Ride-Hailing Order Dispatching with A Mixture of On-Demand and Pre-Booked Requests via Reinforcement Learning

Zijian ZHAO[1], Jing GAO[2*], Sen Li[1,3]

[1] Department of Civil and Environmental Engineering, The Hong Kong University of Science and Technology, Hong Kong, China; E-Mail: zzhaock@connect.ust.hk
[2] (Corresponding Author) Department of Logistics and Maritime Studies, The Hong Kong Polytechnic University, Hong Kong, China; E-Mail: jinggao@polyu.edu.hk
[3] Intelligent Transportation Thrust, The Hong Kong University of Science and Technology (Guangzhou), Guangzhou, China; E-Mail: cesli@ust.hk

ABSTRACT

This paper addresses the order dispatch problem in ride-hailing systems with both on-demand and pre-booked requests using a reinforcement learning-based framework. We consider a platform that assigns both types of orders to drivers in real time, optimizing a balanced objective for passengers and the platform while satisfying operational constraints. To better utilize pre-booked requests, we introduce a sequential two-stage order assignment mechanism with pre-booked order reassignment. The dispatch problem is formulated as a Markov Decision Process with action decomposition: pre-booked orders are dispatched first, followed by system updates and on-demand order dispatch. Our approach employs decentralized learning to approximate state-action values for individual vehicles and centralized decision making to determine the optimal fleet-wide assignment using a double deep Q-network (DDQN) and bipartite matching. Experiments with New York City data demonstrate that the proposed DDQN-based method outperforms optimization-based and heuristic baselines, with the order swapping mechanism yielding a 17% increase in total reward.

INTRODUCTION

Ride-hailing platforms such as Uber, Lyft, and Didi have transformed urban transportation by centrally coordinating vehicle fleets and matching passengers with drivers in real time. These platforms offer users convenient, flexible, and cost-effective mobility through mobile apps, resulting in widespread adoption worldwide. For example, Uber operated in over 10,000 cities across 70 countries in 2023, serving 156 million users and completing 9.45 billion trips annually.

To meet diverse travel needs, ride-hailing platforms have expanded their services to include both on-demand and pre-booked rides. While on-demand rides provide immediate pickup, pre-booked rides allow passengers to reserve trips in advance, offering greater reliability and planning flexibility. Most major platforms now support both options to accommodate varying user preferences.

Pre-booked rides benefit both customers and platforms. Customers can secure rides for planned trips, reducing uncertainty and wait times, especially in areas with limited supply or during peak demand. For platforms, advance bookings provide valuable foresight into future demand, enabling more efficient vehicle scheduling and improved system utilization compared to purely on-demand models.

However, integrating pre-booked requests introduces new challenges for real-time fleet management. When both on-demand and pre-booked requests arrive simultaneously, platforms must quickly decide which requests to accept and how to allocate limited vehicle resources. Assessing the profitability of each request is difficult due to dynamic supply and uncertain demand (Chen et al., 2024). Moreover, pre-booked assignments constrain future vehicle availability, reducing dispatch flexibility for on-demand requests (Engelhardt et al., 2022). The coexistence of both request types increases operational complexity and calls for advanced algorithmic solutions.

This paper addresses the real-time order dispatch problem in ride-hailing systems with mixed on-demand and pre-booked requests. We formulate the problem as a Markov Decision Process (MDP) and propose a reinforcement learning-based framework to optimize order assignment, balancing passenger and platform objectives under operational constraints. A sequential two-stage order assignment mechanism with pre-booked order reassignment is proposed, which enables flexible reallocation of pre-booked orders among drivers and reducing assignment complexity through action decomposition. A reinforcement learning-based approach is developed with decentralized learning for individual vehicles and centralized decision making via a double deep Q-network (DDQN) and bipartite matching. Extensive experiments using New York City ride-hailing data demonstrate that our DDQN-based method outperforms optimization-based and heuristic baselines in both total reward and computational efficiency, with the order swapping mechanism yielding a 17% increase in total reward.

METHODOLOGY

Problem setup

We consider a ride-hailing platform managing a fleet of homogeneous drivers and offering four types of services: (i) on-demand pooling, (ii) on-demand non-pooling, (iii) pre-booked pooling, and (iv) pre-booked non-pooling. On-demand requests require immediate pickup, while pre-booked requests allow passengers to schedule

rides in advance. Pooling services enable ride-sharing, whereas non-pooling services guarantee private rides. The platform receives order requests at regular intervals (e.g., every 1 minute) and must assign a mix of order types to drivers, optimizing a system-level objective that balances customer experience (measured by wait time, detour time, service rate, late rate, and delay) and platform revenue.

This order dispatching problem is a high-dimensional sequential decision-making challenge with complex dynamics and operational constraints, making traditional optimization approaches intractable. Modeling the problem as a Markov Decision Process (MDP) and applying reinforcement learning (RL) is a promising alternative. In RL-based dispatching, Q-learning methods estimate the Q-value for each driver-order pair at each time step, and bipartite matching (Shi et al., 2020) is used to maximize the global Q-value for fleet-wide assignments.

However, the coexistence of on-demand and pre-booked requests introduces nuanced constraints. On-demand orders can only be assigned to available vehicles: non-pooling orders to idle vehicles, and pooling orders to vehicles with empty seats and no non-pooling customers. Assigned on-demand orders are executed immediately. In contrast, pre-booked orders can be assigned to any vehicle, regardless of current status, since execution occurs closer to the scheduled pickup time. Vehicles may be assigned both pre-booked and on-demand orders simultaneously, raising questions about assignment timing and execution. Dispatching vehicles too early for pre-booked pickups reduces utilization, while late arrivals diminish customer satisfaction. These factors expand the action space and complicate decision-making, making standard Q-learning with bipartite matching insufficient.

To address these challenges, we propose a mechanism that divides order dispatching into two sequential stages within each interval and enables reassignment of pre-booked orders. First, pre-booked orders are assigned to vehicles (pre-booked order assignment stage), followed by on-demand order assignment to available vehicles (on-demand order assignment stage). This decomposition allows operational constraints for each order type to be addressed separately, and bipartite matching is applied at both stages. During the pre-booked stage, orders already assigned but not yet executed can be reassigned in subsequent intervals, leveraging the temporal flexibility of pre-booked requests and allowing dynamic adjustment as system conditions evolve. Based on this mechanism, we formulate the dispatching problem as an MDP and develop a DDQN-based RL algorithm to derive the optimal policy. The following sections detail the mechanism, MDP formulation, and algorithm.

Mechanism design

During each time interval, the platform receives a set of ride requests and performs order assignment in two sequential stages:

Stage 1 (Pre-booked order assignment): A key challenge in pre-booked order assignment is determining the optimal timing for assignment. To avoid the complexity of explicitly selecting the assignment time, we propose that the platform assigns pre-booked orders in every interval but defers its execution until it approaches the scheduled pickup time. Additionally, a reassignment strategy is employed to dynamically adjust the assignment as system conditions evolve. Specifically, at the beginning of each time interval, the platform gathers information on newly arrived pre-booked orders as well as those previously assigned but not yet executed, evaluates the payoff for each order-vehicle pair, and determines the fleet-wide optimal assignment to maximize the global payoff, subject to exclusivity constraints: (a) each vehicle is matched with at most one order, and (b) each order is assigned to at most one vehicle. The payoff evaluation for each order-vehicle pair is based on the state-action (Q) value function, and the fleet-wide optimal assignment is obtained by bipartite matching, the details of which will be presented in subsequent sections.

To determine when to execute the assignment and dispatch a vehicle to pick up a customer, the platform adopts an As Late As Possible (ALAP) principle. Under this approach, a pre-booked order is only executed (i.e., the vehicle is dispatched to pick up the customer) when the scheduled pickup time is imminent. Otherwise, the order remains in the matching pool and may be reassigned in the next interval as the system state evolves. In doing so, the platform estimates the earliest possible arrival time of the assigned vehicle at the pickup location and compares it with the scheduled pickup time. This estimation depends on the vehicle's current state, as a vehicle can only begin the pickup process once it becomes available. For non-pooling orders, it requires the vehicle to be idle with no ongoing assignments. For pooling orders, the vehicle is considered available either when it is idle or when it is already transporting pooling customers with at least one empty seat. If the estimated arrival time falls within a specified threshold of the scheduled pickup time (e.g., within 5 minutes) or has already passed, the assignment is executed, and the vehicle is dispatched to pick up the customer as soon as it becomes available. Otherwise, the assignment is not exercised, and the order may be reassigned to other vehicles in subsequent intervals. The rationale behind the ALAP principle is to prevent vehicles from arriving too early, which would reduce fleet utilization, while still ensuring timely pickups for pre-booked customers. This approach enables vehicles to serve more on-demand orders, thereby improving overall fleet efficiency without compromising customer experience.

Stage 2 (On-demand order assignment): After the assignment and execution of pre-booked orders, the platform updates each vehicle's state to reflect the newly assigned pre-booked orders. It then proceeds to assign on-demand orders based on the updated vehicle states, which include vehicle location, ongoing on-demand orders, and assigned pre-booked orders. The payoff for each on-demand order-vehicle pair is

evaluated, and the fleet-wide optimal assignment is determined to maximize the global payoff at this stage, still subject to the two exclusivity constraints. Unlike pre-booked orders, on-demand orders are exercised immediately, with vehicles dispatched to pick up customers upon they are assigned. Therefore, only vehicles that are currently available are considered for assignment. Additionally, if assigning an on-demand order would result in a late pickup for an already assigned pre-booked order, such an assignment is deemed infeasible. These additional operational constraints are incorporated into the fleet-wide assignment problem for on-demand orders, which can be also efficiently solved by bipartite matching.

MDP formulation for order dispatching

We formulate the order dispatching problem as a Markov Decision Process (MDP) and employ a reinforcement learning algorithm to solve it. The operating horizon is discretized into H intervals of identical length Δ, i.e., $\mathcal{H} = \{0, \Delta, 2\Delta, \dots, H\Delta\}$, which constitutes an episode in the MDP. At each interval, the platform receives order requests and makes order assignment decisions.

To capture the sequential two-stage order assignment within each interval, we further divide each interval into two steps, resulting in a set of time steps $\mathcal{T} = \{0,1,2, \dots, 2H+1\}$. For each interval h, the time steps $t = 2h$ and $t = 2h+1$ correspond to the pre-booked and on-demand order assignment stages, respectively. It is important to note that these two time steps do not represent specific timestamps within the interval. Instead, they indicate that the assignment action is decomposed into two stages for modeling and optimization purposes. In practice, the execution of both assignments is implemented simultaneously, but the sequential formulation allows for separate optimization of fleet-wide pre-booked and on-demand order assignments within the MDP framework.

- **State Space:** At each time step t, the state of the entire vehicle fleet is denoted by $s_t \in \mathcal{S}$. This state is represented as a tuple $s_t = [s_{1,t}, s_{2,t}, \dots, s_{N,t}]$, where N is the total number of vehicles in the fleet and $s_{i,t}$ is the state vector of individual vehicle i at time t. The state vector of each individual vehicle encompasses both self states and global states. The self states include the current location of the vehicle $L_{i,t}$, its capacity C_i, and the number of empty seats $c_{i,t}$. Additionally, it contains information about assigned but not yet executed pre-booked orders, denoted as $o_{i,t}^{\text{pre}}$, as well as the set of ongoing orders $o_{i,t}^{\text{on}} = [o_{i,1,t}, o_{i,2,t}, \dots, o_{i,m_{i,t},t}]$, where $m_{i,t}$ is the number of ongoing orders for vehicle i at time t. Each ongoing order $o_{i,j,t}$ is represented by its destination location $L_{i,j,t}^{\text{dest}}$, the estimated pickup time $\tau_{i,j,t}^{\text{pickup}}$, and

the order type $k_{i,j,t}$.

The global states provide information relevant to the entire fleet and the current set of orders. These include the proportion of pre-booked orders P_1, the proportion of pooling orders P_2, and the set of orders to be assigned at time t, represented as $o_t^{\text{assign}} = [o_{1,t}, o_{2,t}, \dots, o_{m_t,t}]$, where m_t is the total number of orders to be assigned at time t. Each order to assigned $o_{j,t}$ is characterized by its origin location $L_{j,t}^{\text{origin}}$, destination location $L_{j,t}^{\text{dest}}$, scheduled pickup time $\tau_{j,t}^{\text{pickup}}$ (for on-demand orders, the scheduled pickup time is defined as the time the order is received), and order type $k_{j,t}$. The state vector for assigned but not yet executed pre-booked orders $o_{i,t}^{\text{pre}}$ follows the same structure as $o_{j,t}^{\text{assign}}$.

- **Action Space:** At each time step t, the action taken by the platform is represented by a vector $a_t \in \mathcal{A}(s_t)$, which specifies the assignment decisions for the entire fleet. This action vector is defined as $a_t = [a_{1,t}, a_{2,t}, \dots, a_{N,t}]$, where $a_{i,t}$ is the action vector for individual vehicle i at time t. For each vehicle, $a_{i,t}$ is a binary vector of length m_t, with each element $a_{i,t}[j]$ indicating whether order $o_{j,t}^{\text{assign}}$ is assigned to vehicle i at time t. Specifically, $a_{i,t}[j] = 1$ if the order is assigned to vehicle i, and $a_{i,t}[j] = 0$ otherwise. This formulation allows the platform to flexibly assign multiple orders to vehicles, subject to operational constraints such as assignment exclusivity, vehicle capacity, and order compatibility.
- **Reward Function:** The reward function is designed to align with the system objective that balances both customer experience and platform revenue. It is defined separately for the on-demand and pre-booked order assignment stages. For the on-demand order assignment stage, the reward for assigning order j to vehicle i at time t is given by:

$$R^{\text{on}}(s_{i,t}, a_{i,t}) = \begin{cases} \beta_1 + \beta_2 d_{j,t} - \beta_3 \tau_{i,j,t}^{\text{pickup}} - \beta_4 \rho_{i,j,t}, & \text{if order } j \text{ is assigned} \\ 0, & \text{if no orders are assigned} \end{cases} \quad (1)$$

where $\beta_1, \beta_2, \beta_3, \beta_4 \geq 0$ are weighting coefficients, $d_{j,t}$ denotes the trip distance for order j (calculated as the shortest path between the origin and destination), $\tau_{i,j,t}^{\text{pickup}}$ is the pickup time for order j, and $\rho_{i,j,t}$ represents the additional travel time incurred for all ongoing orders when order j is added to the route of vehicle i at time t. This reward structure directly reflects the dual objectives of the system. The terms $\beta_1 + \beta_2 d_{j,t}$ capture the platform's revenue model, which typically consists

of a base fare plus a distance-based fare, thus aligning with the platform's financial interests. On the other hand, $-\beta_3 \tau_{i,j,t}^{\text{pickup}}$ penalizes longer customer waiting times, thereby promoting better customer experience, while $\rho_{i,j,t}$ accounts for additional detour time due to ride-pooling and $-\beta_4 \rho_{i,j,t}$ penalizes prolonged detour time and minimizes customer inconvenience. By appropriately tuning the weighting coefficients, the platform can balance the trade-off between maximizing revenue and enhancing customer satisfaction.

For the pre-booked order assignment stage, the reward function accounts for both the execution and timeliness of the assignment. The reward of assigning pre-booked order j to vehicle i at time t is defined as:

$$R^{\text{pre}}(s_{i,t}, a_{i,t}) = \begin{cases} \alpha_1 + R^{\text{on}}(s_{i,t}, a_{i,t}), & \text{if order } j \text{ is executed without delay} \\ -\alpha_2(t - \tau_{j,t}) + R^{\text{on}}(s_{i,t}, a_{i,t}), & \text{if order } j \text{ is executed with delay} \\ -\alpha_3, & \text{if order } j \text{ is not executed but is already late} \\ -\alpha_4, & \text{if order } j \text{ is not executed but may be late} \\ 0, & \text{otherwise} \end{cases} \quad (2)$$

where $\alpha_1, \alpha_2, \alpha_3, \alpha_4 \geq 0$ are weighting coefficients, and $\tau_{j,t}$ is the scheduled pickup time for order j. This reward structure is designed to reflect both the importance of fulfilling pre-booked orders and the timeliness of their execution. The term $\alpha_1 + R^{\text{on}}(s_{i,t}, a_{i,t})$ rewards the successful and timely execution of pre-booked orders, where the reward for execution follows the same structure as the on-demand order assignment stage to ensure consistency in evaluating both types of requests. If a pre-booked order is executed with delay, a penalty proportional to the delay duration, $-\alpha_2(t - \tau_{j,t})$, is applied in addition to the on-demand reward, discouraging late pickups. If the order is not executed yet but is already late (the current time is late than the scheduled pickup time), a fixed penalty $-\alpha_3$ is imposed, while a smaller penalty $-\alpha_4$ is assigned if the order is not executed but is estimated to be late (the earliest possible arrival time at the pickup location exceeds the schedule pickup time). This design encourages the platform to prioritize the timely fulfillment of pre-booked orders, penalize delays, and maintain a balance between operational efficiency and customer satisfaction.

The fleet-wide rewards on both stages are assumed to be the sum of the rewards from all vehicles:

$$\begin{cases} R^{\text{on}}(s_t, a_t) = \sum_{i=1}^{N} R^{\text{on}}(s_{i,t}, a_{i,t}) \\ R^{\text{pre}}(s_t, a_t) = \sum_{i=1}^{N} R^{\text{pre}}(s_{i,t}, a_{i,t}) \end{cases} \quad (2)$$

Solution

In this section, we present a solution method that integrates Double Deep Q-Network (DDQN) reinforcement learning (Van et al., 2016) with bipartite matching to derive the optimal order dispatching policy under the MDP formulation, while satisfying the fleet-wide operational constraints. At each time step t, the fleet-wide optimal order assignment action is obtained by solving the following bipartite matching problem:

$$\max_{a_{i,j}} \sum_{i\in\mathcal{I}, j\in\mathcal{J}} Q_{i,j} \cdot a_{i,j} \tag{3}$$

subject to

$$\begin{gathered} \sum_{j\in\mathcal{J}} a_{i,j} \le 1 \\ \sum_{i\in\mathcal{J}} a_{i,j} \le 1 \\ a_{i,j} \in \{0,1\} \\ \sum_{i\in\mathcal{I}, j\in\mathcal{J}} a_{i,j} = \min\left(|\mathcal{I}|, |\mathcal{J}|\right) \end{gathered} \tag{4}$$

where $\mathcal{I}, \mathcal{J}$ denote the sets of vehicles and orders to be assigned at time step t, respectively. Q is the state-action value matrix with entry Q_{ij} representing the state-action value of assigning order j to vehicle i. A is the assignment matrix, where $a_{i,j} = 1$ indicates that order j is assigned to vehicle i, and $a_{i,j} = 0$ otherwise. The first three constraints ensure that each vehicle is assigned at most one order, and each order is assigned to at most one vehicle. The final constraint is designated for the pre-booked order assignment stage which ensures that all available assignments are made, preventing indefinite delays of pre-booked orders due to negative Q values. Besides, during the on-demand order assignment stage (i.e., $t = 2h + 1$), the Q values for all unavailable vehicles and infeasible vehicle-order pairs are set to a sufficiently negative value ($-\bar{Q}$) to exclude them from assignment.

To approximate the optimal state-action value function for each vehicle, we employ a neural network that takes the state and action as input and outputs the corresponding state-action value. This neural network is denoted as $Q_{NN}(s_{i,t}, a_{i,t}; \theta)$, where θ represents the network parameters. We assume a shared state-action function across all vehicles and utilize a single neural network with shared parameters to estimate the optimal state-action values for different vehicles given their specific states and actions. During training, we use two networks—a main network and a target network—and adopt experience replay to stabilize the learning process. The loss function is defined as the Bellman optimality error:

$$L = \mathbb{E}\left[\left(R(s_{i,t}, a_{i,t}) + \gamma Q_{NN}(s_{i,t+1}, a_{i,t+1}^{\text{opt}}; \theta_T) - Q_{NN}(s_{i,t}, a_{i,t}; \theta)\right)^2\right] \tag{5}$$

where $a_{i,t+1}^{\text{opt}}$ is the optimal order assignment for vehicle i at time step $t+1$, obtained from the bipartite matching problem (3), and θ_T denotes the parameters of the target network, which are updated at a slower rate than those of the main network.

For the network architecture, we follow our previous work: two encoders are used to extract features from vehicle and order information, respectively, followed by a QK-Attention module to estimate the Q value based on the combined embeddings. Further details of the network structure can be found in (Zhao & Li, 2025).

EXPERIMENTS AND RESULTS

Experimental setup

We evaluate our approach using yellow taxi data from Manhattan, New York City, collected on July 1, 2024, between 18:00 and 19:00, comprising 4,870 valid records. In our simulation, the pickup time is defined as the appearance time for on-demand orders and the scheduled pickup time for pre-booked orders. The simulation spans one hour, with the first 30 minutes consisting exclusively of on-demand orders, while all pre-booked orders are introduced in advance of their scheduled pickup times. In the final 30 minutes, pre-booked orders constitute $p\%$ of the total orders. Throughout the simulation, $q\%$ of all orders (including both pre-booked and on-demand) are pooling orders, while $1-q\%$ are non-pooling orders. For each training episode, the values of p and q are randomly sampled from a uniform distribution $U(0,100)$ to ensure the learned strategy is robust under varying conditions. Additionally, the advance time for pre-booked orders (i.e., the interval between the request time and the scheduled pickup time) is randomly sampled from $U(10,30)$, indicating that customers submit pre-booked requests 10 to 30 minutes prior to their scheduled pickup.

The detailed configuration of our method is provided in Table 1. The neural network is implemented using PyTorch and trained on a workstation running Windows 11, equipped with an Intel(R) Core(TM) i7-14700KF processor and an NVIDIA RTX 4080 GPU. During training, the model utilized approximately 1.1 GB of GPU memory.

To assess the performance of the proposed RL-based order assignment framework, we benchmark it against two baselines: a rule-based method and a myopic optimization-based method. In the rule-based approach, each order—whether pre-booked or on-demand—is assigned to the nearest available driver at both pre-booked and on-demand order assignment stages. In the optimization-based method, order assignments are determined by solving a bipartite matching problem that maximizes the total immediate reward for the fleet at each time step. This myopic assignment problem is formulated identically to (3), except that $Q_{i,j}$ is replaced by $R_{i,j}$, the immediate reward for assigning order j to vehicle i. To further illustrate the advantages

of the proposed mechanism—namely, the sequential two-stage order assignment with pre-booked order reassignment—we compare the performance metrics of various methods both with and without this mechanism. In the absence of the proposed mechanism, pre-booked orders are assigned and executed immediately upon submission. Consequently, if the assigned vehicle arrives at the pickup location before the scheduled pickup time, it must wait until the designated time to pick up the customer.

Table 1. A Summary of Training Settings

Configuration	Our setting	Configuration	Our setting
Batch size	512	Exploration rate	0.99 to 0.0005
Optimizer	Adam	Training episodes	1,000
Scheduler	ExponentialLR	Number of vehicles	1,000
Initial learning rate	0.0005	Number of parameters	157,200

Results

The performance comparison of the proposed reinforcement learning-based method and baselines with and without the sequential two-stage order assignment mechanism is shown in Table 2. The experimental results demonstrate that the proposed DDQN-based method outperforms the model-based baselines by effectively capturing long-term uncertainties and system dynamics inherent in the order dispatching process. While the optimization-based method yields superior performance for on-demand services, it fails to account for the downstream impact on pre-booked orders. This oversight leads to a higher rate of late pickups and increased customer wait times for pre-booked orders, as a disproportionate number of drivers are allocated to on-demand requests, thereby compromising service quality for pre-booked customers. In contrast, our DDQN-based approach achieves a more equitable allocation of resources between on-demand and pre-booked orders, resulting in improved overall system performance as measured by total reward. This balanced strategy ensures that the service requirements of both order types are met, reducing late pickups and wait times for pre-booked customers without sacrificing efficiency in on-demand service.

Furthermore, the optimization-based method incurs substantial computational overhead, as it necessitates evaluating the reward for all possible order-vehicle assignments at each decision epoch. For our one-hour simulation, this approach required approximately 21 hours of computation time, rendering it impractical for real-time deployment. In contrast, the DDQN-based method completed the same simulation in approximately 2.4 minutes, highlighting its computational efficiency and suitability for real-world applications.

Finally, a comparison of performance metrics for the proposed DDQN-based method, both with and without the developed mechanism, demonstrates the

effectiveness of the sequential two-stage order assignment and pre-booked order reassignment strategy. Specifically, incorporating this mechanism yields a 17% improvement in the overall cumulative reward, highlighting its significant contribution to enhanced fleet efficiency and service quality.

Table 2. Performance Comparison of the Proposed Method and Baselines With and Without the Sequential Two-Stage Order Assignment Mechanism

Methods / Metrics	w/ Our Mechanism Design			w/o Our Mechanism Design		
	DDQN	Optimization	Greedy	DDQN	Optimization	Greedy
Total Reward	**39.35**	33.49	-3.81	33.69	29.61	4.31
On-demand Reward	21.00	**29.51**	17.86	12.63	16.88	12.68
Pre-booked Reward	18.35	3.98	-21.67	**21.06**	12.73	-8.37
Platform Profit	58.92	**64.08**	52.42	43.84	50.95	42.03
Simulation Time (hour)	**0.04**	21.12	0.05	**0.04**	20.89	**0.04**
Average Detour Time (min)	12.84	**6.78**	15.28	12.86	16.98	18.82
Pooling & Pre-booked Orders						
Service Rate (%)	**100**	99.49	98.97	**100**	98.36	99.71
Late Rate (%)	15.62	28.89	54.49	**5.12**	13.39	27.73
Average Late Time (min)	0.47	>1.22	>4.21	**0.17**	>0.51	>2.83
Non-pooling & Pre-booked Orders						
Service Rate (%)	**100**	74.26	44.29	**100**	78.99	47.71
Late Rate (%)	**25.94**	69.97	88.31	25.95	47.29	74.57
Average Late Time (min)	**0.93**	>5.74	>9.89	3.64	>1.51	>7.62
Pooling & On-demand Orders						
Service Rate (%)	91.45	**99.52**	80.92	51.34	74.25	49.73
Average Pickup Time (min)	6.63	**2.71**	10.32	8.01	6.92	10.67
Non-Pooling & On-demand Orders						
Service Rate (%)	65.43	**99.12**	64.24	31.73	50.08	46.90
Average Pickup Time (min)	6.44	**3.80**	10.51	9.69	9.42	10.65

CONCLUSION

This paper presents a multi-agent reinforcement learning framework for real-time order dispatching in ride-hailing systems that accommodate both on-demand and pre-booked requests. By formulating the dispatch problem as a Markov Decision Process and introducing a sequential two-stage assignment mechanism with pre-booked order reassignment, our approach effectively balances operational constraints and optimizes fleet performance. Experimental results using real-world data demonstrate that the proposed DDQN-based method significantly outperforms optimization-based and heuristic baselines in both total reward and computational efficiency. The proposed mechanism further enhances system performance, yielding a substantial improvement in cumulative reward. These findings highlight the potential of reinforcement learning-based solutions for advancing the efficiency and reliability of modern ride-hailing platforms.

REFERENCES

Chen, Y., Liu, Y., Bai, Y., & Mao, B. (2024). Real-time dispatch management of shared autonomous vehicles with on-demand and pre-booked requests. *Transportation research part A: policy and practice*, 181, 104021.

Engelhardt, R., Dandl, F., & Bogenberger, K. (2022). Simulating ride-pooling services with pre-booking and on-demand customers. *arXiv preprint arXiv*:2210.06972.

Shi, J., Gao, Y., Wang, W., Yu, N., & Ioannou, P. A. (2019). Operating electric vehicle fleet for ride-hailing services with reinforcement learning. *IEEE Transactions on Intelligent Transportation Systems*, 21(11), 4822-4834.

Zhao, Z., & Li, S. (2025). The impacts of data privacy regulations on food-delivery platforms. *Transportation Research Part C: Emerging Technologies*, 181, 105364.

Van Hasselt, H., Guez, A., & Silver, D. (2016, March). Deep reinforcement learning with double q-learning. *In Proceedings of the AAAI conference on artificial intelligence* (Vol. 30, No. 1).