Optimizing Ride-Pooling Operations with Extended Pickup and Drop-Off Flexibility

Hao Jiang, Yixin Xu, Pradeep Varakantham

Singapore Management University
haojiang.2021@phdcs.smu.edu.sg, yixinxu@smu.edu.sg, pradeepv@smu.edu.sg

Abstract

The core of efficient on-demand ride-pooling lies in solving the Ride-Pool Matching Problem (RMP), which involves assigning multiple customer requests to single vehicles under various service constraints (e.g., pickup windows, detour allowances, and vehicle occupancy). A significant missed opportunity in most current RMP approaches is the assumption that passengers must be picked up and dropped off exactly at their requested locations. Allowing passengers to walk even short distances to meet vehicles could unlock substantial improvements in ride-pooling operations.

Building upon the limitations of existing Ride-Pool Matching Problem (RMP) solutions that neglect passenger walkability, this paper introduces a novel matching method that strategically incorporates flexible pickup and drop-off locations. Our approach simultaneously determines the optimal assignment of vehicles to requests (one vehicle to potentially multiple requests and each request to at most one vehicle), identifies advantageous meeting points for passengers, and plans efficient vehicle routes. This comprehensive optimization respects all service constraints and considers the long-term implications of routing decisions. To achieve this integrated solution, we first employ a tree-based approach to enumerate all feasible pairings between passengers and vehicles. Then, we calculate an optimal route for each of these feasible matches. Finally, we evaluate the quality of all possible assignments using reinforcement learning and select the most advantageous matching for implementation.

In our experimental evaluation on city-scale taxi datasets, we demonstrate that our method improves the number of served requests by up to 13% and reduces the average vehicle travel distance by up to 21%. By serving more passengers with less driving distance, our approach achieves greater efficiency in a more sustainable manner — using fewer resources to deliver better service and creating a win-win outcome for all stakeholders, including customers, drivers, the aggregator, and the environment.

Introduction

Convenient and flexible on-demand ride-pooling services (e.g., UberPool, LyftLine, GrabShare) have transformed urban travel. By allowing route-sharing, they enhance vehicle efficiency, lower environmental impact and congestion, decrease passenger expenses, and increase driver income, promoting a more sustainable urban transportation system. Optimally assigning vehicles to passenger requests (with each request assigned to at most one vehicle and each vehicle potentially serving multiple requests) under operational constraints (pickup delays, detour limits, capacity) is the core of the Ride-pool Matching Problem (RMP) (Alonso-Mora et al. 2017; Shah, Lowalekar, and Varakantham 2020) in ride-pooling. The goal is to maximize revenue or the number of served requests. Efficient and scalable RMP solutions are essential for managing the complexity of large-scale urban ride-pooling systems with numerous locations and varying vehicle sizes.

Existing RMP solutions assume pickup and drop-off occur precisely at the locations requested by customers (Alonso-Mora et al. 2017; Shah, Lowalekar, and Varakantham 2020; Hao and Varakantham 2022), thus offering no flexibility for passengers to move to nearby points. This assumption limits the potential for increased flexibility and efficiency in ride-pooling services. In this paper, we explore a more flexible scenario where passengers can walk to nearby pickup and drop-off points. This flexibility not only reduces waiting times but also improves match success rates and minimizes unnecessary detours, ultimately enhancing system performance. However, allowing flexible pickup and drop-off locations significantly increases the potential combinations and complicates the assignment. Existing work (Shah, Lowalekar, and Varakantham 2022) has shown that providing theoretical bounds is feasible for stylized versions of the ride-pool matching problem only and not for the real RMP. Our setting with flexible pickup and drop-off points is therefore also at least NP-Hard due to its additional flexibility and combinatorial complexity.

While recent research (Gao et al. 2024) has started to investigate flexible pickup and drop-off options in ride-pooling, these approaches mainly focus on short-term gains, neglecting the long-term consequences of matching decisions on future vehicle availability and overall system performance. Our approach, FlexiPool, overcomes these limitations by employing reinforcement learning to optimize ride-pooling matches with consideration for both immediate and future impacts, resulting in a substantial enhancement of the system's operational efficiency.

In summary, our work makes the following contributions:

- We incorporate flexible pickup and drop-off locations for RMP and propose an objective for optimally assigning passenger requests to vehicles in this flexible environment, considering both short-term and long-term effects.
- We implement an efficient algorithm to identify all feasible matches. Our approach comprehensively explores various assignment possibilities while maintaining computational efficiency.
- We compute the optimal pickup and drop-off locations for passengers, along with the optimal routes, by modeling the task as a Routing Planning Problem (RPP).
- To validate our approach, we conducted experiments on a large, real-world taxi dataset (NYYellowTaxi 2016). Results demonstrate that FlexiPool outperforms leading RMP solvers (Shah, Lowalekar, and Varakantham 2020; Hao and Varakantham 2022) by achieving up to a 13% improvement in the number of served requests and a 21% reduction in average travel distance. This improvement is significant for on-demand transportation services, where even a 0.5% increase in efficiency translates into notable gains in operational performance (Zhou et al. 2020).

Related Work

Earlier research on RMP has employed a range of optimization and planning methods, including branch-and-price algorithms, insertion heuristics, and column generation techniques (Ropke and Cordeau 2009; Ritzinger, Puchinger, and Hartl 2016; Parragh, Doerner, and Hartl 2008). These approaches have demonstrated effectiveness in small-scale applications, but they often encounter difficulties when scaling up to city-wide scenarios.

As ride-pooling problems expanded to larger settings, researchers started to focus on strategies for city-scale applications (Ma, Zheng, and Wolfson 2013; Huang et al. 2013; Tong et al. 2018; Lowalekar, Varakantham, and Jaillet 2019; Alonso-Mora et al. 2017). While these city-scale matching approaches can effectively handle large datasets, their focus on short-term optimization limits their ability to adapt to future matching needs, resulting in myopic performance over the long term.

To consider the long-term impacts of current assignments, reinforcement learning-based (RL-based) methods such as Neural Approximate Dynamic Programming (NeurADP) are proposed (Shah, Lowalekar, and Varakantham 2020; Hao and Varakantham 2022). Although existing RL-based methods can efficiently assign requests to vehicles in dynamic environments while considering long-term effects, they do not account for flexible pickup and drop-off points and overlook the route optimization of vehicles.

Recent research (Dessouky and Mahtab 2023; Dessouky et al. 2024) has demonstrated the advantages of incorporating flexible pickup and drop-off points in ride-pooling systems. Allowing passengers to walk a short distance to nearby meeting points can significantly reduce vehicle detours and improve travel efficiency. For instance, research indicates that flexible pickup and drop-off locations can lead to up to an 18% reduction in travel times, as well as a significant decrease in passenger waiting and in-vehicle time (Dessouky and Mahtab 2023). Furthermore, introducing such flexibility has been shown to improve the scalability of ride-pooling systems, making them better suited for large and dynamic urban settings (Dessouky et al. 2024). However, these methods overlook the long-term rewards of current assignments, whereas our approach considers this aspect using reinforcement learning in dynamic, multi-vehicle scenarios. A recent work utilizes Graph Convolution Network (GCN) to compute optimal routes with flexible pickup and drop-off points (Gao et al. 2024). Their work uses a supervised model and requires pre-training with real-world network data, which may limit its applicability in different settings. In contrast, our method is independent of specific datasets and can be applied across various contexts.

An important consideration in ride-pooling is planning vehicle routes to serve all passengers while adhering to their service constraints. With flexible pickup and drop-off points, a vehicle must visit a set of areas. Specifically, for each passenger, the vehicle must visit one point within its pickup area and one point within its drop-off area. The task of designing a vehicle's route in our setting can be formulated as the Routing Planning Problem (RPP). Modern methods have leveraged machine learning models, such as graph neural networks, to enhance the solution quality of the VRP (Lin et al. 2024; Dornemann 2023). However, these traditional VRP approaches are tailored for static settings, where target locations to be visited are predefined and remains fixed. They are not applicable to our problem, where the target locations are dynamically changing and subject to the locations of incoming requests. We develop a Routing Planning Problem (RPP) framework that supports flexible pickup and drop-off points while accounting for the specific service constraints of the ride-pooling problem (RMP). (Fielbaum, Bai, and Alonso-Mora 2021) propose a heuristic-based method that combines request matching and routing optimization. While this approach achieves computational efficiency, it remains myopic: each matching decision is evaluated using immediate operational cost without considering long-term effects. In contrast, FlexiPool introduces approximate dynamic programming to optimize long-term performance via learned value functions.

Problem Definition

In this section, we formally define the ride-pooling matching problem (RMP) and the extension of pickup/drop-off areas

Ride-pool Matching Problem (RMP)

Formally, traditional Ride-pool Matching Problem(RMP) can be formulated as the tuple: $\langle \mathcal{G}, \mathcal{R}, \mathcal{V}, \mathcal{D}, \Delta, \mathcal{O} \rangle$

$\mathcal{G}$: $\mathcal{G} = \langle I, E \rangle$ denotes the underlying graph of the road network. I are the set of intersections and E are the edges connecting intersections. Without loss of generality, pickups and drop-offs are assumed to occur at intersections.

$\mathcal{R}$: denotes the set of user requests. $\mathcal{R}_t$ is the set of all the requests collected in epoch t, where $\mathcal{R} = \cup_t \mathcal{R}_t$. Each request r_j in $\mathcal{R}$ is represented using the tuple $\langle p_j, e_j, t_j \rangle$,

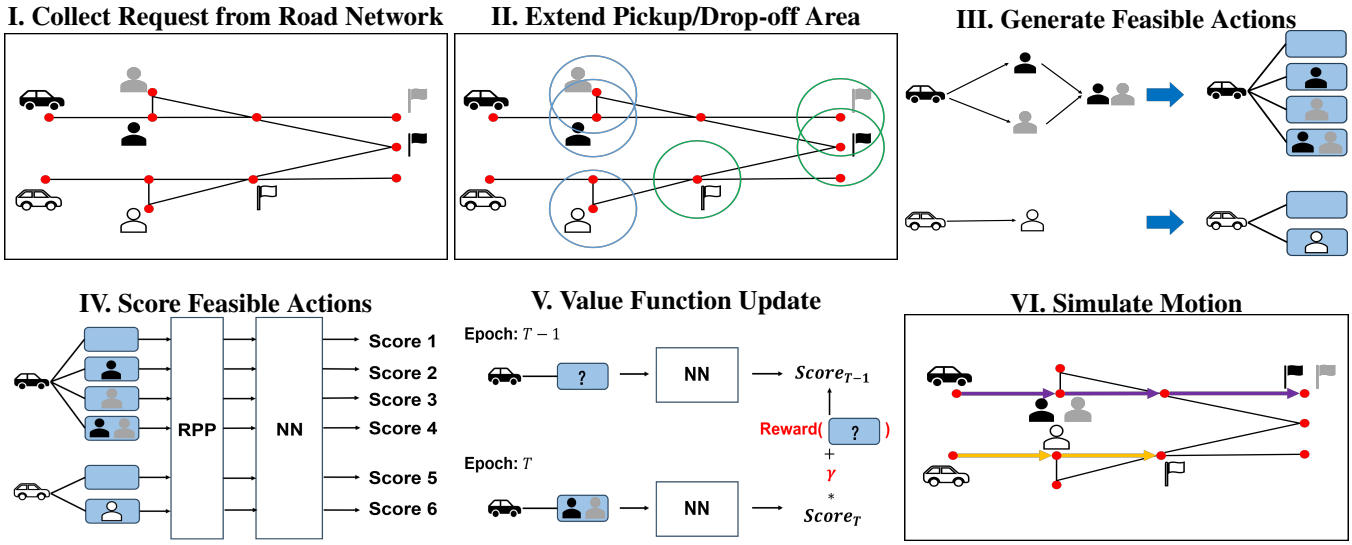

Figure 1: Overview of FlexiPool.

where p_j denotes the pickup location, e_j denotes the drop-off location, t_j denotes the arrival time of the request.

$\mathcal{V}$: denotes the set of vehicles. Each vehicle v_i in $\mathcal{V}$ is represented using the tuple $\langle c_i, l_i, \mathcal{L}_i \rangle$, where c_i denotes the capacity of the vehicle, l_i denotes the current location of the vehicle, $\mathcal{L}_i$ denotes the list of remaining locations that the vehicle must visit to complete its currently assigned requests. $\mathcal{L}_i$ is null if no request is assigned to vehicle v_i.

$\mathcal{D}$: every request has two delay constraints: 1) δ, the maximum allowed pickup delay of the request, and 2) λ, the maximum allowed detour delay of the request.

Δ : denotes the duration of time window (epoch) and refers to the time between each assignment.

$\mathcal{J}$: denotes the optimization objective. Suppose that vehicle i is assigned to a set of requests f during epoch t, we denote the corresponding objective value incurred by this assignment as $\mathcal{J}_t^{i,f}$. This objective can be either number of requests served or distance traveled by vehicles. The goal in RMP is to determine optimal matches between a set of requests and a set of vehicles, so as to optimize the cumulative objective $\sum_{t \in \mathcal{T}} \sum_{i \in \mathcal{V}} \mathcal{J}_t^{i,f}$.

Extension of Pickup/Drop-off Area

In real-world ride-pooling scenarios, passengers may have multiple potential pickup and drop-off points rather than fixed locations. Passengers can walk to nearby pickup points to meet their assigned vehicles and similarly walk from nearby drop-off points after their rides. Formally, for a request r with pickup point p and drop-off point e:

We first conduct a preliminary feasibility check to filter out unsuitable pickup points. Only locations within a reasonable walking distance from the passenger's current position can be considered feasible pickup points. Specifically, a point p' is a feasible pickup point for request r only if the walking time from p to p' does not exceed the passenger's maximum allowable pickup delay δ. For simplicity, we use a fixed walking threshold for all passengers. In practice, this could be dynamic, adapting to factors like time of day, weather, neighborhood, or passenger preferences.

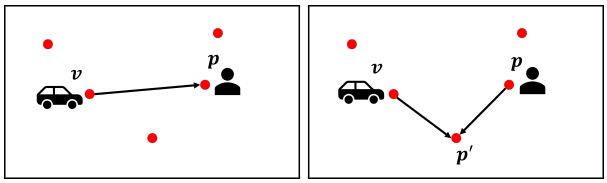

Figure 2: In the original setting (left), vehicle v must travel to the passenger's original pickup location p. With flexible pickup and drop-off (right), passengers can walk to an alternative pickup point p' that is more accessible for the vehicle.

Given a feasible pickup location p' for request r, a vehicle can be assigned to pick up the passenger at p' only if it can arrive p' on time. Specifically, the driving time from the vehicle's current location l^i to the selected pickup location p' must not exceed the passenger's maximum allowable pickup delay δ.

For example, as shown in figure 2, assume the passenger's maximum pickup delay is 5 mins. The passenger can only walk to locations within a 5-minute walking time. Similarly, a vehicle must arrive at the designated pickup location within 5 mins to ensure it can pick up the passenger in time.

We convert time constraints into distance constraints by using the passenger's walking speed: a passenger has a maximum allowed walking distance $d_r = \delta * walking\_speed$. A location p' is a feasible pickup point for request r only if the distance between p' and the original pickup location p does not exceed r's maximum allowed walking distance.

We define the pickup area P of request r as the set of all

feasible pickup locations of r, formally expressed as:

$$d(p, p_i') \leq d_r, \forall p_i' \in P$$

where $d(p, p_i')$ denotes the distance between the original pickup location p and feasible location p_i' in the pickup area P. If walking is not allowed, the pickup area reduces to p.

The maximum allowed walking distance d_r is also used to define feasible drop-off locations of r and its drop-off area. A point is considered a feasible drop-off point for r if and only if it lies within a walking distance of d_r from $r's$ original drop-off location.

The goal of RMP with extended pickup/drop-off area is to determine the optimal pickup and drop-off locations for all requests, as well as the optimal assignment of requests to vehicles, in order to minimize the cumulative objective function.

FlexiPool

Overview of FlexiPool

Figure 1 presents the overall flow of FlexiPool:

- Step I: collect the information of all requests and vehicles. For instance, there are two available vehicles (black and white) and three passengers (black, gray and white).
- Step II: compute the pickup and drop-off areas.
- Step III: generate all the feasible assignments. Each feasible assignment assigns a specific vehicle to serve one or multiple requests. For example, the black taxi can either pick up the black passenger or not pick up anyone, while the white taxi can pick up either passenger or choose not to pick up anyone.
- Step IV: compute an optimal route for each feasible assignment and determine the optimal pickup and drop-off locations for each passenger along the route by modelling the problem as a RPP problem.
- Step V: evaluate the quality of each assignment using reinforcement learning. We compute a score for each assignment and use these scores to figure out the optimal policy and update the parameters. Figure 1 shows the update process for the black taxi.
- Step VI: the agents execute the optimal assignments derived from Step V by moving to the designated locations.

In Figure 3, we provide an example to illustrate the advantage of flexible pickup and drop off. Since Steps I and II only involve state collection and area computation defined earlier, we begin the detailed exposition from Step III.

Step III: Feasible Combination Generation

Next, we present an efficient method to generate all feasible assignments. The goal is to find all feasible assignments that match a set of requests $R = \{r_1, r_2, ..., r_n\}$ to a set of vehicles $V = \{v_1, v_2, ..., v_m\}$ while adhering to all constraints. In this stage, an assignment is valid if at least one feasible pickup and drop-off pair exists. The optimal pickup and drop-off locations is selected later during routing optimization.

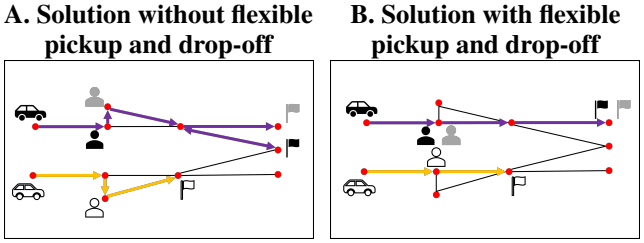

Figure 3: Impact of Flexibility on Solution Quality.

By generating all feasible assignments, we can ensure that the matching process accounts for all possible options and selects the most promising ones. We propose an optimized assignment algorithm that aims to minimize unnecessary expansions in the search space, thereby reducing computational complexity. The detailed algorithm is provided in the appendix.

We observe that any larger combination containing an infeasible assignment will also be infeasible. For example, if $[r_1, r_2]$ is an infeasible assignment for v_1, then any combination that contains $[r_1, r_2]$, such as $[r_1, r_2, r_3]$ or $[r_1, r_2, r_4, r_5]$, will also be infeasible for v_1. Therefore, once a combination is identified as infeasible, it can be eliminated from further consideration, making the search for feasible assignment more efficient.

Based on this observation, we propose to generate combinations incrementally using a tree structure. We start by considering single requests (First Level), then we move on to combinations of two requests (Second Level), followed by combinations of three requests (Third Level), and so forth. If a combination violates any constraint—such as exceeding the vehicle's capacity or violating the pickup/detour delay—it is pruned and stored in memory to avoid considering it in further combinations at later levels. By assessing the feasibility of smaller sets before progressing to larger combinations, our method efficiently eliminates infeasible options early on, thus significantly reducing the search space and improving overall efficiency.

Figure 4 illustrates the process of generating the tree using an example. Considering the matching of a vehicle v_i to four requests r_1, r_2, r_3, and r_4, the tree begins from a root node v_i, indicating the vehicle is not assigned to any requests. The first level of the tree examines the assignment of a single request. Any infeasible request will be pruned and stored in the memory, so that it cannot be further expanded and be considered for further combinations. For example, if r_3 is infeasible for v_i, then r_3 will be pruned and can not be considered for further combinations. As a result, the first level will generate feasible assignments:$[r_1], [r_2], [r_4]$.

The second level of the tree examines combinations of two requests by pairing only the remaining feasible requests, specifically $[r_1], [r_2]$, and $[r_4]$. Initially, this level generates all combinations of two: $[r_1, r_2]$, $[r_1, r_4]$, and $[r_2, r_4]$. If any combination is deemed infeasible, such a combination will be pruned and not considered in subsequent levels. For example, if $[r_1, r_2]$ is found to be infeasible, it will be pruned and cannot be included for further combinations. After prun-

ing out $[r_1, r_2]$, the second level will generate feasible assignments $[r_1, r_4]$ and $[r_2, r_4]$.

The third level of the tree examines combinations of three requests. Based on the results of the previous level, there is only one combination of three: $[r_1, r_2, r_4]$. However, this combination includes $[r_1, r_2]$, which has already been identified as infeasible and stored in memory. As a result, $[r_1, r_2, r_4]$ is added into memory and the generation process stops here. The final feasible assignment for v_i includes the combinations generated from all levels: $\{\emptyset, [r_1], [r_2], [r_4], [r_1, r_4], [r_2, r_4]\}$. r_3 cannot be assigned to v_i, but it can be assigned to other vehicles and included in their respective matches.

Our pruning strategy keeps the search space manageable, given the limited capacity of each vehicle (≤ 4). Once all feasible assignments are generated, we then optimize the routes to complete the assignments in the next step.

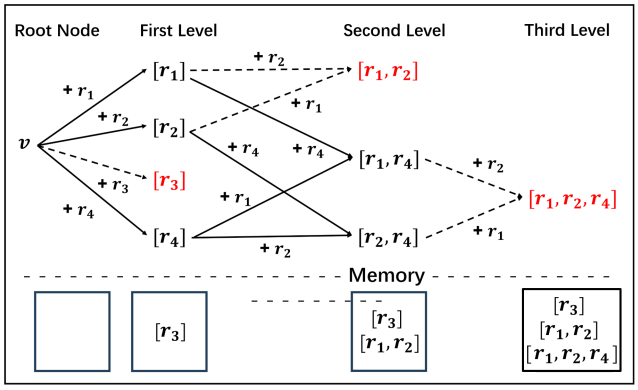

Figure 4: Example of Combination Generation. Black solid lines represent feasible combinations, while black dashed lines represent infeasible combinations.

Step IV: Route Optimization

After identifying all feasible matches, the next step is to plan an optimal route for each feasible match. A feasible match may assign a vehicle to multiple requests. For each request, it must first visit the pickup area and then the drop-off area. The optimal route must visit both the pickup and drop-off locations for all assigned requests, satisfy the service constraints associated with each request (such as time windows or delays), and minimize the total travel time. A passenger's pickup delay is calculated as the maximum of the passenger's walking time and the vehicle's arrival time.

We define this problem as Routing Planning Problem (RPP). The input to RPP is a feasible assignment (including the current location of vehicle, the pickup and drop-off areas for all assigned requests and service constraints). The output of the RPP is an optimal route that satisfies all service constraints.

We formulate this problem as a Mixed-Integer Linear Program (MILP) with constraints and provide the formulas in appendix. MILP produces an optimal vehicle route, which is represented as an ordered sequence of locations. It ensures at least one point within each assigned pickup and drop-off area is visited within its specified time window. If two pickup or drop-off areas overlap, MILP may choose a single location within the shared region and include it twice in the route. In this case, the same physical location serves separately as both the pickup and the drop-off points.

Step V: Future aware Matching

In the next step, we assess the quality of all feasible assignments considering their long-term effects. The best assignments will then be chosen and executed to optimize long-term value.

In this part, we apply the Neural Approximate Dynamic Programming algorithm (Shah, Lowalekar, and Varakantham 2020), the leading approach for solving RMP. The Approximate Dynamic Programming (ADP) is commonly applied in large-scale vehicle routing and resource allocation problems. In the context of Ride-Pool Matching (RMP), ADP provides an effective method for managing dynamic vehicle assignments by predicting future rewards associated with each assignment. This prediction is based on an approximation of the system's state and the decisions made by the agents.

Formally, The ADP problem for RMP is formulated using the tuple $\langle S, A, \xi, T, \mathcal{J} \rangle$, where :

S : denotes current state of the system. At decision epoch t, the state $S_t \in S$ includes the state of all vehicles v_t and the set of available requests r_t. The status of vehicles and requests are detailed in Section 3.1. The state is collected in Step I of Figure 1.

A : denotes the set of actions, where each action corresponds to assigning a set of requests to a vehicle. The generation of feasible combinations for each vehicle i at time t, denoted as $\mathcal{F}_t^i$, is computed in Step III of Figure 1.

ξ : denotes the exogenous information, which is the source of randomness in the system.

T : denotes the transition function. In an ADP, the system evolution happens as

$$(s_0, a_0, s_0^a, \xi_1, s_1, a_1, s_1^a, \cdots, s_t, a_t, s_t^a, \cdots),$$

where s_t denotes the pre-decision state at decision epoch t and s_t^a denotes the post-decision state (Powell 2007). The transition from state s_t to s_{t+1} depends on the action vector a_t and the exogenous information ξ_{t+1}. Therefore,

$$s_{t+1} = T(s_t, a_t, \xi_{t+1})$$
$$s_t^a = T^a(s_t, a_t); s_{t+1} = T^\xi(s_t^a, \xi_{t+1})$$

$\mathcal{J}$: denotes the reward function.

At each decision epoch, the goal of ADP is to find the optimal action a_t by solving the following Bellman equation:

$$V(s_t) = \max_{a_t \in A_t} (\mathcal{J}(s_t, a_t) + \gamma \mathbb{E}[V(s_{t+1}) | s_t, a_t, \xi_{t+1}]) \quad (1)$$

where $V(s_t)$ denotes the value function at state s_t, γ is the discount factor.

Using post-decision state, the Bellman equation can be reformulated into two steps:

$$V(s_t) = \max_{a_t \in A_t} (\mathcal{J}(s_t, a_t) + \gamma V^a(s_t^a)) \quad (2)$$

$$V^a(s_t^a) = \mathbb{E}[V(s_{t+1}) | s_t^a, \xi_{t+1}] \quad (3)$$

To efficiently handle the complexity of post-decision states, we apply value decomposition. The joint value function $V^a(s_t^a)$ is broken down into individual vehicle values:

$$V^a(s_t^a) = \sum_i V^{i,a}(s_t^{i,a})$$

where individual value function $V^{i,a}(.)$ is modeled as a neural network. This network is updated over time, using exogenous information and the best action determined by the ILP solution.

To incorporate the value decomposition, the following constraints for joint actions, $a_t \in \mathcal{A}_t$ is introduced:

1. Each vehicle i can only be assigned at most one request combination f.
2. At most one vehicle i can be assigned to a request j.
3. A vehicle i can either be assigned to a request combination or not.

To mathematically formalize these constraints, let $z_t^{i,f}$ denote the decision variable that indicates whether vehicle i takes action f (a combination of requests) at decision epoch t. The following constraints are then applied to ensure that the above conditions are met:

$$\sum_{f \in \mathcal{F}_t^i} z_t^{i,f} = 1 ::: \forall i \in \mathcal{V} \tag{4}$$

$$\sum_{i \in \mathcal{V}} \sum_{f \in \mathcal{F}_t^i; j \in f} z_t^{i,f} \leq 1 ::: \forall j \in \mathcal{R}_t \tag{5}$$

$$z_t^{i,f} \in \{0,1\} ::: \forall i, f \tag{6}$$

Neural Approximate Dynamic Programming (NeurADP) provides a good estimate of the joint value function. HIVES (Hao and Varakantham 2022) further improves the accuracy by introducing a hierarchical mixing neural network. This network clusters agents and combines their individual value functions more effectively. We use the value function structure of HIVES in our reinforcement learning framework.

Experiment

In the experiments, we aim to demonstrate the effectiveness of ride-pooling with flexible pickup and drop-off points. We will also conduct ablation study to evaluate various configurations—extending only the pickup locations, extending only the drop-off locations, and extending both—to assess their impact on overall improvement.

Setup

The experiment is performed with the New York Yellow Taxi Dataset (NYYellowTaxi 2016). Following the settings of similar works (Shah, Lowalekar, and Varakantham 2020), we exclude locations without outgoing edges and focus on areas with the highest frequency of requests, resulting in a network consisted of 4373 locations and 9540 edges. Travel times between locations are included in the dataset. We assume that all vehicles have the same capacity and are initialized at random locations. The maximum detour delay is set to $\lambda = 2\delta$, and the time window for assignments was 60 seconds. We examine the effect of four parameters:

- Capacity: varies from 2 to 4, with a default value of 4.
- Total number of vehicles: varies from 1000 to 2000, with a default value of 1000.
- Maximum pickup delay: varies from 300 seconds to 420 seconds, with a default value of 300 seconds.
- Maximum allowed walking distance: varies from 0.3 km to 0.5 km.

Note that the maximum allowed walking distances d_r are related to the value of δ. Specifically, d_r values of 0.3 km, 0.4 km, and 0.5 km correspond to δ values of 300s, 360s, and 420s, respectively. Our experiments reveal that the average number of neighbors for each node is 10, 14, and 18, corresponding to d_r values of 0.3 km, 0.4 km, and 0.5 km, respectively. Note that the pickup and drop-off areas for some requests may overlap at the default speed. In such cases, we adjust the size of the areas to eliminate any overlap.

We compare our method, FlexiPool, with state-of-the-art methods for solving RMP: NeurADP and HIVES. FlexiPool allows flexible pickup and drop-off options, whereas NeurADP (Shah, Lowalekar, and Varakantham 2020) and HIVES (Hao and Varakantham 2022) assume fixed pickup and drop-off points.

Experimental Results

We first compare the number of served requests of FlexiPool with those of NeurADP and HIVES . As original NeurADP and HIVES do not consider flexibility, we assume pickup and drop-off could be done in the nearest locations to the original pickup and drop-off points within the maximum allowed travel distance. We have averaged the results over three runs for each method. Table 1 provides the overall results with different settings of capacity, vehicle number and pickup delay. Here are the key observations regarding the improvement in the number of served requests:

1. FlexiPool consistently outperforms both NeurADP and HIVES across all settings. Our average improvement over NeurADP and HIVES is 13% and 6% respectively, which is considered a significant improvement in city-scale taxi ride-pooling problems (Xu et al. 2018).
2. When the capacity increases from 2 to 4, the percentage of improvement over NeurADP decreased from 16.75% to 13.36%. However, the improvement over HIVES increased from 5.42% to 6.39% over HIVES. This is because larger vehicle capacity provides more available resources, reducing the advantage over NeurADP, while in HIVES, the increase in capacity weakens the effect of clustering and neighboring, allowing FlexiPool to benefit more from the increased flexibility.
3. When the number of vehicles increases from 1000 to 2000, the improvement percentage over NeurADP decreased, while the improvement over HIVES increased. This is for the same reason as in the second observation: more available resources reduce the advantage over NeurADP but weaken the effect of clustering and neighboring in HIVES, allowing FlexiPool to better utilize the additional vehicles.

	Variants	NeurADP	HIVES	FlexiPool	% over NeurADP	% over HIVES
Capacity	2	165063	183093	190842	15.62	4.23
	4	243253	260553	271882	11.77	4.35
Vehicles	1000	243253	260553	271882	11.77	4.35
	1500	302602	310531	324446	7.22	4.48
Pickup Delay	300 (d_r=0.3 km)	243253	260553	271882	11.77	4.35
	360 (d_r=0.4 km)	241397	259121	275194	13.99	6.21
	420 (d_r=0.5 km)	241543	261850	279665	15.77	6.81

Table 1: Number of Served Requests and Improvement over baselines.

	Variants	NeurADP	HIVES	FlexiPool	% over NeurADP	% over HIVES
Capacity	2	345.34	311.84	285.49	17.33	8.45
	4	501.71	470.84	405.44	19.19	13.89
Vehicles	1000	501.71	470.84	405.44	19.19	13.89
	1500	415.93	405.92	352.58	15.23	13.14
Pickup Delay	300 (d_r=0.3 km)	501.71	470.84	405.44	19.19	13.89
	360 (d_r=0.4 km)	502.43	468.69	402.78	19.83	14.07
	420 (d_r=0.5 km)	503.64	465.26	399.52	20.69	14.10

Table 2: Average Travel Distance Reduction over baselines.

Method	Served Requests
HIVES	255562
FlexiPool without Flexibility	256498
FlexiPool without Route Planning	269365
FlexiPool	271882

Table 3: Ablation Study

4. When the pickup delay increases from 300 to 420 seconds, the improvement over NeurADP increased from 13.36% to 16.16% , and the improvement over HIVES increased from 6.39% to 7.31%. That is because the maximum allowed travel distance is directly related to the pickup delay. A larger pickup delay increases the extended pickup and drop-off areas, allowing for more feasible options and improving the overall assignment.

Sustainability - Reduction in Distance Traveled

We evaluate average travel distance with the same experiment setup in Table 2. Despite serving more requests, FlexiPool achieves a lower average travel distance than both baselines due to its flexibility and route optimization. Shorter travel distance improves vehicle efficiency, lowers fuel consumption, and enhances user experience.

Ablation Study

In this section, we evaluate the individual contribution of the two key components of FlexiPool: pickup and drop-off flexibility and route planning.

Table 3 presents the ablation study evaluating the contribution of different components within our framework. 'FlexiPool without Flexibility' applies route optimization only, 'FlexiPool without Route Planning' applies flexibility only, and removing both yields HIVES. When all components of FlexiPool are removed, the framework reduces to HIVES. The results indicate that incorporating flexibility alone yields greater improvement compared to route planning alone. However, optimizing routes together with flexibility achieves the best overall performance.

Conclusion

In this paper, we addressed a Ride-Pool Matching Problem (RMP) that incorporates flexible pickup and drop-off locations for passengers. We leveraged passengers' mobility by allowing them to walk to nearby locations to meet vehicles, while considering the long-term impacts of assignments and vehicle routes to achieve optimal matching outcomes.

Our method first employed a tree-based approach to efficiently generate feasible combinations of requests for each vehicle. By incrementally creating combinations from smaller sizes to larger ones, infeasible matches are pruned at early stages to enhance efficiency. Then, for each possible assignment, we applied route planning to determine the optimal routes for picking up and dropping off the assigned passengers. Finally, we utilized a reinforcement learning-based method to evaluate the matches and identify the optimal matches considering the long-term effects.

Experiments on real-world city-scale datasets demonstrated that our approach outperforms existing leading methods by achieving up to a 13% improvement in the total number of served requests and 21% improvement in the average total distance. This result highlights the potential of incorporating passenger mobility and dynamic routing to enhance ride-pooling services in high-demand urban environments.

In the future, we plan to develop a pricing model for ride-pooling with flexible pickup and drop-off points. By offering discounts to passengers who walk to nearby pickup or drop-off locations, the pricing model has the potential to incentivize passengers to adjust their locations and lead to more optimized outcomes.

References

Alonso-Mora, J.; Samaranayake, S.; Wallar, A.; Frazzoli, E.; and Rus, D. 2017. On-demand high-capacity ride-sharing via dynamic trip-vehicle assignment. *Proceedings of the National Academy of Sciences*, 114(3): 462–467.

Dessouky, M.; and Mahtab, Z. 2023. The Ridesharing Routing Problem with Flexible Pickup and Drop-off Points. Research report, University of Southern California, National Center for Sustainable Transportation.

Dessouky, M.; Mahtab, Z.; Center, M. T.; et al. 2024. Stochastic Rideshare System with Flexible Pickup and Drop-Off Points. Technical report, National Center for Sustainable Transportation (NCST)(UTC).

Dornemann, J. 2023. Solving the capacitated vehicle routing problem with time windows via graph convolutional network assisted tree search and quantum-inspired computing. *Frontiers in applied mathematics and statistics*, 9: 1155356.

Fielbaum, A.; Bai, X.; and Alonso-Mora, J. 2021. On-demand ridesharing with optimized pick-up and drop-off walking locations. *Transportation research part C: emerging technologies*, 126: 103061.

Gao, Y.; Ma, L.; Yu, Z.; Zhang, S.; Fang, J.; Gao, X.; and Chen, G. 2024. Lightweight GCN Encoder and Sequential Decoder for Multi-Candidate Carpooling Route Planning in Road Network. In *Companion Proceedings of the ACM on Web Conference 2024*, 302–310.

Hao, J.; and Varakantham, P. 2022. Hierarchical value decomposition for effective on-demand ride-pooling. In *Proceedings of the 21st International Conference on Autonomous Agents and Multiagent Systems*, 580–587.

Huang, Y.; Jin, R.; Bastani, F.; and Wang, X. S. 2013. Large scale real-time ridesharing with service guarantee on road networks. *arXiv preprint arXiv:1302.6666*.

Lin, Z.; Wu, Y.; Zhou, B.; Cao, Z.; Song, W.; Zhang, Y.; and Jayavelu, S. 2024. Cross-problem learning for solving vehicle routing problems. *arXiv preprint arXiv:2404.11677*.

Lowalekar, M.; Varakantham, P.; and Jaillet, P. 2019. ZAC: A zone path construction approach for effective real-time ridesharing. In *Proceedings of the International Conference on Automated Planning and Scheduling*, volume 29, 528–538.

Ma, S.; Zheng, Y.; and Wolfson, O. 2013. T-share: A large-scale dynamic taxi ridesharing service. In *2013 IEEE 29th International Conference on Data Engineering (ICDE)*, 410–421. IEEE.

NYYellowTaxi. 2016. New york yellow taxi dataset.

Parragh, S. N.; Doerner, K. F.; and Hartl, R. F. 2008. A survey on pickup and delivery problems. *Journal für Betriebswirtschaft*, 58(2): 81–117.

Powell, W. B. 2007. *Approximate Dynamic Programming: Solving the curses of dimensionality*, volume 703. John Wiley & Sons.

Ritzinger, U.; Puchinger, J.; and Hartl, R. F. 2016. A survey on dynamic and stochastic vehicle routing problems. *International Journal of Production Research*, 54(1): 215–231.

Ropke, S.; and Cordeau, J.-F. 2009. Branch and cut and price for the pickup and delivery problem with time windows. *Transportation Science*, 43(3): 267–286.

Shah, S.; Lowalekar, M.; and Varakantham, P. 2020. Neural approximate dynamic programming for on-demand ride-pooling. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, 507–515.

Shah, S.; Lowalekar, M.; and Varakantham, P. 2022. Joint Pricing and Matching for City-Scale Ride-Pooling. In *Proceedings of the International Conference on Automated Planning and Scheduling*, volume 32, 499–507.

Tong, Y.; Zeng, Y.; Zhou, Z.; Chen, L.; Ye, J.; and Xu, K. 2018. A unified approach to route planning for shared mobility. *Proceedings of the VLDB Endowment*, 11(11): 1633.

Xu, Z.; Li, Z.; Guan, Q.; Zhang, D.; Li, Q.; Nan, J.; Liu, C.; Bian, W.; and Ye, J. 2018. Large-scale order dispatch in on-demand ride-hailing platforms: A learning and planning approach. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, 905–913.

Zhou, T.; Zhang, F.; Tang, P.; and Wang, C. 2020. Multi-Agent Reinforcement Learning with Graph Clustering. *arXiv preprint arXiv:2008.08808*.