time also includes the detour delay, which refers to the additional time caused by shared routes, as multiple passengers share a single vehicle. Therefore the primary components of the reward function of ride-hailing are the waiting time for passengers to be matched and the time passengers wait for the driver to pick up after being matched. For ride-pooling, the reward function should also include the detour delay. The agent is incentivized to minimize the total waiting times, thus improving the system's overall efficiency and passenger satisfaction.

The reward $R^h(s_t, a_t)$ at state s_t after taking action a_t for ride-hailing is defined as:

$$R^h(s_t, a_t) = -(\phi R_m(s_t, a_t) + R_w(s_t, a_t)), \tag{4}$$

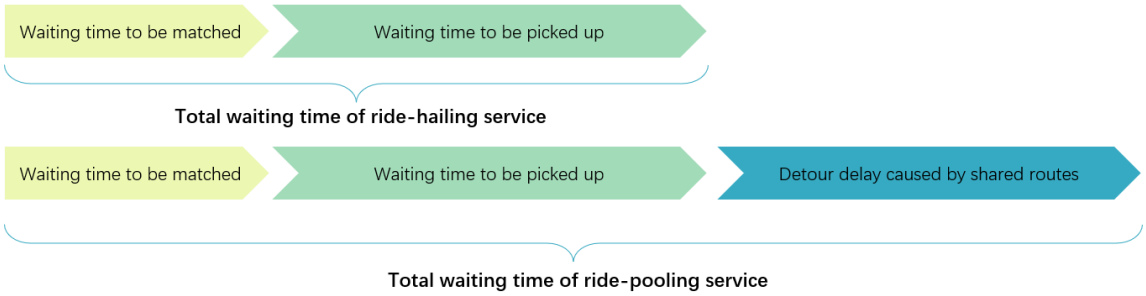

Figure 3: Components of Passenger Waiting Time in Ride-Hailing and Ride-Pooling Services

$$R_m(s_t, a_t) = \Delta t N_p^{unmatch}(s_t, a_t) \tag{5}$$

$$R_w(s_t, a_t) = \begin{cases} 0, & a_t = 0 \\ f_{pick}(s_t), & a_t = 1 \end{cases} \tag{6}$$

- $R_m(s_t, a_t)$ represents the incremental waiting time for all unmatched passengers at the current state s_t, after taking action a_t. This value is calculated at each time step, regardless of the action taken. Over time, the waiting time accumulates for unmatched passengers, meaning the sum of all previous $R_m(s_t, a_t)$ values ultimately represents the total waiting time for a passenger before being matched. Once a passenger is matched, the cumulative sum of $R_m(s_t, a_t)$ becomes their final waiting time;
- Δt represents the length of the time step;
- $N_p^{unmatch}(s_t, a_t)$ represent the number of unmatched passenger orders at state s_t after taking action a_t;
- $R_w(s_t, a_t)$ represents the waiting time for matched passengers to be picked up by drivers at the current states_t, after taking action a_t;
- $f_{pick}(s_t)$ represents the function to calculate the sum of the waiting time for matched passengers to be picked up by drivers at the current states_t, assuming a matching decision is made. The specific calculation method will be detailed in the Spatial Matching Algorithm section;
- ϕ is a weighting coefficient reflecting passengers' higher tolerance for waiting for the driver to arrive compared to the time spent waiting to be matched [38].

This weighting coefficient ϕ adjusts the significance of pre-match waiting time in the reward function, as passengers are generally more tolerant of waiting for a driver after being matched than waiting to be matched [38].

The reward structure for ride-pooling services is more complex compared to conventional ride-hailing services. This is because, in addition to the standard considerations, the system must account for the detour delays caused by combining two passenger orders into a single shared trip. This detour may result in increased delays for both passengers. Thus, the reward function $R^p(s_t, a_t)$ for the ride-pooling service at state s_t is defined as:

$$R^p(s_t, a_t) = -(\phi R_m(s_t, a_t) + R_d(s_t, a_t) + R_w(s_t, a_t)) \tag{7}$$

$$R_d(s_t, a_t) = \begin{cases} 0, & a_t = 0 \\ f_{detour}(s_t), & a_t = 1 \end{cases} \tag{8}$$

Where:

- $R_d(s_t, a_t)$ represents the detour delay caused by ride-pooling for all matched passengers at state s_t, after taking action a_t;
- $f_{detour}(s_t)$ represents the function to calculate the sum of the detour delay for matched passengers at the current states_t, assuming a matching decision is made. The specific calculation method will be detailed in the Spatial Matching Algorithm section.

Reward Shaping with Potential-Based Mechanism

A key challenge arises because $R_w(s_t, a_t))$ and $R_d(s_t, a_t)$ are only calculated when the agent chooses to initiate a matching action (i.e., $a_t = 1$). When the agent decides to wait (i.e., $a_t = 0$), it does not receive any immediate feedback regarding $R_w(s_t, a_t)$ and $R_d(s_t, a_t)$. This leads to a sparse reward problem, where the agent tends to repeatedly choose waiting as it lacks feedback on the effectiveness of matching.

To tackle this issue, *Potential-based Reward Shaping* is employed. This method was proposed by Marthi et al.[39]. This technique adjusts the immediate reward $R(s, a)$ of RL based on a potential function $\Phi(s)$, providing additional learning signals that help the agent converge faster to an optimal policy:

$$R'(s_t, a_t) = R(s_t, a_t) + \gamma\Phi(s_{t+1}) - \Phi(s_t) \tag{9}$$

Where:

- $\Phi(s)$ is the potential function, designed to reflect the desirability of the state.

This adjustment allows the agent to receive intermediate reward signals during waiting periods, helping it better navigate the trade-offs between waiting and matching. However, the introduction of Potential-based Reward Shaping can affect the total return for the entire episode:

$$\begin{aligned} G &= R(s_0, a_0) + \gamma R(s_1, a_1) + \cdots + \gamma^{T-1} R(s_{T-1}, a_{T-1}) && (10)\\ &= \sum_{t=0}^{T-1} \gamma^t R(s_t, a_t) && (11) \end{aligned}$$

$$\begin{aligned} G' &= \sum_{t=0}^{T-1} \gamma^t \left(R(s_t, a_t) + \gamma\Phi(s_{t+1}) - \Phi(s_t)\right) && (12)\\ &= G + \gamma \sum_{t=1}^{T} \gamma^{t-1}\Phi(s_t) - \Phi(s_0) - \sum_{t=0}^{T-1} \gamma^t \Phi(s_t) && (13)\\ &= G - \Phi(s_0) + \left(\gamma \sum_{t=1}^{T} \gamma^{t-1}\Phi(s_t) - \sum_{t=1}^{T} \gamma^{t-1}\Phi(s_t)\right) && (14)\\ &= G - \Phi(s_0) + \gamma^{T-1}\Phi(s_T) && (15) \end{aligned}$$

Where G represents the original total return, and G' is the total return after introducing Potential-based Reward Shaping. The difference between the original and shaped total returns lies in the potential function value $\Phi(s_t)$ at the initial state. This value is a constant and does not depend on the agent' s behavior. Therefore, during the optimization process, this shaping reward does not affect the quality of the strategy or the selection of the optimal strategy.

While Potential-based Reward Shaping helps mitigate the sparse reward problem and provides additional learning signals, it only causes a shift in the total return based on the potential values at the initial and terminal states, as shown in equation (15). This bias term introduces a certain degree of error in finite episodes, but it does not affect policy selection or the quality of the optimal policy. Therefore, PBRS can still ensure that the difference in potential values between consecutive states maintains consistency with the original reward structure, enhancing the learning process by providing more frequent reward signals.

In order to apply Potential-based Reward Shaping, the matching process can be viewed as a sequence of several actions, as shown in Figure 4. Instead of providing all the reward at the matching step, the reward is distributed over each action in the sequence.

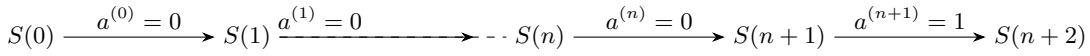

Figure 4: State transition diagram of matching process

The rewards for ride-hailing and ride-pooling after applying PBRS,$R'^h(s_t, a_t)$ and $R'^p(s_t, a_t)$, are as follows:

$$R'^h(s_t, a_t) = R^h(s_t, a_t) + \gamma\Phi^h(s_{t+1}) - \Phi^h(s_t) \tag{16}$$

$$\Phi^h(s_t) = -f_{pick}(s_t) \tag{17}$$

$$R'^p(s_t, a_t) = R^p(s_t, a_t) + \gamma\Phi^p(s_{t+1}) - \Phi^p(s_t) \tag{18}$$

$$\Phi^p(s_t) = -(f_{pick}(s_t) + f_{detour}(s_t)) \tag{19}$$

Where:

- $\Phi^h(s_t)$ represents the potential function at state s_t for ride-hailing;
- $\Phi^p(s_t)$ represents the potential function at state s_t for ride-pooling.

When $a_t = 0$:

$$\begin{aligned} R'^h(s_t, a_t=0) &= R^h(s_t, a_t=0) + \gamma\Phi^h(s_{t+1}) - \Phi^h(s_t) && (20)\\ &= -(\phi R_m(s_t, a_t=0) + R_w(s_t, a_t=0)) + \gamma\Phi^h(s_{t+1}) - \Phi^h(s_t) && (21)\\ &= -\phi R_m(s_t, a_t=0) - \gamma f_{pick}(s_{t+1}) + f_{pick}(s_t) && (22) \end{aligned}$$

$$\begin{aligned} R'^p(s_t, a_t=0) &= R^p(s_t, a_t=0) + \gamma\Phi^p(s_{t+1}) - \Phi^p(s_t) && (23)\\ &= -(\phi R_m(s_t, a_t=0) + R_w(s_t, a_t=0) + R_d(s_t, a_t=0) + \gamma\Phi^p(s_{t+1}) - \Phi^p(s_t) && (24)\\ &= -\phi R_m(s_t, a_t=0) - \gamma f_{pick}(s_{t+1}) - \gamma f_{detour}(s_{t+1}) + f_{pick}(s_t) + f_{detour}(s_t) && (25) \end{aligned}$$

When $a_t = 1$

$$\begin{aligned} R'^h(s_t, a_t=1) &= R^h(s_t, a_t=1) + \gamma\Phi^h(s_{t+1}) - \Phi^h(s_t) && (26)\\ &= -(\phi R_m(s_t, a_t=1) + R_w(s_t, a_t=1)) + \gamma\Phi^h(s_{t+1}) - \Phi^h(s_t) && (27)\\ &= -\phi R_m(s_t, a_t=1) - f_{pick}(s_t) - \gamma f_{pick}(s_{t+1}) + f_{pick}(s_t) && (28)\\ &= -\phi R_m(s_t, a_t=1) - \gamma f_{pick}(s_{t+1}) && (29) \end{aligned}$$

$$\begin{aligned} R'^p(s_t, a_t=1) &= R^p(s_t, a_t=1) + \gamma\Phi^p(s_{t+1}) - \Phi^p(s_t) && (30)\\ &= -(\phi R_m(s_t, a_t=1) + R_w(s_t, a_t=1) + R_d(s_t, a_t=1)) + \gamma\Phi^p(s_{t+1}) - \Phi^p(s_t) && (31)\\ &= -\phi R_m(s_t, a_t=1) - f_{pick}(s_t) - f_{detour}(s_t) - \gamma f_{pick}(s_{t+1}) - \gamma f_{detour}(s_{t+1}) + && (32)\\ & f_{pick}(s_t) + f_{detour}(s_t) && (33)\\ &= -\phi R_m(s_t, a_t=1) - \gamma f_{pick}(s_{t+1}) - \gamma f_{detour}(s_{t+1}) && (34) \end{aligned}$$

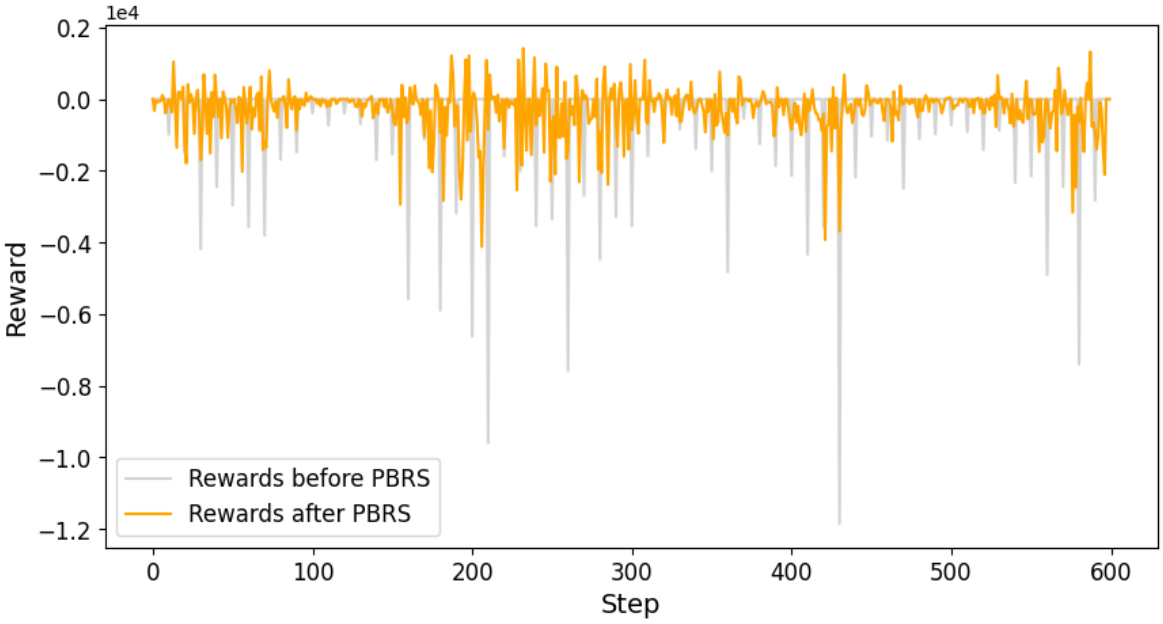

Figure 5: Comparison of Rewards Before and After PBRS

At the final time step $t = T$, the reward will also compensate for the return loss caused by PBRS to ensure that the total return remains unchanged.

Figure 5 illustrates the change in reward values before and after applying Potential-Based Reward Shaping. The gray line represents the rewards before PBRS, while the orange line shows the rewards after PBRS. As observed, the reward signals before PBRS are sparse, with large negative fluctuations at certain time steps. This sparse reward pattern can hinder the model's ability to converge quickly during training. However, after applying PBRS, the reward signals become smoother and more frequent, effectively alleviating the sparse reward issue and allowing the model to receive feedback more consistently.

This design ensures that the reward is distributed across the entire process, providing the agent with more frequent and consistent feedback. By doing so, the agent is encouraged to make more informed decisions throughout the waiting and matching process, ultimately improving the performance of the ride-hailing and ride-pooling systems.

3.3 Simulator

Following the design of the reinforcement learning framework, the next step is to create a realistic and dynamic environment in which the proposed matching time interval optimization strategy can be tested and validated. Deep reinforcement learning agents will also be trained in this environment. To achieve this, we construct a comprehensive simulator that replicates the complex dynamics of ride-hailing and ride-pooling services. The simulator serves as a critical tool for evaluating how the deep reinforcement learning (DRL) model performs in various real-world-like scenarios, where the supply of drivers and demand from passengers fluctuate over time and across different geographic regions.

The primary functionality of the simulator is to generate realistic passenger order data and driver distribution data, thereby modeling the actual supply and demand conditions in urban settings. By leveraging real historical data, the simulator captures both the spatial and temporal fluctuations in passenger demand, while also simulating driver-related variables such as their initial locations, availability, and movement patterns. This creates a robust and dynamic service network that mimics real-world complexities.

In addition to simulating the environment, the simulator incorporates a spatial matching algorithm that dynamically allocates vehicles to passengers based on factors such as geographic proximity, waiting times, and vehicle statuses. This allows the DRL-based model to be trained and tested under different matching time intervals, helping it to learn optimal matching strategies that minimize passenger wait

times and maximize vehicle utilization. Through this simulation, the system can be evaluated under various conditions, enabling the identification of the best-performing matching strategies that enhance the overall operational efficiency of both ride-hailing and ride-pooling services.

3.3.1 Assumptions of the Simulator

To construct a realistic yet computationally efficient simulation environment, several assumptions have been made in the design of the simulator. These assumptions help simplify the complex dynamics of ride-hailing and ride-pooling services, enabling the model to focus on the core aspects of the matching process. Below are the key assumptions implemented in the simulator:

- **Fixed Vehicle Speed**
  Due to lack of traffic data, all vehicles are assumed to travel at a constant speed of 40 km/h throughout the simulation. This assumption helps simplify the calculation of travel times between pick-up and drop-off locations. While real-world traffic conditions may cause variations in vehicle speeds, this fixed-speed assumption provides a reasonable approximation for evaluating the matching strategies.

- **Fixed Pick-up and Drop-off Points**
  All passenger orders are assumed to be picked up and dropped off at designated fixed points. These points serve as hubs where passengers and drivers meet. By using fixed locations, the simulator avoids the complexity of dynamically determining pick-up and drop-off locations, which streamlines the matching process and reduces computational overhead.

- **Order Cancellation Policy**
  If a passenger' s order is not matched with a driver within 5 minutes, the passenger will cancel the order. This assumption reflects the real-world impatience of passengers, who may switch to alternative transportation options if their wait time exceeds a reasonable threshold. The 5-minute window is set to balance passenger satisfaction with system efficiency.

- **Idle Driver Behavior**
  Drivers who are not currently matched with a passenger will wait at the fixed pick-up points for a maximum of 10 minutes before they begin to roam or reposition themselves within the service area. This assumption models typical driver behavior in ride-hailing services.

- **Ride-Pooling Limitation**
  ride-pooling is restricted to trips involving only two passenger orders. This simplification reduces the complexity of the ride-pooling process, making it easier to calculate detour times and matching efficiency. While real-world ride-pooling services may accommodate more than two passengers, limiting the simulator to two orders allows for a more focused study of ride-pooling dynamics and detour management.

These assumptions serve to balance the need for a realistic representation of ride-hailing operations with the computational constraints of simulating large-scale systems. Although they simplify certain aspects of the real-world system, they still capture the essential dynamics needed for evaluating and optimizing the matching process.

3.3.2 Passenger and Driver Data Generation

The simulator utilizes the publicly available historical ride-hailing order dataset (including ride-pooling order) for the Manhattan area, released by the New York City government, to generate the necessary data. This dataset includes specific details for each order, such as request time, pickup time, drop-off time, and the origin and destination zones.

Based on the dataset, a comprehensive analysis of historical data was conducted to examine the distribution patterns of driver availability and passenger demand over various time intervals and across different geographical zones. This analysis aimed to uncover temporal and spatial trends that could inform the development of a realistic simulation environment. The results showed that the occurrence of ride requests and the availability of idle drivers in any given minute is relatively infrequent within most zones. This observation suggests that such events are sporadic and do not follow a uniform

distribution throughout the day, particularly in regions with lower passenger density or outside peak hours.

To effectively model this randomness, the Poisson distribution was chosen, as it is well-suited for representing the probability of discrete events—such as ride requests or driver availability—occurring independently over a fixed time or spatial domain. The Poisson distribution is widely used in scenarios where events happen at random but at a known average rate, which aligns closely with the characteristics of the ride-hailing scenario. The sporadic nature of ride requests and driver appearances makes the Poisson distribution an ideal fit, as it can capture the unpredictable yet pattern-driven dynamics of these occurrences, particularly when confined to narrow time windows and low probabilities.

In this simulation, it is assumed that each geographical zone follows its own distinct Poisson distribution for every minute across the 24-hour period. This reflects the reality that different areas in a city, such as commercial centers, residential neighborhoods, and less dense suburbs, experience varying levels of demand and driver availability throughout the day. The parameters of the Poisson distribution, specifically the average event rate (λ), are calibrated using historical data to ensure that the simulator accurately reflects real-world conditions. This λ value dictates the expected frequency of ride requests and idle driver occurrences in each zone, adjusting dynamically based on the time of day and location.

By leveraging these calibrated Poisson distributions, the simulator can generate a realistic number of new passenger orders and idle drivers for each zone every minute. For each new event, the simulator randomly assigns them to one of the predefined pickup points within the zone, simulating the randomness of real-world ride-hailing scenarios.

Moreover, by adjusting the λ parameter, the simulator can also replicate different urban settings and test how the ride-hailing system performs under various conditions. For example, during peak demand hours, the event rate for passenger orders in commercial areas would increase, reflecting the real-world surge in ride requests, while in off-peak hours, the event rate would decrease, simulating quieter periods. Similarly, driver availability can be adjusted based on the patterns observed in the historical data, ensuring that the supply of drivers fluctuates in a realistic manner.

This combination of spatial and temporal event generation ensures that the simulator is not only dynamic but also closely mimics the real-world complexity of ride-hailing systems, where demand and supply constantly shift throughout the day and across different zones.

In addition to the origin information, accurate modeling of the destination data for passenger orders is crucial for creating a realistic simulation of ride-hailing services. Passenger trips are inherently directional, meaning that where a ride begins and ends plays a significant role in shaping both the demand patterns and the operational efficiency of the system. To account for this, a transition probability matrix was developed based on an in-depth analysis of the historical order data.

The transition probability matrix captures the likelihood of passengers traveling from one geographical zone to another within the city. Each entry in the matrix represents the proportion of trips that originate in a specific zone and end in another zone, calculated as the ratio of trips between these two zones relative to the total number of trips from the origin zone. This approach not only reflects the spatial distribution of demand but also incorporates the varying popularity of different destinations, such as high-demand commercial districts, transportation hubs, or residential areas.

By analyzing the historical data, trends such as common commuting routes, frequent trips to business districts during peak hours, or popular evening destinations can be identified and encoded within the matrix. The matrix allows for the assignment of probability-weighted destinations to each newly generated passenger order in the simulation, ensuring that the movement patterns of passengers mirror those observed in the real-world dataset. For example, if a specific origin zone has a higher probability of trips ending in a neighboring commercial area, that likelihood is reflected in the destination assigned to each new ride request originating from that zone. This transition probability matrix thus enables the simulator to generate realistic destination data for each passenger order, ensuring that the flow of traffic in the simulation mirrors real-world patterns.

Furthermore, the simulation accounts for the dwell time of idle drivers at pickup points and the waiting time of passenger orders. This helps simulate the roaming behavior of idle drivers in search of passengers and the possibility of order cancellations due to extended wait times. In the current iteration of the simulator, the idle driver dwell time is assumed to be 10 minutes, while passenger orders have a waiting time of 5 minutes.

The final generated passenger data includes the precise order request time, the potential future

cancellation time, the specific pickup point, and the designated destination. Driver data includes the current location, the exact time of arrival at that location, and the expected time the driver will become unavailable.

This data generation method equips the reinforcement learning algorithm with a dynamic and uncertain environment, which is critical for optimizing decision-making models. The simulator provides second-by-second data on orders and drivers, creating continuous decision points based on fluctuating supply and demand conditions. This allows the reinforcement learning algorithm to be trained and evaluated under various scenarios. Additionally, by adjusting parameters, this setup creates a rich array of demand-supply fluctuation scenarios, facilitating the testing and evaluation of the algorithm's adaptability to real-world conditions.

3.3.3 Spatial Matching Algorithm

This study will address the spatial matching problems for both conventional ride-hailing and ride-pooling services, with distinct matching algorithms employed for each. In ride-pooling, compared to conventional ride-hailing, the challenge lies in matching passenger orders that have similar origins and destinations, whereas conventional ride-hailing only focuses on matching a single passenger order with a nearby driver. Therefore, the ride-pooling process is divided into two stages in this study: passenger-passenger matching and passenger-driver matching. The passenger-driver matching stage is consistent with the process used in conventional ride-hailing and will be described uniformly in the subsequent sections.

Passenger-Passenger Match (ride-pooling)

In this phase of the simulation, the algorithm enumerates all viable matches between passenger orders within the current time step. The primary objectives are twofold: to maximize the number of successfully matched passenger orders while minimizing the negative impact of detours caused by shared rides on passenger satisfaction. The feasibility of matching two distinct passenger orders is evaluated based on two key criteria.

Firstly, the intersection of their respective time windows must include the current time step, as defined by the simulator. This ensures that both passengers are available for pickup and that the timing aligns for a feasible ride-sharing arrangement. If the time windows do not overlap sufficiently, the match will be deemed unfeasible, as it would either result in excessive wait times or missed pickups.

Secondly, the algorithm assesses the additional delay that would be introduced due to detouring for ride-sharing purposes. In ride-pooling systems, a key challenge is managing the additional travel time introduced by detouring to accommodate multiple passengers. Minimizing this added travel time is essential, as significant detours can lead to passenger dissatisfaction, particularly if the detour results in a much longer trip than originally anticipated. To effectively measure the efficiency of a ride-sharing arrangement, the **Detour Delay Rate (DDR)** is introduced. This metric quantifies the additional delay caused by detouring in relation to the direct travel distance that would have been taken without sharing the ride. The DDR is calculated using the following formula:

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

In this formula, the "Distance without detour" represents the total distance the vehicle would have traveled if it had only served one passenger without any additional stops, while the "Distance after detour" refers to the actual distance traveled with multiple passengers, considering the detours required to pick up and drop off each passenger.

The DDR value inversely reflects the efficiency of the detour:

- **A higher DDR value** indicates a favorable detour scenario, where the added travel distance is minimal. In this case, the detour has a relatively low impact on the passengers' overall trip time, making the shared ride experience more efficient and less disruptive. This generally leads to higher passenger satisfaction, as the detour minimally impacts their expected arrival time.
- **A lower DDR value** suggests that the detour has introduced a significant additional travel distance. This can result in longer delays for one or more passengers, making the ride-sharing

experience less attractive. Such matches are less efficient, as the detour negatively affects the passengers' experience.

The DDR serves as a guiding metric for the ride-sharing algorithm, allowing it to prioritize matches that result in higher DDR values. This ensures that the system focuses on minimizing the negative effects of detouring while still maximizing the benefits of ride-sharing, such as reduced costs and lower vehicle emissions. By optimizing for high DDR values, the algorithm effectively balances operational efficiency with passenger satisfaction. Utilizing the DDR metric enables the ride-sharing system to intelligently manage detours, ensuring that the added travel time remains within acceptable limits for passengers while optimizing the overall efficiency of the service.

The passenger-passenger matching process in the simulation follows a series of structured steps, designed to ensure that potential matches are systematically evaluated and optimized. The steps include:

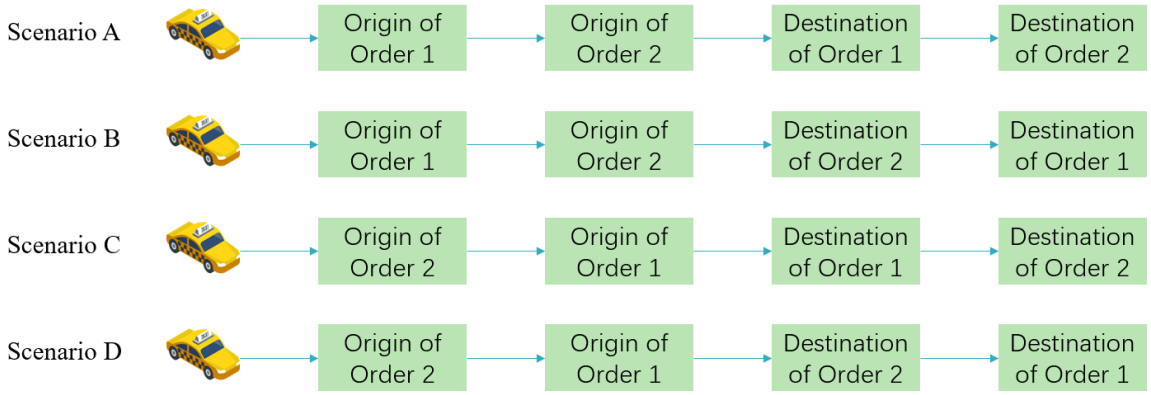

Figure 6: All possible pickup and drop-off sequences for two passenger orders

a. Extract Relevant Passenger Orders: At the beginning of the passenger-passenger matching process, all active passenger orders whose time windows encompass the current timestep, as defined by the simulator, are extracted from the passenger order database. Along with each order, the relevant details—such as the origin, destination, requested pickup time, and current passenger location—are also retrieved. This ensures that only passengers who are available for a feasible match during the current simulation cycle are considered, optimizing computational efficiency and ensuring real-time responsiveness.

b. Once the relevant passenger orders are identified, the algorithm generates all possible passenger-passenger pairings within this set. For each pair, the **Detour Delay Rate (DDR)** is calculated to evaluate both the feasibility and the efficiency of the shared ride.

The DDR value helps to assess how much additional travel distance a detour will introduce when combining two passenger orders. Lower DDR means greater detour distance, while a higher DDR indicates that the two shared passenger orders have similar routes. To determine whether the detour remains within acceptable limits, the DDR is calculated based on the distances involved in different potential routes. DDR can be categorized into four distinct scenarios, evaluated based on the relative positions of passengers and their destinations, to determine the pickup and drop-off routes between two orders (as shown in Figure 6).

The formula for calculating DDR is as follows:

$$\begin{aligned}\mathrm{DDR}((O_1,D_1),(O_2,D_2)) = \max\Bigg(&\min\left(\frac{d(O_1,D_1)}{d(O_1,O_2,D_1)},\frac{d(O_2,D_2)}{d(O_2,D_1,D_2)}\right),\\ &\frac{d(O_1,D_1)}{d(O_1,O_2,D_2,D_1)},\frac{d(O_2,D_2)}{d(O_2,O_1,D_1,D_2)},\\ &\min\left(\frac{d(O_1,D_1)}{d(O_1,D_2,D_1)},\frac{d(O_2,D_2)}{d(O_2,O_1,D_2)}\right)\Bigg)\end{aligned} \tag{35}$$

Where:

- O_1 and D_1 represent the origin and destination of passenger 1, and O_2 and D_2 represent the origin and destination of passenger 2.
- $d(\cdot)$ represents the total shortest path distance between these points, depending on the sequence of the pick-up and drop-off locations.

Each part of the formula evaluates a different potential route for the shared ride. Below are the specific considerations for each case:

(a) Situation A: The first part of the equation, $\min\left(\frac{d(O_1,D_1)}{d(O_1,O_2,D_1)}, \frac{d(O_2,D_2)}{d(O_2,D_1,D_2)}\right)$, compares the detour incurred when the vehicle first picks up passenger 1 and then passenger 2, followed by completing both trips. It measures how efficiently the route minimizes detour for both passengers.

(b) Situation B: The second term, $\frac{d(O_1,D_1)}{d(O_1,O_2,D_2,D_1)}$, evaluates the efficiency of a route where passenger 2' s destination is visited before completing passenger 1's trip. This sequence assesses how much delay is introduced if passenger 2 is dropped off first.

(c) Situation C: $\frac{d(O_2,D_2)}{d(O_2,O_1,D_1,D_2)}$, similarly evaluates the situation where passenger 1's destination is reached before completing passenger 2's trip.

(d) Situation D: Finally, the $\min\left(\frac{d(O_1,D_1)}{d(O_1,D_2,D_1)}, \frac{d(O_2,D_2)}{d(O_2,O_1,D_2)}\right)$ term compares the detour incurred if the two trips follow a different sequence of drop-offs, where each route attempts to minimize the detour for one passenger relative to the other.

Once the DDR value is calculated, the algorithm can evaluate whether the detour is reasonable, ensuring that both passengers experience minimal additional delay. If the DDR value is too low, it indicates that the detour introduces significant inefficiencies, and the ride-pooling match may not be feasible. The goal is to select passenger pairs with high DDR values to ensure that the shared ride remains efficient and satisfactory for both passengers.

For each selected pair, the simulator records the optimal sequence of pick-up and drop-off points, the shared time window, and the calculated DDR value. This information is then used in subsequent steps to finalize the matching and optimize the overall route for the shared ride.

c. The enumeration of all feasible passenger-passenger pairs inevitably leads to situations where certain passenger orders can be matched with multiple other orders. In such cases, it is crucial to ensure that as many passengers as possible are successfully paired while also prioritizing the pairs with higher DDR values, as these represent more efficient and satisfactory ride-sharing arrangements. This challenge can be formulated as an integer linear optimization problem.

The set of enumerated feasible passenger-passenger pairs can be represented as a larger undirected graph $G = (P, E)$, where P denotes the set of vertices corresponding to passenger orders, and E represents the set of edges, corresponding to the feasible passenger-passenger pairs. Each edge is assigned a weight $\omega(e)$, which corresponds to the DDR value of the passenger pair. The objective of the optimization is to decompose the undirected graph into as many edge pairs as possible while maximizing the total weight of the selected edges, ensuring that the most efficient matches (with the highest DDR values) are prioritized.

To achieve this, we introduce a binary decision variable $x(e)$ for each edge e. If the edge e is selected as part of the decomposition, then $x(e) = 1$; otherwise, $x(e) = 0$. The optimization problem is thus focused on maximizing the total weight of the selected edges while ensuring that no passenger order is matched more than once.

The specific mathematical model is as follows:

$$\max \sum_{e \in E} \omega(e) \cdot x(e) \tag{36}$$

$$\sum_{e \in \delta(v)} x(e) \leq 1, \tag{37}$$

$$x(e) \in \{0, 1\}, \tag{38}$$

$$\forall v \in P, \forall e \in E \tag{39}$$

The solution result is all the optimal passenger-passenger pairs PP. The detour delay for each passenger's order within the passenger-passenger pairs will also be calculated, and their sum constitutes the output of the $f_{\text{detour}}()$ function.

The Passenger-Passenger Match algorithm ultimately outputs the matching results for all ride-pooling services at the current time step, identifying all possible pairs of passenger orders that can be matched. For each successful match, the algorithm records the detour delay experienced by the passengers due to sharing the trip. This detour delay is a key metric that helps evaluate the overall efficiency of the ride-sharing process and ensures that the additional time incurred is within acceptable limits.

Moreover, the resulting passenger-passenger pairs are treated as single entities and will proceed to the next stage, the Passenger-Driver Match phase, where they will be matched with available drivers. In this phase, each passenger pair is treated as a single request, with their combined origin and destination considered in the driver matching process. This ensures that the algorithm can seamlessly integrate the results from the passenger matching phase into the larger ride-hailing system, optimizing the overall efficiency of both shared and individual rides.

Passenger-Driver Match

This phase encompasses the entire matching process for regular ride-hailing services and serves as the second stage of ride-pooling. The primary objective of this phase is to efficiently assign passengers (or passenger pairs, in the case of ride-pooling) to the nearest available driver, minimizing the wait time for passengers. The challenge of matching passengers to drivers in real-time is formulated as an Integer Linear Programming (ILP) problem, which seeks to optimize the spatial allocation of available drivers to waiting passengers.

In this phase, the algorithm aims to match each passenger or passenger pair to the most suitable driver based on proximity, considering factors such as the travel distance between the driver and the pickup location and the availability of drivers in nearby zones. By framing the problem as an ILP, the algorithm can systematically evaluate multiple possible driver-passenger assignments, selecting the one that results in the most efficient spatial distribution of resources. This approach allows the system to account for real-world complexities, such as the dynamic nature of supply and demand, traffic fluctuations, and varying trip lengths, ensuring that the ride-hailing service operates smoothly even in high-demand scenarios.

The specific formulation of this optimization problem is presented as follows:

$$\min \sum_{i=1}^{n} \sum_{j=1}^{n} T_{ij} \cdot x_{ij} \tag{40}$$

$$\sum_{i=1}^{n} x_{ij} \leq 1, \quad \forall j \in P \tag{41}$$

$$\sum_{j=1}^{n} x_{ij} \leq 1, \quad \forall i \in D \tag{42}$$

$$x_{ij} \in \{0, 1\}, \quad i \in D,\ j \in P \tag{43}$$

where T_{ij} is the pickup time between driver $i \in D$ and passenger (passenger pair) $j \in P$, estimated by their Manhattan distance divided by the average pickup speed v. x_{ij} is a binary decision variable. When driver i is selected to pick up passenger (passenger pair) j, $x_{ij} = 1$; otherwise, $x_{ij} = 0$. The optimized result, $\sum_{i=1}^{n} \sum_{j=1}^{n} T_{ij} \cdot x_{ij}$, represents the output of the $f_{\text{pick}}()$ function.

3.3.4 Verification Of the Simulator

This study will visualize both Passenger-Passenger Match (ride-pooling in ride-pooling services) and Passenger-Driver Match to verify the spatial matching algorithm's correctness and effectiveness under different scenarios.

Firstly, for the Passenger-Driver Match, visualizing the matching results between passengers and drivers can intuitively display the spatial relationship between matched passengers and their corresponding drivers. Using a map or two-dimensional coordinate system, the initial locations of passengers and drivers will be represented by different symbols, and the matching results will be connected by lines. This visualization clearly illustrates the distances and distributions between the two during the matching process, helping to verify the algorithm' s correctness.

Secondly, for the Passenger-Passenger Match (ride-pooling in ride-pooling), the visualization will further demonstrate the situation where multiple passengers share the same vehicle. By visualizing the routes of ride-pooling passengers, not only can the pick-up and drop-off points of each passenger be displayed, but the detours caused by ride-pooling can also be clearly seen. The focus of ride-pooling matching visualization includes:

- **Ride-pooling route rationality**: By displaying the travel paths of multiple passengers, the system can be verified to determine whether it has selected the optimal ride-pooling combination and whether passengers with similar destinations have been reasonably matched to reduce detour-related delays.
- **Detour costs**: After visualizing the ride-pooling route, the extra delay caused by the detour can be intuitively observed, and it can be checked whether this delay falls within an acceptable range.

These visualized results will intuitively reveal the spatial layout of matching decisions, thereby verifying the algorithm's effectiveness and correctness in both ride-hailing and ride-pooling environments.

The visualization of the Passenger-Driver Match (as shown in Figure 7) provides clear evidence of the spatial matching algorithm's effectiveness in optimizing ride-hailing services. The results show that passengers are matched with geographically proximate drivers, minimizing the initial distance between them and reducing both passenger wait time and driver idle time. The red routes connecting passengers and drivers demonstrate efficient path planning, with no significant detours or excessive travel distances, indicating that the algorithm optimally selects routes to further improve system efficiency. The even distribution of matched pairs across the map suggests that the algorithm can handle varying demand across different geographic regions effectively. Overall, the visualization supports the algorithm's capability to deliver efficient, real-time matching in dynamic urban environments.

The visualization of the Passenger-Passenger Match (as shown in Figure 8) illustrates the effectiveness of the ride-pooling algorithm in combining two passenger trips into a single, efficient route. The blue and orange markers represent the origins and destinations of the two passengers. The red line shows the shared route connecting both trips. The visualization indicates that the algorithm has successfully minimized detours by selecting passengers with similar travel patterns. The proximity between the origins and destinations demonstrates the algorithm's efficiency in matching passengers whose trips align spatially. Furthermore, the shared route is relatively direct, suggesting that the ride-pooling process added minimal extra travel distance, ensuring an efficient and cost-effective ride for both passengers. Overall, the visualized route confirms the algorithm's capability to optimize shared trips by reducing detour times and enhancing route efficiency.

To validate the feasibility of the simulator, the next step is to implement a fixed time interval strategy within the simulator. The performance of the simulator will be evaluated by calculating and analyzing several key metrics: the total waiting time for all passenger orders to be matched, the total waiting time for passengers to be picked up by drivers, and, in the case of ride-pooling services, the total detour delay caused by ride-sharing. These metrics will be compared against real-world data to assess whether the simulator can accurately represent the dynamic conditions of both ride-hailing and ride-pooling services. By comparing the calculated values with actual operational data, it will be possible to determine if the simulator accurately reflects real-world scenarios, thus validating its feasibility as a research tool.

In particular, the fixed time interval strategy will accumulate passenger orders and available drivers over a set time period before performing the matching process. This allows for the collection of

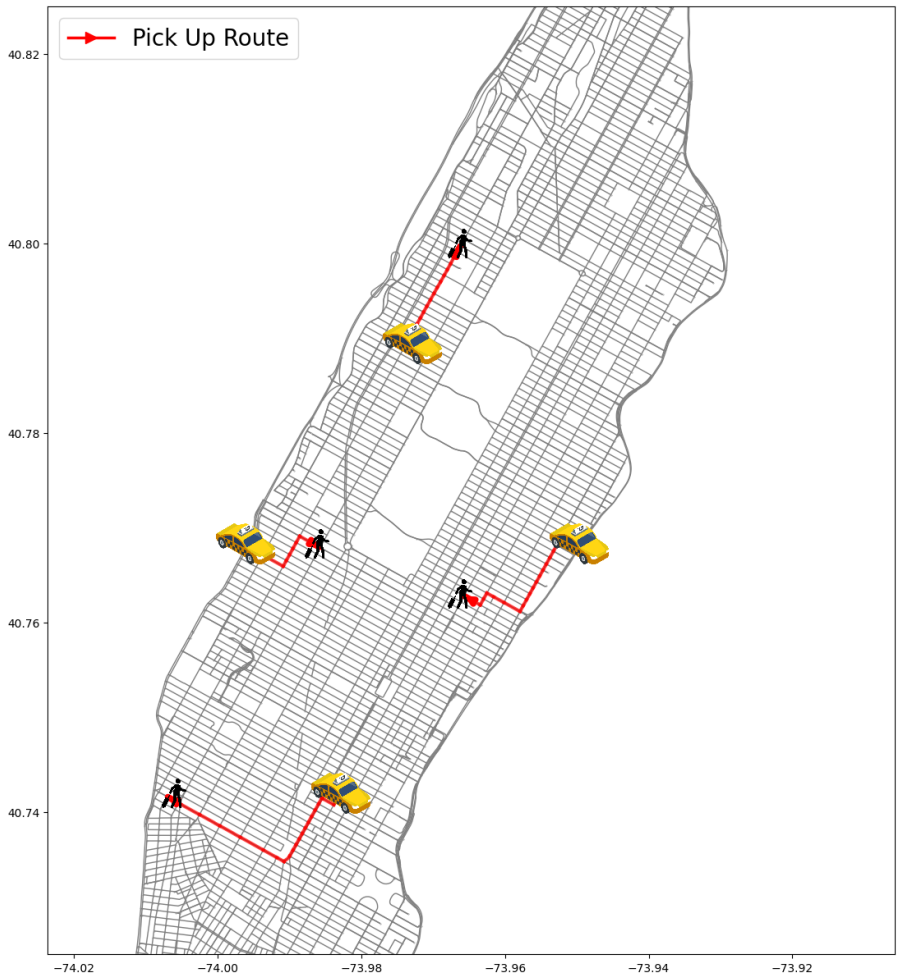

Figure 7: Visualization of Passenger-Driver Match results

sufficient data to compute the total matching wait time, the time passengers spend waiting for their drivers after being matched, and, in ride-pooling scenarios, the additional delay due to detours when multiple passengers share the same vehicle. By ensuring that the simulation outputs align closely with real-world data, the simulator' s ability to model complex ride-hailing systems can be confirmed. This process will demonstrate that the simulator is not only capable of generating realistic matching and detour outcomes but is also a valid platform for testing and optimizing different ride-hailing strategies.

The results from Figures 9 and 10 show how different fixed time intervals for matching impact key performance metrics in both ride-hailing and ride-pooling simulations.

For ride-hailing (Figure 9), the simulator results show gradual changes in total waiting time across different matching time intervals, reflecting the simulator' s dynamic response to the supply-demand relationship in ride-hailing services. As seen in the figure, the impact of matching intervals on waiting time follows the expected trend: as the matching interval increases, the system can accumulate more orders, thereby reducing fluctuations in total waiting time. This suggests that the simulator can, to some extent, replicate the optimization processes found in real ride-hailing services under dynamic supply-demand conditions, aligning with real-world operational logic. The figure also shows that as the matching interval increases, the waiting time to be matched gradually rises, consistent with the order accumulation process in real scenarios. Meanwhile, the waiting time for a driver to arrive decreases, although this decline gradually slows with longer intervals, which is also in line with actual service dynamics. This reasonable distribution of waiting times demonstrates the simulator' s ability to accurately represent different components of waiting time in a dynamic ride-hailing environment, validating its capability to handle complex waiting time distributions. Overall, the ride-hailing simulator effectively replicates real-world dynamic ride-hailing scenarios, including the impact of matching

Figure 8: Visualization of Passenger-Passenger Match results

intervals on waiting time and the reasonable distribution of waiting times. These features confirm the simulator' s capability to accurately model dynamic ride-hailing services.

For ride-pooling (Figure 10), the simulator results show that as the matching time interval increases, the total waiting time generally decreases, with a particularly noticeable effect in the initial few seconds of interval adjustment. This aligns with the dynamics of real ride-pooling services, where increasing the time interval allows the system to accumulate more orders, resulting in better matches through batch processing. As the matching interval increases, the waiting time to be matched gradually rises, while the waiting time to be picked up shows a gradual downward trend, and the detour delay, specific to ride-pooling, also decreases. These trends are consistent with real-world logic. The figure further shows that as the matching interval increases, the total waiting time gradually stabilizes, reaching an optimal saturation point around 16 seconds. This reflects the marginal optimization effect observed in real ride-pooling scenarios: initially, increasing the matching interval significantly improves efficiency, but beyond a certain threshold, the gains diminish. This pattern suggests that the simulator' s matching strategy optimization closely aligns with real-world needs, balancing efficiency with passenger waiting experience. In sum, the ride-pooling simulator effectively replicates real-world dynamic ride-pooling scenarios, including dynamic responses to supply-demand fluctuations, a reasonable distribution of waiting times, marginal optimization effects, and realistic detour delays. These features demonstrate the simulator' s strong capability to accurately model supply-demand changes and optimization effects in dynamic ride-pooling services.

Overall, the results confirm that the simulator realistically models both ride-hailing and ride-pooling services. The fixed matching time intervals can indeed reduce passengers' total waiting time to some extent.

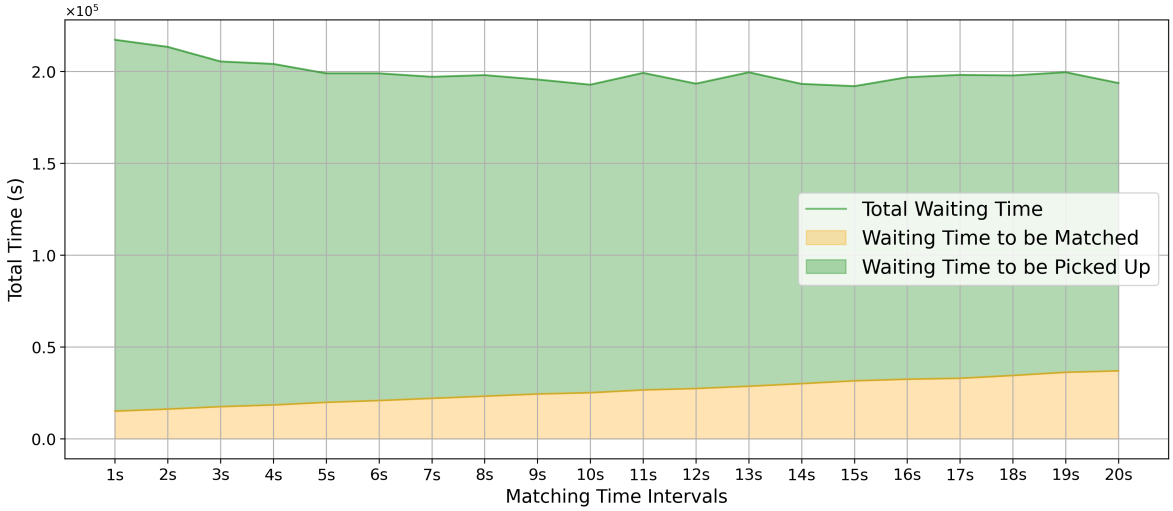

Figure 9: Fixed Time Interval Strategy in the Ride-Hailing Simulator

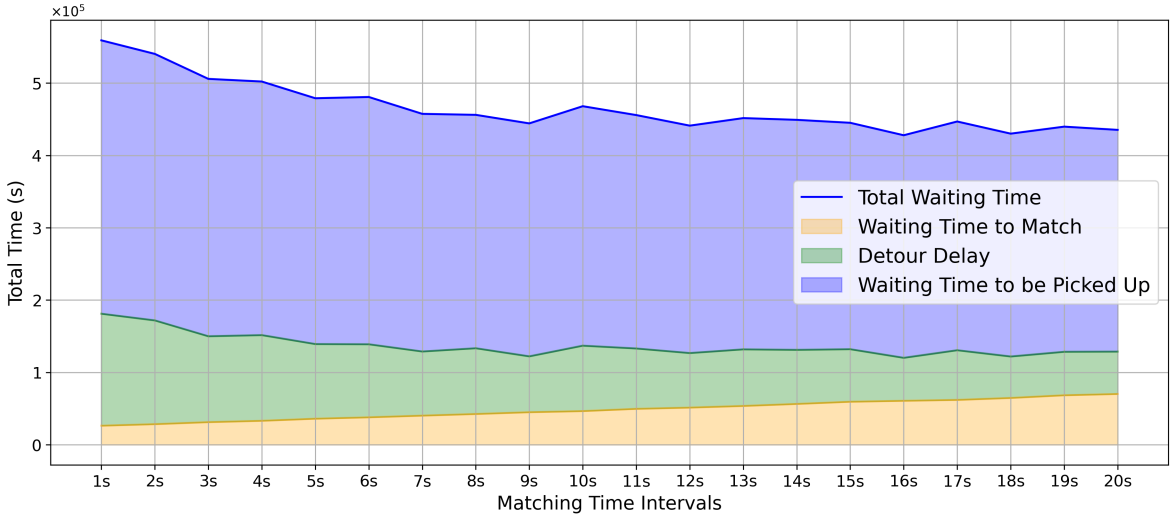

Figure 10: Fixed Time Interval Strategy in the Ride-Pooling Simulator

3.4 The Proximal Policy Optimization (PPO) Algorithm

The Proximal Policy Optimization (PPO) algorithm is employed in this study to address the optimization of matching time intervals in (shared) ride-hailing services. PPO is a reinforcement learning algorithm proposed by Schulman et al.[40]. PPO is chosen due to its balance between simplicity, computational efficiency, and robust performance, making it particularly suitable for dynamic and complex environments such as ride-hailing systems.

PPO is a reinforcement learning algorithm that provides several key advantages. First, it ensures stability through its clipped objective function, which prevents overly large updates to the policy. This feature is crucial in highly dynamic environments like ride-hailing services, where rapid changes in supply and demand conditions require stable learning. Second, PPO promotes efficient exploration while controlling risk, enabling the learning agent to adapt its decisions to a wide range of scenarios and achieve optimal performance under varying conditions. Additionally, PPO' s simplicity in implementation, combined with its ability to scale to large problems, makes it a practical choice for optimizing large-scale simulations such as those involving ride-hailing services.

In the context of this study, PPO is used to train the agent to optimize the decision-making process for determining the best matching time intervals. The agent operates in a dynamic environment where it adjusts its actions based on real-time information about passenger demand, driver availability, and

system delays. The goal is to minimize passenger wait times, reduce driver idle time, and handle the complexities introduced by ride-pooling, such as additional detour delays.

The implementation of PPO in this study is based on a neural network architecture within the actor-critic framework, forming a deep reinforcement learning model that facilitates both policy learning and evaluation simultaneously. By integrating PPO with a deep neural network, the agent can effectively handle high-dimensional and complex spatiotemporal features, making it well-suited for optimizing the dynamic environment of ride-hailing and ride-pooling services. The neural network architecture enables the agent to approximate both the policy (actor) and value function (critic) more accurately, capturing intricate patterns in supply-demand fluctuations.

The agent' s neural network consists of two primary components: an actor network and a critic network, both structured as multi-layer perceptrons (MLPs) with three hidden layers of 64 units each, activated by the Tanh function. The critic network estimates the value function for a given state, outputting a single scalar value that represents the expected cumulative reward. The actor network outputs a probability for the agent' s actions, which is derived from a sigmoid activation function and then bounded between a minimum and maximum threshold to avoid extreme values. This probability determines the action distribution, where the agent samples its actions from a Bernoulli distribution to choose dynamically between waiting and matching actions.

In this setup, the agent' s probabilistic action selection enables it to make robust decisions at each decision point by exploring various policies and adjusting its actions based on observed outcomes. The actor-critic framework ensures that policy updates remain stable and efficient, with the critic providing feedback to guide the actor's policy improvements, while PPO' s clipped objective ensures controlled updates to the policy for stable learning.

Furthermore, performance metrics are tracked using tools like Weight & Bias to monitor the evolution of the agent's policy over time. This tracking not only enables observation of key metrics, such as the reduction in passenger delays, but also ensures that the model continuously balances trade-offs and achieves higher system efficiency. By combining PPO with this neural network architecture, this deep reinforcement learning approach is able to learn optimal matching strategies that adapt to dynamic changes, offering a significant advantage over traditional reinforcement learning methods.

Overall, PPO Combined with MLPs is well-suited for optimizing dynamic and large-scale problems like ride-hailing, where the goal is to enhance system efficiency by learning adaptive, real-time strategies.

The basic flow of the PPO algorithm as implemented in this study is outlined in the following pseudocode:

This pseudocode highlights the overall structure and flow of the PPO algorithm, illustrating how it operates within the dynamic environment of ride-hailing services. The combination of policy optimization, value estimation, and gradient updates enables the agent to effectively learn strategies for optimizing matching time intervals and improving system efficiency.

4 Result

4.1 Experimental Setup and Metrics

To validate the practical effectiveness of the proposed reinforcement learning-based dynamic matching time interval optimization strategy, three experiments were designed, each addressing distinct research objectives and system performance evaluations:

Experiment 1: Training Performance Evaluation of the RL Strategy
This experiment aims to assess the training performance of the reinforcement learning (RL) strategy under different service modes (Ride-Hailing and Ride-Pooling). By analyzing learning curves and comparing final strategy outcomes, the role of Potential-Based Reward Shaping (PBRS) in accelerating convergence and enhancing strategy performance is examined. The detailed experimental setup and results for this experiment are presented in **Section 5.2**.

Experiment 2: Comparison with Baseline Strategies
To demonstrate the superiority of the RL strategy in practical scenarios, this experiment compares the performance of three matching strategies: fixed-time interval matching, real-time matching, and the RL strategy. Key performance metrics are analyzed to explore how the RL strategy significantly

Algorithm 1 PPO Algorithm

for episode = 1 to E **do**
  Reset the ride-hailing simulator and get initial state $s_0 = \{N_t^D, N_t^R\}$
  Initialize episode return $R = 0$ and $done = False$
  Compute noise factor $p = \frac{\text{episode}}{\text{max\_episodes}}$
  for $t = 1$ to T **do** ▷ Action selection
    $a_t = p \cdot \mu_\theta(s_t) + (1 - p) \cdot \text{noise}$ ▷ Compute primal action
    $rescaled\_a_t = \text{scale\_action}(a_t)$ ▷ Re-scale action
    Pass $rescaled\_a_t$ to simulator
    ▷ Ride-matching logic
    Create matching pool with $rescaled\_a_t$
    Execute matching
    Transition to next state $s_{t+1} = \{N_{t+1}^D, N_{t+1}^R\}$
    Observe reward r_t and done status d_t
    ▷ Store transitions
    Normalize states s_t and s_{t+1}
    Store $(s_t, a_t, r_t, s_{t+1}, d_t)$ in replay buffer R
    if $r_t >$ good_action_threshold **then**
      Store $(s_t, a_t, r_t, s_{t+1}, d_t)$ in good replay buffer R'
    end if
    ▷ Training logic
    if memory size > warm-up size **then**
      Sample minibatches from R and R'
      Update critic network by minimizing value loss
      Update actor network using policy gradient
      Update target networks:
      $\theta' \leftarrow \tau \cdot \theta + (1 - \tau) \cdot \theta'$
      $\omega' \leftarrow \tau \cdot \omega + (1 - \tau) \cdot \omega'$
    end if
    if *done* **then**
      break
    end if
  end for
  Reset noise
end for

improves system efficiency. The experimental setup and results for this experiment are elaborated in **Section 5.3**.

Experiment 3: Cross-Comparison of Ride-Hailing and Ride-Pooling
This experiment focuses on comparing the efficiency and user experience of the two service modes. Using several experimental scenarios (based on driver-to-order ratios and travel modes), the unique characteristics of the two services and the RL strategy's effectiveness in optimizing their matching efficiency are analyzed. The primary goal is to conduct a cross-comparison of Ride-Hailing and Ride-Pooling while evaluating the generalizability of the RL strategy, particularly under imbalanced supply-demand conditions. The details of this experiment are provided in **Section 5.4**.

Through these three experiments, this study aims to comprehensively evaluate the performance of the RL strategy, verify its advantages in dynamic supply-demand environments, and explore its adaptability to different service modes.

A baseline strategy based on an optimal fixed time interval is employed in all above three experiments to compared with the RL strategy. In the traditional ride-hailing simulator, the baseline strategy has a fixed matching interval of 15 seconds, while in the ride-pooling simulator, the interval is set to 20 seconds. These intervals were determined through testing within the simulator, ensuring that the baseline strategy is representative across performance metrics.

In the traditional ride-hailing service, model performance is evaluated using the following 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 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 Waiting Time**: The cumulative waiting time of all passengers, measuring overall waiting.

In the ride-pooling service, two additional metrics are introduced to account for the extra impact of ride-pooling:

- **Average Detour Delay**: The average delay time per passenger due to detours, evaluating the effectiveness of ride-pooling strategies.
- **Total Detour Delay**: The total delay caused by detours in ride-pooling.

These evaluation metrics allow a comprehensive analysis and comparison of the model performance under different reward designs, revealing how the batched matching strategy optimizes system efficiency and improves user experience in both traditional ride-hailing and ride-pooling environments.

In all reinforcement learning experiments described in this study, an *episode* refers to a single simulation cycle that begins from an initial system state and ends upon meeting predefined termination conditions. For the ride-hailing and ride-pooling scenarios examined in this research, an episode spans a fixed simulation period representing 10 minutes of real-world ride-hailing or ride-pooling service, during which there are 600 discrete time steps. Throughout the episode, the RL agent interacts with the simulated environment, making decisions on whether and when to perform matching operations. The agent's performance in each episode is evaluated using the metrics outlined earlier.

The concept of an episode is fundamental to training RL agents. The agent learns optimal strategies by repeatedly interacting with the environment across multiple episodes. Each episode provides the agent with new opportunities to explore and refine its decision-making policies, with the ultimate goal of maximizing system performance under varying supply-demand conditions. When testing the strategies developed during training, the agent's performance is assessed over multiple episodes to ensure robustness and generalizability.

To introduce variability and randomness, different experiments generate distinct datasets of passenger orders and driver distributions, creating diverse episodes for the agent to interact with based on specific experimental requirements. This variability ensures that the agent is exposed to a wide range of supply-demand conditions, enabling it to develop robust and generalized decision-making strategies.

Furthermore, by evaluating the agent's performance across multiple episodes, the experiments provide insights into the consistency and reliability of the learned policies, ensuring that the strategies are not only effective in isolated cases but also adaptable to diverse and dynamic environments.

4.2 Training Performance

The experiments in this study simulate traditional Ride-Hailing and Ride-Pooling services in Manhattan during peak traffic hours (8:30–8:40 a.m.). Each service mode is trained separately over 4,800 episodes (equivalent to 2,880,000 steps). This time period and episode length were chosen to reflect a highly dynamic supply-demand environment, ensuring the model learns under realistic and challenging conditions. Passenger requests and idle driver locations are generated based on a Poisson distribution.

During training, the agent experiences variability across episodes. While passenger requests and driver data are generated based on the Poisson distribution, inherent randomness ensures that no two episodes are identical, allowing the agent to adapt to different supply-demand conditions. This variability enhances the robustness of the learned strategy.

To evaluate the impact of Potential-Based Reward Shaping (PBRS) on learning efficiency, two reward designs are compared: one incorporating PBRS and one without.

To ensure the reliability and generalizability of the results, the training experiments were repeated five times using different random seeds (10, 20, 30, 40, and 50). The use of random seeds plays a critical role in mitigating the inherent randomness in ride-hailing environments and reinforcement learning (RL) training processes. Specifically:

- **Reproducibility**: By setting random seeds, the environment, data sampling, and initialization of neural network weights are controlled, ensuring that the same sequence of random numbers is used during execution. This guarantees that the results can be accurately reproduced, enabling consistent evaluation of the model's performance under identical experimental conditions.
- **Comprehensive Evaluation**: Different random seeds simulate variations in the ride-hailing system, such as driver and passenger distributions, request times, and other stochastic factors. These variations reflect real-world dynamics, providing a robust assessment of the model's adaptability and stability across diverse scenarios.
- **Reduction of Bias**: Averaging results across multiple random seeds helps mitigate anomalies or outliers that may arise due to the stochastic nature of RL processes. This provides a more reliable estimate of the model's true performance.

At the code level, random seeds are applied in several key areas:

- **Environment Initialization**: Random seeds are used to initialize the sampling of driver and passenger data files within the ride-hailing and ride-pooling environments.
- **Data Sampling**: Functions such as `sample` and `choice` leverage the set random seed to ensure deterministic behavior during data preprocessing and simulation.
- **Neural Network Training**: Random seeds are employed to initialize neural network weights and control the stochasticity of learning algorithms, such as random actions during exploration.

The results from repeated experiments are averaged, and confidence intervals are calculated to provide statistically robust evaluations. This multi-seed approach not only enhances the validity of the findings but also ensures that the proposed RL strategies perform consistently across a wide range of simulated conditions.

To provide a comparative reference, the baseline strategy is evaluated over 1,000 random episodes in the same simulation environment. These baseline episodes are configured identically to those used for training the agent, including the Poisson-based data generation and randomness. This ensures a fair and consistent comparison between the RL strategy and the baseline. The number of episodes is sufficient to generate stable results, serving as a benchmark to highlight the performance improvements achieved by the RL strategy.

To evaluate the model' s learning progression and effectiveness, this section presents the training curves for both traditional ride-hailing and ride-pooling services over 4,800 episodes. These curves

illustrate how the model' s performance improves over time as it optimizes the matching process within each service environment. By visualizing the changes in total reward across training iterations, the training curves offer insight into the model' s convergence behavior and the relative impact of different reward designs. According to the reward design, the total return of each episode can reflect the size of the passenger's total waiting time.

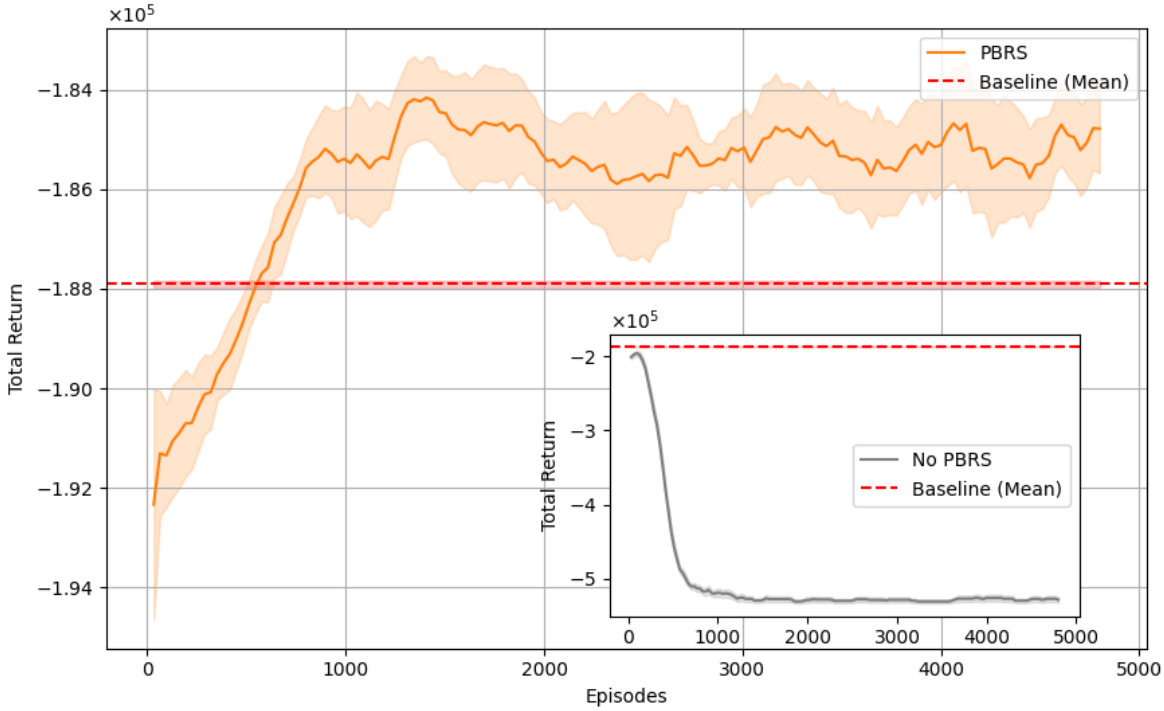

Figure 11: Training curve of Traditional Ride-Hailing Service

Figure 11 illustrates the trend in Total Return for two different reward designs (with and without PBRS) in the traditional ride-hailing simulator. Each curve represents the average Total Return over the training process, with the shaded area indicating the confidence interval based on the standard error, reflecting the model' s variability.

The gray curve represents the reward design without Potential-Based Reward Shaping (PBRS). In the initial few hundred episodes, the Total Return does not increase with training; instead, it rapidly decreases before gradually stabilizing at a low level in the later stages. Upon analyzing the actions taken by the agent, it was observed that due to the issue of sparse rewards, the agent does not receive $R_w(s_t, a_t)$ signals consistently. As a result, in the later stages of training, the agent repeatedly chooses to wait rather than initiate matching actions, causing the waiting time for all passengers to increase continuously.

The orange curve shows the performance of the PPO algorithm with PBRS. Compared to the design without PBRS, the PBRS-enhanced PPO curve demonstrates a faster convergence rate, with a rapid increase in Total Return during the early episodes. After approximately 600 episodes, the Total Return rises over the baseline and eventually stabilizes at a significantly lower level (around 186,000). This result suggests that the PBRS reward design effectively facilitates model learning, enabling the agent to optimize matching efficiency in a shorter time frame, thereby increasing the total return and enhancing system performance.

The shaded area surrounding the curves represents the confidence interval based on standard error, reflecting the variation in total return across different runs. During the initial phase (first 500 episodes), the shaded area is relatively wide, indicating greater variability as the model has yet to converge. As training progresses into the middle and later stages, the standard error decreases, and the shaded area narrows, signaling that the model' s performance is becoming more stable and the strategy is converging.

Overall, the PBRS reward design yields a strategy that surpasses the non-PBRS design in both

learning speed and optimization effectiveness, highlighting the significant role of PBRS in enhancing the efficacy of reinforcement learning algorithms within the traditional ride-hailing environment. Compared to the baseline, the PBRS-based PPO strategy demonstrates superior performance in increasing total return, ultimately offering an improved service experience for passengers. Additionally, PBRS effectively addresses the sparse reward problem, providing more frequent feedback to the agent and enabling better convergence during the training process.

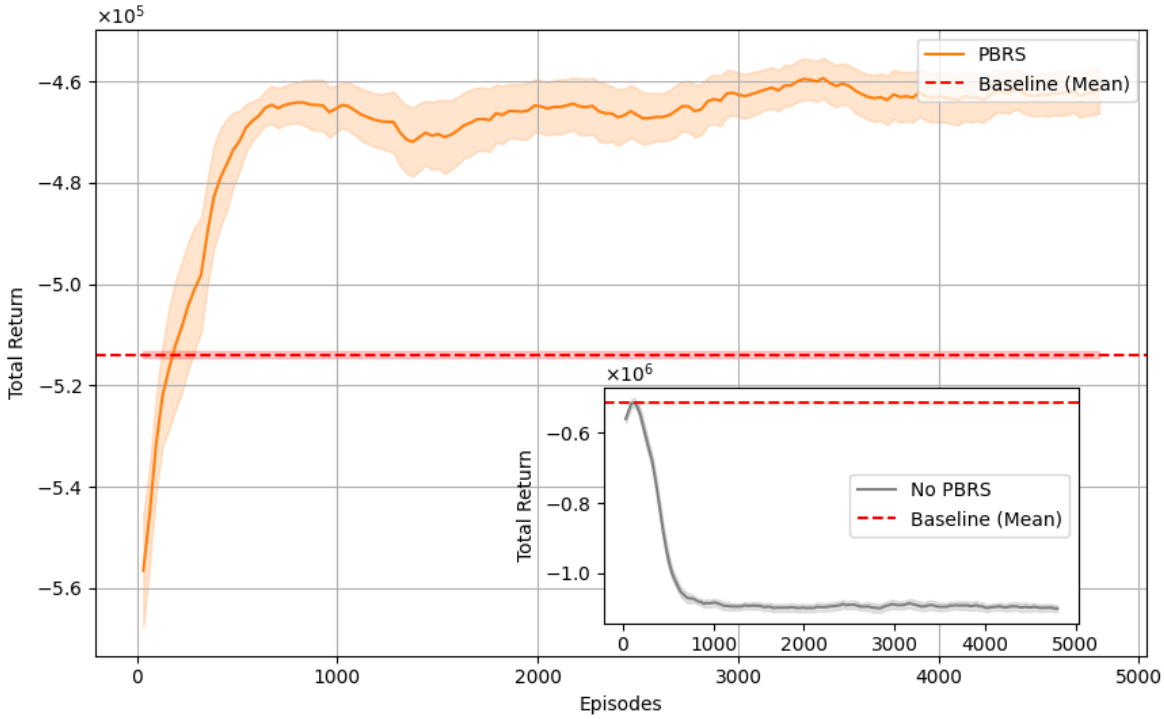

Figure 12: Training Curve of Ride-Pooling Service

Figure 12 illustrates the training process of the PPO strategy with PBRS compared to the PPO strategy without PBRS in the ride-pooling environment. In the early stages of training, the Total Return of the PBRS-enhanced curve exhibits a significant upward trend. Within the first 500 episodes, the Total Return rapidly rises from a high level, indicating that the strategy is effectively learning matching methods and quickly adapting to the demands of ride-pooling tasks. After approximately 1,000 episodes, the Total Return stabilizes, reaching a relatively steady level well above the baseline strategy, marked by the red dashed line at around -520,000. In contrast, the gray curve, which represents the reward design without PBRS, shows that while the Total Return briefly increases at the beginning of training, it sharply declines after the 50th episode, eventually falling to around -1,100,000. Training logs further reveal that the agent frequently opts to wait. This result highlights the superiority of the PPO strategy with PBRS in the ride-pooling setting.

During the first 500 episodes, the orange curve' s shaded area is relatively wide, indicating considerable variability in model performance as the strategy has not yet fully converged. In this phase, the strategy is still exploring optimal actions and gradually learning how to efficiently allocate drivers and passengers, leading to fluctuations in total return. Around episode 1,000, the curve becomes more stable, and the shaded area narrows. This phenomenon suggests that, as training progresses, the strategy' s performance across multiple runs converges, with improved stability and consistency. This stability implies that the model can increase total return consistently across different random environments.

Compared to the baseline strategy, the PPO strategy with PBRS demonstrates a clear advantage once it converges. Throughout the training process, the orange curve consistently remains above the baseline level and eventually stabilizes around a total return of approximately -460,000. Overall, the PPO strategy with PBRS exhibits strong learning capability and stability in the ride-pooling environment. It quickly increase total return in the early stages and, in the later stages, converges

to a level far below the baseline. The decreasing standard error further indicates the consistency and robustness of the strategy across different runs.

Comparing the training curves in Figure 11 and Figure 12, the PPO strategy with PBRS consistently demonstrates a rapid convergence rate. In both the traditional ride-hailing and ride-pooling environments, Total Return increases significantly during the early stages of training, indicating that the strategy effectively learns to optimize matching and dispatching from the outset. This result suggests that PBRS has a general benefit in enhancing the learning efficiency of the strategy. In both scenarios, the Total Return of the PBRS-enhanced strategy converges to levels over the baseline. This indicates that the PBRS reward design not only accelerates the learning process but also significantly improves final performance, leading to better passenger waiting times and enhanced system efficiency across both service environments.

In the traditional ride-hailing environment, the PPO curve with PBRS shows a certain level of fluctuation post-convergence, as evidenced by the relatively wide shaded area. In contrast, the ride-pooling curve exhibits greater stability, with a narrower confidence interval after convergence. This difference may be attributed to the increased complexity of ride-pooling, which requires the strategy to make more careful decisions involving multiple passengers, thereby promoting greater stability and consistency. In traditional ride-hailing, where matching conditions are more straightforward, the strategy tends to show more adjustments and fluctuations. Additionally, in the initial training phase, the Total Return for ride-pooling increases more rapidly than for traditional ride-hailing. This may be because the ride-pooling strategy optimizes the matching result between passenger orders, which leads to a swift increase in total return. In contrast, the simpler matching conditions in traditional ride-hailing result in a comparatively slower increase in waiting time.

4.3 Comparison of Strategies

To further validate the effectiveness of the trained strategy, this study conducted a detailed comparison of the RL strategy enhanced by Potential-Based Reward Shaping (PBRS), the baseline strategy, and the immediate matching strategy. Subsequently, these strategies were evaluated across 1,000 episodes within the same simulation environment.

During the evaluation phase, each strategy was tested under consistent conditions, where passenger requests and driver availability were randomly generated based on a Poisson distribution calibrated for the Manhattan area. While the episode settings were identical to those used for training the RL strategy in Section 5.2, the data generated differed due to the stochastic nature of the Poisson distribution. In each episode, the total waiting time for all passengers was recorded, and the average total waiting time for each strategy was calculated, enabling a comprehensive assessment of their performance in optimizing matching efficiency and reducing passenger waiting time.

By comparing the performance of these strategies, this section aims to illustrate the impact of the trained RL strategy on both ride-hailing and ride-pooling services. Such analysis provides critical insights for selecting the most effective strategy for real-world applications.

In the left-side graph of Figure 13 for the traditional ride-hailing scenario, the trained strategy (blue point) demonstrates the lowest average total waiting time, approximately 188,000, with a relatively small standard error. This indicates that the strategy consistently maintains a low waiting time across various test runs, reflecting high stability and reliability. The small standard error highlights the trained strategy's effectiveness in optimizing waiting time under random conditions, significantly reducing passenger waiting time. In contrast, the baseline strategy (yellow point) has a slightly higher average total waiting time of around 190,000. This suggests that, while the baseline strategy performs fairly consistently in the traditional ride-hailing scenario, it lacks the optimization efficiency of the trained strategy and is less adaptable to variations, resulting in higher waiting times in some cases. The instant matching strategy (green point) shows the highest average total waiting time, close to 200,000, indicating its limited effectiveness in this context.

In the right-side graph of Figure 13 for the ride-pooling scenario, the trained strategy again performs well, with an average total waiting time of approximately 475,000 and a small standard error. This result demonstrates the strategy's adaptability and consistency in the more complex ride-pooling environment, where it effectively optimizes matching efficiency and minimizes total waiting time across multiple test runs. This performance suggests that the trained strategy is robust, even in scenarios with diverse passenger needs. The baseline strategy in the ride-pooling scenario does not perform as well as the trained strategy, with an average total waiting time of around 600,000. Although it has

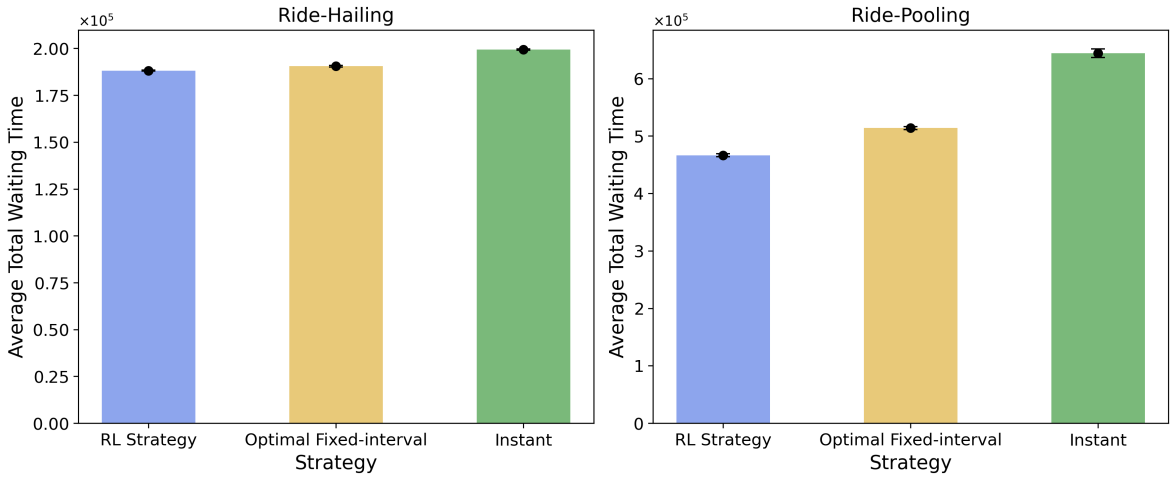

Figure 13: Comparison of Average Total Waiting Time with Standard Error Across Different Strategies in Ride-Hailing and Ride-Pooling

a small standard error, indicating stability, its relatively high waiting time shows its limitations in handling ride-pooling tasks, likely due to a lack of flexibility in complex shared ride scenarios. The instant matching strategy, in the ride-pooling scenario, exhibits the highest average total waiting time, close to 650,000, with a larger standard error, indicating substantial variability. This suggests that the instant matching strategy has considerable fluctuations in waiting time across different test runs, reflecting a lack of stability when facing diverse supply and demand conditions. Its high average waiting time demonstrates its inability to leverage delayed matching windows to optimize trip scheduling in ride-pooling contexts effectively.

Overall, the trained strategy shows the lowest average total waiting time and the smallest standard error in both scenarios, highlighting its superior optimization capabilities and strong stability. While the baseline strategy performs consistently, it is less efficient in the both scenarios. The instant matching strategy performs poorly in both scenarios, with high waiting times that make it less suitable for practical applications.

To validate the effectiveness of the training strategy and analyze the evolution of policies during the training process, this study selected five key strategies from the training curve at episodes 50, 500, 1000, 2000, and 4800. These checkpoints represent different stages of training, capturing the progression from initial learning to gradual optimization and eventual convergence. By comparing the performance of these strategies at different stages, the gradual improvements throughout the training process can be effectively illustrated.

In this experiment, the five checkpoint strategies, along with the baseline strategy, were tested over 1,000 episodes. The episodes were configured consistently with the previous experiments to ensure fair comparison within the same simulation environment. During the tests, several key metrics were recorded and analyzed, including Average Pickup Time, Average Matching Time, Average Detour Delay, and Average Total Waiting Time. These metrics provide detailed insights into the variations in passenger order waiting times and offer a comprehensive analysis of the performance of each strategy at different stages of training.

The experimental results clearly demonstrate the performance of traditional ride-hailing and ride-pooling services, with key metrics for each strategy shown in Figure 14 and Figure 15, respectively. Additionally, system-wide metrics, including Total Pickup Time, Total Matching Time, Total Detour Delay, and Total Waiting Time, are presented in the appendix in Figure 21 and Figure 22.

Figure 14 and Figure 21 reveal a clear trend: as the number of training episodes increases, the Average Pickup Time per Passenger gradually decreases from 258 seconds in the initial stages to 247 seconds by the 4800th episode. This improvement indicates that with more training, the strategy becomes increasingly effective at reducing the time passengers wait to be picked up after being matched with a driver. Similarly, the Total Pickup Time for the system follows the same downward trajectory, decreasing from 156,000 seconds to 150,000 seconds over the same period. While the reduction in Average Pickup Time (approximately 11 seconds) might appear modest, it is important to consider

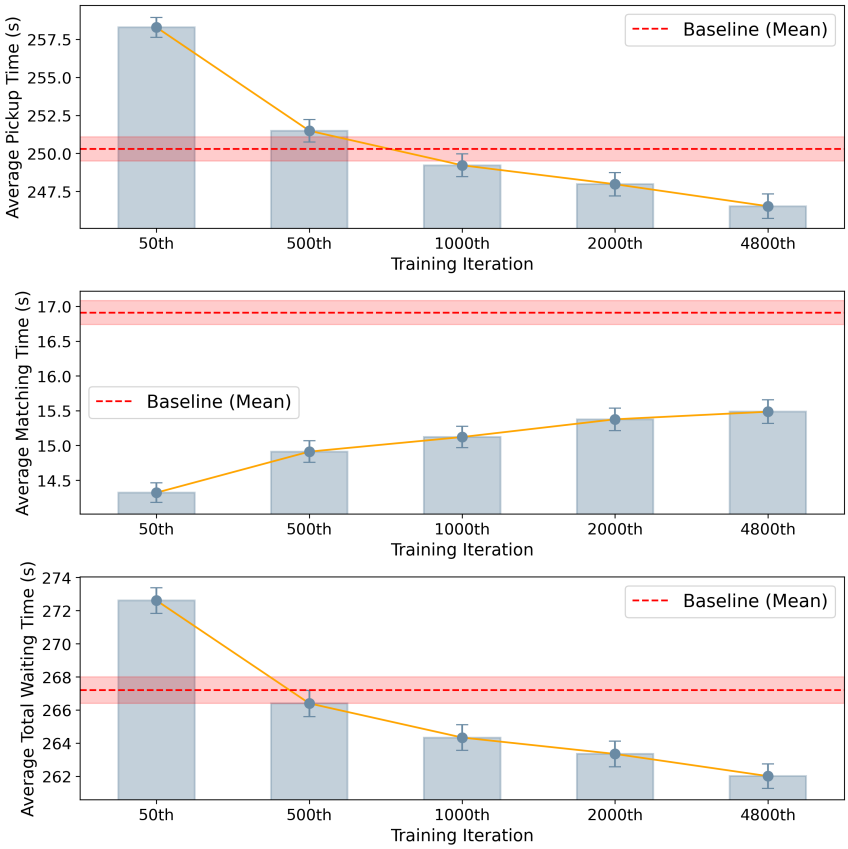

Figure 14: Comparison of Metrics Across Training Episodes and Baseline Strategy for Ride-Hailing Services

that this is an average across all passengers. For systems handling large volumes of passengers, even small per-passenger improvements translate into significant reductions in total system-wide pickup time. This synchronized decline in both metrics underscores the effectiveness of the trained strategy in enhancing overall system efficiency by optimizing pickup times for individual passengers. In contrast, the baseline strategy, although relatively stable in terms of Average Pickup Time, demonstrates limited capacity to optimize system efficiency or reduce passenger pickup times. This comparison highlights the superior performance of the trained strategy in addressing key operational inefficiencies.

From Figure 14, it can be observed that the Average Matching Time per Passenger shows a slight increase as training progresses, rising from 14.4 seconds at the 64th episode to 15.7 seconds at the 4800th episode. Although this increase is minor, it contrasts with the observed decrease in pickup waiting time. Similarly, the Total Matching Time (Figure 21) increases from approximately 34,500 seconds initially to about 37,100 seconds by the end of training, following a trend consistent with the average matching time. These metrics suggest that while the trained strategy effectively reduces pickup waiting time, this improvement comes at the cost of a slight increase in matching time. The rise in both average and total matching time implies that the system likely prioritizes optimizing the pickup process during training, which may result in longer delays during the matching phase. However, it is worth noting that the increase in matching time remains relatively modest and still falls below the matching time observed in the baseline strategy. Moreover, this trade-off yields significant improvements in pickup waiting time compared to the baseline. As a result, the overall system performance demonstrates substantial gains, reflecting the effectiveness of the trained strategy in balancing these competing objectives.

The Average Total Waiting Time per Passenger reflects the average waiting time each passenger experiences throughout the entire ride process (including matching and pickup). As the number of training episodes increases, the average waiting time decreases from 270 seconds at the 64th episode to 262 seconds by the 4800th episode. This suggests that the trained strategy performs well in enhancing

the overall ride experience by progressively reducing the average waiting time for passengers. The Total Waiting Time exhibits a similar trend to the average waiting time, decreasing from 191,000 seconds at the 64th episode to 186,000 seconds by the 4800th episode. The baseline strategy' s total waiting time is around 189,000 seconds, which is higher than the final performance of the trained strategy. The synchronized decline in both the average waiting time and total waiting time reflects an improvement in overall system efficiency. Through the optimization of the trained strategy, not only has each passenger's waiting experience improved, but the total waiting time across the entire system has also significantly decreased. This synchronized improvement shows that the trained strategy better balances individual passenger experience and overall system operational efficiency, minimizing total waiting time while enhancing service quality.

These analyses indicate that although the trained strategy may involve some trade-offs (e.g., a slight increase in matching waiting time), overall, it achieves significant improvements in passenger experience and system efficiency. Especially in large-scale system operations over the long term, the trained strategy demonstrates strong stability and adaptability.

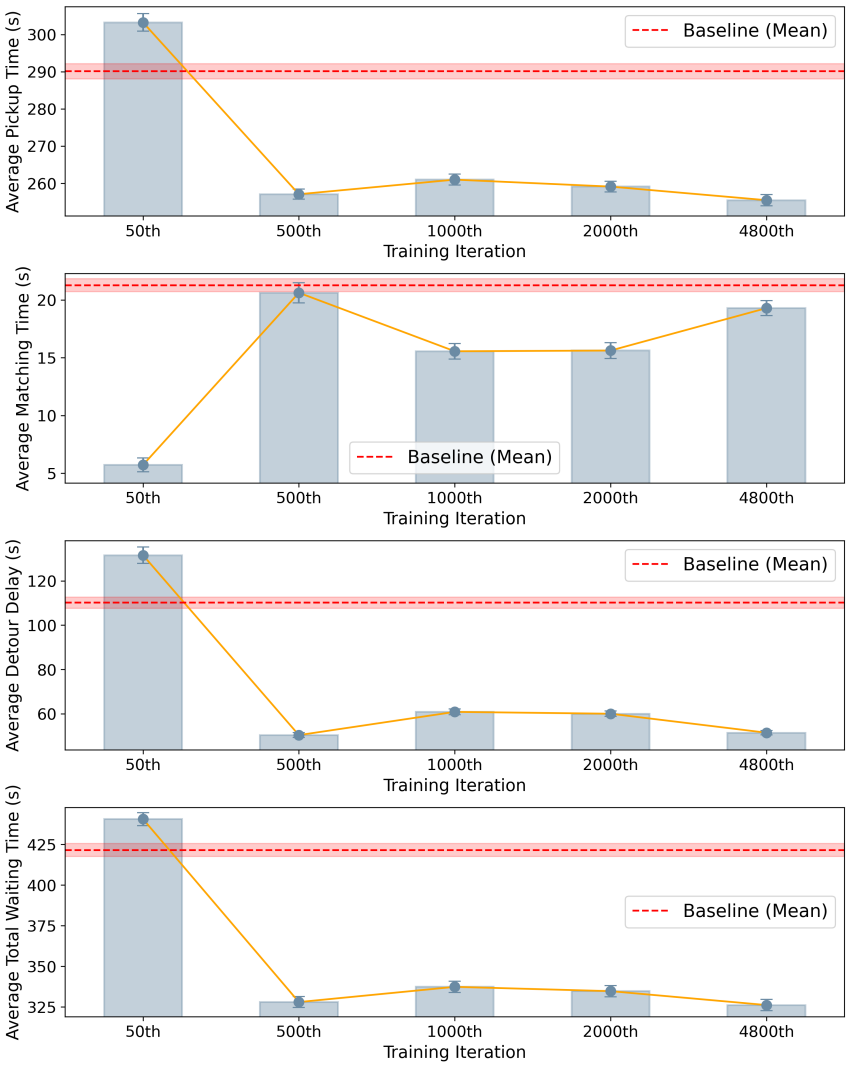

Figure 15: Comparison of Metrics Across Training Episodes and Baseline Strategy for Ride-Pooling Services

Figures 15 illustrate that the Average Pickup Time per passenger begins at approximately 31 seconds during the 50th episode but decreases rapidly as training progresses, stabilizing at around 25 seconds by the 500th episode. In contrast, the baseline strategy exhibits a significantly higher Average Pickup Time of approximately 29 seconds, highlighting its inefficiency compared to the trained strategy.

Similarly, the Total Pickup Time shows a substantial reduction with increasing training episodes, dropping from an initial 365,000 seconds to approximately 310,000 seconds. The baseline strategy, however, maintains a much higher Total Pickup Time of about 350,000 seconds. This synchronized downward trend in both metrics underscores the effectiveness of the trained ride-pooling strategy in improving overall system performance, particularly by reducing passenger wait times after matching. As training continues and the system becomes more optimized, pickup times not only decrease but also stabilize, reflecting enhanced efficiency. In contrast, the baseline strategy, reliant on a fixed matching interval, remains significantly less effective throughout.

In Figure 15, the Average Matching Time per passenger starts at around 6.0 seconds during the 50th episode, then increases with more training, peaking at approximately 20 seconds before stabilizing around 18.0 seconds. By comparison, the baseline strategy shows a worse performance, with an average matching time of about 22 seconds. The Total Matching Time follows a similar trend, increasing from around 18,000 seconds at the 50th episode to about 90,000 seconds by the 4800th episode. The baseline strategy shows an even higher total waiting time, exceeding 110,000 seconds. The slight increase in both average and total matching times suggests that while the trained strategy optimizes the pickup process, the matching process involves longer waiting times. This could be due to the complexity of decision-making involved in ride-pooling scenarios, where drivers are assigned to multiple passengers. Nevertheless, the trained strategy still performs better than the baseline, showing greater adaptability in handling complex scenarios.

The Average Detour Delay per passenger starts at around 125 seconds during the 50th episode but rapidly decreases through further training, reaching around 55 seconds by the 4800th episode. This indicates that the system becomes more efficient at reducing detour times as the strategy evolves. The Total Detour Delay follows a similar pattern, dropping from 160,000 seconds to around 6,000 seconds. The baseline strategy, however, shows a much higher total detour delay of about 130,000 seconds. These two metrics demonstrate that the trained ride-pooling strategy significantly reduces detour times, which is crucial in ride-pooling services. As training progresses, the system becomes more effective at route optimization, substantially lowering both individual detour delays and the total detour delay for the system as a whole, contributing to improved passenger satisfaction.

The Average Total Waiting Time per passenger decreases steadily from approximately 440 seconds during the 64th episode to around 330 seconds by the 4800th episode. This indicates that passengers' overall waiting experience improves significantly as training progresses. The baseline strategy, however, maintains a higher average waiting time of around 420 seconds, showing inferior performance. The Total Waiting Time decreases from an initial 550,000 seconds to around 460,000 seconds by the 4800th episode, whereas the baseline strategy' s total waiting time remains higher, at approximately 540,000 seconds.

Overall, the trained ride-pooling strategy shows significant improvements in both passenger experience and system efficiency. Although the matching time slightly increases, the optimizations in pickup time and detour delays result in excellent overall system performance. The reductions in total waiting time and improved system performance further demonstrate the effectiveness of reinforcement learning in ride-pooling scenarios, as the system continues to optimize and perform more efficiently as training progresses.

An additional validation experiment was designed to demonstrate that the strategy can dynamically adjust matching time intervals based on the current system state, rather than relying on fixed intervals. In this experiment, the fully trained RL strategy was deployed in several episodes with the same settings as in the previous experiments. The distribution of dynamic matching time intervals in both traditional ride-hailing and ride-pooling scenarios was systematically recorded and analyzed. Additionally, the number of passenger requests and available drivers at each matching decision was tracked to reveal the system's flexibility in responding to varying supply and demand conditions, especially in cases of imbalance.

Analysis of the matching time interval distribution clearly illustrates how the trained strategy dynamically adjusts the intervals based on real-time supply-demand conditions. This finding highlights the strategy's ability to adapt to different supply-demand situations without relying on fixed, predetermined intervals, thereby improving service efficiency and reducing passenger waiting times. The results further validate the robustness and adaptability of the trained strategy in dynamic environments, showcasing its potential effectiveness and applicability in real-world scenarios and providing a strong theoretical foundation for broader practical deployment.

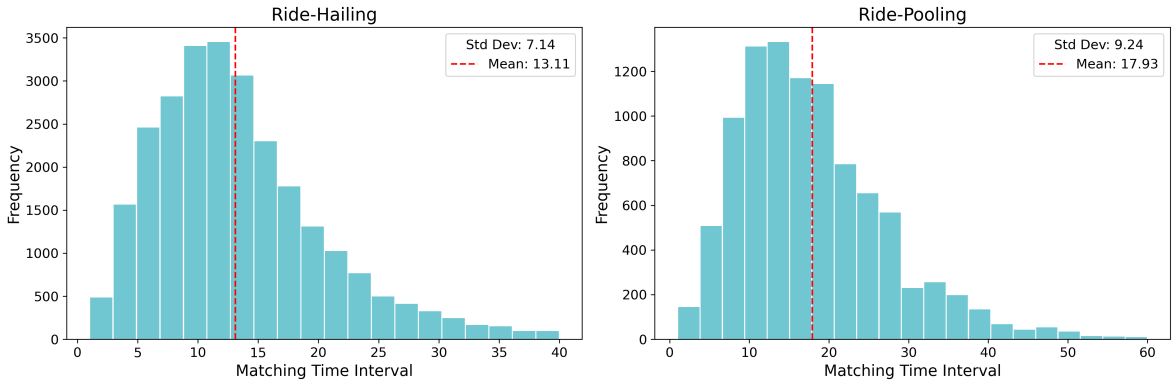

Figure 16: Distribution of Dynamic Time Intervals

Figure 17: Matching Interval Dynamics with Order and Driver Counts

Figure 16 presents the distribution of matching time intervals under the trained strategy for both traditional ride-hailing and ride-pooling scenarios. The histograms show how the matching time intervals dynamically change, rather than remaining fixed, demonstrating the adaptability of the trained strategy in adjusting to real-time system conditions.

In the traditional ride-hailing scenario, the distribution of matching intervals is concentrated within shorter time ranges, with most matches occurring between 10 to 20 seconds. However, there are also instances where the matching time is longer, extending to around 40 seconds. In the ride-pooling scenario, the distribution of matching intervals is more dispersed, reflecting the complexity of coordinating multiple passengers with compatible routes. The strategy exhibits a wider range of matching intervals, frequently extending beyond 30 seconds. This longer interval allows the system to accumulate enough passenger requests to identify optimal ride-pooling opportunities and driver matches. By dynamically adjusting the matching intervals based on the current number of passenger requests and available drivers, the strategy improves ride-pooling efficiency, reduces detour delays, and shortens driver pick-up distances, thus enhancing overall system performance.

Figure 17 shows the number of passenger orders (X-axis) and available drivers (Y-axis) at each matching action, with comparisons for both ride-hailing and ride-pooling scenarios. The color on the heatmap represents the length of time since the last matching action, with darker colors indicating longer intervals and lighter colors indicating shorter intervals.

In the ride-hailing scenario (left chart), as the number of passenger orders increases, the matching time interval also extends (darker colors), especially when the number of passenger orders exceeds 30, and the matching interval significantly increases. On the diagonal line, there is obviously longer matching time interval. This might be because the system chooses to wait for more orders and drivers to improve matching efficiency. It can be understood that, in a supply-demand balanced situation, the system delays matching to accumulate more resources and thus make better matching decisions. At the same time, when the number of available drivers is lower (below 30 on the Y-axis) and close to the y axis, the matching time interval is shorter, indicating that the system tends to match more quickly in this case to avoid prolonged passenger wait times. This reflects the system's preference for shorter matching intervals when driver supply is tight to reduce passenger wait times. Additionally, when the number of passenger orders exceeds the number of available drivers, the matching time interval is also shorter, implying that there are more idle drivers near each passenger request, enabling more efficient passenger-driver matches. This suggests that when driver supply is abundant, the system also tends to choose shorter matching intervals.

For the ride-pooling scenario (right chart), the matching intervals are generally longer compared to the ride-hailing scenario (darker colors), which aligns with the complexity of ride-pooling. As the number of passenger orders and drivers increases, the matching intervals extend significantly, especially when the number of passenger orders exceeds 50, with intervals extending beyond 30 seconds. This indicates that in the ride-pooling scenario, the system needs to wait for more orders to find the optimal pooling combinations. Therefore, in balanced supply-demand situations, the system intentionally delays matching to accumulate enough passenger orders to make better ride-pooling decisions. However, in situations where supply and demand are imbalanced, the system, similar to traditional ride-hailing, chooses shorter matching intervals to quickly process the accumulation caused by the imbalance.

Overall, whether in the ride-hailing or ride-pooling scenario, the system adjusts the matching intervals based on actual supply and demand conditions. Particularly in situations of imbalance, the system shortens the matching intervals to handle the accumulated orders and drivers. On the other hand, in balanced situations, the system extends the matching intervals to accumulate more orders and drivers for better decisions. This dynamic matching strategy shows that the system can flexibly adjust the matching intervals according to current conditions, rather than relying on fixed matching intervals, thus improving the overall efficiency and service quality of the system.

4.4 Ride-Hailing vs. Ride-Pooling

After analyzing the dynamic adjustments in matching time intervals, a further step in this research involves an experiment designed to compare the performance of ride-hailing and ride-pooling modes in the same environment. This experiment aims to provide a comprehensive evaluation of these two transportation modes under varying conditions, focusing specifically on key metrics such as matching time, pickup time, and detour delays.

One of the primary objectives of this experiment is to compare how ride-hailing and ride-pooling perform in different scenarios, providing insights into the unique challenges and advantages of each mode. By analyzing their respective performances, this study aims to highlight the differences in efficiency, particularly in terms of how effectively each mode handles fluctuating supply and demand conditions.

Additionally, this experiment evaluates the improvements introduced by the trained matching strategy. The goal is to demonstrate how the dynamic adjustment capabilities of the trained strategy outperform fixed-interval matching methods traditionally used in ride-hailing and ride-pooling services. The comparison will focus on whether the trained strategy can reduce overall waiting times and detour delays, while also optimizing the matching process by dynamically adjusting to real-time conditions.

Through this comparative analysis, the study seeks to provide a deeper understanding of how dynamic matching strategies can be applied to both ride-hailing and ride-pooling services, enhancing system performance and ultimately improving passenger experience. By establishing eight distinct scenarios, this experiment aims to highlight the advantages and limitations of each service model under varying resource constraints and matching strategies. Specifically, the goal is to evaluate how traditional ride-hailing and ride-pooling compare in terms of efficiency metrics, such as passenger wait times and detour delays. This horizontal comparison offers insights into which model is more effective under different demand conditions, providing a foundation for optimizing urban mobility solutions and informing future policy or operational adjustments in shared mobility services.

In this experiment, eight scenarios are set up to compare different transportation modes and matching strategies, as shown in Table 3.

Table 3: Experimental Scenario Design

Scenario	Service Mode	Driver-to-Order Ratio	Strategy
A	Ride-Hailing	1:1	Baseline
B	Ride-Hailing	1:1	RL
C	Ride-Hailing	0.5:1	Baseline
D	Ride-Hailing	0.5:1	RL
E	Ride-Pooling	1:1	Baseline
F	Ride-Pooling	1:1	RL
G	Ride-Pooling	0.5:1	Baseline
H	Ride-Pooling	0.5:1	RL

These eight scenarios provide a systematic framework for evaluating the effects of service modes, driver-to-order ratios, and matching strategies on system efficiency. Each scenario focuses on either ride-hailing (matching single-passenger requests) or ride-pooling (matching two passenger orders for shared rides) and is tested under two supply conditions: a 1:1 driver-to-order ratio (sufficient supply) and a 0.5:1 ratio (limited supply). This comprehensive approach enables a detailed analysis of strategy performance across varied resource availability. All scenarios were trained over 250 episodes, with the passenger order dataset remaining consistent across episodes and representing simulated supply-demand data for Manhattan from 8:30 AM to 8:40 AM. For scenarios with a 0.5:1 driver-to-order ratio, only half of the driver dataset used in the 1:1 scenarios was utilized. This design ensures a controlled comparison of strategy effectiveness under different resource constraints.

Specifically, comparisons between scenarios A and B, C and D, E and F, and G and H highlight the differences between the baseline and RL strategies. Comparisons between scenarios A and C, B and D, E and G, and F and H reveal how each strategy performs under different supply-demand conditions. Finally, comparisons between scenarios A and E, B and F, C and G, and D and H illustrate the differences between service modes. These distinctions are assessed based on key performance indicators for each passenger order, including Average Pickup Time, Average Matching Time, Average Detour Delay, and Average Total Waiting Time, as shown in Table 4. Further details on system performance metrics can be found in Appendix Table 5.

Figure 18 visually illustrates the differences in Average Total Waiting Time per passenger order across different scenarios. Under varying supply-demand conditions, regardless of whether the service mode is ride-hailing or ride-pooling, the RL strategy consistently achieves lower waiting times, demonstrating its ability to match passengers with drivers more efficiently and thereby reduce waiting time. Combined with Table 4, the RL strategy also shows significant advantages over the baseline in metrics such as Average Pickup Time, Average Matching Time, and Average Detour Delay.

There are notable differences between ride-hailing and ride-pooling under the two supply-demand conditions. When the Driver-to-Order Ratio is 1:1, ride-hailing achieves a supply-demand balance. However, when the supply is halved to a 0.5:1 ratio, ride-hailing faces a demand surplus, causing a significant increase in Average Total Waiting Time for passengers. Although the Average Pickup Time decreases in this situation, it is mainly because drivers prioritize matching with nearby passenger orders, leaving many passengers unmatched. This leads to a significant increase in Average Matching Time and a high rate of order cancellations due to prolonged waiting times. Although previous analysis shows that the RL strategy can optimize performance in unbalanced supply-demand conditions by shortening the matching interval, this adjustment only brings a slight improvement and does not resolve the issue of numerous unmatched passenger orders.

The situation is different for ride-pooling. Since each driver can serve two passenger orders, a 1:1 Driver-to-Order Ratio in ride-pooling actually represents an oversupply. In this case, passengers can be matched with closer drivers, significantly reducing the Average Pickup Time, which also results in some idle drivers. The RL strategy dynamically adapts by shortening the matching interval, thus reducing Average Matching Time. When the Driver-to-Order Ratio is 0.5:1, the increase in Average Total Waiting Time for ride-pooling is less drastic than for ride-hailing, showing only a slight rise. This is because, in ride-pooling, a 0.5:1 ratio still represents a balanced supply-demand state, and there is no significant increase in order cancellations due to prolonged matching times. In this case, the RL

Table 4: Comparison of Metrics Across Different Ride-Hailing Scenarios

Scenario	Metric	Mean (s)	St. Error
A	Average Pickup Time	282.992	1.601
	Average Matching Time	16.6908	0.317
	Average Detour Delay	-	-
	Average Total Waiting Time	299.6828	1.322
B	Average Pickup Time	247.176	1.629
	Average Matching Time	15.306	0.318
	Average Detour Delay	-	-
	Average Total Waiting Time	262.482	1.452
C	Average Pickup Time	54.302	0.611
	Average Matching Time	506.124	0.346
	Average Detour Delay	-	-
	Average Total Waiting Time	560.426	1.221
D	Average Pickup Time	51.539	0.604
	Average Matching Time	449.337	0.344
	Average Detour Delay	-	-
	Average Total Waiting Time	500.876	1.225
E	Average Pickup Time	243.534	0.494
	Average Matching Time	20.008	0.589
	Average Detour Delay	177.47	0.841
	Average Total Waiting Time	441.012	1.163
F	Average Pickup Time	203.9952	0.462
	Average Matching Time	12.7528	0.235
	Average Detour Delay	85.894	0.727
	Average Total Waiting Time	302.642	0.917
G	Average Pickup Time	314.614	1.278
	Average Matching Time	22.8872	0.374
	Average Detour Delay	144.089	1.757
	Average Total Waiting Time	481.5902	1.669
H	Average Pickup Time	270.634	0.936
	Average Matching Time	21.555	0.436
	Average Detour Delay	76.162	0.821
	Average Total Waiting Time	368.351	1.249

strategy responds by slightly extending the matching interval, leading to a minor increase in Average Matching Time while keeping Average Pickup Time and Average Detour Delay relatively stable.

When the Driver-to-Order Ratio is 1:1, ride-hailing indeed has an advantage over ride-pooling, with both the baseline and RL strategies resulting in lower Average Total Waiting Times compared to ride-pooling. This is primarily because ride-hailing does not require a passenger-to-passenger matching phase, which reduces the Average Matching Time and eliminates Detour Delay. However, a comparison of the metrics between Scenario F and Scenarios A and B reveals that, in the RL-enhanced ride-pooling scenario, the Average Total Waiting Time is reduced to a level close to that of Scenarios A and B. This result reflects the adaptability and flexibility of the RL strategy within ride-pooling. Due to the increased complexity of ride-pooling, where multiple passenger needs must be met simultaneously, the introduction of RL has evidently enhanced the system's capability to handle complex scenarios.

With a 0.5:1 Driver-to-Order Ratio, the performance of ride-hailing is less satisfactory. In this case of constrained supply, ride-hailing struggles to meet demand under limited resources, highlighting the significant advantage of ride-pooling, particularly in effectively meeting passenger demand when resources are limited. Notably, even with only half the driver supply of Scenario A, ride-pooling in Scenarios G and H maintains system efficiency. Specifically, in Scenario H, the trained RL strategy optimizes resource allocation more effectively. Despite the reduction in driver supply by half, the Average Total Waiting Time and Average Pickup Time in Scenario H do not double as would be expected in the traditional ride-hailing model. This demonstrates that the pooling mode, even with limited driver availability, can effectively match demand by combining multiple passenger trips.

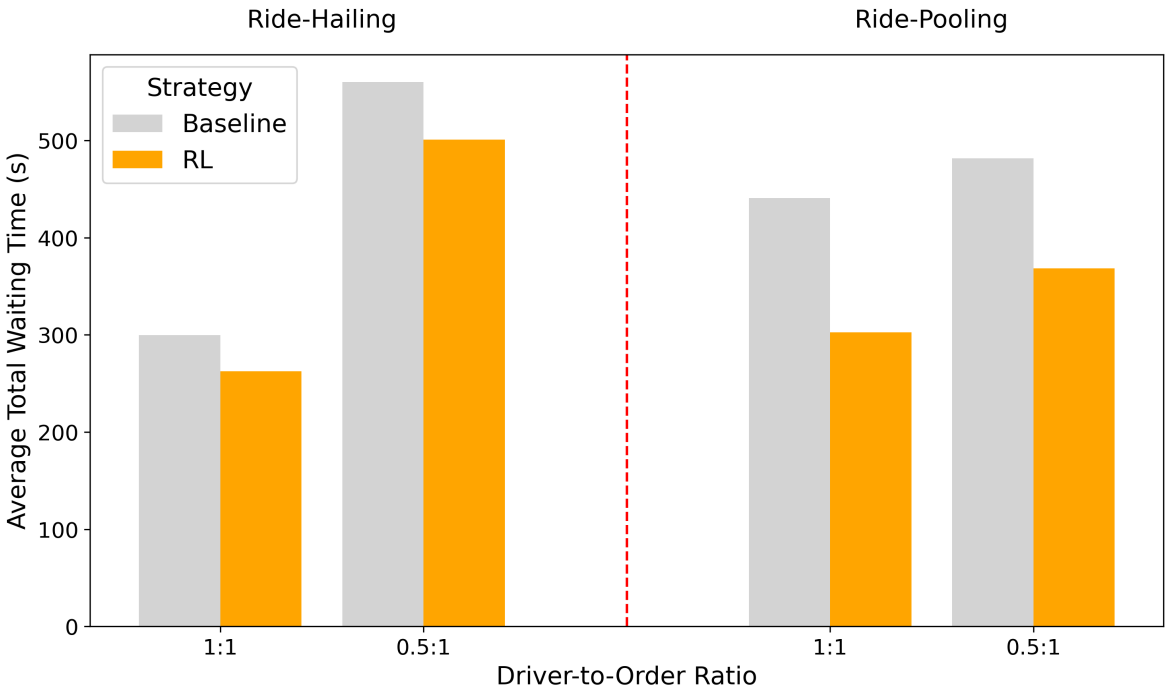

Figure 18: Average Total Waiting Time in 8 Scenarios

Overall, the main advantage of ride-pooling lies in its ability to maintain efficient system operation even with fewer drivers by optimizing algorithms and matching strategies. This efficient resource utilization not only reduces total system waiting time but also effectively lowers passenger detour delays. Even when driver supply is tight, the system can meet passenger demand through pooling combinations. This means that in situations of supply-demand imbalance, ride-pooling can improve the overall service level of the system by integrating multiple orders without significantly increasing the waiting time for each passenger.

5 Conclusion

5.1 Research Contributions

The objective of this study is to leverage deep reinforcement learning techniques, specifically a dynamic strategy based on the Proximal Policy Optimization (PPO) algorithm, to optimize matching time intervals, thereby reducing passenger wait times and improving vehicle utilization. Additionally, to address the challenge of sparse rewards, Potential-Based Reward Shaping (PBRS) is employed to enhance learning efficiency. By dynamically adjusting the matching strategy, this research provides optimized solutions not only for traditional ride-hailing services but also for the complexity of multi-passenger matching and detour delays in ride-pooling.

This study introduces several key innovations that improve upon existing matching strategies for both ride-hailing and ride-pooling services. The specific contributions of this research include the following:

- **Dynamic Optimization of Matching Time Intervals**
  This study develops a reinforcement learning-based dynamic matching strategy that continuously learns and adjusts matching time intervals according to fluctuations in supply and demand. Compared to traditional fixed or real-time matching methods, dynamic optimization of matching intervals finds a better balance between matching efficiency and passenger wait time, thus enhancing overall system performance.

- **Introduction of Potential-Based Reward Shaping (PBRS)**
  This study applies PBRS in reward design to tackle the challenge of sparse rewards. Traditional

reinforcement learning methods may struggle with low learning efficiency when faced with sparse rewards. With the introduction of PBRS, the model converges to the optimal strategy more quickly, significantly improving the learning performance in the ride-hailing service environment.

- **Addressing the Complexity of Multi-Passenger Matching in Ride-Pooling**
  For the ride-pooling service, this study proposes an efficient algorithm framework to consider similarities in passengers' origins and destinations during the matching process, thereby reducing detour time. This optimized strategy not only improves matching efficiency but also reduces detour delays for passengers, enhancing overall passenger satisfaction.

- **Development of an Efficient Simulator for Method Validation**
  This research designs and builds a realistic simulator for both ride-hailing and ride-pooling services to simulate the spatio-temporal dynamics of the system. The simulator can accurately generate passenger orders and driver distribution data and dynamically adjust matching strategies, allowing the evaluation of various matching methods under practical conditions. This provides a reliable testing platform for exploring the application of reinforcement learning in dynamic mobility services.

Through these contributions, this study introduces a more flexible and adaptive optimization method to improve existing ride-hailing service matching strategies. It demonstrates the potential of reinforcement learning to enhance system efficiency, reduce passenger wait times, and optimize shared mobility experiences in dynamic urban transportation environments.

5.2 Key Findings and Implications

Via a series of experiments and simulations, this study thoroughly analyzed the performance of different matching strategies in both conventional and ride-pooling services, leading to the following main conclusions:

- **Dynamic Matching Strategy Significantly Reduces Passenger Wait Time**
  The experimental results indicate that the PPO-based dynamic matching strategy demonstrates significant advantages over traditional fixed-interval matching in reducing total passenger wait time. After 4,800 training episodes, the dynamic matching strategy with PBRS successfully minimized both the total and average matching time for passengers. The findings reveal that this strategy' s ability to dynamically adjust matching intervals allows it to better accommodate real-time supply-demand fluctuations, thereby optimizing match quality during high-demand periods and effectively reducing passenger wait times.

- **PBRS Reward Design Enhances Model Learning Efficiency**
  Compared to models without PBRS, the reinforcement learning model with PBRS exhibited faster convergence and more stable performance. PBRS provides more frequent reward feedback in sparse reward environments, enabling the model to learn more effective strategies within the first few hundred episodes. In the ride-pooling environment, PBRS further improved strategy stability and reduced training volatility, allowing the model to find strategies that minimize detour delays more quickly.

- **Effectively Managing Detour Delays in ride-pooling Scenarios**
  This study specifically addresses the multi-passenger matching problem in ride-pooling. The experimental results show that the dynamic matching strategy not only reduces passenger wait time but also effectively controls detour delays during multi-passenger matching. Data indicates that the average detour delay with dynamic matching is significantly lower than with fixed-interval strategies. During high-demand periods, dynamic matching can aggregate more passenger requests by delaying matching intervals slightly, thus optimizing matching and reducing detour costs. These results highlight the potential of dynamic matching strategies to enhance service quality in ride-pooling.

- **Ride-Pooling with Dynamic Matching Outperforms Under Limited Resources**
  The experimental results show that the traditional ride-hailing mode exhibits higher system efficiency when driver resources are abundant, with significantly lower total waiting and matching

times compared to the ride-pooling mode. However, in scenarios with limited driver resources, the ride-pooling mode effectively reduces the system burden through carpooling. Notably, the dynamic matching strategy trained with reinforcement learning can still significantly reduce passenger wait times and detour delays, even in resource-constrained conditions.

- **Simulator Validates Practical Effectiveness of Different Strategies**
  The simulator designed in this study accurately replicates real-world supply-demand fluctuations and allows for effective evaluation of strategies across various interval settings. Through comparative analysis with fixed-interval and random matching strategies, the results demonstrate that while fixed-interval matching can provide stability, it is unable to adapt to dynamic supply-demand changes. Conversely, the PPO-based dynamic matching strategy flexibly responds to varying conditions, thereby improving overall system efficiency.

In conclusion, this study illustrates the potential of reinforcement learning in optimizing matching time intervals in ride-hailing services. The experimental results suggest that dynamically optimizing matching intervals not only improves the efficiency of ride-hailing services but also effectively balances the trade-off between response speed and match quality.

Moreover, the dynamic optimization of matching time intervals offers a valuable improvement direction for current ride-hailing platforms such as Uber and Lyft. By utilizing reinforcement learning-based dynamic matching, the system can respond in real time to fluctuations in supply and demand, thereby reducing passenger wait times and driver idle times. This not only enhances travel efficiency but also improves overall user satisfaction, fostering greater user dependency on and loyalty to shared mobility services.

Additionally, this study' s multi-passenger matching strategy for ride-pooling scenarios demonstrates how to efficiently group passengers during high-demand periods to reduce detour delays. By matching more passengers with similar routes at appropriate times, the strategy optimizes resource utilization and helps platforms provide a more sustainable mobility solution. Especially during peak hours or in congested areas, aggregating multiple passenger requests can help reduce total vehicle miles traveled, alleviate traffic congestion, and further decrease carbon emissions, contributing to a more environmentally friendly transportation service.

The efficient simulator developed in this study provides shared mobility platforms with a tool for testing and validating new strategies. The simulator not only replicates complex dynamic scenarios in real-world mobility services but also allows researchers and operators to test the impact of different matching time intervals and strategies on system efficiency. This flexible simulation capability enables platforms to fine-tune and optimize new strategies before implementation, improving their effectiveness and reliability and supporting continuous innovation.

In summary, the methods and models proposed in this study offer new technical references for future intelligent mobility services. In practical applications, combining dynamic optimization with reinforcement learning models will enable ride-hailing platforms to more accurately predict and respond to supply-demand variations, effectively reducing costs and enhancing service efficiency. This innovation holds practical significance not only for shared mobility platforms but also for broader urban intelligent transportation management, supporting the digital transformation and sustainable development of urban transportation.

5.3 Limitation and Future Work

While this study has made significant strides in developing an effective simulator and applying reinforcement learning to optimize matching strategies in ride-hailing and ride-pooling services, there remain several potential avenues for further research and improvement. These areas include enhancements to the simulator, exploration of various reinforcement learning algorithms, and refinement of state representation. Future research directions are outlined as follows:

- **Incorporating Real-Time Traffic Conditions:** The current simulator does not account for real-time traffic data, which affects vehicle speeds and passenger wait times. Integrating real-time traffic data in future versions of the simulator would allow for a more accurate representation of dynamic traffic conditions encountered in real-world mobility services. This enhancement would enable the reinforcement learning algorithm to develop strategies that are more adaptable to actual traffic scenarios.

- **Simulating a Hybrid Service Environment:** Currently, the simulator models conventional ride-hailing and ride-pooling services separately. Future work could benefit from integrating both services within a single simulation environment, allowing for a comprehensive evaluation of different strategies in a mixed-service setting. This would provide a more accurate assessment of the model' s efficiency and adaptability within complex urban mobility environments, enhancing the practical applicability of the research.

- **Expanding Ride-Pooling Combinations:** At present, the simulator limits ride-pooling to combinations of two passenger orders. Extending this to allow for more than two passengers per vehicle would better represent real-world demand patterns and enable analysis of multi-passenger detour optimizations. This approach could improve detour path selection strategies, ensuring a balanced experience for passengers while enhancing overall system efficiency.

- **Exploring Alternative Reinforcement Learning Algorithms:** This study applied the Proximal Policy Optimization (PPO) algorithm to optimize matching intervals with promising results. However, future work could explore other reinforcement learning algorithms, such as Deep Q-Network (DQN), Deep Deterministic Policy Gradient (DDPG), and Advantage Actor-Critic (A3C), to compare their performance in the context of dynamic matching strategy optimization. Such comparisons would provide deeper insights into the adaptability and efficiency of various algorithms.

- **Enhancing State Representation:** The state representation in this study views the ride-hailing and ride-pooling processes as a partially observed Markov decision process (POMDP), which inherently limits the amount of information captured about the system. Future research could work towards enhancing the state representation by incorporating a graph-based structure to represent the relative locations of passenger requests and available drivers. Additionally, feature engineering could be employed to retain critical information while filtering out less relevant data, enabling more effective decision-making that minimizes wait times and detours.

Overall, advancing the simulator' s capabilities and exploring a variety of reinforcement learning algorithms could enrich the research findings and provide valuable technical support for future ride-hailing optimization strategies. These developments would further enhance the practical applicability and scalability of the study, building a strong foundation for intelligent optimization in next-generation mobility 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] G. Feng, G. Kong, and Z. Wang, "We are on the way: Analysis of on-demand ride-hailing systems," *TransportRN: Transportation Modes*, 2017.

[4] C. Rodier and J. Michaels, "The effects of ride-hailing services on greenhouse gas emissions," 2019.

[5] S. Shaheen, "Shared mobility: The potential of ride hailing and pooling," pp. 55–76, 2018.

[6] J. Jacob and R. Roet-Green, "Ride solo or pool: Designing price-service menus for a ride-sharing platform," *TransportRN: Transportation Costs (Topic)*, 2018.

[7] Y. Sun and L. Zhang, "Potential of taxi-pooling to reduce vehicle miles traveled in washington, d.c." *Transportation Research Record*, vol. 2672, pp. 775 – 784, 2018.

[8] J. Ke, H. Yang, and Z. Zheng, "On ride-pooling and traffic congestion," *Transportation Research Part B-methodological*, vol. 142, pp. 213–231, 2020.

[9] G. Feng, G. Kong, and Z. Wang, "We are on the way: Analysis of on-demand ride-hailing systems," *Manuf. Serv. Oper. Manag.*, 2021.

[10] 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. [Online]. Available: https://www.pnas.org/doi/abs/10.1073/pnas.1611675114

[11] A. Sundt, Q. Luo, J. Vincent, M. Shahabi, and Y. Yin, "Heuristics for customer-focused ride-pooling assignment," *ArXiv*, vol. abs/2107.11318, 2021.

[12] J. Gao, X. Li, C. Wang, and X. Huang, "Bm-ddpg: An integrated dispatching framework for ride-hailing systems," *IEEE Transactions on Intelligent Transportation Systems*, vol. 23, pp. 11 666–11 676, 2022.

[13] G. Shakya and M. Yokoo, "Balancing fairness and efficiency in 3d repeated matching in rideshar-ing," pp. 2121–2128, 2023.

[14] Uber. (2023) How batch matching works. Accessed: October 5, 2024. [Online]. Available: https://www.uber.com/us/en/marketplace/matching/

[15] M. Baccara, S. Lee, and L. Yariv, "Optimal dynamic matching," *Theoretical Economics*, 2020.

[16] B. Zheng, L. Ming, Q. Hu, Z. Lü, G. Liu, and X. Zhou, "Supply-demand-aware deep reinforcement learning for dynamic fleet management," *ACM Transactions on Intelligent Systems and Technology (TIST)*, vol. 13, pp. 1 – 19, 2022.

[17] X. Wang, X. Li, and L. Lai, "On improving the learning of long-term historical information for tasks with partial observability," *2020 IEEE Fifth International Conference on Data Science in Cyberspace (DSC)*, pp. 232–237, 2020.

[18] S. S. Ramesh, P. G. Sessa, Y. Hu, A. Krause, and I. Bogunovic, "Distributionally robust model-based reinforcement learning with large state spaces," *ArXiv*, vol. abs/2309.02236, 2023.

[19] V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. A. Riedmiller, A. Fidjeland, G. Ostrovski, S. Petersen, C. Beattie, A. Sadik, I. Antonoglou, H. King, D. Kumaran, D. Wierstra, S. Legg, and D. Hassabis, "Human-level control through deep rein-forcement learning," *Nature*, vol. 518, pp. 529–533, 2015.

[20] Z. Wang, Y. Zhang, B. Jia, and Z. Gao, "Comparative analysis of usage patterns and underlying determinants for ride-hailing and traditional taxi services: A chicago case study," *Transportation Research Part A: Policy and Practice*, vol. 179, p. 103912, 2024. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0965856423003324

[21] I. O. Olayode, A. Severino, F. Justice Alex, E. Macioszek, and L. K. Tartibu, "Systematic review on the evaluation of the effects of ride-hailing services on public road transportation," *Transportation Research Interdisciplinary Perspectives*, vol. 22, p. 100943, 2023. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S2590198223001902

[22] N. Alisoltani, L. Leclercq, and M. Zargayouna, "Can dynamic ride-sharing reduce traffic congestion?" *Transportation Research Part B: Methodological*, vol. 145, pp. 212–246, 2021. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0191261521000114

[23] S. Feng, J. Ke, F. Xiao, and H. Yang, "Approximating a ride-sourcing system with block matching," *Transportation Research Part C: Emerging Technologies*, vol. 145, p. 103920, 2022. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0968090X22003333

[24] 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. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0191261521001004

[25] T. Dong, Q. Luo, Z. Xu, Y. Yin, and J. Wang, "Strategic driver repositioning in ride-hailing networks with dual sourcing," *Transportation Research Part C: Emerging Technologies*, vol. 158, p. 104450, 2024. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0968090X23004400

[26] 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. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S2192437623000031

[27] 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. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0968090X18302882

[28] J. Long, W. Tan, W. Szeto, and Y. Li, "Ride-sharing with travel time uncertainty," *Transportation Research Part B: Methodological*, vol. 118, pp. 143–171, 2018. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0191261518303151

[29] J. Wang, X. Wang, S. Yang, H. Yang, X. Zhang, and Z. Gao, "Predicting the matching probability and the expected ride/shared distance for each dynamic ridepooling order: A mathematical modeling approach," *Transportation Research Part B: Methodological*, vol. 154, pp. 125–146, 2021. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0191261521001880

[30] S. M. Meshkani and B. Farooq, "A generalized ride-matching approach for sustainable shared mobility," *Sustainable Cities and Society*, vol. 76, p. 103383, 2022. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S2210670721006569

[31] 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. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0377221720305816

[32] 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. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0968090X21002989

[33] 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. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0968090X23003650

[34] 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. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0191261518311731

[35] 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. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1568494623009833

[36] C. Mao, Y. Liu, and Z.-J. M. Shen, "Dispatch of autonomous vehicles for taxi services: A deep reinforcement learning approach," *Transportation Research Part C: Emerging Technologies*, vol. 115, p. 102626, 2020. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0968090X19312227

[37] G. Qin, Q. Luo, Y. Yin, J. Sun, and J. Ye, "Optimizing matching time intervals for ride-hailing services using reinforcement learning," *Transportation Research Part C: Emerging Technologies*, vol. 129, p. 103239, 2021. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0968090X21002527

[38] S. Guo, Y. Liu, K. Xu, and D. M. Chiu, "Understanding passenger reaction to dynamic prices in ride-on-demand service," in *2017 IEEE International Conference on Pervasive Computing and Communications Workshops (PerCom Workshops)*, 2017, pp. 42–45.

[39] B. Marthi, "Automatic shaping and decomposition of reward functions," *Proceedings of the 22nd National Conference on Artificial Intelligence*, pp. 601–608, 2007.

[40] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, "Proximal policy optimization algorithms," *arXiv preprint arXiv:1707.06347v2*, 2017. [Online]. Available: https://arxiv.org/abs/1707.06347

Appendix

A Ride-Hailing Simulator Working Flow

```
# Ride-hailing Simulation Environment (PPO)

# Initialize environment with seed and directories for data files
Initialize environment with `data_directory`, `seed`, `driver_files`, `passenger_files`,
    action and observation spaces

# Load data from driver and passenger files and preprocess
load_data():
    Read driver and passenger data from Parquet files
    Remove missing values and reset indices for drivers and passengers
    Load point shapefile and distance matrix for location and distance calculations

# Reset environment for each new episode
reset():
    Randomly choose a driver and passenger file for the episode
    Load data and filter drivers and passengers by start and end times
    Initialize current and historical order and driver dataframes for tracking
    Set initial time and primary pick-up distance for reward calculation
    Return initial state and episode info

# Step through environment based on action taken by the agent
step(action):
    Reset current match dataframes for new step
    If `action == 1`:
        Match drivers with passengers using distance matrix
        Update matched and unmatched orders and drivers
        Log matched results for rewards and return calculations
    Else:
        Increase waiting time for unmatched orders
    Update `current_time` by 1 second
    Retrieve new orders and drivers based on `current_time`
    Identify canceled orders if their waiting time exceeds threshold
    Calculate primary pick-up distance for reward calculation
    Update state, reward, and episode return
    Return new state, reward, done status, and episode info

# Calculate reward based on waiting time and matching distance
calculate_reward():
    Calculate reward based on:
        - Waiting pick-up time reduction
        - Waiting match penalty for remaining unmatched orders
    Return total reward, waiting pick reward, and waiting match penalty

# Get current state representation of the environment
get_state():
    Extract current time, number of orders and drivers, average and max waiting time
    Return state array for agent input

# Render environment (print current time)
render():
    Print current simulation time

# Close environment and release resources
close():
    Print message indicating environment close
```

B Ride-Pooling Simulator Working Flow

```
# Ride-Pooling Simulation Environment (PPO)

# Initialize environment with seed and directories for data files
Initialize environment with `data_directory`, `seed`, `driver_files`, `passenger_files`,
    action and observation spaces

# Load data from driver and passenger files and preprocess
load_data():
    Read driver and passenger data from Parquet files
    Remove missing values and reset indices for drivers and passengers
    Assign unique indices to passengers and drivers
    Load point shapefile and distance matrix for location and distance calculations

# Reset environment for each new episode
reset():
    Randomly choose a driver and passenger file for the episode
    Load data and filter drivers and passengers by start and end times
    Initialize current and historical order and driver dataframes for tracking
    Set initial time, primary pick-up, and detour distances for reward calculation
    Return initial state and episode info

# Step through environment based on action taken by the agent
step(action):
    Reset current match dataframes for new step
    If `action == 1`:
        Match drivers with passengers using distance matrix
        Update matched and unmatched orders and drivers
        Log matched results for rewards and return calculations
    Else:
        Increase waiting time for unmatched orders
    Update `current_time` by 1 second
    Retrieve new orders and drivers based on `current_time`
    Identify canceled orders if their waiting time exceeds threshold
    Calculate primary pick-up and detour distances for reward calculation
    Update state, reward, and episode return
    Return new state, reward, done status, and episode info

# Calculate reward based on waiting time, matching distance, and detour
calculate_reward():
    Calculate reward based on:
        - Waiting pick-up time reduction
        - Waiting match penalty for remaining unmatched orders
        - Detour distance reduction
    Return total reward, waiting pick reward, waiting match penalty, and detour penalty

# Get current state representation of the environment
get_state():
    Extract current time, number of orders and drivers, average and max waiting time
    Return state array for agent input

# Render environment (print current time)
render():
    Print current simulation time

# Close environment and release resources
close():
    Print message indicating environment close
```

C Developing Environment

This project was developed using Python 3.12.4 as the primary environment, along with several key Python libraries and tools to support the implementation of the simulator and Proximal Policy Optimization (PPO) algorithm. The details are as follows:

- **Python Version**
  - **Python 3.12.4**: This version provides stable and updated language features and is compatible with most mainstream machine learning and data science libraries.
- **Basic Utility Packages**
  - `os`: Used for file path operations.
  - `random`: Provides random number generation, used for randomization in the simulator and algorithms.
  - `time`: Used for calculating program runtime and adding delays.
  - `dataclasses`: Provides data class support for defining simple data structures.
- **Data Processing and Mathematical Computations**
  - `pandas`: Mainly used for data import, processing, and storage, facilitating data cleaning, preprocessing, and analysis.
  - `numpy`: Supports multidimensional arrays, suitable for mathematical and statistical computations.
- **Reinforcement Learning Environment**
  - `gymnasium`: Provides a standardized interface for defining and managing reinforcement learning environments. `SyncVectorEnv` is used for parallelized environment creation.
  - `torch` and `torch.nn`: Core modules of PyTorch, used for creating neural network layers and defining model architecture.
  - `torch.optim`: Contains various optimization algorithms used in training neural network models.
  - `torch.distributions.categorical` and `torch.distributions.bernoulli`: Used for defining the discrete action distribution and Bernoulli distribution in the PPO algorithm.
- **Optimization and Graph Theory Analysis**
  - `scipy.optimize`: Supports optimization algorithms, including solving linear programming (`linprog`) and linear assignment problems (`linear_sum_assignment`).
  - `pulp`: Used for modeling and solving linear programming problems.
  - `igraph` and `networkx`: Used for constructing and analyzing graph structures, useful in the simulator for road network generation and shortest path calculation.
  - `osmnx`: Supports importing road networks from OpenStreetMap and provides visualization capabilities.
- **Geospatial Data Processing and Visualization**
  - `geopandas`: Used for handling geospatial data, including reading and manipulating geographic data files.
  - `matplotlib.pyplot` and `seaborn`: Used for data visualization, including generating graphs, scatter plots, and heatmaps.
  - `wandb`: Used for experiment tracking and visualization, allowing for logging of model training progress and performance metrics.
- **Logging and Debugging**

- `SummaryWriter`: A PyTorch logging tool used to record and visualize metrics during training.
- `datetime` and `timedelta`: Used for handling and computing time, especially for timestamped data processing.

In summary, the Python 3.12.4 environment and the packages listed above form the core of the development environment, supporting the simulator ' s operation, data processing, reinforcement learning algorithm implementation, and debugging processes. These libraries not only improve development efficiency but also enhance the model's performance and interpretability.

D 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.

E Map of Pick-up and Drop-off Points

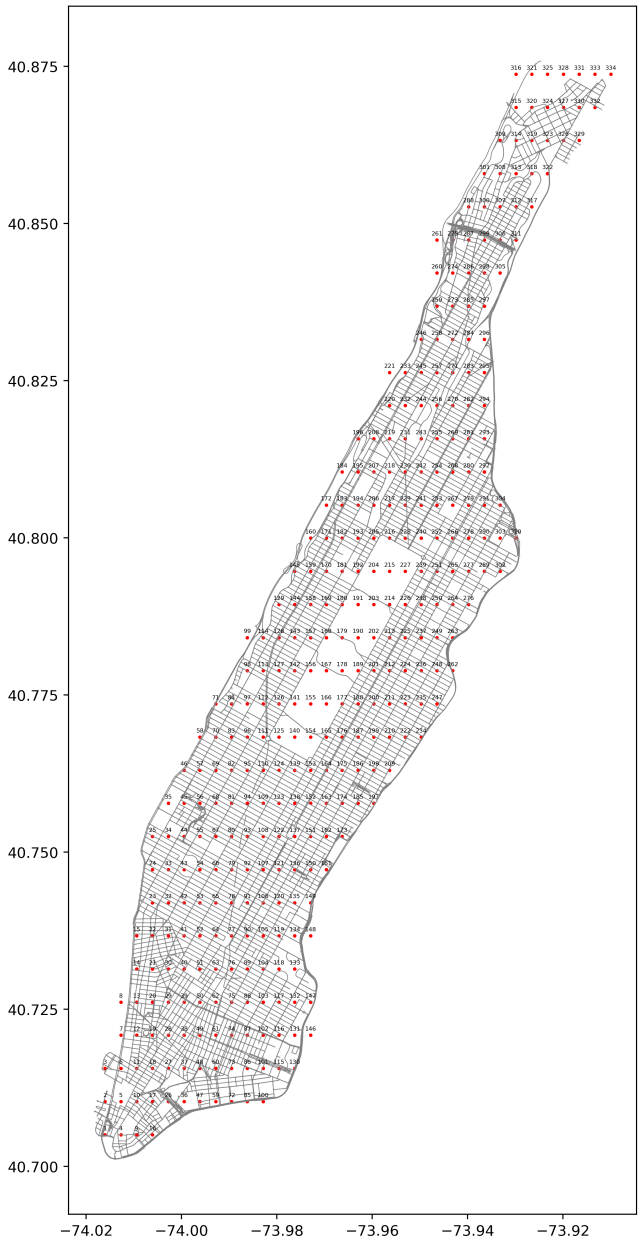

Figure 19: Map of Pick-up and Drop-off Points

F Distribution of Passenger Orders in the Experiment

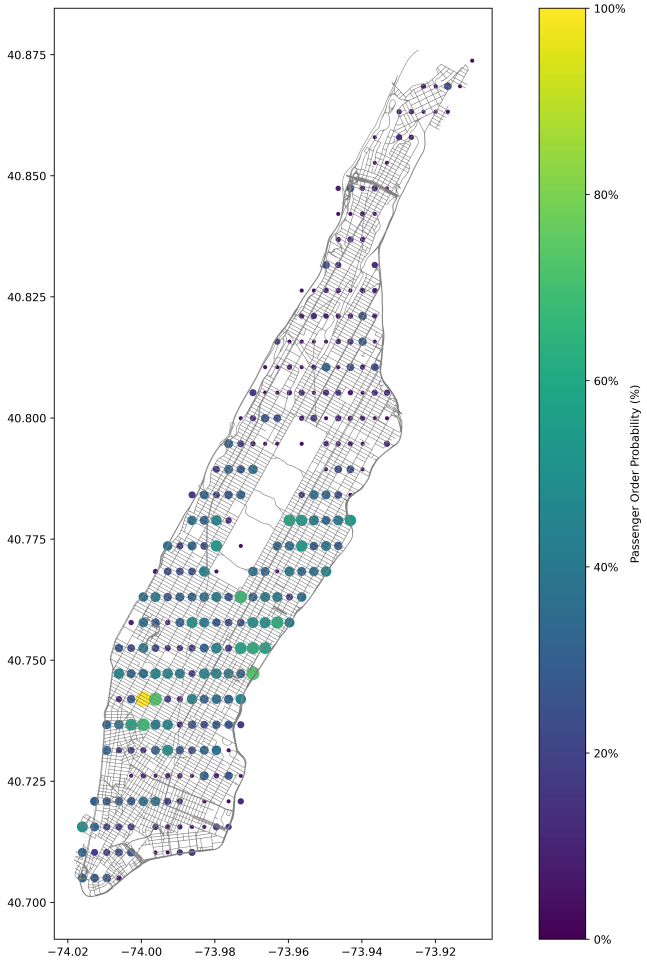

Figure 20: Distribution of Passenger Orders in the Experiment

G Comparison of System Strategy Performance Metrics

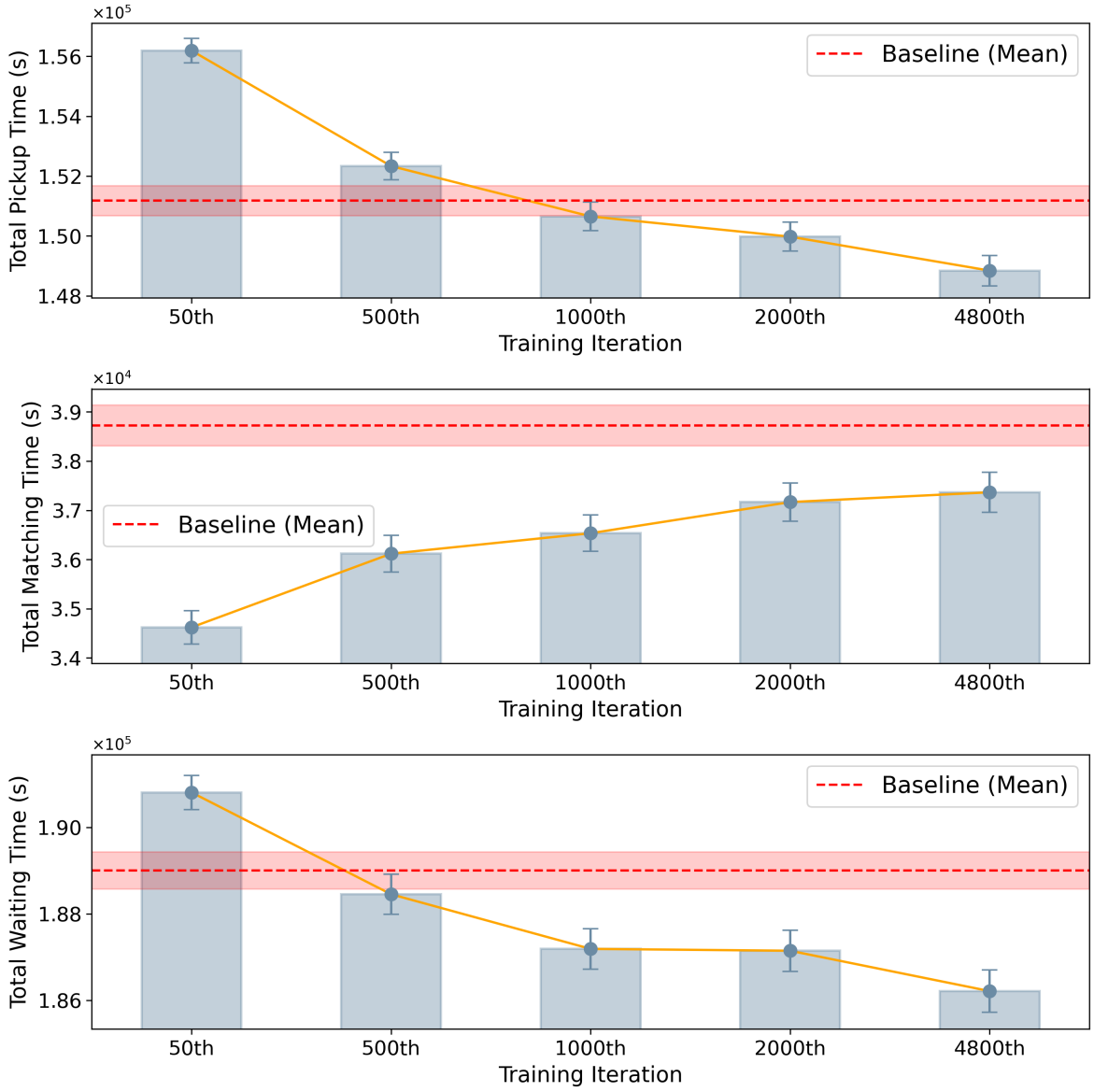

Figure 21: Comparison of System Waiting Times Across Training Episodes and Baseline Strategy for Ride-Hailing Services

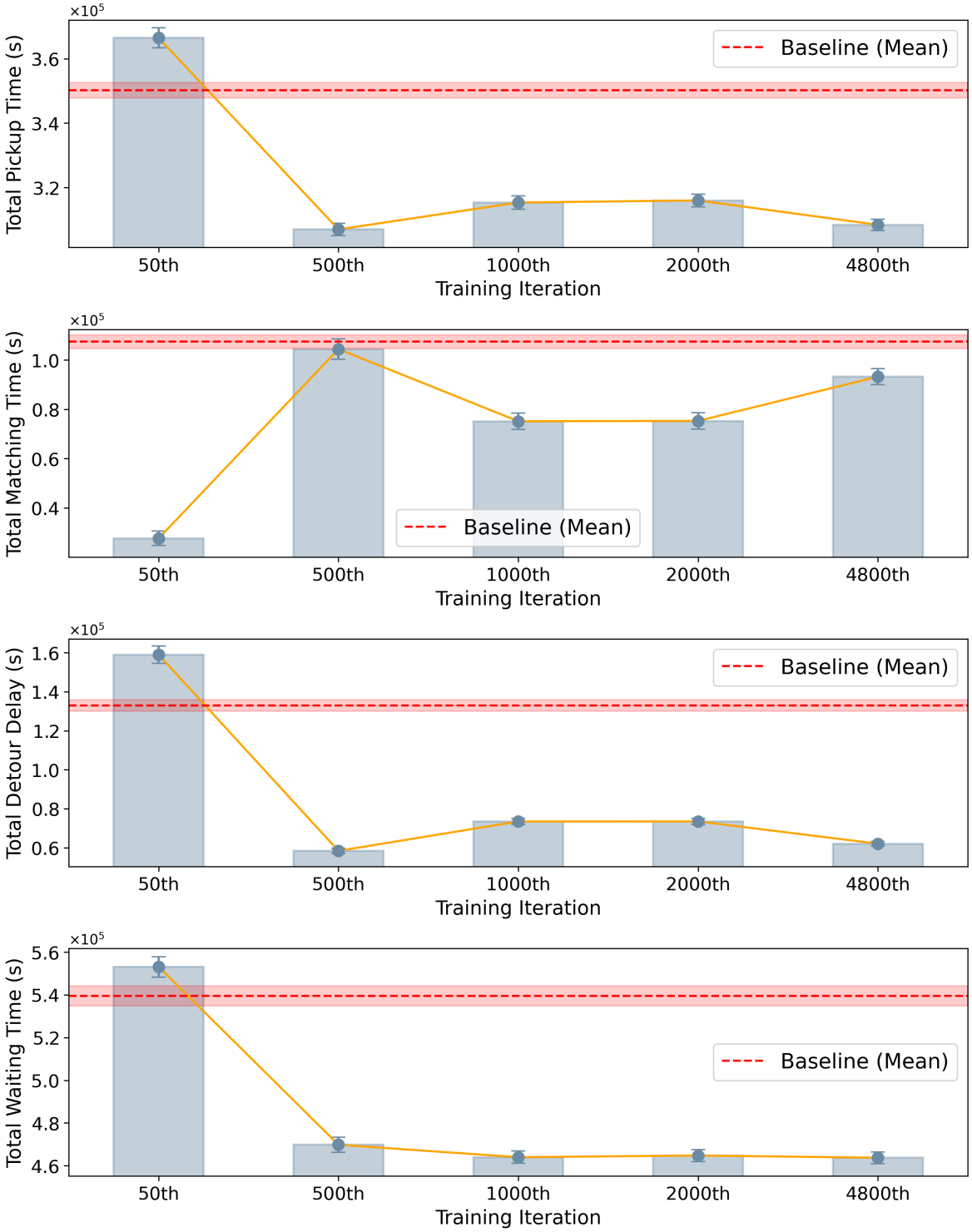

Figure 22: Comparison of System Waiting Times Across Training Episodes and Baseline Strategy for Ride-Pooling Services

H Metrics of Ride-Hailing vs. Ride-Pooling

Table 5: All Metrics of Ride-Hailing vs. Ride-Pooling

Scenario	Metric	Mean (s)	St. Error
A	Total Waiting Time	379437.571	836.411
	Total Pickup Time	360169.971	989.928
	Total Matching Time	19267.600	770.111
	Total Detour Delay	-	-
	Average Pickup Time	282.992	1.601
	Average Matching Time	16.691	0.317
	Average Detour Delay	-	-
	Average Total Waiting Time	299.683	1.322
B	Total Waiting Time	186906.831	911.524
	Total Pickup Time	149792.719	997.843
	Total Matching Time	37114.112	775.760
	Total Detour Delay	-	-
	Average Pickup Time	247.176	1.629
	Average Matching Time	15.306	0.318
	Average Detour Delay	-	-
	Average Total Waiting Time	262.482	1.452
C	Total Waiting Time	1260031.525	909.419
	Total Pickup Time	32902.309	369.068
	Total Matching Time	1227129.216	978.789
	Total Detour Delay	-	-
	Average Pickup Time	54.302	0.611
	Average Matching Time	506.124	0.346
	Average Detour Delay	-	-
	Average Total Waiting Time	560.426	1.221
D	Total Waiting Time	1120662.612	912.412
	Total Pickup Time	31229.316	365.411
	Total Matching Time	1089433.296	975.037
	Total Detour Delay	-	-
	Average Pickup Time	51.539	0.604
	Average Matching Time	449.337	0.344
	Average Detour Delay	-	-
	Average Total Waiting Time	500.876	1.225
E	Total Waiting Time	303618.043	717.689
	Total Pickup Time	147588.364	327.914
	Total Matching Time	48502.144	15.056
	Total Detour Delay	107527.535	501.566
	Average Pickup Time	243.534	0.494
	Average Matching Time	20.008	0.589
	Average Detour Delay	177.470	0.841
	Average Total Waiting Time	441.012	1.163
F	Total Waiting Time	206593.257	586.545
	Total Pickup Time	123634.231	309.869
	Total Matching Time	30900.646	564.160
	Total Detour Delay	52058.379	446.713
	Average Pickup Time	203.995	0.462
	Average Matching Time	12.753	0.235
	Average Detour Delay	85.894	0.727
	Average Total Waiting Time	302.642	0.917
G	Total Waiting Time	333496.330	1068.749
	Total Pickup Time	190680.652	816.571
	Total Matching Time	55513.562	909.496
	Total Detour Delay	87302.116	1061.663
	Average Pickup Time	314.614	1.278
	Average Matching Time	22.887	0.374
	Average Detour Delay	144.089	1.757
	Average Total Waiting Time	481.590	1.669
H	Total Waiting Time	262445.809	840.485
	Total Pickup Time	164033.804	617.049
	Total Matching Time	52273.424	1062.285
	Total Detour Delay	46138.581	493.162
	Average Pickup Time	270.634	0.936
	Average Matching Time	21.555	0.436
	Average Detour Delay	76.162	0.821
	Average Total Waiting Time	368.351	1.249