RG-CQL: A Reward-Guided Conservative Q-Learning Framework for the Coordination of Ride-Pooling and Public Transit Services

Yulong Hu*[1] and Sen Li[1,2]

[1]Department of Civil and Environmental Engineering, The Hong Kong University of Science and Technology, Hong Kong, China
[2]Intelligent Transportation Thrust, Systems Hub, The Hong Kong University of Science and Technology (Guangzhou), Guangzhou, China

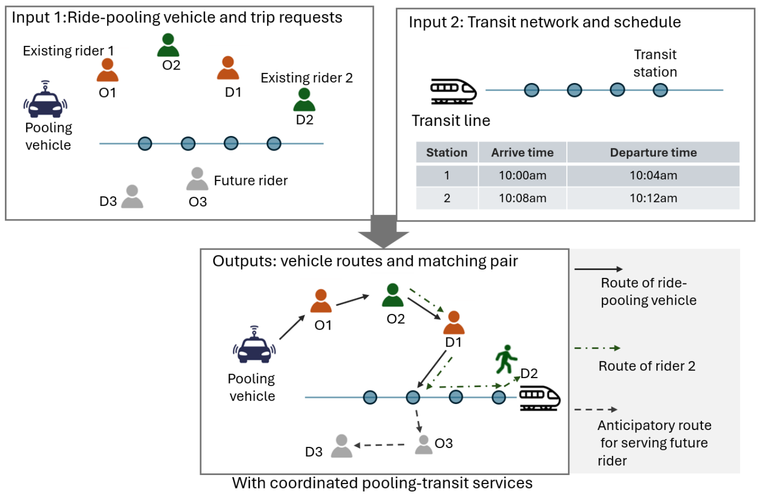
Short summary

This paper presents the Reward-Guided Conservative Q-learning (RG-CQL), a novel Reinforcement Learning (RL) framework designed to optimize the coordination between ride-pooling and public transit in multimodal transportation systems. By modeling the problem as a Markov Decision Process (MDP), RG-CQL employs a two-phase approach: offline learning and online fine-tuning. In the offline phase, the Conservative Double Deep Q Network (CDDQN) as potential action executor and a Guider Network as rewards estimator are trained directly from past noisy data trajectories. During the online fine-tuning phase, the Guider Network assists the CDDQN in exploring new state-action pairs, merging conservative training with optimistic online strategies. Extensive numerical experiments in Manhattan demonstrate that RG-CQL improves the operational performance of multi-modal transportation systems by 4.3%, reduces training sample complexity by 81.3%, and effectively addresses the Offline to Online (O2O) RL challenge in large-scale ride-pooling systems.
Keywords: Multimodal Transportation, Ride-Pooling, Public Transit, Offline Reinforcement Learning, Online Fine-tuning, Offline to Online.

1 Introduction

In recent years, the widespread use of transportation network companies, such as Uber, Lyft, and Didi, has raised concerns about their negative impacts on traffic and the environment. For these ride-hailing platforms, a pressing concern is that ride-hailing services may compete with public transit systems and divert a substantial number of riders away from public transportation. As shown in Qin et al. (2022), this could lead to a vicious cycle that undermines public transit development. To address the aforementioned concerns, substantial research has been conducted on how to promote collaborative relationships between ride-hailing and public transportation systems, like in Feng et al. (2022); Gao & Li (2024). The underlying premise is that ride-hailing services can either complement or compete with public transit. To maximize the complementary effects between these two, strategies have been proposed to utilize ride-pooling services for the first and/or last leg of a journey, while leveraging public transit for the middle leg between transport hubs.

However, operating ride-pooling services within multimodal transportation networks that consider inter-modal transfers poses significant research challenges. Firstly, the platform must determine how to pool passengers, match them to vehicles, route these vehicles, and select appropriate pickup and drop-off points for inter-modal transfers. These tasks are intricately intertwined and must be addressed in real-time amid significant uncertainties, rendering traditional model-based algorithms like in Gu & Liang (2024) inadequate. This has led to the exploration of Reinforcement Learning (RL) algorithms. However, in the realm of RL, training models from scratch is not only time-consuming but also prone to yielding low-quality local solutions due to the enormous decision space and the trial-and-error nature illustrated in Sutton et al. (1998). To address these issues, some researchers have proposed adopting off-policy RL techniques like in Yu & Gao (2022), to learn value functions from batches of past vehicle transitions. However, these methods often suffer from significant overestimation issues due to out-of-distribution (OOD) data transitions during the offline training phase, as demonstrated by Kumar et al. (2020). Moreover as shown in Nakamoto et al. (2024), there is a persistent gap from offline learning to online learning (O2O), where current


Input 1:Ride-pooling vehicle and trip requests
Existing rider 1
O2
O1
D1
Existing rider 2
D2
Pooling vehicle
Future rider
D3
O3
Input 2: Transit network and schedule
Transit station
Transit line
Outputs: vehicle routes and matching pair
O1
O2
D1
D2
Pooling vehicle
D3
O3
Route of ride-pooling vehicle
Route of rider 2
Anticipatory route for serving future rider
With coordinated pooling-transit services

Station	Arrive time	Departure time
1	10:00am	10:04am
2	10:08am	10:12am

Figure 1: Problem setup with coordinated ride-pooling and transit services

offline RL methods also experience slow learning and initial unlearning during subsequent online training.

Acknowledging the challenges mentioned above, we hereby propose Reward Guided Conservative Q-learning (**RG-CQL**). This approach leverages an offline training and online fine-tuning RL framework to effectively coordinate ride-pooling with transit, complemented with a Guider to improve the efficiency of exploration. In particular, we first formulate the ride-pooling problem in multimodal transportation networks as a Markov Decision Process (MDP), and then proceed to train an offline learning Conservative Double Deep Q Network (CDDQN) policy, which serves as the potential action executor, alongside a supervised learning reward estimator during the offline training phase to maximize insights from batches of past environmental transitions. Subsequently, during the online fine-tuning phase, we employ the reward estimator as the exploration 'Guider', aiding CDDQN in effectively and conservatively exploring unknown state-action pairs. Our extensive numerical experiments in Manhattan validate that the proposed RG-CQL framework not only markedly enhances the operational performance of multimodal transportation systems but also well handle the O2O challenges in RL for large-scale ride-pooling system.

2 Methodology

In this section, we first discuss our problem setup and then present how we formulate the problem of coordinating ride-pooling with public transit as a Markov Decision Process (MDP) under bipartite match. Following the formulation, we introduce our RG-CQL, an offline RL pretraining and reward-guided online RL fine-tuning framework, for both effectively and efficiently solving the proposed MDP model.

Problem Setup for Coordinating Ride-pooling with Public Transit

As shown in Fig. 1, we consider using ride-pooling services to address the first-mile problem of transit services. Riders requesting trip services are assumed to be willing to use ride-pooling services on their first leg of trips and transit services on the second leg of their trips. A ride-pooling vehicle could pick up a rider from her origin and deliver the rider directly to her destination (like Rider 1) or to a transit station so that the rider can take transit vehicles (like Rider 2). Riders who continue their trip using transit would get off at the stations closest to their destinations that are accessible via walking. Transit vehicles are assumed to follow fixed routes and schedules. Riders dropped at transit stations are expected to follow the shortest transit route to minimize travel time to their destinations, potentially transferring between transit lines. The centralized platform is tasked with optimally dispatching ride-pooling vehicles to riders, deciding whether to drop them off at their destinations or transit stations. This decision-making should rely not only on current vehicle request information but also anticipates future incoming orders, like Rider 3.

Following two assumptions common in the literature of multi-agent RL like Feng et al. (2022); Wang et al. (2023) for ride-pooling systems, we consider each ride-pooling vehicle as homogeneous and an independent agent. The MDP problem is formulated by a tuple $\langle S, A, P, R, \gamma \rangle$. At time t, vehicle agent $n \in N$ observes the environment state $s_{n,t} \in S$ representing number of vacant seats, information of passengers on board, potential bipartite matched order, and current time. According to the policy π mapping from states to actions $a_{n,t} = \pi(a_{n,t}|s_{n,t})$, the vehicle agent chooses an action $a_{n,t} \in A$ to drop off passenger at zone $a_{n,t} = z \in Z$ to get on the transit like subway to finish the remaining journey or directly drive the passenger towards its destination $a_{n,t} = 0$. After taking the action, the agent receives reward $r_{n,t} \in R$ reprsenting the trade-off between order and passenger detour compared to direct ride-sourcing transfer. Subsequently, the agent shifts to the next state $s_{n,t+1}$, which is sampled from transit function $p(s_{n,t+1}|s_{n,t}, a_{n,t})$. Assuming to share the same policy due to homogeneity, the agents' objective is to find the optimal policy π^* to maximize the long term accumulative discounted rewards represented via the Q-value function below:

$$\begin{aligned} \pi^* &= \arg\max_{\pi} \mathbb{E}_{\pi} \left[\sum_{n \in \mathcal{N}} \sum_{\tau \in K_t} \gamma^{\tau} \cdot r_{n,t+\tau} \mid s_{n,t}, , a_{n,t} \right] \\ &= \arg\max_{\pi} \sum_{n \in \mathcal{N}} Q_{\pi}(s_{n,t}, a_{n,t}) \end{aligned} \tag{1}$$

where γ is the discounted factor and $K_t = \{0, 1, 2, T - t\}$ denotes the set of time steps afterwards t until the end of planning horizon at terminal time T.

To determine the matched order while fully considering the uncertainty and complexity of coordinated ride-pooling with transit, we formulate a bipartie match based on Q-value function. As shown in Fig. 2, we represent available ride-pooling vehicles and riders waiting to be matched as two sets of nodes $\mathcal{N}$ and $\mathcal{M}$, respectively. Edges connect each vehicle node n in set $\mathcal{N}$ to each rider node m in set $\mathcal{M}$. A weight $w(n, m)$ is associated with the edge connecting nodes n and m, which measures the gains for matching ride-pooling vehicle n with rider m. Inspired by Sutton et al. (1998), we use an ϵ-greedy strategy to encourage exploration of vehicle agents for finding optimal policy in large-scale multi-modal transportation system. Specifically, for exploitation, we select the action $a_{n,t}$ that maximizes the agent's expected return $Q_{\pi}(s_{n,t}, a_{n,t})$ for given agent n at state $s_{n,t}$. Correspondingly, weight $w(n, m)$ takes value $\max_{a_{n,t}} Q_{\pi}(s_{n,t}, a_{n,t})$. For exploration, the agent n is assigned a random action $a_{n,t} \in \mathcal{Z} \cup 0$, meaning that the ride-pooling vehicle n would drop off a rider m at a randomly selected zone. The corresponding value of $w(n, m)$ is set as a large positive number $\overline{Q}$, driving the agent to take such a random action after being matched. The trade-off between exploitation and exploration is controlled by a parameter $\epsilon \in (0, 1)$ that specifies the exploration rate, and the corresponding weight values can be expressed as:

$$\begin{cases} w(n, m) = \max\limits_{a_{n,t}} Q_{\pi}(s_{n,t}, a_{n,t}), & \text{with probability } 1 - \epsilon \\ w(n, m) = \overline{Q}, & \text{with probability } \epsilon \end{cases} \tag{2}$$

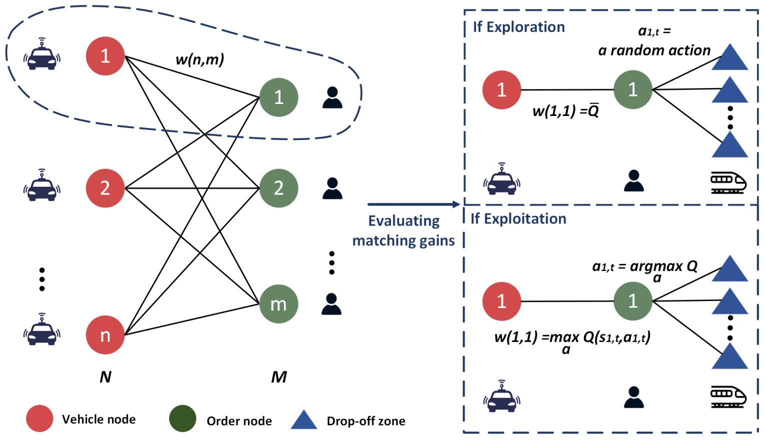


Figure 2: Visualization of bipartite matching graph

With the bipartite graph defined above, the action of agents $a_{n,t}$ and the resultant expected returns can be uniquely determined at a decision time t once the selected edges linking vehicle nodes and rider nodes are established. To this end, we introduce a variable $x_{n,m}$ for each edge, which equals 1 if the edge connecting nodes m and n is selected and 0 otherwise. The following ILP program is formulated for decisions on order-dispatching and drop-off locations, which merges reinforcement learning's policy function with a bipartite matching process:

$$\max_{X} \sum_{n:n\in\mathcal{N}} w(n,m)\cdot x_{n,m}, \tag{3a}$$

$$\text{s.t.} \quad \sum_{n:n\in\mathcal{N}} x_{n,m} \leq 1, \quad \forall m \in \mathcal{M}, \tag{3b}$$

$$\sum_{m:m\in\mathcal{M}} x_{n,m} \leq 1, \quad \forall n \in \mathcal{N}, \tag{3c}$$

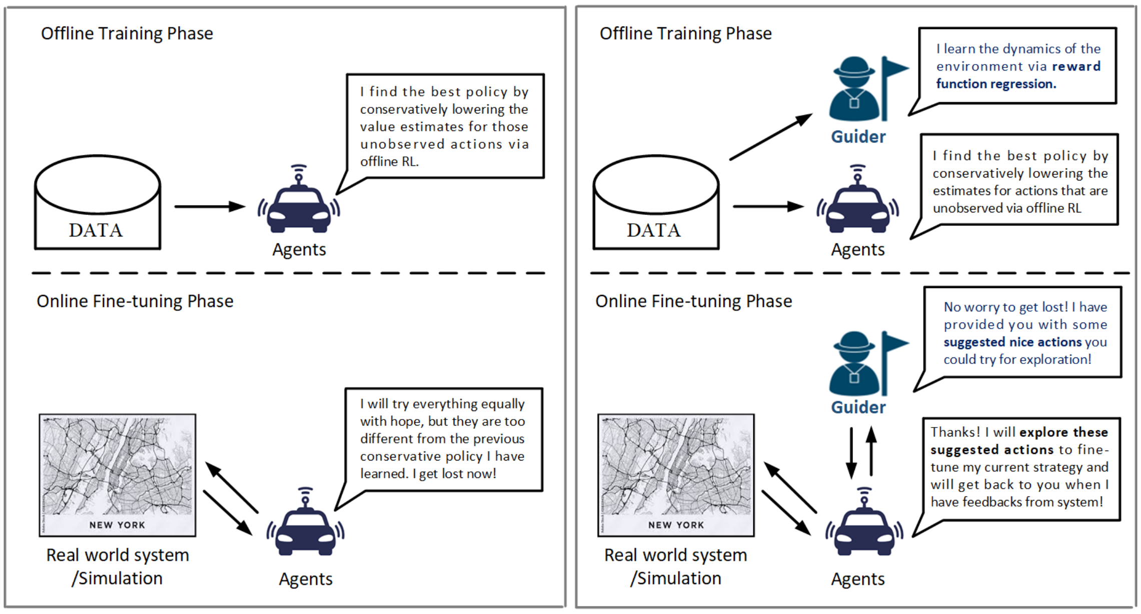
$$\sum_{n:n\in\mathcal{N}} x_{n,m}\cdot d_{n,m} \leq R_{match}, \quad \forall m \in \mathcal{M}, \tag{3d}$$

$$x_{n,m} \in \{0,1\}, \quad \forall n \in \mathcal{N}, m \in \mathcal{M}, \tag{3e}$$

where $X = \{x_{n,m}\}_{n\in\mathcal{N},m\in\mathcal{M}}$ denotes the set of decisions variables. The objective (3a) is to maximize the platform's expected profits. Constraint (3b) ensures that each order can be matched with at most one ride-pooling vehicle while Constraint (3c) guarantees that each ride-pooling vehicle is matched with at most one order at the decision time. Constraint (3d) guarantees that a ride-pooling vehicle and an order can be matched only if the distance between them $d_{n,m}$ is within a maximum matching distance R_{match}.

RG-CQL for Effective and Efficient Value Function and Policy Learning

This subsection introduces RG-CQL, a novel offline RL pretraining and reward-guided online RL fine-tuning framework for effectively and efficiently solving the proposed MDP model. The overall architecture of our RG-CQL framework is given in Fig. 3(b), divided into two stages respectively: offline training stage and online fine-tuning stage.


Offline Training Phase
I find the best policy by conservatively lowering the value estimates for those unobserved actions via offline RL.
DATA
Agents
Online Fine-tuning Phase
I will try everything equally with hope, but they are too different from the previous conservative policy I have learned. I get lost now!
NEW YORK
Real world system /Simulation
Agents
Offline Training Phase
I learn the dynamics of the environment via reward function regression.
Guider
I find the best policy by conservatively lowering the estimates for actions that are unobserved via offline RL
DATA
Agents
Online Fine-tuning Phase
No worry to get lost! I have provided you with some suggested nice actions you could try for exploration!
Guider
Thanks! I will explore these suggested actions to fine-tune my current strategy and will get back to you when I have feedbacks from system!
NEW YORK
Real world system /Simulation
Agents

(a) Naive CQL family (b) RG-CQL: CQL with a Guider

Figure 3: Overall architecture of RG-CQL (on the right) and its comparison with existing state-of-the art Offline RL family (on the left)

First at the offline training stage, a batch of observations $\mathcal{D}$ regarding state transition are obtained from existing on-demand ride data, with $\mathcal{D}$ containing a series of trajectories $\tau = (s, a, r, s')$. The batch data is not required to be a complete coverage of the environment and could include not only ride-pooling with transit data but also data from pooling-only and non-pooling services under any sample policy (The definitions of vehicle state s, reward r, and action a remain consistent with those outlined in coordinated ride-pooling and transit services). To learn the value function and find the best policy in support of our batch data without any agent interactions with the environment, we leverage the concept of "conservatism" introduced in CQL by Kumar et al. (2020) and formulate the loss function L_c of our CDDQN as follows:

$$\begin{aligned} L_c = \mathbb{E}_{\mathcal{T}\sim\mathcal{D}}\left[\left(r + \gamma Q\left(s', \arg\max_{a'} Q(s', a'; \theta); \theta^-\right) - Q(s, a; \theta)\right)^2\right] \\ + C\left(\mathbb{E}_{s\sim\mathcal{D}}[\max_a Q(s, a; \theta)] - \mathbb{E}_{(s,a)\sim\mathcal{D}}[Q(s, a; \theta)]\right), \end{aligned} \tag{4}$$

where $Q(s, a; \theta)$ is the Q-value estimated by the training Q-network whose neural network parameter is θ, and $Q(s, a; \theta^-)$ is the Q-value estimated by the target network θ^-. The first term has its roots in DDQN by Van Hasselt et al. (2016). C is a hyper-parameter and dictates the extent to which the second regularization term should be accounted for. This additional conservative regularization term penalizes Q-values associated with unobserved state-action pairs in the dataset, encourages the Q-values for unobserved state-action pairs to be minimized, particularly if these Q-values mistakenly emerge as the highest among all actions for a given state in the dataset. The purpose is to mitigate the risk of overestimating Q-values for state-action pairs not present in the dataset. The training and target network parameters θ and θ^- will be updated via gradient descent and Polyak Average by Fujimoto et al. (2018) repectively.

In addition to CDDQN, we introduce an innovative module, referred to as "Guider", to resolve the issue caused by the pessimism-optimism gap in later online-finetuning stage. Specifically, we train a Guider network using supervised learning in addition to the CDDQN training during offline training stage. The Guider network is a neural network that learns a function approximator for reward r using existing data, which would be used in the online stage. The loss function for this Guider network is defined as follows:

$$L_g = \mathbb{E}_{\mathcal{T}\sim\mathcal{D}}\left[\left(r - G(s, a; \phi)\right)^2\right], \tag{5}$$

where L_g is the loss function, $G(s, a; \phi)$ represents the Guider's estimation for the true reward r of state-action pair s-a in data batch $\mathcal{D}$, parameters ϕ denotes the weights used by the Guider network and is updated via gradient descent method during training.

Subsequently during online finetuning stage, as illustrated in Fig. 3(b), the Guider aims to enhance agents' exploration by suggesting less blindly optimistic actions compared with naive CQL family by Kumar et al. (2020); Nakamoto et al. (2024) in Fig. 3(a) while maintaining the potential to discover long-term optimal policies during the fine-tuning phase. To accomplish this, the Guider employs its reward estimation learned previously in offline training stage as the foundational model-based dynamics metric to guide agents' action choices when exploration strategy is used in previous order-matching in Equation (2) to have novel exploration strategy as below:

$$A(s_{n,t}) = \{a \mid G(s_{n,t}, a_{n,t}) > \widehat{r}\}. \tag{6}$$

$$w(n, m) = \begin{cases} \max\limits_{a_{n,t}} Q_\pi(s_{n,t}, a_{n,t}), & \text{with probability } 1-\epsilon, \\ \overline{Q}, & \forall a_{n,t} \in A(s_{n,t}), \text{with probability } \epsilon. \end{cases} \tag{7}$$

where $\widehat{r}$ is a predefined reward threshold. Now for exploration (with probability $1-\epsilon$), an agent would be assigned a random action in set $A(s_{n,t})$, instead of an arbitrary action described in Equation (2). Correspondingly, only the action whose estimated reward is larger than $\widehat{r}$ is assigned to a large upper bound Q-value. This modifies the previously blindly optimistic exploration strategy to a more wisely optimistic exploration strategy. The underlying premise is that state-action pairs generating significant negative short-term rewards, especially during peak demand periods, are likely to lead to a high number of rejected orders. Consequently, these pairs are unlikely to contribute to optimal long-term operational decisions, even when considering the trade-off between immediate rewards and long-term objectives. Therefore, by eliminating these unreasonable decisions from the exploration process, we can significantly guide agents to adopt a more conservative stance during online fine-tuning, while simultaneously improving the efficiency of exploration.

The overview of our RG-CQL framework is depicted in Algorithm 1 in Appendix, which embeds the key innovative concepts of our RG-CQL method delineated.

3 Simulation results and discussion

The overall architecture of our Pooling with Transit (PwT) simulator is shown in the Fig. 4. The simualtor is built based on trip request data extracted from the dataset presented in Haliem et al. (2021), which is sourced from the taxi trips of New York City in Taxi & Commission (2024). From this dataset, we extract data for trips occurring during the morning peak hour (8:00 AM - 9:00 AM) on May 4, 2016, with an average order density of around 271 trips per minute. For online fine-tuning, each training episode involves a sample of 95% of these trips, totaling approximately 15,300 orders. The study area is Central Manhattan, which is partitioned into smaller grid zones with a resolution of 800m x 800m. In the simulation study, fifty-seven zones are selected for simulation. Our routing optimization and vehicle navigation are based on the road network of Manhattan, which is obtained from OpenStreetMap. The information on transit services is obtained from the open-source project in Parrott (2017) based on MTA schedules in Authority (2024). In total, there are 29 subway lines in the entire New York City, encompassing over 380 unique subway stations. We set the number of ride-pooling vehicles as 600 and the seat capacity of each vehicle as 3. The length of interval Δt for order matching is 1 minute. The maximum matching distance R_{match} is set to be 1.2km. Our CDDQN and Guider Networks utilize a Multi-layer Perceptron (MLP) with a six-layer configuration respectively.

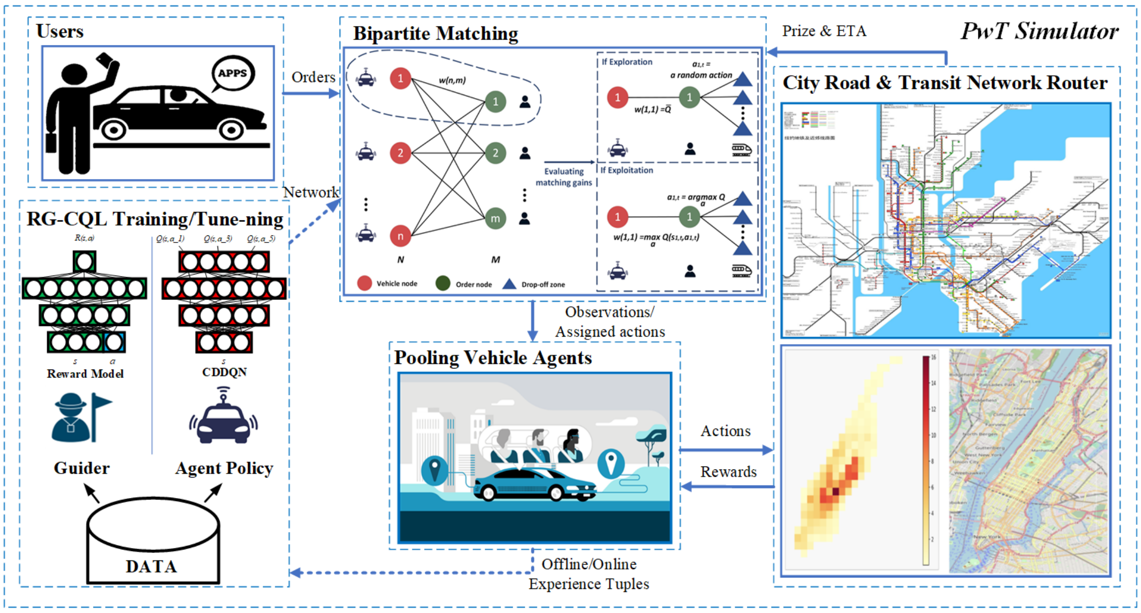


Figure 4: Simulator Design for Coordinating Ride-Pooling with Transit

We first compare our approach with four baseline methods that vary in ride-hailing service modes and RL methods. The aim is to demonstrate the superiority of the proposed coordinated pooling-transit services and RG-CQL framework in enhancing ride-hailing system performance. To distinguish between our comparison cases, we use "A_B" notation, where A represents the ride-hailing service mode ("PwT" for the proposed coordinated pooling-transit services, "P" for purely ride-pooling services, and "NPwT" for coordinated non-pooling and transit services) and B represents the RL method (RG-CQL, "Online RL", and "Greedy" method). The following outlines the comparison cases and discusses how our framework adapts to each baseline method for evaluation:

- **PwT_Online RL**: The service mode in this benchmark aligns with the description above. The MDP model and order matching model are consistent with those previously introduced. However, only the online RL algorithm DDQN in Van Hasselt et al. (2016) is executed to learn the value function and optimal policy.
- **PwT_Greedy in Gu & Liang (2024)**: We train a reward model for the environment through online iteration using a neural network whose architecture is identical to our Guider.

During exploitation, the agent selects actions that maximize the estimated reward. The platform optimizes total rewards in order matching without considering long-term gains.

- **P_Online RL in Al-Abbasi et al. (2019)**: Ride-pooling vehicles must deliver riders to their destinations, limiting an agent's action to door-to-door services. Online RL algorithm DDQN is executed to learn value function and optimal policy.
- **NPwT_Online RL in Feng et al. (2022)**: Pooling is prohibited so that a vehicle has to complete its current trip before becoming available for matching. Passengers can be dropped off at intermediate transit stations or their final destinations. Online RL algorithm DDQN is used for training and we also enhance the baseline by giving generous exploration budget.
- **PwT_RG-CQL**: We aggregate vehicle trajectories from the mixture of four policies above as synthetic data and adopt our proposed approach outlined in Algorithm 1 to train the policy.
- **PwT_Cal-q**: We train the CDDQN policy according to the proposed less conservative regularization in Nakamoto et al. (2024) bounded with reference policy value function in the offline stage, but do not utilize Guider to guide agent's exploration in the online fine-tuning.
- **PwT_Hybrid-q**: As proposed in Song et al. (2023), we train the DDQN policy directly in the online stage, but preload and keep offline data batch in the experience memory batch.

Fig. 5 shows the variations in the accumulative total reward with training progression, where the shaded area represents the range between the maximum and minimum values observed across the nearest 50 episodes. As shown in Fig. 5, the proposed RG-CQL demonstrates superior learning efficiency and effectiveness compared with all other baseline methods. The proposed PwT_RG-CQL method yields the highest total accumulative rewards among all comparison cases. Specifically, for the coordinated service mode, the reward under PwT_RG-CQL mode surpasses those under NPwT_Online RL and PwT_Online RL by 17% and 22%, respectively. In terms of RL framework, compared with PwT_Online RL, thanks to the novel offline training and online finetuninng pipeline, our PwT_RG-CQL not only accelerates the online training time by 81.3% but also improves overall performance by 4.3%. Moreover, we show that compared with state of the art RL algorithm for O2O like PwT_Cal-q and PwT_Hybrid-q, our RG-CQL manages to effectively address the O2O dilemma in the large-scale context of coordinating ride-pooling with public transit by having 6.7% and 16.2% improvement in total rewards respectively.

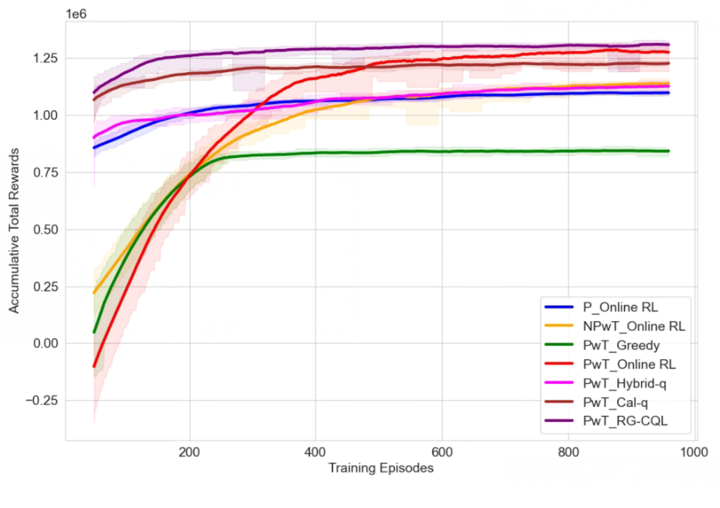


Figure 5: Training comparison of different Ride-hailing modes and RL methods

4 Conclusions

This paper introduces the Reward-Guided Conservative Q-learning (RG-CQL) framework to improve coordination between ride-pooling and public transit in multimodal transportation networks. We formulate the ride-pooling problem in multimodal transportation networks as a Markov Decision Process (MDP). To find the optimal policy, the RG-CQL involves offline learning from past data trajectories using a Conservative Double Deep Q Network (CDDQN) for action execution and a supervised learning-based Guider Network for reward estimation. In the online fine-tuning phase, the Guider Network aids CDDQN in exploring new state-action pairs, effectively balancing conservative offline training with optimistic online tactics. Our extensive numerical experiments validate that the proposed RG-CQL framework not only markedly demonstrates and enhances the operational performance of multi-modal transportation systems but also well handle the Offline to Online (O2O) challenges in RL for large-scale ride-pooling system.

Appendix: Overview of RG-CQL Algorithm

Algorithm 1 PwT_RG-CQL Framework

```
 1: Neural Network Initialization: CDDQN Training Net Parameter $\theta$, CDDQN Target
    Net Parameter $\theta^{-}$, Guider Net Parameters $\phi$.

 2: **Offline Training Stage:**
 3: Dataset Initialization: past data transitions $D$ and sample Size $M$
 4: Training hyper-parameter initialization: Conservative coefficient $C$, CDDQN update
    rate $\alpha_c$ and $\rho$, guider learning rate $\alpha_g$, number of training steps $T$
 5: **for** $t = 0$ to $T$ **do**
 6:     Sample $M$ experience tuples $(s, a, r, s')$ in $D$.
 7:     Use Equation (4) to calculate $L_c$ to update $\theta$ and $\theta^{-}$.
 8:     Use Equation (5) to calculate $L_g$ to update $\phi$
 9: **end for**

10: **Online Fine-tuning Stage:**
11: Initialization: Episode order requirements, vehicle router model, transit simulator,
    matching distance $R_{match}$, number of ride-pooling vehicles $N$.
12: Hyper-parameters Initialization: Conservative Term $C$ as much smaller value, CDDQN
    Fine-tune Rate $\alpha$ and $\rho$, Guider Network Fine-tune Rate $\alpha_g$, Online Phase Exploration
    Rate $\epsilon$, $\epsilon_T$ with Exponential Decay Rate $\beta$, Memory Capacity $D$ and Memory Sample
    Size $M$.
13: **for** $e = 1$ to Episodes **do**
14:     Perform Exponential Exploration Decay.
15:     **for** $t = 0$ to $t_{\text{terminal}}$ by $\Delta t$ **do**
16:         Central platform updates order information, each vehicle's location, and on-board
            passenger situations.
17:         Central platform assigns orders to vehicle agents according to ILP formulation
            in Equation (3) and (7) with the value estimation of the training network and
            guidance from Guider.
18:         Vehicles observe their orders and perform the assigned actions in the simulation
            platform and add every agent's new experience tuple $(s, a, r, s')$ into the memory.

19:         **if** memory size larger than $D$ **then**
20:             Sample $M$ experience tuples $(s, a, r, s')$ in as mini-batch
21:             Adopt Equation (4) to update $\theta$ and $\theta^{-}$.
22:             Use Equation (5) to calculate $L_g$ to update $\phi$ if needed.
23:         **end if**
24:         Based on the chosen action, central platform calculates the new route and esti-
            mated time of pickup, drop off, and transit.
25:     **end for**
26: **end for**
```

References

Al-Abbasi, A. O., Ghosh, A., & Aggarwal, V. (2019). Deeppool: Distributed model-free algorithm for ride-sharing using deep reinforcement learning. *IEEE Transactions on Intelligent Transportation Systems*, *20*(12), 4714–4727.

Authority, M. T. (2024). *Mta schedules, https://new.mta.info/schedules.* Retrieved from `https://new.mta.info/schedules`

Feng, S., Duan, P., Ke, J., & Yang, H. (2022). Coordinating ride-sourcing and public transport services with a reinforcement learning approach. *Transportation Research Part C: Emerging Technologies*, *138*, 103611.

Fujimoto, S., Hoof, H., & Meger, D. (2018). Addressing function approximation error in actor-critic methods. In *International conference on machine learning* (pp. 1587–1596).

Gao, J., & Li, S. (2024). Regulating for-hire autonomous vehicles for an equitable multimodal transportation network. *Transportation Research Part B: Methodological*, *183*, 102925.

Gu, Q.-P., & Liang, J. L. (2024). Algorithms and computational study on a transportation system integrating public transit and ridesharing of personal vehicles. *Computers & Operations Research*, *164*, 106529.

Haliem, M., Mani, G., Aggarwal, V., & Bhargava, B. (2021). A distributed model-free ride-sharing approach for joint matching, pricing, and dispatching using deep reinforcement learning. *IEEE Transactions on Intelligent Transportation Systems*, *22*(12), 7931–7942.

Kumar, A., Zhou, A., Tucker, G., & Levine, S. (2020). Conservative q-learning for offline reinforcement learning. *Advances in Neural Information Processing Systems*, *33*, 1179–1191.

Nakamoto, M., Zhai, S., Singh, A., Sobol Mark, M., Ma, Y., Finn, C., ... Levine, S. (2024). Cal-ql: Calibrated offline rl pre-training for efficient online fine-tuning. *Advances in Neural Information Processing Systems*, *36*.

Parrott, N. (2017). *Nyc subway travel time map, https://subway.nateparrott.com.* Retrieved from `https://subway.nateparrott.com`

Qin, X., Ke, J., & Yang, H. (2022). Government regulations for ride-sourcing services as substitute or complement to public transit. *Available at SSRN 4129034*.

Song, Y., Zhou, Y., Sekhari, A., Bagnell, D., Krishnamurthy, A., & Sun, W. (2023). Hybrid rl: Using both offline and online data can make rl efficient. In *International conference on learning representations.*

Sutton, R. S., Barto, A. G., et al. (1998). *Introduction to reinforcement learning* (Vol. 135). MIT press Cambridge.

Taxi, N., & Commission, L. (2024). *Nyc taxi and limousine commission-trip record data nyc, https://www1.nyc.gov/.* Retrieved from `https://www1.nyc.gov/`

Van Hasselt, H., Guez, A., & Silver, D. (2016). Deep reinforcement learning with double q-learning. In *Proceedings of the aaai conference on artificial intelligence* (Vol. 30).

Wang, D., Wang, Q., Yin, Y., & Cheng, T. (2023). Optimization of ride-sharing with passenger transfer via deep reinforcement learning. *Transportation Research Part E: Logistics and Transportation Review*, *172*, 103080.

Yu, X., & Gao, S. (2022). A batch reinforcement learning approach to vacant taxi routing. *Transportation Research Part C: Emerging Technologies*, *139*, 103640.