Multi-Objective Distributional Reinforcement Learning for Large-Scale Order Dispatching

Fan Zhou, Chenfan Lu
Shanghai University of Finance and Economics
zhoufan@mail.shufe.edu.cn
chenyufanxin@gmail.com

Xiaocheng Tang, Fan Zhang, Zhiwei Qin, Jieping Ye, Hongtu Zhu
DiDi AI Labs
{xiaochengtang, feynmanzhangfan, qinzhiwei,
yejieping, zhuhongtu}@didiglobal.com

***Abstract*—The aim of this paper is to develop a multi-objective distributional reinforcement learning framework for improving order dispatching on large-scale ride-hailing platforms. Compared with traditional RL-based approaches that focus on drivers' income, the proposed framework also accounts for the spatiotemporal difference between the supply and demand networks. Specifically, we model the dispatching problem as a two-objective Semi-Markov Decision Process (*SMDP*) and estimate the relative importance of the two objectives under some unknown existing policy via Inverse Reinforcement Learning (IRL). Then, we combine Implicit Quantile Networks (IQN) with the traditional Deep Q-Networks (DQN) to jointly learn the two return distributions and adjusting their weights to refine the old policy through on-line planning and achieve a higher supply-demand coherence of the platform. We conduct large-scale dispatching experiments to demonstrate the remarkable improvement of proposed approach on the platform's efficiency.**

***Index Terms*—Order Dispatching, Multi-Objective Reinforcement Learning, Distributional Reinforcement Learning, Inverse Reinforcement Learning, Semi-Markov Decision Process**

I. Introduction

In the past few years, the rapid development of mobile internet service brings the advent of large scale online ride hailing services such as Uber and DiDi, which substantially transforms the transportation landscape of human beings. With advanced data storage and processing technologies, platforms can continuously collect and analyze real-time traveling information, dynamically updating the platform policies [7], [11], [22]. In this work, we consider the problem of driver-passenger dispatching [2], [9], [15], [23], [27], which can be treated as a sequential decision making problem by assigning available drivers to nearby passengers over a large-scale spatio-temporal region. Each single driver makes serving decisions, guided by some underlying order dispatching policy as opposed to each driver learning his/her own policy [16], [19].

Traditional dispatching policy follows the 'first-come first-served' rule [8], [24], which is non-optimal from both spatial and temporal perspectives. Zhang et al. [23] makes an improvement by solving a combinatorial optimization problem , but still ignores the long-term effects. Three recent studies [16], [18], [20] introduce the idea of reinforcement learning to ensure the spatio-temporal optimality. They solve the real-time matching by using the spatio-temporal value function obtained from off-line deep Q-learning.

Unfortunately, none of these methods are optimal from the perspective of passengers since they all aim to maximizing the total driver income (TDI) other than the supply-demand equilibrium which is crucial for passenger-side metrics, such as order response rate (ORR) and order completion rate (OCR). In practice, the supply-demand equilibrium is achieved when the spatio-temporal distributions of supplies and demands are well aligned to each other. In this case, customer requests can be more quickly answered by nearby idle drivers such that ORR can be increased. OCR is also increased as the long-distance dispatching becomes less frequent which reduces the average waiting time of customers and pick-up cost of drivers. Furthermore, without taking the spatio-temporal supply-demand differences into consideration, the expected TDI may be overestimated by the existing RL based methods since they model each single driver as an independent agent and ignore the interaction effects when multiple drivers are transferring to the same state. When the future demand has already been fulfilled by drivers re-allocated by previous completed trips, assigning more drivers there might decrease the original value of the target location. Therefore, the healthiness of a ride-hailing platform, which is a two-sided marketplace, hinges on good experiences for both drivers and passengers. Simply maximizing TDI may exacerbate the misalignment of supply and demand distributions, leading to longer average waiting time for passengers and less desirable ridership experience.

In this paper, we propose a multi-objective reinforcement learning (MODRL) framework to jointly maximize drivers' revenues and the overall supply-demand coherence. We model the service trajectory of each single driver as a *multi-objective semi-Markov decision process* and the scalarized reward function can be seen as the weighted sum of order fee and supply-demand difference at the destination of either a charged trip or an idle movement. We use Inverse Reinforcement Learning (IRL) to learn this reward function based on the observed moving trajectories of platform drivers. To capture the intrinsic randomness of the dispatching process, which arises from the environment stochasticity, we combine a recent distributional reinforcement learning (DRL) algorithm, Implicit Quantile Networks (IQN) [3], with the traditional value-based Deep Q-Networks (DQN) to jointly learn the quantile functions for the two separate objectives and the expectation of their weighted sum under the background policy. With the obtained

spatio-temporal value functions, we search for the optimal combination of the two objectives in evaluating each driver-passenger pair and optimally assign collected orders to nearby idle drivers within each small dispatching window to maximize the total utility scores [19]. We compare our method with other state-of-art strategies using a realistic dispatching simulator. According to the experiment results, our method can not only improve the TDI on the supply side but also increase the ORR and OCR on the demand side.

II. A Multi-Objective Semi-MDP Formulation

We model the dispatching system as a *multi-objective semi-Markov decision process* (*MOSMDP*) with a set of temporally extended actions, known as *options*. Under the framework of *MOSMDP*, each driver (*agent*) interacts episodically with the ride-hailing platform (*environment*) at some discrete time scale, $t \in \mathcal{T} := \{0, 1, 2, ..., T\}$ until *termination* T is reached. Within each action window t, the driver perceives the state of the environment, described by the feature vector $s_t \in \mathcal{S}$, and on that basis chooses an option (temporally extended action) $o_t \sim \pi(\cdot|s_t) \in \mathcal{O}$ following an underlying dispatching policy $\pi : \mathcal{S} \times \mathcal{O} \to [0, 1]$ that terminates in $s_{t+\Delta_t} \in P(\cdot|s_t, o_t)$. As mentioned in the introduction part, each single driver follows the dispatching policy π till the end of a day although he or she is allowed to move around before the next trip assignment is activated. As a response, the environment produces a set of numerical rewards $(R_1(s_t, o_t), R_2(s_t, o_t))$ for each intermediate step t. We highlight the following specifics:

State, **Option**, same as [16], thus the details are omitted.

Reward, a vector of two different rewards $R(s_t, o_t) = (R_1(s_t, o_t), R_2(s_t, o_t))^\top$ received by executing option o_t at state s_t, where $R_1(s_t, o_t)$ denotes the order fee and $R_2(s_t, o_t)$ denotes the spatio-temporal relationship at destination $s_{t+\Delta_t}$. To be specific, the demand-supply difference (number of demands - number of idle supplies) in a target spatio-temporal grid $s_{t+\Delta_t}$ is used to represent $R_2(s_t, o_t)$, which is one the most important metrics to quantify the supply-demand coherence of ride-sharing platforms and should always be considered together with TDI. $o_t = 1$ results in both nonzero R_1 and R_2, while $o_t = 0$ leads to a transition with zero R_1 but non-zero R_2. For order dispatching, both $R_1(s_t, o_t)$ and $R_2(s_t, o_t)$ are spread uniformly across the trip duration induced by the discount factor γ, we can thus introduce the discounted accumulative reward $\hat{R}_i$ for each i as

$$\begin{aligned}\hat{R}_i(s_t, o_t) &= \frac{R_i(s_t, o_t)}{\Delta_t} + \gamma\frac{R_i(s_t, o_t)}{\Delta_t} + \cdots + \gamma^{\Delta_t - 1}\frac{R_i(s_t, o_t)}{\Delta_t} \\ &= \frac{R_i(s_t, o_t)(\gamma^{\Delta_t} - 1)}{\Delta_t(\gamma - 1)}, \quad \text{where } 0 < \gamma < 1,\ \Delta_t \geq 1\end{aligned}$$

Policy, $\pi(o|s)$ specifies the probability of taking option o in state s regardless of the time step t. Executing π in the environment generates a history of driver trajectories denoted as $\{\tau_k\} \in \mathcal{H} := \{(s_{kt_0}, o_{kt_0}, \ldots, s_{kt_{T_{\tau_k}}}\}$ where each t_j is the time index of the j-th activated state along the trajectory τ_k. We use $Z^\pi(s) = (Z_1^\pi(s), Z_2^\pi(s))^\top$ to denote the random variable of the cumulative rewards the driver gains from s by following policy π. The expectation of $Z^\pi(s)$ is $V^\pi(s) = \mathbb{E}_{\pi,p,R}(Z^\pi(s))$, and we have the multi-objective distributional Bellman equation for for Z^π

$$\begin{aligned}Z^\pi(s_t) &:\overset{D}{=} \hat{R}(s_t, o_t) + \gamma^{\Delta_t} Z^\pi(s_{t+\Delta_t}) \\ s_{t+\Delta_t} &\sim P(\cdot|s_t, o_t), o_t \sim \pi(\cdot|s_t)\end{aligned} \tag{1}$$

where $:\overset{D}{=}$ here denotes distributionally equivalence.

III. Multi-Objective Distributional Reinforcement Learning

In this section, we describe the Multi-Objective Distributional Reinforcement Learning (MODRL) framework to learn the return distribution $Z^\pi(s) = (Z_1^\pi(s), Z_2^\pi(s))^\top$ and its expectation $V^\pi(s)$ under some unknown existing policy π the platform currently uses in the real world. We assume that the *MOSMDP* specified in this paper employs scalarization functions [17] to define a scalar utility over a vector-valued policy which reduces the dimensionality of the underlying multi-objective environment. The importance of the two objectives in the scalarized reward function $\tilde{w}_1$ and $\tilde{w}_2$ under the existing policy π can be obtained through an IRL-based approach using the observed historical trajectories, which is described in Section III-A. With the learned weights $\tilde{W} = (\tilde{w}_1, \tilde{w}_2)$, we then propose a joint RL based estimation approach in Section III-B to learn value functions $V_1^\pi(s)$ and $V_2^\pi(s)$, which incorporates the distributional reinforcement learning based method IQN with traditional DQN to not only capture the uncertainty within the total returns but also fully utilize the joint information of the two objectives obtained from IRL.

A. Multi-Objective Inverse Reinforcement Learning

Most existing reinforcement learning methods for multi-objective tasks rely on single-policy algorithms [5], [10] which transform the reward vector into a scalar. For any given policy π, we let the scalarization f_π be a function that projects $\hat{R}$ to a scalar by a weighted linear combination $f_\pi(\hat{R}; W) = \sum_{i=1}^2 w_i \hat{R}_i$ where $W = (w_1, w_2)^\top$ is a weight vector parameterizing f_π.

IRL has been widely used to learn reward function of an MDP in the past decade. The key idea of IRL methods is to find a reward function such that the estimations of action-state sequences under an underlying policy matches the observed historical trajectories [13], [25], [26]. The cumulative reward for each objective $i \in \{1, 2\}$ along a trajectory τ is defined as $\hat{R}_i(\tau) = \sum_{j=0}^{T_\tau} \gamma^{t_j} \hat{R}(s_{t_j}, o_{t_j})$ Let $\hat{R}(\tau) = (\hat{R}_1(\tau), \hat{R}_2(\tau))^\top$, the expected return $J(\pi)$ under policy π can be written as a linear function of the two reward expectations

$$J(\pi) = \sum_{\tau \in \mathcal{H}} P(\tau|\pi, T) f_\pi(\hat{R}(\tau); W) \tag{2}$$

where $\mathcal{H}$ denotes the set of all possible driver trajectories and $f_\pi(\hat{R}(\tau); W) = \sum_{i=1}^2 w_i \hat{R}_i(\tau)$. $P(\tau|\pi, T)$ is the probability of trajectory τ being generated by the *MOSMDP* such that

$$P(\tau|\pi, T) = \sum_{(s,o) \in \tau} \pi(o|s) T(s'|s, o) \tag{3}$$

where $\pi(o|s)$ is the probability of taking action o at state s according to the background policy. $T(s'|s,o)$ is the probability of transferring s to s' after taking action o. Let $P(\tau|\pi,T)=P(\tau)$ for simplification when π and T are given, the equation $\sum_{\tau\in\tilde{H}}P(\tau)\hat{R}(\tau)=\tilde{R}$ holds where $\tilde{R}$ is the empirical expectation of $\hat{R}(\tau)$ based on some observed drivers' trajectories $\tilde{H}$. Most existing methods [25], [26] can obtain the maximum likelihood estimate of W by using gradient decent where the gradient is given by $g=\tilde{R}-\sum_{\tau\in\tilde{H}}P(\tau)\hat{R}(\tau)$.

In our case, it is difficult to estimate the gradient g above since the transition function T in $P(\tau)$ cannot be easily computed considering the system complexity and the limited observed trajectories. To address this issue, we employ Relative Entropy Inverse Reinforcement Learning (REIRL) to learn the weight vector W. REIRL is a model free method, which incorporates Relative Entropy Policy Search (REPS) [14] with Generalized Maximum Entropy methods [4]. By using REIRL, we can get rid of T and estimate $F(W)=\sum_{\tau\in\hat{h}}P(\tau)\hat{R}(\tau)$ as follows:

$$F(W)=\sum_{\tau\in\hat{h}}\frac{\mathcal{U}(\tau)\Pi(\tau)^{-1}e^{f_\pi(\hat{R}(\tau);W)}}{\sum_{\tau'\in\hat{h}}\mathcal{U}(\tau')\Pi(\tau')^{-1}e^{f_\pi(\hat{R}(\tau');W)}}\hat{R}(\tau) \quad (4)$$

where $\hat{h}$ is a small batch sampled from the whole collective trajectory set $\tilde{H}$. $\mathcal{U}(\tau)$ is the uniform distribution and $\Pi(\tau)$ is the trajectory distribution from the underlying policy π which is defined as $\Pi(\tau)=\prod_{j=1}^{T_\tau}\pi(o_{t_j}|s_{t_j})$. In practice, $\pi(o|s)$ is usually estimated via a deep neural network, and in this work it share shares the same main architecture with Cerebellar Value Networks (CVNet) [16]. Following the derivations in [1], the gradient $g(W)=\tilde{R}-F(W)$ can be estimated by

$$g(W)=\tilde{R}-\sum_{\tau\in\hat{h}}\frac{\mathcal{U}(\tau)\Pi(\tau)^{-1}e^{f_\pi(\hat{R}(\tau);W)}}{\sum_{\tau'\in\hat{h}}\mathcal{U}(\tau')\Pi(\tau')^{-1}e^{f_\pi(\hat{R}(\tau');W)}}\hat{R}(\tau). \quad (5)$$

We can finally obtain the weight vector $\widetilde{W}=(\tilde{w}_1,\tilde{w}_2)^T$ corresponding to the underlying policy by iteratively applying the gradient descent defined in (5) until convergence reached.

B. A Joint Estimation Approach using Distributional Reinforcement Learning

Assuming $\tilde{w}_1$ and $\tilde{w}_2$ are the learned weights of the two objectives using drivers' trajectories collected by running some underlying policy π, the scalarization function f_π defined in Section III-A can also be applied to the distribution of the cumulative rewards $Z^\pi(s)$ to obtain a single return $SZ^\pi(s;\widetilde{W})=\sum_{i=1}^{2}\tilde{w}_iZ_i^\pi(s)$, which is the weighted-sum of $Z_1^\pi(s)$ and $Z_2^\pi(s)$. $SV^\pi(s;\widetilde{W})$ is then given by

$$SV^\pi(s;\widetilde{W})=E(SZ^\pi(s;\widetilde{W}))=\sum_{i=1}^{2}\tilde{w}_iV_i^\pi(s) \quad (6)$$

We now introduce a novel estimation approach which combines IQN and DQN to jointly learn the value functions $V_1^\pi(s)$ and $V_2^\pi(s)$ of the two objectives, which are the expectations of the two return distributions $Z_1^\pi(s)$ and $Z_2^\pi(s)$. IQN is from the distribution perspective and DQN directly estimates the value functions. In particular, IQN focuses on $Z_1^\pi(s)$ and $Z_2^\pi(s)$ while DQN pays attention to the weighted sum of the two value functions SV^π. By utilizing the distributional information and capturing the stochasticity, the joint training process has a much more robust performance in practice than simply estimating $V_1^\pi(s)$ and $V_2^\pi(s)$ using DQN. Abusing the notation, we use V_i and Z_i from now on to represent the value function and the corresponding distribution for the i-th objective when the underlying policy π is provided.

Individual IQN loss for each objective.

In this paper, we model the distribution of Z_i by a weighted mixture of N Diracs [21]:

$$Z_{i,q,\hat{\tau}}(s):\overset{D}{=}\sum_{j=0}^{N-1}(\tau_{j+1}-\tau_j)\delta_{q_{ij}}(s) \text{ for } i\in\{1,2\}, \quad (7)$$

where δ_z denote a Dirac at $z\in R$, $\tau_1,\ldots,\tau_{N-1}$ represent $N-1$ adjustable fractions satisfying $\tau_{j-1}<\tau_j$ with $\tau_0=0$ and $\tau_N=1$. For simplicity, we denote $\hat{\tau}_j=\frac{\tau_j+\tau_{j+1}}{2}$, and the optimal corresponding quantile values q_{ij} is given by $q_{ij}=F_{Z_i}^{-1}(\hat{\tau}_j)$ where $F_{Z_i}^{-1},i=1,2$ is the inverse function of cumulative distribution function $F_{Z_i}(z)=Pr(Z_i<z)$. In this paper, we employ IQN to train the quantile functions.

We follow the main architecture of CVNet to learn the state embedding $\psi:\mathcal{S}\to R^d$, which is shared by the two objectives. We also compute the embedding of quantile level τ, denoted by $\phi_i(\tau)\in R^d$ for each i, and its j-th dimension $\phi_{i,j}(\tau)$ is parametrized as follows,

$$\phi_{i,j}(\tau):=\text{ReLU}(\sum_{k=0}^{n-1}\cos(k\pi\tau)w_{ijk}+b_{ij}), \quad (8)$$

where w_{ijk} and b_{ij} are network parameters. Now we can build the quantile approximation $F_{Z_i,\theta_i}(\tau)\approx\mathcal{D}(\psi(s;\xi)\odot\phi_i(\tau;\eta_i))$ where $\mathcal{D}:R^d\to R$ is a fully-connected layer and the input is an element-wise (Hadamard) product of $\psi(s;\xi)$ and $\phi_i(\tau;\eta_i)$. $\theta_i=(\xi,\eta_i)$ contains all the parameters to be learned. Let Z_i and Z_i' be the random variables of the total return at s and s' for the i-th objective, the weighted temporal difference (TD) error for quantile locations τ and τ' at step t is defined by

$$\delta_{i,\tau,\tau'}^t=\hat{R}_{it}+\gamma^{\Delta_t}F_{Z_i',\theta_i}^{-1}(\tau)-F_{Z_i,\theta_i}^{-1}(\tau'),\forall i=1,2 \quad (9)$$

The quantile value networks can be trained by minimizing the Huber quantile regression loss [6],

$$\rho_\tau^\kappa(\delta_{i,\tau,\tau'})=|\tau-\mathbb{1}\{\delta_{i,\tau,\tau'}<0\}|\frac{\mathcal{L}_\kappa(\delta_{i,\tau,\tau'})}{\kappa}$$

where $\mathbb{1}$ is the indicator function and $\mathcal{L}_\kappa$ is the Huber loss,

$$\mathcal{L}_\kappa(x)=\begin{cases}\frac{1}{2}x^2, & \text{if } x\le\kappa;\\ \kappa(|x|-\frac{1}{2}\kappa), & \text{otherwise.}\end{cases} \quad (10)$$

The final IQN loss for the i-th objective is given by

$$L_i(s_t,\hat{R}_{it},s_{t+\Delta_t};\theta_i)=\frac{1}{N}\sum_{k=0}^{N-1}\sum_{j=0}^{N-1}\rho_{\hat{\tau}_k}^\kappa(\delta_{i,\hat{\tau}_k,\hat{\tau}_j'}^t), \quad (11)$$

for $i\in\{1,2\}$, where $\hat{\tau}_k=\frac{\tau_k+\tau_{k+1}}{2}$ and $\hat{\tau}_j'=\frac{\tau_j'+\tau_{j+1}'}{2}$. $\tau_k,\tau_j'\sim\mathcal{U}([0,1])$ are iid samples used to estimate the loss.

DQN loss of the scalarized value function SV.

The equation (6) shows that SV can be factorized as the weighted sum of V_i. We would like the learning of distributional RL to exploit this structure directly. To that end, we make use of the observation that the expectation of a random variable can be expressed as an integral of the quantiles, e.g., $V_i = \int_0^1 F_{Z_i}^{-1}(\tau)d\tau$. Applying it to (6) and using the Monte Carlo estimate, we obtain $SV(s;\widetilde{W}) = \sum_{i=1}^{2}\tilde{w}_i V_i(s) \approx \sum_{i=1}^{2}\tilde{w}_i \frac{1}{N}\sum_{j=1}^{N} F_{Z_i(s)}^{-1}(\tau_j)$ where N is the Monte Carlo sample size and τ_j's are the uniformly sampled quantile points from above. The temporal difference (TD) error for SV is defined by

$$\begin{aligned} TD_{SV}^t &= f_\pi(\hat{R}_t;\widetilde{W}) + \gamma^{\Delta_t} SV(s_{t+\Delta_t};\widetilde{W}) - SV(s_t;\widetilde{W}) \\ &= f_\pi(\hat{R}_t;\widetilde{W}) + \gamma^{\Delta_t}\sum_{i=1}^{2}\tilde{w}_i\frac{1}{N}\sum_{k=1}^{N} F_{Z_i(s_{t+\Delta_t})}^{-1}(\tau_k) \\ &\quad - \sum_{i=1}^{2}\tilde{w}_i\frac{1}{N}\sum_{j=1}^{N} F_{Z_i(s_t)}^{-1}(\tau_j), \end{aligned} \tag{12}$$

based on which we can build the loss function $L_{SV} = \frac{1}{2}TD_{SV}^2$ for the DQN part.

Joint training of IQN and DQN.

The final joint training objective regarding Z_1, Z_2 and SV is thus given by

$$\begin{aligned} L(\theta) &= \lambda_1\{L_1(\theta_1) + L_2(\theta_2)\} + \lambda_2 L_{SV}(\theta) + \lambda_3\mathcal{R}(\theta) \\ &= \frac{\lambda_1}{N}\sum_{i=1}^{2}\sum_{k=0}^{N-1}\sum_{j=0}^{N-1}\rho_{\hat{\tau}_k}^{\kappa}(\delta_{i,\hat{\tau}_k,\hat{\tau}'_j}^t) + \frac{\lambda_2}{2}{TD_{SV}^t}^2 + \lambda_3\mathcal{R}(\theta) \end{aligned} \tag{13}$$

where θ is the concatenation of θ_1 and θ_2. $\lambda_1, \lambda_2, \lambda_3 > 0$ are hyper-parameters. $\mathcal{R}(\theta)$ is an added penalty term to control the global Lipschitz constant in $\psi(s)$. The joint loss given in (13) accounts for both the individual uncertainty and the mutural information of the two objectives. Some empirical comparison between the proposed loss and the way to minimize L_1, L_2 separately is presented later in the experiment section.

C. Planning With Multi-driver Dispatching

The order-dispatching system of ride-hailing platforms is a multi-agent environment with multiple drivers making sequential decisions. The platform keeps assigning passengers to nearby idle drivers within a continuous set of small time intervals. Each ride request cannot be paired with multiple drivers to avoid assignment conflicts. A utility score μ_{jk} is used to indicate the value of matching each driver k to an order j, and the global order dispatching algorithm in each decision window is equivalent to solving a bipartite matching problem as follows:

$$\begin{aligned} &\arg\max_{x_{ij}} \sum_{j=0}^{M}\sum_{k=0}^{N}\mu_{jk}x_{jk}, \\ &\text{s.t. } \sum_{j=0}^{M} x_{jk} \le 1 \quad \forall k; \ \sum_{k=0}^{N} x_{jk} \le 1 \quad \forall j; \\ &\quad x_{jk} = 0 \text{ if } c_{jk} > \epsilon \quad \forall j,k, \end{aligned}$$

where

$$x_{jk} = \begin{cases} 1 & \text{if order } j \text{ is assigned to driver } k; \\ 0 & \text{if order } j \text{ is not assigned to driver } k. \end{cases}$$

which can be solved by the Hungarian algorithm [12]. We denote the *Temporal Difference* between the expected return a driver k accepts order j and that the driver stays idle as $A_i(j,k)$ for the i-th objective. In this case, the utility function μ_{jk} could be computed as below,

$$\mu_{jk} = w_1 A_1(j,k) + w_2 A_2(j,k) + \Omega \cdot U_{jk} \tag{14}$$

where

$$A_i(j,k) = \hat{R}_{i,jk} + \gamma^{k_{jk}} V_i(s_k) - V_i(s_j) \text{ for } i \in \{1,2\} \tag{15}$$

and $\hat{R}_{i,jk} = R_{i,jk}\frac{(\gamma^{k_{jk}}-1)}{k_{ij}(\gamma-1)}$, $i \in \{1,2\}$. $R_{1,jk}$ here denotes the trip fee driver k receives by serving order j, $R_{2,jk}$ is the estimated demand-supply relationship of the location where the trip ends. k_{jk} represents the trip duration, U_{jk} characterizes the user experience, and $\Omega \ge 0$ is the hyper-parameter. In practice, U_{jk} is usually set to be the negative distance between driver k and order j, which aims to reducing the waiting time of the passenger. Our goal is to find the optimal $W^* = (w_1^*, w_2^*)^T$ to maximize the platform metrics of interest, such as TDI, ORR and OCR, which reflect the market equilibrium. In practice, we perform grid search within a wide range of (w_1, w_2) to find the optimal combination. To be specific, we first determine a wide range of w_2 with $w_1^* = \tilde{w}_1$ being fixed and then keep updating w_2 until we find the optimal choice of w_2^* which maximizes averaged TDI, ORR and OCR for some representative days.

IV. Experiments

To provide a deeper insight of the proposed MODRL method, we evaluate it offline on a realistic multi-driver simulation environment using real data collected from a large-scale ride-hailing platform to validate its effectiveness in improving TDI and users' experience. Both demands and supplies are initialized at the beginning of the day, and then evolves following the simulator's transition dynamics (including drivers getting online/offline, driver movement with passengers and idle driver random movement). The simulated environment captures all the essential elements of the online system, with its results highly relevant (less than 2% difference from the real world) and insightful to production deployment.

A. Experiment Setting

We collect demand-supply data from 09/01/2019 to 11/10/2019 in three cities served by the ride-hailing platform, A (balanced supply and demand), B (relatively under-supplied) and C (relatively over-supplied). City B has higher chance of supply shortage than the other two cities. City A is more balanced with more hexagon cells having demand-supply differences close to zero. We split each city area into thousands of non-overlapping hexagonal grids (radius = 700m), and compute the spatio-temporal relationship R_2 as the demand-supply differences in each 10-minute time interval. The first

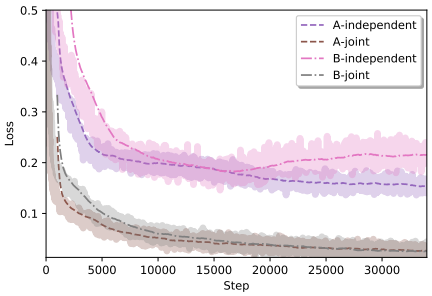

Fig. 1: Loss curves of the joint training described in Section III-B and the separate training for cities A and B

two months of the dataset, from 09/01/2019 to 10/31/2019, were used for model training, while four weekdays and one weekend from the last week were for testing. The daily number of drivers in Cities A, B, C are around 26K, 24K and 11K, that contribute in average 538K, 370K and 166K transitions per day. We assume that all drivers as identically distributed such that the dispatching policy models the population mean.

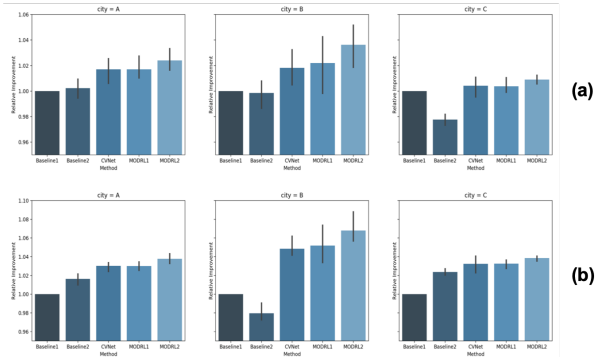

Fig. 2: (a) TDI and (b) ORR improvement of compared methods over baseline across three different cities

We process the raw data to generate an efficient training sample for MOIRL. First, we sort all the charging and idle movements of each driver by time within a daily cycle from 4:00 a.m. to 4:00 a.m. the next day. Small vacancy records between any two trip records that are less than one hour apart are combined together to build a complete idle movement. We add the grid-based demand-supply difference (supply subtracted from demand) at the destination for each transition, the length of which is converted into 10-minute intervals with rounding. Finally, we use a neural network with a CVNet-based architecture to learn the probability of each state-action pair under policy π. Without loss of generality, we fix $\tilde{w}_1 = 1$ in training and iteratively update w_2 until convergence is reached.

We give a brief description below of the policies that we compare in this paper. The variations come from the way the utility scores μ_{jk} in the bipartite graph are computed: (1) **Baseline 1**, μ_{jk} is the negative distance between driver k and order j; (2) **Baseline 2**, μ_{jk} is the immediate earning by assigning order j to driver k; (3) **CVNet** [16], awhich only optimizes the expected future income of each agent (driver) individually ($w_2 = 0$); (4) **MODRL1**, using weights $\widetilde{W}$ obtained from MOIRL in planning; (5) **MODRL2**, using optimal weights W^* obtained by grid search in planning. According to the empirical results, the background weight $\tilde{w}_2$ is close to zero for three cities which ignores the effect of demand-supply differences while a significantly nonzero w_2^* (City A: 1, City B: 0.7, City C: 0.5) is determined via on-line searching to improve the dispatching efficiency.

To be fair, we fix $\gamma = 0.94$ for all the compared methods, which is in use by the real-world dispatching system. Since $\hat{R}_{2,jk}$ is unknown at the decision making point, we take the average of the same weekday or weekend from the four training weeks as an prediction of $\hat{R}_{2,jk}$. MODRL follows most of the main CVNet setting in [16], and use $\lambda_1 = \lambda_2 = 1, \lambda_3 = 0.001$ in (13), which achieves the greatest robustness in model training. Figure 1 shows that the joint training approach given by (13) achieves a faster convergence rate and more stable loss control than individually learning the two return distributions L_1 and L_2.

	TDI	ORR	OCR
City A	0.79 ± 0.36	0.82 ± 0.25	0.72 ± 0.29
City B	1.47 ± 0.53	1.66 ± 0.45	1.34 ± 0.38
City C	0.57 ± 0.25	0.60 ± 0.22	0.53 ± 0.17

TABLE I: (%) improvement of MODRL2 over MODRL1

B. Dispatching Results

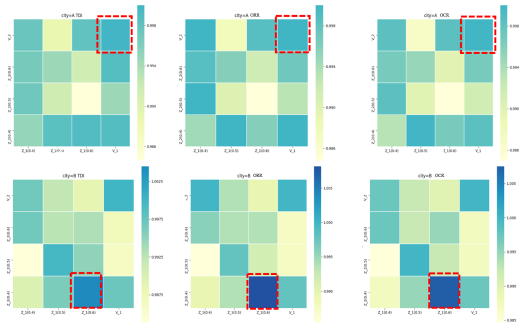

Fig. 3: TDI, ORR, and OCR improvement of Z1 and Z2 at 0.4, 0.5, and 0.6 quantile levels over V1 and V2. The best combinations are marked by red.

Figure 2(b), (c) plot the simulation results averaged over five testing days. For all the three metrics, the results by the five compared methods are standardized with respect to Baseline1. We notice that, MODRL1 achieves close or slightly better results than CVNet across cities and days since MODRL1 pays very limited attention to the supply-demand equilibrium by using a small $\tilde{w}_2$ in planning. MODRL2 significantly outperforms MODRL1 in all three cities, especially in City B. As mentioned, City B suffers from more severe supply-demand mismatch (higher proportions of non-zero demand-supply differences across all spatio-temporal hexagons) and

thus benefits more from the MODRL2 policy, which helps allocate supply resources from a more global view to improve future supply-demand equilibrium. A detailed comparison between MODRL1 and MODRL2 are presented in Table I.

We notice in the results that MODRL2 maintains a consistent improvement over the second best policy MODRL1 across all three experimental cities. Compared to the SOTA order dispatching algorithm CVNet, MODRL2 can achieve up to 2% improvement, which in practice means tens of million dollars increase in profit per year for a large-scale ride-sharing company such as Uber and DiDi. As Table I shows, although the (%) improvement varies across days which is largely due to the supply-demand relationship of a certain day, a positive increase in all three measurements is ensured. Among the three measurements, ORR always achieves the largest improvement when replacing MODRL1 with MODRL2, which verifies our hypothesis that MODRL2 tries to optimize the supply-demand equilibrium and subsequently increases the probability a customer request being quickly answered.

All results provided above is entirely based on the value functions, e.g. the expectations of two return distributions. We also explore the effects of taking 'risk-seeking' policies by replacing V_1, V_2 in (15) with quantile values of Z_1, Z_2 at different levels.

Figure 3 visualizes TDI, OAR and OFR improvement of different $Z_i(\tau)$ and V_i combinations over the baseline V_1, V_2 results in City A and B. We can clearly see that City A prefers the 'risk-neutral' policy, while City B achieves the optimal platform efficiency by choosing Z_1 at 0.6 quantile level and Z_2 at 0.4 quantile level.

V. Conclusion

This paper proposes a novel two-objective *SMDP* formulation for the ride-hailing order dispatching problem. We employ a joint RL algorithm to simultaneously learn distributions over two returns and the expectation of their weighted sum. We present empirical results to demonstrate the advantage of our methods over prior order dispatching methods and the effects of quantile values on platform efficiency.

Our proposed approach is still two-stage, while a more ideal way is to combine learning and planning via an end-to-end model. We would also like to see the application of the proposed algorithm to other real-world matching problems. We leave these ideas as potential future directions.

References

[1] A. Boularias, J. Kober, and J. Peters, "Relative entropy inverse reinforcement learning," in *Proceedings of the Fourteenth International Conference on Artificial Intelligence and Statistics*, 2011, pp. 182–189.

[2] B. Chen and H. H. Cheng, "A review of the applications of agent technology in traffic and transportation systems," *IEEE Transactions on intelligent transportation systems*, vol. 11, no. 2, pp. 485–497, 2010.

[3] W. Dabney, G. Ostrovski, D. Silver, and R. Munos, "Implicit quantile networks for distributional reinforcement learning," *arXiv preprint arXiv:1806.06923*, 2018.

[4] M. Dudík and R. E. Schapire, "Maximum entropy distribution estimation with generalized regularization," in *International Conference on Computational Learning Theory*. Springer, 2006, pp. 123–138.

[5] Z. Gábor, Z. Kalmár, and C. Szepesvári, "Multi-criteria reinforcement learning." in *ICML*, vol. 98, 1998, pp. 197–205.

[6] P. J. Huber, "Robust estimation of a location parameter," in *Breakthroughs in statistics*. Springer, 1992, pp. 492–518.

[7] B. Li, D. Zhang, L. Sun, C. Chen, S. Li, G. Qi, and Q. Yang, "Hunting or waiting? discovering passenger-finding strategies from a large-scale real-world taxi dataset," in *2011 IEEE International Conference on Pervasive Computing and Communications Workshops (PERCOM Workshops)*. IEEE, 2011, pp. 63–68.

[8] Z. Liao, "Real-time taxi dispatching using global positioning systems," *Communications of the ACM*, vol. 46, no. 5, pp. 81–83, 2003.

[9] M. Maciejewski, J. Bischoff, and K. Nagel, "An assignment-based approach to efficient real-time city-scale taxi dispatching," *IEEE Intelligent Systems*, vol. 31, no. 1, pp. 68–77, 2016.

[10] S. Mannor and N. Shimkin, "A geometric approach to multi-criterion reinforcement learning," *Journal of machine learning research*, vol. 5, no. Apr, pp. 325–360, 2004.

[11] F. Miao, S. Han, S. Lin, J. A. Stankovic, D. Zhang, S. Munir, H. Huang, T. He, and G. J. Pappas, "Taxi dispatch with real-time sensing data in metropolitan areas: A receding horizon control approach," *IEEE Transactions on Automation Science and Engineering*, vol. 13, no. 2, pp. 463–478, 2016.

[12] J. Munkres, "Algorithms for the assignment and transportation problems," *Journal of the society for industrial and applied mathematics*, vol. 5, no. 1, pp. 32–38, 1957.

[13] A. Y. Ng, S. J. Russell *et al.*, "Algorithms for inverse reinforcement learning." in *Icml*, vol. 1, 2000, p. 2.

[14] J. Peters, K. Mulling, and Y. Altun, "Relative entropy policy search," in *Twenty-Fourth AAAI Conference on Artificial Intelligence*, 2010.

[15] K. T. Seow, N. H. Dang, and D.-H. Lee, "A collaborative multiagent taxi-dispatch system," *IEEE Transactions on Automation science and engineering*, vol. 7, no. 3, pp. 607–616, 2009.

[16] X. Tang, Z. T. Qin, F. Zhang, Z. Wang, Z. Xu, Y. Ma, H. Zhu, and J. Ye, "A deep value-network based approach for multi-driver order dispatching," in *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. ACM, 2019, pp. 1780–1790.

[17] P. Vamplew, R. Dazeley, C. Foale, S. Firmin, and J. Mummery, "Human-aligned artificial intelligence is a multiobjective problem," *Ethics and Information Technology*, vol. 20, no. 1, pp. 27–40, 2018.

[18] Z. Wang, Z. Qin, X. Tang, J. Ye, and H. Zhu, "Deep reinforcement learning with knowledge transfer for online rides order dispatching," in *IEEE International Conference on Data Mining*. IEEE, 2018.

[19] Z. Xu, Z. Li, Q. Guan, D. Zhang, W. Ke, Q. Li, J. Nan, C. Liu, W. Bian, and J. Ye, "Large-scale order dispatch in on-demand ride-sharing platforms: a learning and planning approach," in *Proceedings of the 24rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. ACM, 2018.

[20] Z. Xu, Z. Li, Q. Guan, D. Zhang, Q. Li, J. Nan, C. Liu, W. Bian, and J. Ye, "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*. ACM, 2018, pp. 905–913.

[21] D. Yang, L. Zhao, Z. Lin, T. Qin, J. Bian, and T.-Y. Liu, "Fully parameterized quantile function for distributional reinforcement learning," in *Advances in Neural Information Processing Systems*, 2019, pp. 6190–6199.

[22] D. Zhang, L. Sun, B. Li, C. Chen, G. Pan, S. Li, and Z. Wu, "Understanding taxi service strategies from taxi gps traces," *IEEE Transactions on Intelligent Transportation Systems*, vol. 16, no. 1, pp. 123–135, 2014.

[23] L. Zhang, T. Hu, Y. Min, G. Wu, J. Zhang, P. Feng, P. Gong, and J. Ye, "A taxi order dispatch model based on combinatorial optimization," in *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. ACM, 2017, pp. 2151–2159.

[24] R. Zhang and M. Pavone, "Control of robotic mobility-on-demand systems: a queueing-theoretical perspective," *The International Journal of Robotics Research*, vol. 35, no. 1-3, pp. 186–203, 2016.

[25] B. D. Ziebart, J. A. Bagnell, and A. K. Dey, "Modeling interaction via the principle of maximum causal entropy," 2010.

[26] B. D. Ziebart, A. Maas, J. A. Bagnell, and A. K. Dey, "Maximum entropy inverse reinforcement learning," 2008.

[27] Q. Zou, G. Xue, Y. Luo, J. Yu, and H. Zhu, "A novel taxi dispatch system for smart city," in *International Conference on Distributed, Ambient, and Pervasive Interactions*. Springer, 2013, pp. 326–335.