Future Aware Pricing and Matching for Sustainable On-Demand Ride Pooling

Xianjie Zhang[1,2], **Pradeep Varakantham**[2], **Hao Jiang**[2]

[1]Key Laboratory for Ubiquitous Network and Service Software of Liaoning Province, School of Software, Dalian University of Technology, China

[2]Singapore Management University, Singapore

zhangxianjie@mail.dlut.edu.cn, pradeepv@smu.edu.sg, haojiang.2021@phdcs.smu.edu.sg

Abstract

The popularity of on-demand ride pooling is owing to the benefits offered to customers (lower prices), taxi drivers (higher revenue), environment (lower carbon footprint due to fewer vehicles) and aggregation companies like Uber (higher revenue). To achieve these benefits, two key interlinked challenges have to be solved effectively: (a) pricing – setting prices to customer requests for taxis; and (b) matching – assignment of customers (that accepted the prices) to taxis/-cars. Traditionally, both these challenges have been studied individually and using myopic approaches (considering only current requests), without considering the impact of current matching on addressing future requests. In this paper, we develop a novel framework that handles the pricing and matching problems together, while also considering the future impact of the pricing and matching decisions. In our experimental results on a real-world taxi dataset, we demonstrate that our framework can significantly improve revenue (up to 17% and on average 6.4%) in a sustainable manner by reducing the number of vehicles (up to 14% and on average 10.6%) required to obtain a given fixed revenue and the overall distance travelled by vehicles (up to 11.1% and on average 3.7%). That is to say, we are able to provide an ideal win-win scenario for all stakeholders (customers, drivers, aggregator, environment) involved by obtaining higher revenue for customers, drivers, aggregator (ride pooling company) while being good for the environment (due to fewer number of vehicles on the road and lesser fuel consumed).

Introduction

The emergence of Uber, LYFT, DiDi and other taxi aggregation companies have improved the utilization rate of taxis and reduced customers' waiting time. One of the most popular services offered by taxi aggregation companies is the on-demand ride pooling service. Apart from being one of the most economical transport methods that allows multiple passengers to share a taxi, on-demand ride pooling also provides higher revenue for drivers and taxi aggregation companies, while having fewer taxis on the roads (lower carbon footprint). Effectiveness and efficiency of ride pooling service is determined by the methods used for pricing of customer requests (referred to as the pricing problem) and matching of taxis to groups of customer requests (referred to as the matching problem).

There are multiple difficult challenges in solving the matching and pricing problems for ride-pooling systems. First, finding taxis to serve requests (matching) determines the price to be set for requests (pricing) and price set for a request (pricing) determines which requests still remain (as customers can drop the request if the price is too high) for the taxis to serve. In summary, there is a cyclic dependency between pricing and matching, where pricing impacts matching and matching impacts pricing. Second, both pricing and matching decisions can have an impact on future requests and recent work (Shah, Lowalekar, and Varakantham 2020) has shown that developing future aware approaches can have a significant impact. Finally, unlike in on-demand ride sharing, on-demand ride pooling requires matching vehicles to trips (combinations of requests), which is no longer a bi-partite matching problem. It is matching on a tripartite graph (Alonso-Mora et al. 2017) between requests, trips (combinations of requests) and vehicles, which is a significantly harder problem.

Due to the cyclic dependency between matching and pricing, recent work has acknowledged the importance of simultaneously optimizing pricing and matching (Shah, Lowalekar, and Varakantham 2022; Özkan 2020) in improving the overall effectiveness and efficiency of on-demand ride sharing systems. However, most existing works have focused on optimizing either pricing or matching individually (Banerjee, Johari, and Riquelme 2015; Bimpikis, Candogan, and Saban 2019; Lowalekar, Varakantham, and Jaillet 2018; Shah, Lowalekar, and Varakantham 2020, 2022) and usually in a myopic manner (without considering the impact on serving future requests), but recently there has been emphasis on future aware methods (Shah, Lowalekar, and Varakantham 2020; Lowalekar, Varakantham, and Jaillet 2019) that have shown to provide significant benefits. There have been few recent studies on joint optimal pricing and matching decisions, however, they are either not scalable (Chen et al. 2019b) or rely on unrealistic assumptions about knowing all (current and future) customer requests in advance (Chen et al. 2019a). Moreover, the above works (Chen et al. 2019b,a) are not for the case of ride pooling.

To that end, in this paper, we make multiple contribu-

tions to simultaneously optimize matching and pricing decisions, while considering the future impact of those decisions for sustainable city scale on-demand ride pooling systems. Specifically, we make the following contributions:

- We provide a precise and formal definition of the overall matching and pricing problem in the context of ride pooling systems.
- We provide a two-layered reinforcement learning approach, that combines mean field Q-learning for pricing and neural approximate dynamic programming for matching in a synergistic manner to optimize both pricing and matching. We employ different RL approaches for matching and pricing due to the special structure present in the two problems.
- Social impact is typically feasible if there is a win-win situation for all stakeholders involved. In our case, the different stakeholders are customers, drivers, aggregator (Uber etc.) and environment. While customers, drivers and aggregator are more focussed on the revenue and availability of taxis, environment will win if fewer vehicles are on the road and amount of fuel consumed is lower (distance travelled is lower). We provide a detailed experimental section to evaluate our approach on a benchmark simulator that is based on a city scale real-world taxi dataset. Our approach is able to improve up to 17% on revenue across a wide range of parameter values and reduce the number of vehicles by up to 14%. As a benchmark, typically 0.5-1% improvements are considered significant in mobility settings (Xu et al. 2018).

Pricing and Matching Problem

In this problem, we have a set of vehicles V, each of which can potentially serve multiple customers (maximum of c) simultaneously (e.g., UberPool or GrabShare). The operator receives a set of requests for rides from customers. Typically, these requests are batched together over a fixed interval (e.g., 60 seconds) and we will refer to this set as R. The aggregation company then searches the set of available vehicles, V that satisfy quality constraints with respect to wait time and time to destination for the customer requests. Once at least one available vehicle is identified for a request, a price for the customer request is set by the aggregation company, which the customers can decide to accept or reject. There is a trade-off between pricing too high and losing customers vs pricing too low and losing revenue. Our goal is to design a pricing strategy that can ensure that the induced matching maximizes the expected revenue. We now formally define the 'pricing' and 'matching' problems.

Pricing Problem

The goal of the pricing problem is to compute a pricing vector, μ, which contains a price μ_r for each customer request, $r \in R$. Formally, the problem of pricing customer requests is formulated as two-level optimization problem:

$$\mu^* = \arg\max_{\mu} \Big[\max_{X \in C(R)} \Big(\sum_{x_v^f \in X} x_v^f \cdot o_v^f(\mu^f) \Big] \Big) \Big] \quad (1)$$

Symbol	Definition
V	Available vehicles in the current time period
R	Batched requests in the current time period.
r	Variable to denote a request
v	Variable to denote a vehicle
f	Variable to denote a combination of requests. R^f corresponds to requests in f
x_v^f	Binary decision variable that is set to 1 if v is assigned to f
μ	Pricing vector for all requests at a time step
$o_v^f(\mu^f)$	Expected reward obtained by assigning request combination f to v given price vector μ^f
$p(.)$	Price-sensitivity function maps the set of prices μ to the probability of acceptance $p(\mu)$. It is assumed to be known apriori. For a given request $r \in R$, $p_r(\mu_r)$ is the probability of acceptance of the customer at the price μ_r.
$C(.)$	Matching constraints required for feasible matching
a_v	Multiplier on base price for serving a request
τ	Pick up delay constraint
λ	Detour delay constraint

Table 1: Symbols and their definition

While the outer maximization is for computing the best pricing vector for all requests, R, the inner maximization is for finding the matching strategy that will maximize the expected revenue (immediate or long term) given the pricing strategy. $o_v^f(\mu^f)$ is the expected immediate (or long term) reward given the pricing vector, μ^f for requests in the request combination, f. There are multiple key challenges in solving this two-level optimization while being future aware (i.e., considering potential future requests):

- There are exponentially many pricing strategies and computing the outer maximum is expensive.
- From Equation 1, pricing and matching are dependent on each other. Pricing affects the active requests (i.e., ones for which customers accept the price) and hence affects the matching process. The matching determines which vehicle is assigned to a request and correspondingly the price associated with the request.
- The matching and pricing at a time step have an impact on the available vehicles (and potentially requests) at the next time step.

Matching Problem

The goal of matching is to create an assignment of vehicles to requests such that the objective o is maximized subject to constraints on matchings. This problem can be seen as a 'maximum weight bipartite matching' in the un-shared case but, bipartite matching doesn't capture the structure of the matching problem when multiple requests can be matched to a single vehicle. For example, consider two requests A and B that can be served by Vehicle 1 individually. However, the vehicle cannot serve them together as it would lead to an unacceptable amount of detours; these kinds of combinatorial constraints cannot be captured in a bipartite graph.

The way to solve this problem is to create an intermediate representation called a 'trip' which is a combination of requests (Alonso-Mora et al. 2017) represented using f. Then, requests are mapped to vehicles via trips such that both vehicles and requests can be assigned at most one trip. Each trip and vehicle mapping has an associated weight of o_v^f, and the solution to the matching problem can be seen as a maximum weight matching in this resulting tripartite graph. We write this optimization using the equation:

$$X^* = \arg \max_{X \in C(R)} \sum_{x_v^f \in X} x_v^f \cdot o_v^f(\mu^f) \tag{2}$$

The matching operation is performed by solving an Integer Linear Program. The specific formulation is given below. While there is an exponentially large number of trips in the worst case, (Alonso-Mora et al. 2017) provided an approximation method for generating feasible trips that work well in practice. This set of feasible trips, $\mathcal{F}$ contains requests that satisfies quality constraints (described in the next section).

$$\begin{aligned} \max \quad & \sum_{x_v^f} x_v^f * o_v^f(\mu^f) \\ s.t. \quad & \sum_{v \in V} x_v^f = 1 ::: \forall f;; \sum_{f|r \in f} \sum_{v \in V^r} x_v^f \leq 1 ::: \forall r \\ & x_v^f \in \{0,1\} ::: \forall t, v \end{aligned}$$

The constraints in the above Integer Linear Program (ILP) are referred to as $C(.)$ in Equation 2.

Matching Objective, $o_v^f(\mu^f)$: In our formulation, we abstract away the objective in terms of $o_v^f(\mu^f)$, but concretely it represents the system-level objective that the operator wants to maximise. This could be an obvious goal like profit or revenue, but may also be a way to incorporate future information (Shah, Lowalekar, and Varakantham 2020) or even fairness metrics (Lesmana, Zhang, and Bei 2019). All these objectives can be modeled as a linear function of the revenue of the trip.

Profit:

$$o_v^f(\mu^f) = [\sum_{r \in R^f} p^r(\mu^r) \cdot \mu^r] - cost_v^f$$

where $cost_v^f$ is a constant denotes the marginal increase in cost incurred by serving a trip f with vehicle v.

NeurADP (Shah, Lowalekar, and Varakantham 2020):

$$o_v^f(\mu^f) = [\sum_{r \in R^f} p^r(\mu^r) \cdot \mu^r] + \gamma \cdot \mathbb{V}_v^f$$

where the second term $\gamma \cdot \mathbb{V}_v^f$ is a learned constant. While we look at determining the optimal pricing and matching for a given batching interval in this paper, NeurADP attempts to match riders to drivers such that it is optimal across multiple batching intervals. Their solution takes the form of adding a 'future value' $\gamma \cdot \mathbb{V}_v^f$ to the matching objective in every batching interval and fits neatly into our generalisation.

Historical Earnings (Lesmana, Zhang, and Bei 2019):

$$o_v^f(\mu^f) = [\sum_{r \in R^f} \mu^r \cdot p^r(\mu^r)] + hist_v$$

where the second term is a constant that denotes the historical earnings for a given driver. In the paper, they attempt to even out driver earnings by adding the driver's historical earnings to the objective.

As a result, we focus on maximising a set of general objectives that can be written as a linear function of a trip's revenue (α_v^f and β_v^f are constants):

$$o_v^f(\mu^f) = \alpha_v^f \cdot \sum\nolimits_{r \in R^f} p^r(\mu^r) \cdot \mu^r + \beta_v^f \tag{3}$$

The overall objective is combinatorial in the requests because each o_v^f component depends on the source and destination of all the requests in trip f, as well as the existing trajectory of vehicle v.

Related Work

The existing work for optimising ride-sharing systems can be categorized based on different dimensions, as shown in Table 2. These dimensions include how the pricing and matching decisions are made, the capacity of the vehicles considered, sequential (one by one) or batched (considering all active requests together) processing of requests and the amount of future information available to the algorithm.

As seen in the table, most of the existing work for ride-sharing systems has studied the decision-making in isolation, i.e., they either perform matching by assuming a heuristic or fixed pricing (Ma, Zheng, and Wolfson 2013; Zheng, Chen, and Ye 2018), or focus on pricing decisions and ignore the optimisation for matching (Banerjee, Riquelme, and Johari 2015; Bimpikis, Candogan, and Saban 2019). Most existing works also consider sequential processing of requests, for at least one of the decision-making components. The sequential solution is faster to compute but is typical of poorer quality than the batched solution as it has less information available to make a decision (Uber 2018).

Banerjee et al. (Banerjee, Riquelme, and Johari 2015; Banerjee, Johari, and Riquelme 2016) provide a threshold-based pricing mechanism where the platform raises the price whenever the available vehicles in the region fall below a threshold. Their focus is on studying the theoretical properties of such threshold-based policies. Bimpikis, Candogan, and Saban (2019) provide a spatial pricing scheme for ride-sharing markets where they set a price for each region. As opposed to our work, these works do not focus on optimising matching decisions, and heuristically match the customer with unit-capacity vehicles available in the region. However, while these region-based simplifications may work well in the unit-capacity case, matching in the multi-capacity case is much more complicated and cannot be approximated in the same way.

Özkan (2020) theoretically prove that optimising only along one dimension is not optimal in general and joint pricing and matching optimisation can significantly increase performance. Unlike this paper, however, they do not provide an explicit algorithm, but rather focus on deriving conditions in which optimising only pricing or matching individually can help.

	Matching	Pricing	Future Demand and Price-Sensitivity	Capacity of Vehicles	Request Processing
Banerjee et al. 2015 Banerjee et al. 2016 Ma et al. 2019	Not optimised	optimised	Known Distribution	Unit Capacity	Region based Pricing Sequential (Matching)
Ma et al. 2013 Zheng et al. 2018	optimised	Not optimised	No Information	Multi-Capacity	Batched (Matching)
Chen et al. 2019a	optimised	optimised	Known Distribution	Unit Capacity	Batched(Matching) Sequential (Pricing)
Ma et al. 2019	optimised	optimised	Exact Information	Unit Capacity	Batched
Our Work Shah et al. 2022	**optimised**	**optimised**	**Known Distribution**	**Multi-Capacity**	**Batched**

Table 2: Summary of Differences Between Our Work and Related Work

There has been some research on providing algorithms for jointly optimising pricing and matching decisions for these systems. Ma, Fang, and Parkes (2019) provide a spatio-temporal pricing mechanism that provides competitive equilibrium prices and can achieve a higher revenue than a myopic pricing scheme. They make an unrealistic assumption that all the information about future requests is exactly known. On the other hand, the only assumption made about requests in this paper is that we know the price-sensitivity of the customers making the request.

Chen et al. (2019a,b) provide frameworks for jointly optimising pricing and matching decisions. However, these formulations either are not suitable to the multi-capacity setting or are not scalable. As shown in our experimental results, our approach can provide decisions in real-time for multi-capacity ride pooling in a city scale problem.

The key difference in our approach and (Shah, Lowalekar, and Varakantham 2022) is that we employ future aware joint matching and pricing, while Shah *et al.* only employ myopic pricing.

Methodology

In this section, we describe our contributions in solving the meta optimization problem of Equation 1 while considering the potential future impact of pricing and matching decisions. A brute-force approach would be to employ a centralized Reinforcement Learning method that makes joint pricing and matching decisions. However, such an approach is not viable and suitable, for multiple reasons. First, there are thousands of vehicles and a few hundred requests at each time step, so the complexity of even solving the single step meta optimization of Equation 1 is very significant. Second, matching decisions are dependent on pricing decisions, so searching for a joint policy that makes both decisions simultaneously can result in searching over many unnecessary policies. Finally, matching problem has a specific structure where transition uncertainty (arising due to customer requests) is external to the action space (matching decisions). Similarly, the pricing problem for a vehicle has a specific structure, where the pricing strategy is primarily dependent on the vehicle neighborhood (requests and vehicles in the neighborhood). A centralized Reinforcement learning method will not be able to exploit such structure. To that end,

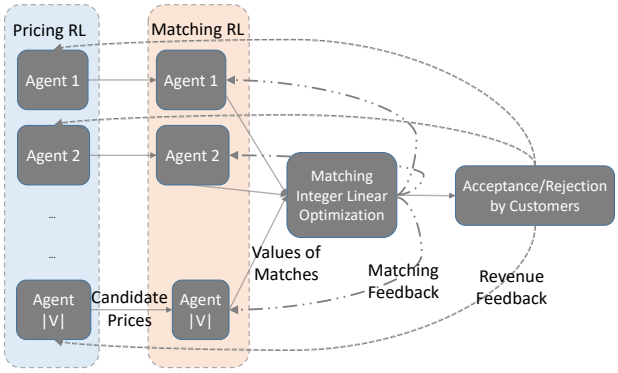

Figure 1: Two layer RL based framework for future aware pricing and matching in on-demand ride pooling

we provide a two-layered Reinforcement Learning approach for each vehicle, as shown in Figure 1. Given the set of requests at a given time step, we have the following overall framework:

- **Pricing RL** for each vehicle/agent (referred to as Agent 1 Pricing RL etc. in the figure) is responsible for computing candidate prices for neighborhood requests of that vehicle;
- Based on the computed candidate prices, **Matching RL** for each vehicle (referred to as Agent 1 Matching RL, etc.) computes the future value for different possible matches;
- Using the future values of different matches from all vehicles, a centralized linear integer optimization program (referred to as the Matching Integer Linear Optimization in the figure) computes the best feasible matching of vehicles to requests;
- Based on the assignment output ("Matching Feedback" dotted line in the figure) by the integer optimization program, weights in the deep neural network (in Matching RL) are used for estimating the values of different matches (actions in Matching RL) are updated for the different vehicles; and
- Finally, customers can accept or reject the price (if high)

and this feedback is used to update the weights in the deep network for estimating the values of different price factors (actions) in Pricing RL for different vehicles.

Pricing RL

One of the key challenges in solving the pricing optimization problem of Equation 1 is the maximization over a combinatorial solution space (of pricing vectors). In the real world taxi setup considered in our experiments, the number of requests per minute is around 300, so this would be a 300 dimension search space. The price for a request depends on the supply (available taxis)-demand (customer requests) imbalance[1] around the request location and we exploit this observation in providing a scalable solution. Instead of centrally deciding on a price vector, as shown in Figure 1, we compute candidate prices for requests in the neighborhood of a vehicle using the Pricing RL box for that vehicle. Specifically, to capture the dependence of pricing on neighborhood vehicles/agents, we use Mean Field Q-Learning (MFQL) in the pricing RL box. MFQL generates for each vehicle a multiplier on top of the base price for serving requests that can be served by the vehicle.

Formally, we initially compute a base price μ_0^r for each request, r based on factors such as source, destination, travel distance, etc. The goal of MFQL is to compute a multiplier, a_v for each vehicle in serving a request in its neighborhood, so as to maximize the overall objective (e.g., revenue). The candidate price, $\tilde{\mu}_v^r$ due to vehicle v for request r will be

$$\tilde{\mu}_v^r = \mu_0^r \cdot a_v$$

$\tilde{\mu}^r = \{\tilde{\mu}_v^r\}_{v \in V}$ refers to the candidate prices for request r by all vehicles that can serve the request.

Mean Field Q-Learning, MFQL: In problems with a large number of agents with actions that can be aggregated in a meaningful way, mean field Multi-Agent Reinforcement Learning methods (MARL) methods (Yang et al. 2018) have been employed successfully. In these methods, interactions between agent populations are approximated as interactions between a single agent and the average influence of neighboring agents. We adapt this broad idea in computing candidate prices for requests in the neighborhood.

The key elements of the learning problem are as follows. Each vehicle is defined as an **agent**. The **observation** space of each vehicle is given by:

$$\omega_v = (l_v, l_{N_v}, R_v)$$

where l_v corresponds to the location (intersection in the road network) of vehicle v, N_v corresponds to the vehicles that are close to v, l_{N_v} corresponds to the location of other vehicles in the neighborhood of v and R_v corresponds to the set of requests that can be served by v. The available **action** set of each agent, a_v is discrete and can take values from set of price factors for that agent, for e.g., $\mathcal{A}_v = \{0.8, 0.9, 1.0, 1.1, 1.2\}$.

[1]It will also depend on the price sensitivity of customers and we consider the price sensitivity as part of the matching objective.

The **state transition** of an agent is determined by the matching phase (of Section) and will take it to the location of the assigned request. Like with state transition, **reward** for an agent, $\mathcal{J}_v$ is determined by the outcome of matching and final pricing phase and specifically the request combination, f assigned to vehicle v and the price sensitivity function of the customer. Thus, reward in expectation is

$$\sum_{r \in f} \tilde{\mu}_v^r \cdot p^r(\tilde{\mu}_v^r)$$

However, for a specific request and a customer, it will depend on whether the customer accepted the price.

The action or price factor a_v for each vehicle v is a discrete categorical variable represented as the one-hot encoding. State provides supply demand imbalance in the neighborhood of the current agent. Actions (price factors) of the neighbor agents provide signal of the supply demand imbalance in the areas around the neighborhood. Specifically, we calculate mean action of the neighbor agents, N_v of the vehicle v. The average response is $\bar{a}_{N_v} = \frac{1}{|N_v|} \sum_{k \in N_v} a_k$. Assuming that vehicle v is only affected by the actions of neighboring vehicles, the Q function becomes $Q_v(\omega_v, \bar{a}_{N_v}, a_v)$.

For action selection, the Boltzmann softmax selector is used to obtain the final action probability:

$$a_v \sim \pi_v(\cdot \mid \omega_v, \bar{a}_{N_v}),$$
$$\pi_v(a_v \mid \omega_v, \bar{a}_{N_v}) = \frac{\exp\left(\beta \cdot Q_v\left(\omega_v, a_v, \bar{a}_{N_v}\right)\right)}{\sum_{a'_v \in \mathcal{A}_v} \exp(\beta \cdot Q_v\left(\omega_v, a'_v, \bar{a}_{N_v}\right))} \quad (4)$$

where the $\mathcal{A}_v$ is the action space of vehicle (agent) v.

Similar to Q-learning, the mean field Q-function is updated as follows:

$$Q_v^{t+1}(\omega, a_v, \bar{a}_v) = (1-\alpha) Q_v^t(\omega, a_v, \bar{a}_v) + \alpha\left[\mathcal{J}_v + \gamma V_{\text{MF}}^t{}_v(\omega')\right] \quad (5)$$

where

$$V_{\text{MF}}^t{}_v(\omega') = \sum_{a_v} \pi_v^t \mathbb{E}_{\bar{a}_v(\boldsymbol{a}_{-v}) \sim \pi_{-v}^t}\left[Q_v^t(\omega', a_v, \bar{a}_v)\right] \quad (6)$$

where the $-v$ is the other agent except agent v.

Matching RL

After computing the candidate prices, we employ future aware (consider matches that will enable better matches in the future) matching to compute the values for different matches given the prices computed in the first step. In this case, we employ an approximate dynamic program as transition uncertainty is external to the action taken. That is to say, uncertainty is associated with customer requests arising at a period and this is not dependent on the action taken. Therefore, we employ an Approximate Dynamic Program (ADP) to exploit this structure, as has been exploited in previous works (Shah, Lowalekar, and Varakantham 2020).

ADP is similar to a Markov Decision Problem (MDP) with the key difference that the transition uncertainty is extrinsic to the system and not dependent on the action. The ADP problem for the matching problem for each vehicle, v is formulated as follows :

S_v : The state of a vehicle considers the location and requests that are in the neighborhood.

A_v : At each time step there are a large number of requests arriving to the taxi service provider, however for an individual vehicle only a small number of such requests is reachable. The feasible set of request combinations for each vehicle v at time t, $\mathcal{F}_t^v$ is given by:

$$\mathcal{F}_v^t = \Big\{ f_v \mid f_v \in \cup_{c'=0}^{c^v} [\mathcal{U}]^{c'}, \text{PickUpDelay}(f_v, v) \leq \tau \quad \text{DetourDelay}(f_v, v) \leq \lambda \}$$

where c_v is the maximum capacity of vehicle, and $c' \leq c_v$ is the optional capacity of vehicle v. We use $[\mathcal{U}]^{c'}$ as a set of permutations and combinations of optional orders under optional capacity c'. $\cup_{c'=0}^{c_v} [\mathcal{U}]^{c'}$ is the union of permutations and combinations under all optional capacities, including the empty set when $c' = 0$.

$x_v^{t,f}$ is the decision variable that indicates whether vehicle v takes action f (a combination of requests) at a decision epoch t.

ξ : denotes the exogenous information – the source of randomness in the system. This would correspond to the user requests or demands. ξ^t denotes the exogenous information at time t.

T_v : denotes the transitions of vehicle state. In an ADP, the system evolution happens as:

$$(s_v^0, x_v^0, \tilde{s}_v^0, \xi_1, s_v^1, x_v^1, \tilde{s}_v^1, \cdots, s_v^t, x_v^t, \tilde{s}_v^t, \cdots)$$

where s_v^t denotes the pre-decision state at decision epoch t for vehicle v and $\tilde{s}_v^t$ denotes the post-decision state (Powell 2007). The transition from state s_v^t to s_v^{t+1} depends on the action vector x_v^t and the exogenous information ξ^{t+1}. Therefore,

$$s_v^{t+1} = T_v(s_v^t, x_v^t, \xi^{t+1}); \qquad \tilde{s}_v^t = \tilde{T}_v(s_v^t, x_v^t);$$
$$s_v^{t+1} = T_v^{\xi}(\tilde{s}_v^t, \xi_{t+1}) \tag{7}$$

It should be noted that $\tilde{T}_v(.,.)$ is deterministic as uncertainty is extrinsic to the system.

o_v : denotes the revenue for vehicle v considering the candidate prices provided by pricing RL. The $o_v^f(\mu_v^f)$ is vehicle v's revenue which can be obtained by selecting the request combination f given candidate prices, μ_v^f from pricing RL. The total revenue for vehicle v is

$$o_v(s_v, f) = o_v^f(\mu_v^f) = \sum_{r \in f} \mu_v^r \tag{8}$$

where μ_r^v is the candidate price provided by pricing RL for request r.

Bellman equations over the joint value of all agents can be provided as follows:

$$V(s^t) = \max_{x^t \in \mathcal{F}^t} (o(s^t, x^t) + \gamma \mathbb{E}[V(s^{t+1}) | s^t, x^t, \xi^{t+1}]) \tag{9}$$

where γ is the discount factor. Using post-decision state, this expression breaks down nicely:

$$V(s^t) = \max_{x^t \in \mathcal{F}^t} (o(s^t, x^t) + \gamma V(\tilde{s}^t));$$
$$V(\tilde{s}^t) = \mathbb{E}[V(s^{t+1}) | \tilde{s}^t, \xi^{t+1}] \tag{10}$$

For the matching problem, Shah *et al.* (Shah, Lowalekar, and Varakantham 2020) introduced a two-step decomposition of the joint value function in the second equation above that converts it into a linear combination over individual value functions associated with each vehicle. This allows us to get around the *combinatorial explosion of the post-decision state of all vehicles.* NeurADP thus has the joint value function :

$$V(\tilde{s}^t) = \sum_v V_v([\tilde{s}_v^t, s_{-v}^t]) \tag{11}$$

We then evaluate these individual V_v values for all possible $\tilde{s}_v^t$ from the individual value neural network and pass it to the matching integer linear optimization problem indicated in Figure 1.

Centralized Matching: Linear Integer Optimization

Given the values for different matches for each vehicle from Matching RL, we compute the best joint matching strategy over all available vehicles and customer requests. The key goal of this module is to ensure matching constraints are satisfied, i.e., (a) each vehicle, v can only be assigned at most one request combination, f; (b) at most one vehicle, v can be assigned to a request r; and (c) a vehicle, v can be either assigned or not assigned to a request combination.
$x_t^{v,f}$ is set to 1 if vehicle v takes action f at time t. Formally, these three constraints can be specified as follows:

$$\sum_{f \in \mathcal{F}_v^t} x_v^{t,f} = 1 ::: \forall v; ; \sum_v \sum_{f \in \mathcal{F}_v^t; r \in f} x_v^{t,f} \leq 1 ::: \forall r$$
$$x_v^{t,f} \in \{0, 1\} ::: \forall v, f \tag{12}$$

The myopic objective over all agents is given by:

$$o(s^t, x^t) = \sum_v \sum_{f \in \mathcal{F}_v^t} o_v^{t,f}(\mu_v^f) \cdot x_{v,f}^t \tag{13}$$

This myopic objective decomposes over the individual vehicles. Similarly, from Equation 11, we have the future value decomposing over individual values. Combining these two equations, we have the following overall Integer Linear Programming (ILP) considering the long term impact:

$$\max_{\mathbf{x}} \quad o(s^t, x^t) + V(\tilde{s}^t)$$
$$\text{s.t. } \textit{constraints in (12) are satisfied} \tag{14}$$

Before solving this integer linear optimization, we retrieve values for each vehicle from the individual neural network.

Experimental Results

We now describe the experimental setup and the performance results of our overall approach on a benchmark simulation that is based on a city scale real dataset.

Setup

Dataset Description We perform our experiments on the demand distribution from the publicly available New York Yellow Taxi Dataset (NYYellowTaxi 2016). The experimental setup is similar to the setup used by (Shah, Lowalekar,

and Varakantham 2020). Street intersections are used as the set of locations $\mathcal{L}$. They are identified by taking the street network of the city from openstreetmap using osmnx with 'drive' network type (Boeing 2017). Nodes that do not have outgoing edges are removed, i.e., we take the largest strongly connected component of the network. The resulting network has 4373 locations (street intersections) and 9540 edges. The travel time on each road segment of the street network is taken as the daily mean travel time estimate computed using the method proposed in (Santi et al. 2014).

Similar to previous work, we only consider the street network of Manhattan as a majority ($\sim$75%) of requests have both pickup and drop-off locations within it. The dataset contains data about past customer requests for taxis at different times of the day and different days of the week. From this dataset, we take the following fields: (1) Pickup and drop-off locations (latitude and longitude coordinates) - These locations are mapped to the nearest street intersection. (2) Pickup time - This time is converted to appropriate decision epoch based on the value of Δ. The dataset contains on an average 322714 requests in a day (on weekdays) and 19820 requests during peak hour.

We evaluate the approaches over 24 hours on different days starting at midnight and take the average value over 5 weekdays (4 - 8 April 2016) by running them with a single instance of the initial random location of taxis [2]. Our approach is trained using the data for 8 weekdays (23 March - 1 April 2016) and it is validated on 22nd March 2016. For the experimental analysis, we consider that all vehicles have identical capacities.

Simulator In standard supervised learning problems, the data is stationary for the algorithm. Our 2 layered Reinforcement learning is different from this learning paradigm. It needs continuous interaction with the environment to learn the pricing and matching strategies. Using real-world data to build traffic simulators is a common method in all previous works (Alonso-Mora et al. 2017; Shah, Lowalekar, and Varakantham 2020; Lesmana, Zhang, and Bei 2019; Lowalekar, Varakantham, and Jaillet 2019). More details on the simulator provided in Appendix C. We use the price sensitive function obtained from Uber data (Yan et al. 2019) to determine the probability of customer acceptance.

$$p_r(\mu_v^r) = \frac{1}{1 + e^{\frac{0.67 \cdot \mu_v^r}{\mu_0^r} - 1.69}} \tag{15}$$

where μ_0^r is the base price, and μ_v^r is the actual price.

Baselines

To demonstrate the utility of our joint pricing and matching framework, we compared against baselines that optimize pricing and matching individually. Specifically, in these baselines, we replaced the component (pricing/matching) that is not being optimized with existing methods:

[2]All experiments are run on 60 core - 3.8GHz Intel Xeon C2 processor and 240GB RAM. The algorithms are implemented in python and optimisation models are solved using CPLEX 20.1

Name	Pricing	Matching
M & N-E	Mean field	NeurADP-Exp return-Eqn 16
M & N-N	Mean field	NeurADP-Norm return-Eqn 8
M & IR	Mean field	Immediate reward
Q & N-E	Q-learning	NeurADP-Exp return-Eqn 16
Q & N-N	Q-learning	NeurADP-Norm return-Eqn 8
F & N-E	Fixed price	NeurADP-Exp return
F & IR	Fixed price	Immediate reward

Table 3: List of all approaches. Fixed price is a myopic pricing approach. Immediate reward is a myopic matching approach. NeurADP is a future aware matching approach. Mean Field is the future aware pricing approach.

Name	Result					
	1000v		1500v		2000v	
	2c	4c	2c	4c	2c	4c
M & N-E	9.46	5.09	4.74	1.75	1.22	1.20
M & N-N	9.35	3.06	4.73	1.27	0.98	1.14
M & IR	6.10	-0.32	2.75	1.10	1.34	0.56
Q & N-E	8.52	4.65	3.96	1.30	0.71	0.66
Q & N-N	-3.30	2.48	1.87	-4.94	0.36	0.02
F & N-E	0.00	0.00	0.00	0.00	0.00	0.00
F & IR	-1.20	-5.11	-1.70	0.37	-0.16	-0.01

Table 4: The impact of different pricing and matching strategies on the results. We show the results of overall revenue for different numbers of vehicles with 1000v referring to 1000 vehicles and 2c referring to a vehicle capacity of 2. The values in the table are calculated as the percentage increase relative to the F & N-E baseline.

(-) To illustrate the role of mean field in our framework, we use Q-learning as a pricing strategy for comparison.
(-) We also set up a baseline pricing algorithm using fixed price which obtains the price through travel distance and time.
(-) To make the algorithm match the vehicles with more reasonable bids, we set up an ADP with the expectation revenue of accepting orders. On the basis of formula 8 multiplied by an acceptance probability, the expected return can be obtained. Expected return of the vehicle v is:

$$o_v^{t,f}(\mu_v^{t,f}) = \sum_{r \in f} a_v^t \cdot \mu_v^{t,r} \cdot p_r(\mu_v^{t,r}) \tag{16}$$

We combined the components, and the abbreviations of the different approaches are shown in Table 3.

	Result				
	F&N-E	M&N-E	M&N-N	Q&N-E	Q&N-N
Uber	0.00	5.09	3.06	4.65	2.48
Conscious	0.00	17.55	4.45	-0.45	-47.11
Very Conscious	0.00	15.67	-34.24	12.41	-38.50

Table 5: Impact of price sensitivity on the performance. The values in the table are calculated as the percentage increase relative to the F & N-E baseline. We set the number of vehicles at 1000 and capacity at 4.

	F & IR	M & N-E
$\sim$ 200K	570v 2c	530v 2c
$\sim$ 250K	750v 2c	670v 2c
$\sim$ 300K	940v 2c	840v 2c
$\sim$ 350K	1140v 2c	990v 2c
$\sim$ 400K	1380v 2c	1180v 2c

Table 6: Number of vehicles required by different algorithms to reach the income level.

	F & IR	M & N-E	Save Distance
1000v 2c	11.27	11.18	0.78%
1000v 4c	11.27	11.09	1.70%
1500v 2c	11.18	10.98	1.85%
1500v 4c	11.08	10.69	3.50%
2000v 2c	10.78	10.44	3.16%
2000v 4c	10.56	9.38	11.13%

Table 7: Average kilometers traveled by all vehicles in one hour (18:00-19:00).

Results: Efficiency, Sustainability

To illustrate the performance of our proposed framework, comparisons are made along different dimensions. While we tried other peak time periods as well with similar results, we show the results for the evening peak period. Results in all tables are averaged over 5 runs and repeated over the testing days. The first metric we use to compare them is revenue value, which is the total revenue for the selected period. The second metric we use is the number of vehicles required to achieve a fixed revenue level. The decision epoch duration is set as 60 seconds. We utilized F & N-E as the baseline strategy, as this performed the worst among all approaches where at least one of pricing or matching was future aware. Here are the key observations:

(*) Our best future aware matching and pricing strategy, i.e., M & N-E provides up to 9% improvement over a method that employs future aware matching but myopic pricing, i.e., F & N-E. There is improvement across all settings and can be attributed to the simultaneous matching and pricing optimization in our approach. As expected, this improvement reduces as more higher capacity vehicles are available.

(*) Our best future aware matching and pricing strategy, i.e., M & N-E provides up to 6% improvement over a method that employs future aware pricing but myopic matching, i.e., M & IR. Improvement is consistent and reduces as higher capacity vehicles are available.

(*) Given a fixed matching strategy, our mean field pricing strategy due to considering neighbor strategies provides consistent improvement (up to 12%) over a Q-learning based pricing strategy.

(*) Using a myopic pricing and matching strategy performs the worst among all approaches.

(*) Using a sample based estimate for matching rewards, i.e., N-N strategy provided slightly lower performance than using expected rewards, i.e., N-E strategy, especially when we used mean field pricing strategy.

Impact on environment: To demonstrate that our approach is more sustainable, we fixed the overall revenue number and computed the number of vehicles required to achieve that revenue for different approaches. Table 6 shows the results. We observe that our M & N-E approach provides a reduction of up to 14.5% (and on average 11.2%)in the number of vehicles over a myopic pricing and matching approach.

We also computed the average number of kilometers traveled by all vehicles during the peak period. Table 7 provides these results. We were able to provide a reduction of up to 11.13% and on average a reduction of 3.8%.

Impact of price sensitivity, p: Customers in different cities can have different levels of price sensitivity. The uber one we have shown so far was for one city. When we made the sensitivity function more price conscious (which is something that can potentially happen in developing or underdeveloped countries), the impact of our approach was very significant as shown in Table 5. We achieve this price consciousness by scaling the coefficient of the final price and the constant term in Equation 15 by a factor of 10 which makes the transition from 'accept' to 'not accept' more dramatic. Once again, M & N-E approach clearly outperformed other approaches and the improvement was up to 17% over the baseline.

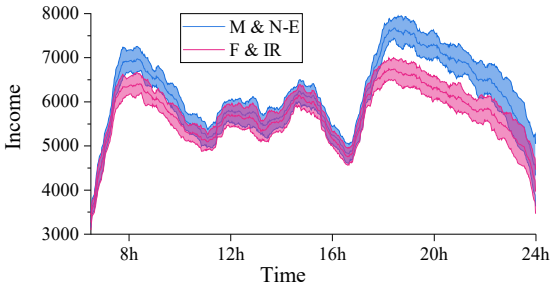

Figure 2: The curve represents the total revenue (mean and one standard deviation over 5 days) of all vehicles at 60-second intervals. We set the number of vehicles at 1000 and capacity at 4.

Conclusion

The emergence of Uber, LYFT, DiDi and other applications has improved service for customers while having higher utilization for taxis. Most existing work has focused on separately optimizing for the matching and pricing problems, and this can reduce the effectiveness, specifically in on-demand ride pooling systems. We have provided a novel 2 layered Reinforcement Learning approach with a centralized optimization for simultaneously optimizing pricing and matching. Our approach consistently improves over existing baselines and in the best case achieves an impressive 17% improvement in revenue and 14% reduction in number of vehicles for a city scale taxi dataset. This is a big improvement considering that even a 1% improvement in revenue is considered a big improvement in similar transportation problems by the industry (Xu et al. 2018).

Acknowledgments

This research/project is supported by the National Research Foundation Singapore and DSO National Laboratories under the AI Singapore Programme (AISG Award No: AISG2-RP-2020-017) and China Scholarship Council (No: 202006060229).

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.

Banerjee, S.; Johari, R.; and Riquelme, C. 2015. Pricing in Ride-Sharing Platforms: A Queueing-Theoretic Approach. In *Proceedings of the Sixteenth ACM Conference on Economics and Computation*, EC '15, 639. New York, NY, USA: Association for Computing Machinery. ISBN 9781450334105.

Banerjee, S.; Johari, R.; and Riquelme, C. 2016. Dynamic pricing in ridesharing platforms. *ACM SIGecom Exchanges*, 15(1): 65–70.

Banerjee, S.; Riquelme, C.; and Johari, R. 2015. Pricing in ride-share platforms: A queueing-theoretic approach. *Available at SSRN 2568258*.

Bimpikis, K.; Candogan, O.; and Saban, D. 2019. Spatial pricing in ride-sharing networks. *Operations Research*, 67(3): 744–769.

Boeing, G. 2017. OSMnx: New methods for acquiring, constructing, analyzing, and visualizing complex street networks. *Computers, Environment and Urban Systems*, 65: 126–139.

Chen, H.; Jiao, Y.; Qin, Z.; Tang, X.; Li, H.; An, B.; Zhu, H.; and Ye, J. 2019a. InBEDE: Integrating Contextual Bandit with TD Learning for Joint Pricing and Dispatch of Ride-Hailing Platforms. In *2019 IEEE International Conference on Data Mining (ICDM)*, 61–70. IEEE.

Chen, M.; Shen, W.; Tang, P.; and Zuo, S. 2019b. Dispatching through pricing: modeling ride-sharing and designing dynamic prices. In *Proceedings of the 28th International Joint Conference on Artificial Intelligence*, 165–171. AAAI Press.

Lesmana, N. S.; Zhang, X.; and Bei, X. 2019. Balancing efficiency and fairness in on-demand ridesourcing. In *Advances in Neural Information Processing Systems*, 5309–5319.

Lowalekar, M.; Varakantham, P.; and Jaillet, P. 2018. Online spatio-temporal matching in stochastic and dynamic domains. *Artificial Intelligence*, 261: 71–112.

Lowalekar, M.; Varakantham, P.; and Jaillet, P. 2019. ZAC: A Zone Path Construction Approach for Effective Real-Time Ridesharing. In *ICAPS*.

Ma, H.; Fang, F.; and Parkes, D. C. 2019. Spatio-temporal pricing for ridesharing platforms. In *Proceedings of the 2019 ACM Conference on Economics and Computation*, 583–583.

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

NYYellowTaxi. 2016. New York Yellow Taxi DataSet. http://www.nyc.gov/html/tlc/html/about/trip_record_data.shtml. Accessed: 2021-12-04.

Özkan, E. 2020. Joint pricing and matching in ride-sharing systems. *European Journal of Operational Research*.

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

Santi, P.; Resta, G.; Szell, M.; Sobolevsky, S.; Strogatz, S. H.; and Ratti, C. 2014. Quantifying the benefits of vehicle pooling with shareability networks. *Proceedings of the National Academy of Sciences*, 111(37): 13290–13294.

Shah, S.; Lowalekar, M.; and Varakantham, P. 2020. Neural Approximate Dynamic Programming for On-Demand Ride-Pooling. In *The Thirty-Fourth AAAI Conference on Artificial Intelligence*, 507–515. AAAI Press.

Shah, S.; Lowalekar, M.; and Varakantham, P. 2022. Joint Pricing and Matching for City-Scale Ride-Pooling. *Proceedings of the International Conference on Automated Planning and Scheduling*, 32(1): 499–507.

Uber. 2018. Uber Matching Solution. https://marketplace.uber.com/matching. Accessed: 2021-12-04.

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 and Data Mining*, KDD '18, 905–913.

Yan, C.; Zhu, H.; Korolko, N.; and Woodard, D. 2019. Dynamic pricing and matching in ride-hailing platforms. *Naval Research Logistics (NRL)*, 67(8): 705–724.

Yang, Y.; Luo, R.; Li, M.; Zhou, M.; Zhang, W.; and Wang, J. 2018. Mean Field Multi-Agent Reinforcement Learning. In *Proceedings of the 35th International Conference on Machine Learning (ICML)*, volume 80, 5571–5580.

Zheng, L.; Chen, L.; and Ye, J. 2018. Order dispatch in price-aware ridesharing. *Proceedings of the VLDB Endowment*, 11(8): 853–865.

Özkan, E. 2020. Joint pricing and matching in ride-sharing systems. *European Journal of Operational Research*, 287(3): 1149–1160.