Reward Design for Driver Repositioning Using Multi-Agent Reinforcement Learning

Zhenyu Shou[a], Xuan Di[a,b,*]

[a]*Department of Civil Engineering and Engineering Mechanics, Columbia University*
[b]*Data Science Institute, Columbia University*

Abstract

A large portion of passenger requests is reportedly unserviced, partially due to vacant for-hire drivers' cruising behavior during the passenger seeking process. This paper aims to model the multi-driver repositioning task through a mean field multi-agent reinforcement learning (MARL) approach that captures competition among multiple agents. Because the direct application of MARL to the multi-driver system under a given reward mechanism will likely yield a suboptimal equilibrium due to the selfishness of drivers, this study proposes an reward design scheme with which a more desired equilibrium can be reached. To effectively solve the bilevel optimization problem with upper level as the reward design and the lower level as a multi-agent system, a Bayesian optimization (BO) algorithm is adopted to speed up the learning process. We then apply the bilevel optimization model to two case studies, namely, e-hailing driver repositioning under service charge and multiclass taxi driver repositioning under NYC congestion pricing. In the first case study, the model is validated by the agreement between the derived optimal control from BO and that from an analytical solution. With a simple piecewise linear service charge, the objective of the e-hailing platform can be increased by 4.0%. In the second case study, an optimal toll charge of $5.1 is solved using BO, which improves the objective of city planners by 7.9%, compared to that without any toll charge. Under this optimal toll charge, the number of taxis in the NYC central business district is decreased, indicating a better traffic condition, without substantially increasing the crowdedness of the subway system.

Keywords: Mean Field Multi-Agent Reinforcement Learning, Reward Design, Bayesian Optimization

1. Introduction

The emergence of transportation network companies (TNCs) or e-hailing platforms (such as Didi and Uber) has revolutionizsed the traditional taxi market and provided commuters a flexible-route door-to-door mobility service. Nonetheless, it is reported that a large portion of the passenger requests remain unserviced because of the imbalance between demand (i.e., passenger requests) and supply (i.e., available drivers) (Lin et al., 2018), resulting in long cruising trips for taxi drivers to find the next passenger (Powell et al., 2011; Di and Ban, 2019). Such cruising behavior has negative impact on urban economy by not only decreasing drivers' income but also generating additional vehicle miles traveled. Thus, repositioning available drivers to potential locations with near-future high demand, i.e., to balance supply and demand, becomes the key challenge faced by the taxi and for-hire market, including e-hailing platforms. Leveraging cutting edge machine learning techniques, this paper aims to improve the efficiency of the taxi and for-hire market.

The essence of the repositioning task is to provide recommendations to idle drivers on where to find the next passenger. Some recommender systems have been proposed for drivers (Ge et al., 2010; Hwang et al., 2015; Yuan et al., 2011; Qu et al., 2014). These studies extracted useful aggregated statistical quantities such as taxi demand and travel time from historical data and recommended a next cruising location (Ge et al., 2010), a sequence of potential pickup points (Hwang et al., 2015), a driving route (Qu et al., 2014), or a route and a location (Yuan et al., 2011).

*Corresponding author. Tel.: +1 212 853 0435;
Email address: sharon.di@columbia.edu (Xuan Di)

Although the aforementioned studies provide effective recommendations of the next cruising route or location to drivers at the immediate next step, they are nearsighted and fall short of capturing the future long-run payoffs. To capture the effect of future rewards on the recommendation at the immediate next step, various Markov decision process (MDP) based approaches have been proposed to model idle drivers' passenger searching process (Rong et al., 2016; Zhou et al., 2018; Verma et al., 2017; Gao et al., 2018; Yu et al., 2019; Shou et al., 2020). In an MDP with a single agent, a driver is the agent who makes decisions of where to go next. The dynamic environment is determined by the stochastic passenger requests and all other traffic information including the road network, distribution of drivers, and traffic conditions. Once the agent makes an action in a state, the agent then transits into a new state and receives an immediate reward by following the dynamics of the environment. The agent aims to derive an optimal policy which maximizes her expected cumulative reward. When the dynamic environment is known to the agent, dynamic programming or value iteration can be used to solve the MDP and derive an optimal policy. When the dynamic environment is unknown to the agent, the agent needs to interact with the environment by the trial and error process and gradually learns an optimal policy by some reinforcement learning (RL) algorithms such as Q-learning and temporal difference learning (Sutton and Barto, 1998).

The competition among multiple agents is, however, neglected in the aforementioned MDP models due to their single-agent setting, resulting in overly optimistic optimal policies. In other words, one agent cannot earn the full amount of the expected reward by following the policy derived in the single-agent setting. In a dynamic environment involving a group of agents, multiple agents interact with both the shared environment and other agents. Multi-agent reinforcement learning (MARL) (Buoniu et al., 2010) thus fits naturally well in this multi-agent system (MAS). Recently, MARL has been attracting significant attention due to its success in tackling high dimensional and complicated tasks such as playing the game of Go (Silver et al., 2016, 2017), Poker (Brown and Sandholm, 2018, 2019), Dota 2 (OpenAI, 2018), and StarCraft II (Vinyals et al., 2019).

MARL tasks can be broadly grouped into three categories, namely, fully cooperative, fully competitive, and a mix of the two, depending on different applications (Zhang et al., 2019): (1) In the fully cooperative setting, agents collaborate with each other to optimize a common goal; (2) In the fully competitive setting, agents have competing goals, and the return of agents sums up to zero; (3) The mixed setting is more like a general-sum game where each agent cooperates with some agents while competes with others. For instance, in the video game *Pong*, an agent is expected to be either fully competitive if its goal is to beat its opponent or fully cooperative if its goal is to keep the ball in the game as long as possible (Tampuu et al., 2017). A progression from fully competitive to fully cooperative behavior of agents was also presented in Tampuu et al. (2017) by simply adjusting the reward.

A key challenge arises in MARL when independent agents have no knowledge of other agents, that is, the theoretical convergence guarantee is no longer applicable since the environment is no longer Markovian and stationary (Matignon et al., 2012; Nguyen et al., 2018). To tackle this issue, one way is to exchange some information among agents. In some contexts, agents actually exchange information with their peers through some coordination. For example, in the game of a team of hunters capturing a team of preys, Tan (1993) proposed multiple ways to enable coordination among agents and concluded that the performance of the hunter agents can be better off through some coordination. However, in other contexts such as the driver repositioning system, agents only have access to their own information. Thus, information exchange among agents involves a central controller which collects the information of all agents and disseminates it to agents. Agents update their value functions and policies based on the provided information from the central controller and their local observations. This is the centralized learning (i.e., based on global information) and decentralized execution (i.e., based on local observation) paradigm, which has become increasingly popular in recent research (Foerster et al., 2016; Lowe et al., 2017; Lin et al., 2018; Li et al., 2019).

While training is stabilized conditioning on the information of other agents such as joint state and joint action in the centralized training paradigm, scalability becomes a critical issue in MARL because the joint state space and joint action space grow exponentially with the number of agents. To make MARL tractable when a large number of agents coexist, Yang et al. (2018) employed the mean field theory to simplify the interaction among agents. The basic idea is, from the perspective of an agent, to treat other agents as a mean agent. Thus, the complexity of interactions among a large number of agents is substantially eased by reducing the dimension in the Q-value function. The large scale MARL with hundreds of or even thousands of agents becomes solvable. To investigate the large-scale order dispatching problem where thousands of agents are present, Li et al. (2019) adopted a mean field approximation and proposed to take the average response from neighboring agents as a proxy of the

interaction between the agent and other agents.

Recent studies have successfully applied MARL to multi-driver repositioning and large scale order dispatching problems (Lin et al., 2018; Li et al., 2019; Zhou et al., 2019). Different from treating each driver as an agent in previous studies, Jin et al. (2019) treated each spatial grid as a worker agent and each region composed of several spatial grids as a manager agent and adopted hierarchical reinforcement learning to tackle the joint task of order dispatching and fleet management. All these studies rely on an underlying assumption that drivers are willing to cooperate under a specifically crafted reward function. For example, embedding the goal of the platform such as improving the gross merchandise volume (GMV) or the order response rate (ORR) into the reward function of a driver encourages cooperation among drivers. Human-drivers are, however, selfish in nature and will only cooperate if the overall return from cooperation is higher than that from competition. This self-interested behavior is utilized to achieve certain degree of cooperation among agents such as adjusting the reward for each agent. However, when the imposed reward function (Lin et al., 2018; Li et al., 2019; Zhou et al., 2019; Jin et al., 2019) is not aligned with the goal of real drivers (e.g., a real driver's goal can simply be maximizing her monetary return), drivers will not follow the derived optimal policy. Thus, in this work, instead of enforcing a reward function for drivers to cooperate, drivers are regarded as selfish and non-cooperative, and the reward for a driver is simply the monetary return that the driver earns.

Although the approaches in Lin et al. (2018); Li et al. (2019); Zhou et al. (2019); Jin et al. (2019) are efficient under a given reward function, the reached equilibrium is very likely to be a suboptimal from the overall perspective of the system. Congestion pricing is a common way to drive the traffic system performance towards a system optimum in transportation network design problems (Yang and H. Bell, 1998; Zhang and Yang, 2004; Meng and Liu, 2012; Di et al., 2014, 2016, 2018). In this paper, we show that by integrating a reward design mechanism which adjusts the monetary return that a driver earns, a desirable equilibrium can be reached in this intrinsically large-scale non-cooperative system. The desirable equilibrium refers to a Nash equilibrium where each independent and selfish agent's strategy is the best-response to other agents' strategies and will produce better overall performance of the system. Mguni et al. (2019) proposed a two-layer architecture with an incentive designer as the upper layer and a potential game as the lower layer and formulated the incentive designers problem as an optimization problem. In contrast, the MARL problem in our context may not be able to be transformed as a potential game, complicating computation of its equilibrium.

In summary, the major contributions of this paper are as follows: (1) With the lower level as the MAS and the upper level as the reward design, this paper formulates a bilevel optimization problem in which a mean field actor-critic algorithm is developed to solve the MAS and a Bayesian optimization algorithm is adopted to efficiently solve the problem. (2) Instead of intentionally crafting a reward function, which aligns with the goal of the platform but may not reflect the intrinsic reward of real drivers, this paper takes the monetary return of a driver as the reward function. It aims to improve the performance of the platform by adjusting the monetary return that one driver can earn through a reward design mechanism of the platform (e.g., platform service charge and incentives). (3) In the case study of taxi driver repositioning under congestion pricing, a multiclass MARL is developed to capture the intrinsic behavioral difference between yellow taxis and green taxis.

The remainder of the paper is organized as follows. Section (2) introduces the single-agent actor-critic algorithm, which is a stepping stone for MARL. Section (3) presents the mean field multi-agent reinforcement learning algorithm. Section (4) presents a reward design mechanism and formulates a bilevel optimization problem. Section (5) presents the result and validates the effectiveness of the proposed reward design. Section (6) concludes.

2. Single agent reinforcement learning

As a stepping stone, we first introduce the single agent reinforcement learning where only one agent interacts with the environment.

2.1. Problem definition

A Markov decision process (MDP) (Puterman, 1994) is typically specified by a tuple (S, A, R, P, γ), where S denotes the state space, A stands for the allowable actions, $R : S \times A \times S \to \mathbb{R}$ collects rewards, $P : S \times A \times S \to [0, 1]$ denotes a state transition probability from one state to another, and $\gamma \in [0, 1]$ is a discount factor. A general MDP proceeds simply as follows. Starting from the initial state, the

agent specifies an action $a \in A$ whenever the agent is in a state $s \in S$. The agent then transits into a new state $s' \in S$ with probability $P(s'|s,a)$ and observes an immediate reward $r(s,a,s')$ by obeying the dynamics of the environment. Then the process repeats until a terminal state is reached. A policy $\pi : S \times A \to [0,1]$ simply maps from state $s \in S$ to the probability of taking action $a \in A$ in state s, i.e., $\pi(a|s)$. The goal of solving an MDP is to derive an optimal policy π^* so that the agent can maximize her long term expected reward by following the policy. In reinforcement learning problems, the transition probability matrix P is commonly unknown, and the agent learns about P from its interaction with the environment.

Denote $V^\pi(s)$ as the state value, which is the expected cumulative reward that an agent can earn by starting from state s and following a policy π. V^π can be recursively given as (Sutton and Barto, 1998) $V^\pi(s) = \mathbb{E}_{a\sim\pi(\cdot|s),s'\sim P(\cdot|s,a)}\left[r(s,a,s') + \gamma V^\pi(s')\right]$. Denote $Q^\pi(s,a)$ as the state-action value, which is the expected cumulative reward that an agent can earn by starting from state s, taking action a, and following a policy π. Q^π is related with V^π through $Q^\pi(s,a) = \mathbb{E}_{s'\sim P(\cdot|s,a)}\left[r(s,a,s') + \gamma V^\pi(s')\right]$.

The optimal value V can then be written as $V(s) = max_\pi V^\pi(s), \forall s \in S$. The Bellman optimality equation is given as (Sutton and Barto, 1998):

$$V(s) = max_a \mathbb{E}_{s'\sim P(\cdot|s,a)}[r(s,a,s') + \gamma V(s')],$$

where the optimal state-action value is $Q(s,a) = \mathbb{E}_{s'\sim P(\cdot|s,a)}[r(s,a,s') + \gamma V(s')]$.

Our task is then to derive an optimal policy π^* (i.e., to solve the MDP) with which the agent can optimize its expected cumulative reward.

To demonstrate how to apply MDPs problems to the context of e-hailing driver reposition, we will use examples on a 2-by-2 grid world throughout the paper every time when models are introduced.

Example 2.1. (Single-Agent 2×2). The single-agent driver reposition is presented in Figure (1). We adopt a grid world setup where the index of each grid (denoted as l) is shown at the upper left corner. The taxi icon denotes the driver, and the person icon is the passenger request with the corresponding fare shown above. The time beneath the driver and the passenger request records the current time of the driver and the appearance time of the passenger request, respectively. The dashed line with arrow shows the origin and destination of the passenger request.

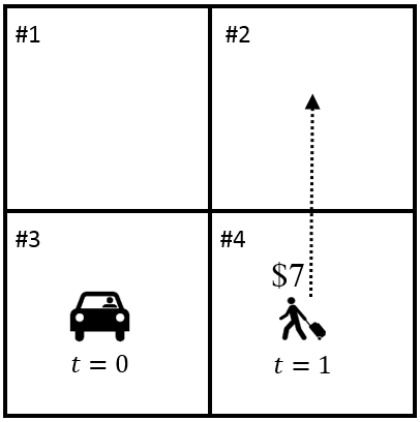


Figure 1: An illustrative example (Single agent)

S. The state of the driver consists of two components, namely, the grid index l and current time t, i.e., $s = (l,t)$. For instance, the current state of the driver is $s = (\#3, 0)$ in this example.

A. The allowable action of the driver is either moving into one of the neighboring grids or staying within the current grid. To be concise, we use the index of grid where the driver chooses to enter as the action. Suppose the driver decides to go rightward in the example, then we can denote $a = \#4$. We further assume it takes the driver one time step to enter grid #4. In other words, the current time of the driver is $t = 1$ when the driver arrives in grid #4.

P. Considering the driver arrives in grid #4 at time $t = 1$, and at the same time a passenger request appears in grid #4 with 80% probability. If this driver is matched to the passenger and picks up the passenger, the driver will transit to the passenger's destination, which is grid #2. Denote the transition time from grid #4 to grid #2 as $\Delta t_{\#4\to\#2}$. We can define the new state $s' = (\#2, 1 + \Delta t_{\#4\to\#2})$.

Then the transition probability from the state s at time 1 to the state s' at time $1 + \Delta t_{\#4 \to \#2}$ is 80%, mathematically, $P(s'|s, a) = 80\%$. If there is no passenger request in grid #4 at time $t = 1$, then the driver ends up in state $s' = (\#4, 1)$. The transition probably becomes $P(s'|s, a) = 20\%$.

R. If we take the fare of the fulfilled passenger request as the reward, $r(s, a, s') = \$7$ in the example. Based on the received reward at this step and the future cumulative reward, the driver chooses an action in the new state s', and the state transition process repeats until a terminal state (i.e., $t = T$ where T is a predefined ending time, say, the end of the driver's work time) is reached. □

2.2. Actor-Critic method

To solve optimal policies, there are two types of methods, namely, value based or critic-only method and policy based or actor-only method. Value based and policy based methods are commonly used terminologies, but from now on we will use critic-only and actor-only methods for the purpose of introducing the actor-critic method.

Critic-only methods aim to output the optimal policy π through optimizing the state-action $Q(s, a)$ or the state value $V(s)$. Actor-only methods directly output an optimal policy π without resorting to stored value functions $Q(s, a)$ or $V(s)$ as an intermediary. Both methods have pros and cons. Critic-only methods enjoy a low variance in the estimate of the state-action value but may lack guarantees on the optimality or near-optimality of the resulting policy if an optimal policy cannot be easily solved from value functions. Actor-only methods work well on continuous and large action spaces but may suffer from high fluctuation in policies (Konda and Tsitsiklis, 2003; Grondman et al., 2012). To overcome the shortcomings of these methods, actor-critic methods are developed to combine strengths of both methods (Konda and Tsitsiklis, 2003).

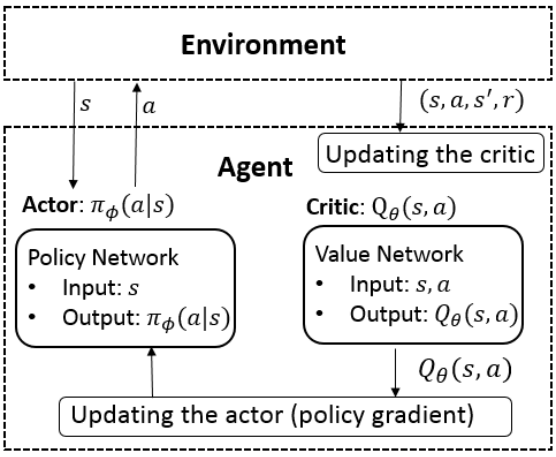


Figure 2: Actor-critic algorithm

Figure (2) presents the architecture of the actor-critic algorithm. One agent, who has an actor and a critic, interacts with the environment. The agent observes its state s from the environment and inputs s to the actor that outputs the policy, i.e., a probability distribution over all possible actions. The agent samples an action a from the probability distribution and takes action a in the environment. Then the agent observes a state transition $s \to s'$ and receives a reward r from the environment. Based on the one-step transition $s \to s'$ as well as action a and reward r, the agent updates its critic. With the updated Q-value $Q_\theta(s, a)$, the agent updates its actor using policy gradient. Now we detail both the critic and the actor, respectively.

Critic. The critic takes as input state s and action a and outputs Q-value $Q(s, a)$. Q-learning is the most commonly used algorithm to update the Q value based on the state transition $s \to s'$ with reward $r(s, a, s')$ and updates the Q-value by

$$Q(s, a) \leftarrow Q(s, a) + \eta[r(s, a, s') + \gamma max_{a'} Q(s', a') - Q(s, a)] \tag{1}$$

where η is the learning rate and $0 < \eta \leq 1$. If η reduces over time properly, the Q-learning update converges (Sutton and Barto, 1998). Equation (1), however, is only applicable to a finite and discrete state and action space. In other words, one needs to maintain a Q table with all possible combinations

of s and a, which is not tractable for a continuous and large state and action space. Therefore we need functional approximation to the original Q-value. Deep neural network, i.e., deep Q network (DQN), is one of the most popular value approximator (Mnih et al., 2015). Denote a deep neural network parameterized by θ as $Q_\theta(s,a)$, to approximate $Q(s,a)$. DQN updates its parameter θ by minimizing the loss

$$\mathcal{L}(\theta) = \mathbb{E}_{s,a,s'}[(\underbrace{r(s,a,s') + \gamma max_{a'} Q_\theta(s',a')}_{\text{target}} - Q_\theta(s,a))^2]. \tag{2}$$

This problem can be solved by the gradient descent method, whose gradient is straightforward to compute as follows: $\nabla_\theta \mathcal{L}(\theta) = \mathbb{E}_{s,a,s'}[-\nabla_\theta Q_\theta(s,a) \times (r(s,a,s') + \gamma max_{a'} Q_\theta(s',a') - Q_\theta(s,a))]$, where the gradient is not taken with respect to the target.

Actor. The actor takes as input state s and outputs a probability distribution on all allowable actions in this state. Similarly to how we use a value network to approximate Q-value, we can also use a deep neural network, i.e., policy network, to approximate the policy π. Denote the policy network parameterized by ϕ as $\pi_\phi(a|s)$. The goal of the actor is to maximize its expected cumulative reward, denoted as $\rho(\pi_\phi) = \sum_{t=0}^{T} \gamma^t r^t$, where r^t is the reward the actor receives at time t. To solve the optimal policy of the actor requires us to know its gradient. The gradient of the policy is complicated to solve and is given as (Sutton et al., 1999)

$$\nabla_\phi \rho(\pi_\phi) = \mathbb{E}_{s,a}[(\underbrace{Q^{\pi_\phi}(s,a) - b^{\pi_\phi}(s)}_{\text{advantage}}) \nabla_\phi log(\pi_\phi(a|s))], \tag{3}$$

where Q^{π_ϕ} denotes the Q-value function following the policy π_ϕ, b^{π_ϕ} is some baseline (e.g., $b^{\pi_\phi} = V^{\pi_\phi}$, i.e., the value function following the policy V^{π_ϕ}), and $Q^{\pi_\phi}(s,a) - b^{\pi_\phi}(s)$ is called the advantage of a taken action a, a measure of the goodness of an action. If it is greater than zero, it means this taken action is generally good, otherwise it may be bad. Naturally, the underlying rationale in computing the policy gradient defined in Equation (3) is to update the policy distribution to concentrate on potentially good action(s). When the chosen action a leads to a positive advantage, i.e., $Q^{\pi_\phi}(s,a) - b^{\pi_\phi}(s) > 0$, the policy is updated towards the direction of favoring action a. When the advantage is negative for action a, the policy is updated in the direction of against action a.

To summarize, in addition to the policy network π_ϕ, the actor-critic algorithm also maintains a value network Q_θ so that the calculation of the gradient of the policy in Equation (3) directly uses the Q-function approximator Q_θ, to ensure stability of policy update. The actor-critic algorithm simultaneously updates critic (by minimizing the loss given in Equation (2)) and the actor (by the gradient given in Equation (3)) as more samples are fed in.

3. Multi-agent reinforcement learning

To tackle a real-world problem with multiple agents, the aforementioned single agent reinforcement learning falls short of capturing the coupling effects or the competition among multiple agents. In this section, we introduce a mean field multi-agent reinforcement learning approach to model the multi-driver repositioning task.

3.1. Problem definition

The multi-agent problem is modeled as a partially observable Markov decision process (POMDP) (Littman, 1994), defined by a tuple $(S, O_1, O_2, \cdots, O_N, A_1, A_2, \cdots, A_N, P, R_1, R_2, \cdots, R_N, N, \gamma)$, where N is the number of agents and S is the environment state space. Environment state $\mathbf{s} \in S$ is not fully observable. Instead, agent i draws a private observation $o_i \in O_i$ which is correlated with $\mathbf{s}$. O_i is the observation space of agent i, yielding a joint observation space $O = O_1 \times O_2, \times \cdots \times O_N$, A_i is the action space of agent $i \in \{1, 2, \cdots, N\}$, yielding a joint action space $A = A_1 \times A_2 \times \cdots \times A_N$, $P : S \times A \times S \to [0,1]$ is the state transition probability, $R_i : S \times A \times S \to \mathbb{R}$ is the reward function for agent i, and γ is the discount factor.

Agent $i \in \{1, 2, \cdots, N\}$ uses a policy $\pi_i : O_i \times A_i \to [0,1]$ to choose actions after drawing observation o_i. After all agents taking actions, the joint action $\mathbf{a}$ triggers a state transition $\mathbf{s} \to \mathbf{s}'$ based on the state transition probability $P(\mathbf{s}'|\mathbf{s}, \mathbf{a})$. Agent i draws a private observation o_i' corresponding to $\mathbf{s}'$ and receives a reward $r_i(\mathbf{s}, \mathbf{a}, \mathbf{s}')$. Agent i aims to maximize its discounted expected cumulative reward by deriving an

optimal policy π_i^* which is the best response to other agents' policies. This process repeats until agents reach their own terminal state.

Due to the existence of other agents, the Q-value function for agent i , i.e., Q_i, is now dependent on the environment state $\mathbf{s} \in S$ and the joint action $\mathbf{a} \in A$ of all agents, i.e,

$$Q_i = Q_i(\mathbf{s}, \mathbf{a}). \tag{4}$$

Similarly, the value function of agent i, i.e., $V_i = V_i(\mathbf{s})$, is dependent on the environment state $\mathbf{s}$.

Subsequently, we will demonstrate how to formulate the multi-driver repositioning problem in MARL, building on the single-agent example developed in the previous section.

Example 3.1. (Multi-Agent 2×2). The multi-agent driver reposition is presented in Figure (3). Same as before, a grid world setup is adopted. Now we have two drivers with their indices shown above the taxi icon and two passenger requests with fare presented above the passenger icon. The time beneath drivers and passenger requests records the current time of the driver and the appearance time of the passenger request, respectively. The dashed line with arrow shows the origin and destination of the passenger request.

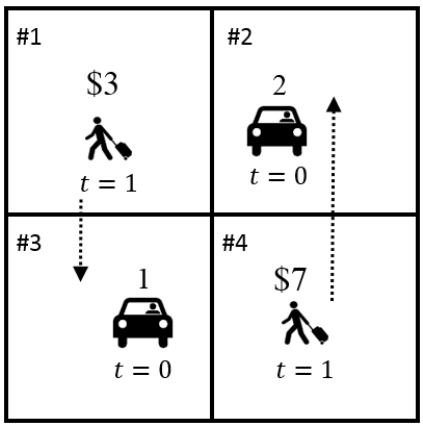


Figure 3: An illustrative example (Multi agent)

N. There are $N = 2$ drivers moving around in the environment. We denote drivers by $\{1, 2\}$.

S. The environmental state consists state information of both drivers. For driver i, her state s_i is composed of her current location l_i (i.e., the grid index based on a grid world setup) and current time t, i.e., $s_i = (l_i, t)$. The joint state of both drivers, i.e., the environment state $\mathbf{s} \in S$, at time t is denoted as $\mathbf{s} = (s_1, s_2)$. In this example, at current time $t = 0$, $\mathbf{s} = ((\#3, 0), (\#2, 0))$.

A. For driver i, her action $a_i \in A_i$ can be any of the five possible actions, i.e., moving into any of her four neighboring grids or staying in the current grid. The same as before, we use the index of grid where the driver chooses to enter as the action. The joint action of both drivers is $\mathbf{a} = (a_1, a_2)$. Assuming driver 1 decides to go rightward (i.e, to enter grid #4) and driver 2 chooses to go leftward (i.e., to enter grid #1), the joint action is $\mathbf{a} = (\#4, \#1)$. We further assume it then takes driver 1 one time step to enter grid #4 and driver 2 one time step to enter grid #1. In other words, after driver 1 arrives in grid #4 and driver 2 arrives in grid #1, the clock ticks one step forward and the current time is now $t = 1$.

P. The joint action $\mathbf{a}$ triggers a state transition $\mathbf{s} \to \mathbf{s}'$ with some probability according to the state transition function, i.e., $P(\mathbf{s}|\mathbf{s}', \mathbf{a})$. Driver 1 gets matched to the passenger request in grid #4 at $t = 1$, loads up the passenger, and drives to the destination of the passenger. Driver 1 then arrives in a new state $s_1' = (\#2, 1 + \Delta t_{\#4 \to \#2})$ where $\Delta t_{\#4 \to \#2}$ is the transition time from grid #4 to grid #2. Driver 2 gets matched to the passenger request in grid #1 at $t = 1$, loads up the passenger, and drives to the destination of the passenger. Driver 2 then arrives in a new state $s_2 = (\#3, 1 + \Delta t_{\#1 \to \#3})$ where $\Delta t_{\#1 \to \#3}$ is the transition time from grid #1 to grid #3. $\mathbf{s}' = ((\#2, 1 + \Delta t_{\#4 \to \#2}), (\#3, 1 + \Delta t_{\#1 \to \#3}))$. In this simple example, $P(\mathbf{s}'|\mathbf{s}, \mathbf{a}) = 1$ due to the deterministic appearance of passenger requests.

R. Along with the state transition, each driver receives a reward, i.e., r_i. The reward function r_i for each agent $i \in \{1, 2\}$ is simply the fare of the fulfilled passenger request, i.e., $r_1 = \$7$ and $r_2 = \$3$. □

This example will be revisited later in this section to illustrate the algorithm.

3.2. Techniques to simplify the Q-value function

The dependency of the Q-value of an agent i on other agents' states and actions, as shown in Equation (4), however, introduces prohibitively high difficulties in learning the optimal Q-value. The main reasons are two-fold. First, although each agent draws its private observation o_i from the environment state $\mathbf{s}$, $\mathbf{s}$ cannot be observed by any agent, i.e., $\mathbf{s}$ is unknown. Second, one agent does not observe the actual actions taken by all agents, i.e., $\mathbf{a}$ is unknown.

To make the Q-value of an agent in the multi-agent system tractable, the dependency of the Q-value on the environment state $\mathbf{s}$ and joint action $\mathbf{a}$ needs to be simplified. A very natural approach, inspired by the single-agent setting, is independent learning where each agent i only has information about its own observation o_i and action a_i but has no information about other agents. Thus, the Q-value function of agent i is reduced to

$$Q_i = Q_i(o_i, a_i). \tag{5}$$

In other words, private observations and joint action of other agents are not used by agent i. After all agents choosing actions, the joint action $\mathbf{a}$ triggers a state transition. Agent i then draws a new private observation o_i' and receives a reward r_i.

The independent learning algorithm, although is intuitive and simple, can be unstable and hard to reach convergence since the environment is no longer Markovian and stationary due to the appearance of other agents (Matignon et al., 2012).

3.2.1. Centralized training and decentralized execution

To make the training more stable and ensure convergence, we employ the centralized training and decentralized execution paradigm (Foerster et al., 2016; Lowe et al., 2017; Lin et al., 2018; Li et al., 2019). In this paradigm, to train the policy of agents, we assume these agents know the global information such as the joint observation and/or joint action. In other words, in addition to observation o_i and action a_i, agent i also has access to the observations and/or actions of other agents during training. While in the execution phase, decentralized testing or execution is implemented, meaning they would not have access to the global information anymore. To realize this paradigm, the aforementioned actor-critic algorithm naturally fits in, because we can apply global information to the critic, i.e., joint observation and joint action in Q_i, in the training phase, while feeding local information to the actor, i.e., o_i in π_i, in the execution phase. Decentralized execution becomes possible because only actors are used in execution.

Then the Q-value function of agent i becomes

$$Q_i = Q_i(o_i, o_{-i}, a_i, a_{-i}), \tag{6}$$

where $o_{-i} = (o_1, \cdots, o_{i-1}, o_{i+1}, \cdots, o_N)$ and $a_{-i} = (a_1, \cdots, a_{i-1}, a_{i+1}, \cdots, a_N)$ denote the joint observation and joint action of all agents except agent i, respectively.

In the context of e-hailing driver repositioning, considering the definition of the action, which is the index of the grid where the driver chooses to enter, the Q-value function of driver i, i.e., Q_i, does not depend on the joint observation of other drivers, i.e., o_{-i}. Explanations are as follows. When driver i chooses action $a_i = l$ based on its observation o_i, driver i then enters grid l. At the same time, other drivers also enter some grid based on their joint action a_{-i} regardless of their joint observation o_{-i}. The Q-value function of driver i only depends on the current distribution of drivers, which has been determined by their joint action a_{-i}. Therefore it is the joint action a_{-i} which affects Q_i. The Q-value function is thus further reduced to

$$Q_i = Q_i(o_i, a_i, a_{-i}). \tag{7}$$

3.2.2. Mean field approximation

The centralized training and decentralized execution paradigm, however, can easily become intractable due to the exponential increase in the joint action space with the increasing number of agents. For example, the size of the joint action space easily blows up for N agents with $|A|$ possible actions (i.e., $|A|^N$ possibilities). To simplify the interaction among agents, we adopt the mean field approximation. The basic idea of the mean field approximation is to simplify the complicated interaction between one agent and all other agents by a pairwise interaction between the agent and a virtual mean agent which is formed by the neighboring agents of the agent. Thus, the complexity of interactions among a large number of agents is substantially eased by reducing the dimension in the input of the Q-value function. Therefore the large scale MARL with hundreds of or even thousands of agents becomes solvable.

To be more precise, we provide brief explanations that lead to the applicability of the mean field approximation in MARL as described in Yang et al. (2018). First, from the perspective of agent i, the multi-agent effect or competition effect mainly comes from its neighboring agents, i.e., $Q_i(o_i, a_i, a_{-i}) \approx \frac{1}{N_i} \sum_{k \in N(i)} Q_i(o_i, a_i, a_k)$, where $N(i)$ denotes the neighboring agents of agent i. However, it is still cumbersome to compute $Q_k, k \in N(i)$ for the neighboring agents of agent i if this number is large. Define a mean action $\bar{a}_i$, which is a proxy of the actions taken by the neighboring agents. Accordingly, $Q_i(o_i, a_i, a_{-i})$ can be further simplified to $Q_i(o_i, a_i, \bar{a}_i)$ when Taylor expansion is applied, which is

$$Q_i \approx \frac{1}{N_i} \sum_{k \in N(i)} Q_i(o_i, a_i, a_k) \approx Q_i(o_i, a_i, \bar{a}_i). \tag{8}$$

Interested readers can refer to Yang et al. (2018) for a detailed explanation and proof.

Example 3.2. (Multi-Agent 2×2). The mean action $\bar{a}_i$ of the neighboring drivers of driver i is defined as the demand to supply ratio in the grid where driver i is entering. Assuming both drivers choose action #4, i.e., $a_1 = a_2 = \#4$ in the multi-agent 2×2 example shown in Figure (3), there are 2 drivers and 1 passenger request in grid #4 after both drivers enter grid #4. The mean action for both drivers is thus $\bar{a}_1 = \bar{a}_2 = \frac{1}{2} = 0.5$. This definition of mean action captures the level of competition in a grid. A larger mean action $\bar{a}_i$ denotes a higher demand to supply ratio and lower level of competition, and vice versa. □

3.3. Mean field actor-critic algorithm

As previously mentioned, each agent $i \in \{1, 2, \cdots, N\}$ maintains a policy network π_i (i.e., the actor) and a Q-value network Q_i (i.e., the critic). For a real-world multi-agent task, there are typically hundreds of or even thousands of agents, indicating that maintaining two deep neural networks (i.e., one for the actor and one for the critic) per agent is not computationally tractable. Considering that for a class of multi-agent tasks where anonymous agents share the same state space, action space, and reward function, agents are thus homogeneous. The multi-agent task can then be largely simplified by sharing both the actor and the critic among drivers, i.e., $Q_1 = Q_2 = \cdots = Q_N = Q$ and $\pi_1 = \pi_2 = \cdots \pi_N = \pi$.

After adopting the mean field approximation, the loss function for the critic, which was presented in Equation (2) for the single-agent setting, now becomes

$$\mathcal{L}(\theta) = \mathbb{E}_{o_i, a_i, o_i'}(r(o_i, a_i, o_i') + \gamma max_{a_i'} \mathbb{E}_{\bar{a}_i'}[Q_\theta(o_i', a_i', \bar{a}_i')] - Q_\theta(o_i, a_i, \bar{a}_i))^2. \tag{9}$$

The only difference is the incorporation of the mean action $\bar{a}$ into the Q-value function approximation. Similarly, the gradient of the policy, which was presented in Equation (3) for single-agent setting, is now

$$\nabla_\phi \rho(\pi_\phi) = \mathbb{E}_{o_i, a_i}[(\mathbb{E}_{\bar{a}}[Q_\theta(o_i, a_i, \bar{a}_i)] - V(o_i)) \nabla_\phi log(\pi_\phi(a_i|o_i))]. \tag{10}$$

Algorithm 1 Mean field actor-critic algorithm

Initialize a deep neural network $Q_\theta(o, a, \bar{a})$, parameterized by θ, for the critic and a deep neural network $\pi_\phi(a|o)$, parameterized by ϕ, for the actor
Initialize $\epsilon = \epsilon_0$, which is the parameter associated with random exploration
Initialize the learning rate $\eta = \eta_0$
repeat
    Randomly initialize a starting grid for all agents, and each agent $i \in \{1, 2, \cdots, N\}$ draws a private observation o_i
    Set $t = 0$
    repeat
        Sample a value x from a uniform distribution which is defined on $[0, 1]$
        if $x < \epsilon$ **then**
            Select an action a_i from the allowable action space randomly for all available agents
        else
            Select an action a_i greedily according to the policy π for all available agents
        end if
        Each available agent takes its action a_i, observes a reward r_i and draws a new observation o_i'
        $t \leftarrow t + 1$
    until $t = T$
    Update the critic Q by minimizing the loss defined Equation (9)
    Update the actor π using the gradient defined in Equation (10)
    Decrease the exploration parameter ϵ
    Decrease the learning rate η
until the algorithm converges
Return the optimal policy π

Example 3.3. (Multi-Agent 2×2). Now we apply the mean field actor-critic algorithm to the multi-driver example shown in Figure (3). Figure (4) presents the architecture of the mean field actor-critic algorithm particularly for the context of multi-driver repositioning. Homogeneous agents, who share a common actor and a common critic, interact with the environment. The shared actor is a multilayer perceptron with 32 neurons in its hidden layer and takes as input observation o_i and outputs a five dimensional vector denoting the probability distribution of taking five actions. Similarly, the shared critic takes as input $(o_i, a_i, \bar{a}_i)$ and outputs the Q-value. During training, agent $i \in \{1, 2, \cdots, N\}$ draws its private observation o_i from the environment and inputs o_i to the actor which outputs a probability distribution over actions. Agent i samples an action a_i from the probability distribution and takes the sampled action in the environment. Joint action of all agents $\mathbf{a}$ triggers a state transition $\mathbf{s} \to \mathbf{s}'$ in the environment. Agent i then observes the mean action $\bar{a}_i$, draws a new observation o_i', and receives a reward r_i from the environment. The agent then uses $(o_i, a_i, o_i', r_i, \bar{a}_i)$ to update the shared critic by minimizing the loss presented in Equation (9). Based on the advantage calculated from the critic, agent i updates the shared actor using the gradient presented in Equation (10).

The aforementioned training process is centralized because the mean action used in the critic is actually some global information. During execution, agents only need to use the updated actor, which only takes as input the local information, i.e., the private observation. In other words, the shared critic is not used in execution.

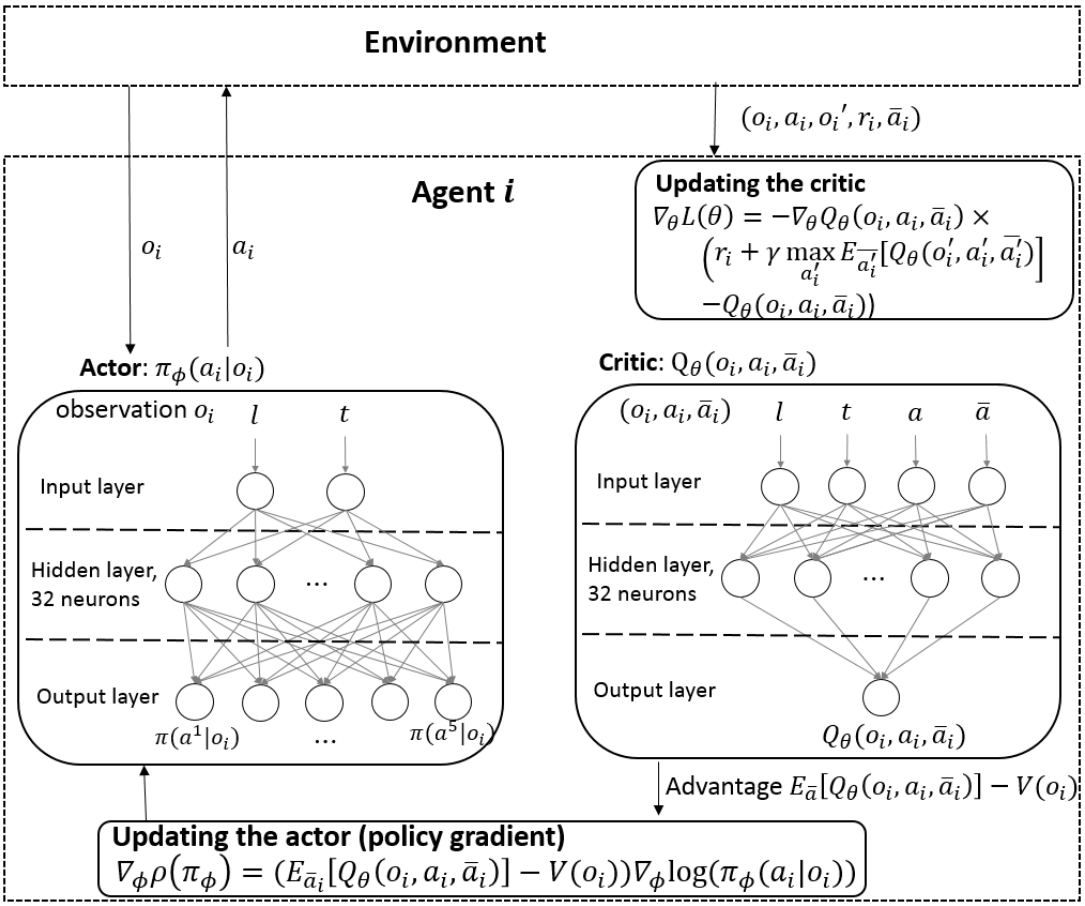
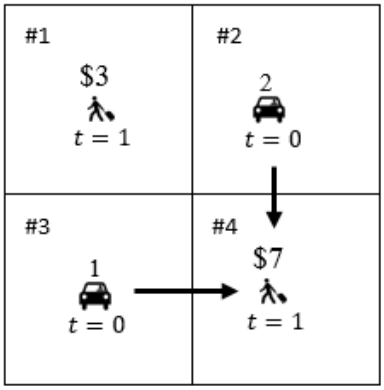


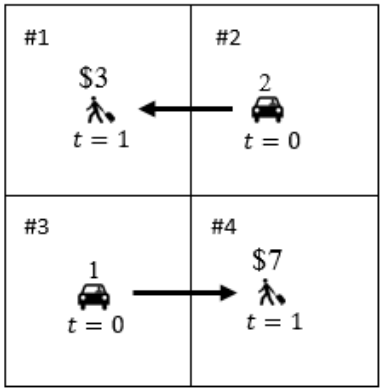
Figure 4: Mean field actor-critic algorithm for multi-driver repositioning

The derived Q values corresponding to four scenarios of interest are presented in Figure (5). In Figure (5a), when both drivers choose action #4, the observed mean action for both of them is the ratio of demand to supply, i.e., $\bar{a}_1 = \bar{a}_2 = \frac{1 \text{ passenger request}}{2 \text{ drivers}} = 0.5$. The resulting expected value for both drivers is \$3.5, i.e., $Q(o_1, a_1, \bar{a}_1) = Q(o_2, a_2, \bar{a}_2) = 3.5$, because both of them have an equal probability $\frac{1 \text{ driver}}{2 \text{ drivers}} = 50\%$ to take the passenger request with \$7. Similarly, the observed mean actions and resulting Q values can be explained in other scenarios. The Q-value bimatrix is presented in Table (1) where driver 1 is the column player and driver 2 is the row player. When driver 1 chooses action #1 and driver 2 chooses action #1, Q-values for them are 1.5 and 1.5, respectively, according to Figure (5d). Similarly, Q-values for both drivers can be read from Figure (5) for other scenarios. Based on the bimatrix, driver 1 always chooses action #4 because action #4 is strictly better than action #1 regardless of the observed mean action, and driver 2 always chooses action #4 for the same reason. Thus, the optimal policy for both drivers is to enter grid #4 with an expected payoff \$3.5.



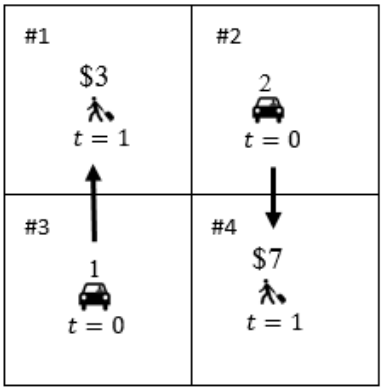
$o_1 = (\#3,0), a_1 = \#4, \bar{a}_1 = 0.5, Q(o_1, a_1, \bar{a}_1) = 3.5$
$o_2 = (\#2,0), a_2 = \#4, \bar{a}_2 = 0.5, Q(o_2, a_2, \bar{a}_2) = 3.5$

(a) Both drivers take action #4



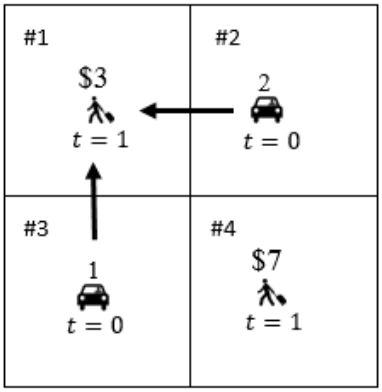
$o_1 = (\#3,0), a_1 = \#4, \bar{a}_1 = 1, Q(o_1, a_1, \bar{a}_1) = 7$
$o_2 = (\#2,0), a_2 = \#1, \bar{a}_2 = 1, Q(o_2, a_2, \bar{a}_2) = 3$

(b) Driver 1 takes action #4 and driver 2 takes action #1



$o_1 = (\#3,0), a_1 = \#1, \bar{a}_1 = 1, Q(o_1, a_1, \bar{a}_1) = 3$
$o_2 = (\#2,0), a_2 = \#4, \bar{a}_2 = 1, Q(o_2, a_2, \bar{a}_2) = 7$

(c) Driver 1 takes action #1 and driver 2 takes action #4



$o_1 = (\#3,0), a_1 = \#1, \bar{a}_1 = 0.5, Q(o_1, a_1, \bar{a}_1) = 1.5$
$o_2 = (\#2,0), a_2 = \#1, \bar{a}_2 = 0.5, Q(o_2, a_2, \bar{a}_2) = 1.5$

(d) Both drivers take action #1

Figure 5: Derived Q values for four scenarios of interest

Table 1: Q-value bimatrix for drivers

		Driver 1	
		#1	#4
Driver 2	#1	1.5, 1.5 (Figure 5d)	7, 3 (Figure 5b)
	#4	3, 7 (Figure 5c)	3.5, 3.5 (Figure 5a)

4. Reward design for multi-agent reinforcement learning

Due to selfishness of each agent, performing MARL under a given reward function in an MAS is very likely to yield an undesirable equilibrium from the perspective of the system. In other words, this equilibrium may not be an optimum with respect to some system objectives. To guide a multi-agent system towards a desirable equilibrium, system planners could resort to reward design mechanisms by modifying the reward function of agents. In this paper, we introduce a new parameter $\alpha \in \mathcal{A}$ into agents' reward, where $\mathcal{A}$ is the feasible domain of α. Parameter α can be either a scalar or a vector. The goal of system planners is to maximize some system performance measure dependent of α, denoted as $f(\alpha)$. The system planner first chooses a value of α and inputs α to the MAS. With the given α which determines the reward, the developed mean field actor-critic algorithm is employed to derive an

optimal policy π, which is dependent on α, for all agents in the system. Some performance measure f, which is calculated by executing the derived optimal policy π for all agents, is then fed into the reward design. The performance measure f is dependent on α through the dependency of π on α. In other words, $f = f(\alpha)$.

In summary, the reward design problem is to select a parameter α to maximize the performance measure $f(\alpha)$ on the upper level, while the distributed agents aim to maximize their individual cumulative rewards on the lower level once α is given as part of their reward. This process can be formulated as a bi-level optimization problem, mathematically,

$$max_{\alpha \in \mathcal{A}} f(\alpha) \tag{11}$$
where
$$max_{\pi} \sum_{k=t}^{T} \gamma^{k-t} r_i^k(\alpha),\ \forall t \in \{0, 1, \cdots, T\},\ \forall i \in \{1, 2, \cdots, N\}$$

The interaction between upper and lower levels through exchange of variables is shown in Figure (6).

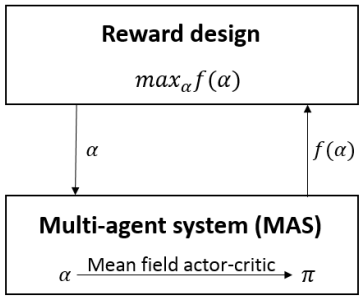


Figure 6: Architecture of the reward design

The optimization problem presented in Equation (11), however, is not straightforward to solve due to the unknown complex structure of f over the parameter α. The traditional gradient based method such as gradient descent is thus no longer applicable.

In this paper, we adopt Bayesian optimization (hereafter we call it BO). The procedure of BO is as follows. First, BO places a statistical model on the objective function f, such as a Gaussian process. Second, BO devises an acquisition function to decide where to evaluate next, i.e., to choose an α based on the statistical model. Third, BO updates the statistical model based on the newly evaluated α, and the process repeats. The pesudo-code of BO is listed in Algorithm (2). Interested readers are referred to (Frazier, 2018) for more details on BO.

Algorithm 2 Bayesian Optimization

```
Initialize a Gaussian process prior on f
Evaluate f at n_0 different αs according to certain rules
Set computational budget 𝒩 and n = n_0
while n ≤ 𝒩 do
    Update posterior probability distribution on f based on all evaluated αs
    Calculate an acquisition function
    Locate the α_n which maximizes the acquisition function
    Evaluate f at α_n
    n ← n + 1
end while
Return α_n which maximizes f
```

To be more concrete, now we use the multi-agent 2×2 example presented in Figure (3) to illustrate the potential of the reward design.

Example 4.1. (Multi-Agent 2×2). We take the order response rate (ORR), i.e. the ratio of the number of fulfilled passenger requests to the total number of passenger requests, as the performance measure

of the system. The direct application of mean field actor-critic algorithm yields a 50% ORR, which is obviously not the desired equilibrium from the perspective of the system. Noticing that the platform typically charges a certain proportion of the fare paid by the passenger as the so-called platform service charge, which is reportedly to be dependent on various factors such as distance, duration, and city. We aim to improve the performance of the system by devising a proper reward design.

In Figure (3), trip fares are shown right above each passenger request, the reached equilibrium for both drivers without any charge are to enter grid #4 and get an expected reward as \$3.5, leading to an oversupply (i.e., a low demand to supply ratio) in grid #4 and an undersupply (i.e., a high demand to supply ratio) in grid #1, which is not beneficial for the system. A reward design which deducts \$1.1 from the passenger request paid to the driver in grid #4 will effectively attract one driver to leave grid #4 for grid #1 to get more monetary return, resulting in a 100% order response rate.

5. Case Study

To test the performance of the proposed bilevel optimization model, we use two datasets including a synthetic dataset and one real-world large-scale taxi dataset downloadable from official website of New York City (NYC) Taxi & Limousine Commission (https://www1.nyc.gov/site/tlc/about/tlc-trip-record-data.page).

5.1. E-hailing driver repositioning under service charge

We first test the bilevel optimization model on a 2-by-2 grid world example, where an analytical solution of the reward design can be derived. Then we compare both values to justify the correctness of our BO algorithm.

The dataset consists of seven deterministic passenger requests in a 2-by-2 grid world setup, as shown in Figure (7). At $t = 0$, there are five idle drivers in grid #2 and five in grid #3. At time $t = 1$, five passenger requests with fare \$10 deterministically appear in grid #4 and two passenger requests with fare \$4.9 appear in grid #1. The observation space for driver i consists of the grid index and current time, i.e., $o_i = (l_i, t)$, and the action space is to enter one of neighboring grids or to stay at the current grid.

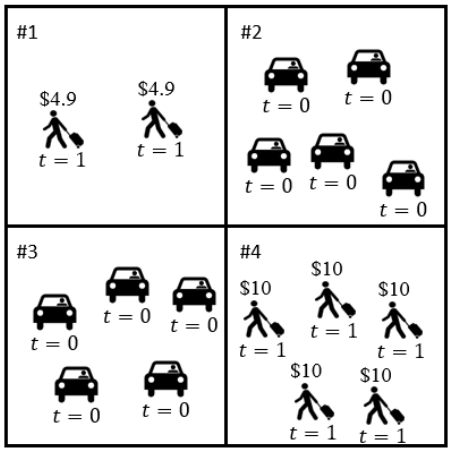


Figure 7: Layout

Without any reward design, the optimal policy for all drivers is to enter grid #4, because the expected return for entering grid #4 is at least \$5 (i.e., 10 drivers compete for 5 orders with \$10 each) while that for entering grid #1 is at most \$4.9 (i.e., the highest fare of an order in #1 is \$4.9). The resulting ORR is $5/7 = 71.43\%$, which is not desirable from the perspective of the platform because it is expected to achieve a 100% ORR in this setting. Actually, the platform can achieve a better ORR by adjusting the reward that drivers earn through the use of a platform service charge (aka the commission fee). The platform service charge used in this study is denoted as a fare percentage. For instance, a 10% service charge means the platform takes 10% of the fare paid by the passenger to the driver as its revenue. In other words, the driver gets less money under a higher service charge while the payment from the

passenger remains the same. To achieve a better ORR, the platform needs to place a high service charge in grids which are oversupplied. Drivers oversupply grid l because on average they can earn more by entering grid l, compared with entering other grids. A high service charge placed in grid l can effectively reduce monetary returns for drivers entering l and make grid l less attractive to drivers. Thus, some drivers choose other grids and take other passenger requests, resulting in an increase in ORR.

Before introducing a functional form of the platform service charge, we formly provide two notations, namely demand to supply ratio (DS) and service charge (SC). We then construct an effctive form of SC as a function of DS. A small DS indicates that the grid is oversupplied, and a large DS means the grid is undersupplied. The goal of the platform is to drive DS close to 1, meaning a balance between demand and supply. In a grid l with DS_l below 1, SC_l is expected to be large to discourage drivers from oversupplying the grid; while in a grid l with DS_l above 1, SC_l is supposed to be small. To illustrate such a relation, we use a piecewise linear function with a parameter α as SC in grid l, i.e.,

$$SC_l = \begin{cases} \alpha \times (1 - DS_l) & \text{if } DS_l \leq 1, \\ 0 & \text{otherwise,} \end{cases} \tag{12}$$

where a relatively high SC is charged to all drivers in the grid with a low DS and no SC is charged to drivers in the grid with DS above 1.

With an adjustable parameter α, the platform aims to maximize some objective f, consisting of two components, namely ORR and overall service charge (OSC), where

$$OSC = \frac{\sum_l \sum_{\text{order} \in l} SC_l \times \text{fare}_{\text{order}}}{\sum_l \sum_{\text{order} \in l} \text{fare}_{\text{order}}}.$$

The rationale of choosing these two components is as follows. First, from the perspective of the platform, it aims to maximize ORR, because a larger ORR typically means a higher revenue and a higher customer satisfaction. To maximize ORR, the platform simply chooses the largest possible value of α. The reason is that with the largest possible α, the platform penalizes drivers heavily for oversupplying a grid, and therefore drivers will be directed to other grids. This strategy, i.e., choosing the largest α, however, is a big threat for the long-term growth of the platform because drivers are very likely to quit under such a high service charge. Thus, the platform also needs to maintain a relatively small OSC. Considering the competition between ORR and OSC, we use a weighted average of ORR and $(1 - \text{OSC})$ as the objective of the platform, i.e.,

$$f = w \times \text{ORR} + (1 - w) \times (1 - \text{OSC}), \tag{13}$$

where $w \in [0, 1]$ is the weight for ORR. In this case study, we set $w = \frac{3}{5}$, meaning that the platform cares more about ORR. We then use two methods, namely, BO and an analytical method, to determine the optimal value of α.

1. **BO**. We first employ BO with the objective function given in Equation (13). For a bilevel optimization problem, first we need to check the convergence of the lower level. As an example to validate the convergence, ORR and (1 - OSC) versus the index of iterations are presented in Figure (8) with $\alpha = 0.6$. ORR increases very fast and (1 - OSC) steadily decreases during the first 1,000 iterations where agents explore the environment and learn the optimal policy. ORR and (1 - OSC) gradually converge after 1,000 iterations when agents mainly exploit the knowledge they have gained through their previous explorations.

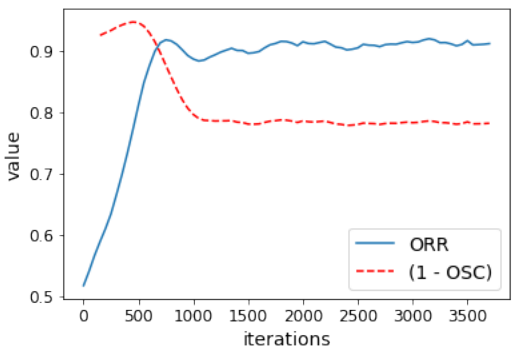


Figure 8: Convergence

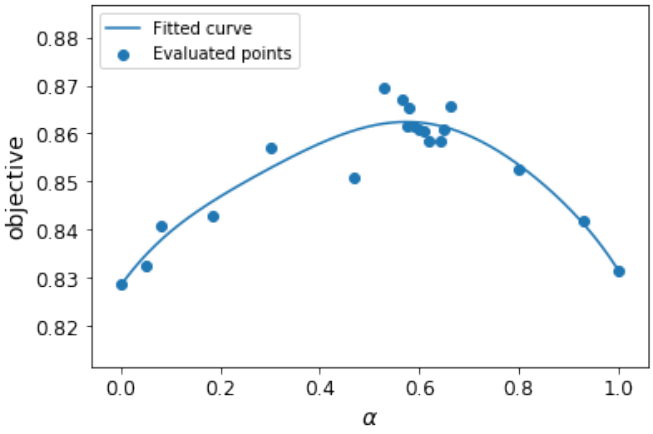


Figure 9: BO result

With the validated convergence of the lower level MAS, we run BO until convergence. The convergence of BO is defined as choosing 5 consecutive αs with the difference between the highest and lowest below a threshold of 0.05. In other words, BO converges when it starts choosing similar αs to evaluate. The result from BO is presented in Figure (9). It is noticeable that the evaluation of the objective on αs seems quite noisy. In other words, the evaluated objective may be slightly different even for the same α. This is expected because there are multiple local optima when solving the lower level MAS. Actually, it is commonly impossible to find a global optimum using deep learning. Thus researchers usually settle for local optima (Goodfellow et al., 2016). Local optima introduce noise into the evaluation of the objective at each α. Although the evaluations are noisy, the fitted curve is able to capture the mean objective f for each α. Due to the flatness near the peak of f, optimal αs are non-unique and determined as $[0.51, 0.64]$. In other words, any $\alpha \in [0.51, 0.64]$ yields the same optimal mean objective, i.e., 0.86. The optimum is 4.0% higher than the objective $f = 0.83$ without any reward design.

2. **Analytical method**. Due to the simplicity of this case, we can analytically derive the optimal value of α and shed some light on the effectiveness of the proposed platform service charge. Recall that the optimal policy for all drivers is to enter grid #4 when $\alpha = 0$. The resulting DS ratio in grid #4 is $5/10 = 0.5$, which is well below 1, meaning that grid #4 is oversupplied. ORR is $5/7 = 71.4\%$. To increase ORR, one needs to increase α to penalize drivers who oversupply a grid. As α gradually increases, grid #4 becomes less attractive, because the expected return one driver can earn decreases as α increases. When the expected return one driver can earn is less than \$4.9, one driver will enter grid #1 instead of grid #4 for a higher monetary return. Note that to ease the analysis, we assume the number of drivers entering a grid is always an integer. Similarly, as one keeps increasing α, the second driver will choose to enter #1 instead of grid #4. Now we present how we calculate the critical value of α below which there is no driver choosing to enter grid #1 while above which there is one driver attracted by grid #1. With one driver entering grid #1, there are 9 drivers entering grid #4, resulting a $5/9 = 0.56$ DS ratio in grid #4. $SC_{\#4} = \alpha \times (1 - 0.56) = 0.44\alpha$, meaning that the expected return for these 9 drivers is $\frac{\$10 \times (1 - 0.44\alpha) \times 5}{9} = 5.56 \times (1 - 0.44\alpha)$. The expected return for the driver entering grid #1 is \$4.9. We then have the critical condition $5.56 \times (1 - 0.44\alpha) = 4.9$, yielding $\alpha = 0.27$. Similarly, we can calculate the critical value of α below which there is one driver choosing to enter grid #1 while above which there are two drivers attracted by grid #1, and the critical value is $\alpha = 0.58$.

Table 2: Values of interest

α	0	0.27	0.58 (optimal)
DS ratio in grid #1 ($t = 1$)	N.A. (supply is zero)	$2/1 = 200\%$	$2/2 = 100\%$
DS ratio in grid #4 ($t = 1$)	$5/10 = 50.0\%$	$5/9 = 55.6\%$	$5/8 = 62.5\%$
ORR	$5/7 = 71.4\%$	$6/7 = 85.7\%$	100%
OSC	0	0.11	0.18
f	0.83	0.87	0.93 (optimum)

Values of interest are presented in Table (2). With $\alpha = 0$, ORR = 71.4% and OSC = 0. The objective is $\frac{3}{5} \times \text{ORR} + \frac{2}{5}(1 - \text{OSC}) = 0.83$. With α increasing to 0.27, there is one driver attracted by grid #1, resulting in a 85.7% ORR. The OSC is calculated as follows. The DS ratio in grid #4 is now 0.56, resulting in $SC_{\#4} = 0.27 \times (1 - 0.56) = 0.12$. Thus, $\text{OSC} = \dfrac{\$10 \times 5 \times 0.12}{\$10 \times 5 + \$4.9} = 0.11$. The objective is $\frac{3}{5} \times \text{ORR} + \frac{2}{5}(1 - \text{OSC}) = 0.87$. Similarly, with α increasing to 0.58, ORR = 100%, OSC = 0.18, and the objective is 0.93. Increasing α further does not improve ORR but increases OSC, resulting in a decrease in the objective. Thus, the analytically derived optimal value of α is 0.58.

The analytically derived optimal value of α, i.e., 0.58, agrees well with the derived optimal range of α from BO, i.e., $[0.51, 0.64]$. The optimum from the analytical solution, i.e., 0.93, however, deviates from its numerical counterpart, i.e., 0.86. One possible explanation is as follows. In the analytical solution, the policy for agents is deterministic and exact two drivers choose grid #1 after increasing α to 0.58; while in BO, the derived optimal policy for agents is stochastic, introducing variance in drivers' actions. For example, the derived optimal policy says each driver has a 20% probability of choosing grid #1 and a 80% probability of choosing grid #4. Although the expected number of agents in grid #1 is 2 and the expected number of agents in grid #4 is 8, the probability of all agents choosing grid #4 is $0.8^{10} = 10.7\%$. This variance reduces both ORR and (1 - OSC), resulting in a lower objective from BO, compared with the objective from the analytical solution.

5.2. Multiclass taxi driver repositioning under congestion pricing

In this case study, we apply the proposed bilevel optimization model to a real-world scenario where city planners aim to mitigate traffic congestion in the central business district (CBD). As an effective way to improve traffic condition, congestion pricing has been adopted by many cities such as London and Stockholm (de Palma and Lindsey, 2011). The basic idea of congestion pricing is to impose a toll charge on all vehicles entering the CBD. Consequently, some drivers may be sensitive to the toll charge and take alternative travel modes such as subway while some drivers can bear the toll charge. To demonstrate the effectiveness of congestion pricing, we use NYC taxi and subway data due to data availability.

In the taxi market, congestion pricing affects both the demand and supply. On the demand side, the toll charge is passed to passengers for taxi drivers carrying passengers into the CBD. In other words, the fare paid by passengers is increased and thus the demand (i.e., number of passenger requests) is decreased. According to Schaller (1999), taxi demand falls by $0.22 \times x$ percent when taxi fare increases by x percent. For example, when taxi fare increases from \$10 to \$12.5 (i.e., increases by 25%) for a trip, the probability of the passenger choosing alternative travel modes is 5.5% (i.e. $0.22 \times 25\%$). On the supply side, the toll charge is paid by taxi drivers when they enter the CBD vacantly, which discourages drivers from entering the CBD without any passenger. The overall effect of congection pricing results in a reduced number of taxis in the CBD, leading to an improved traffic condition. This, however, may direct too many passengers, whose taxi requests are unfulfilled, to the public transit which is already running at full pressure during rush hours (Plitt, 2020). Thus, there exists a tradeoff between reducing the number of taxis in the CBD and maintaining a reasonable level of crowdedness in the public transit system.

The objective of city planners thus consists of two components, namely, the number of taxis in the CBD and the crowdedness of the public transit system. Considering the accessibility of data (i.e., NYC taxi data and subway data), we make two assumptions: (1) The proposed congestion pricing scheme only affects the behavior of taxi drivers, as previously mentioned; (2) The subway system is used as a proxy of the public transit of the city.

To be more precise, Figure (10) presents objectives of city planners and how planners derive the best control. City planners impose a toll charge on taxis entering the CBD. Adaptive taxis learn the optimal policy by the mean-field actor-critic algorithm under the toll charge. With fewer taxis searching for passengers in the CBD, more unfulfilled passenger requests are directed to the subway system. City planners observe the number of taxis in the CBD and crowdedness in the subway and adjust the toll charge to achieve a better balance between these two objectives. This process repeats until city planners reach a satisfactory balance.

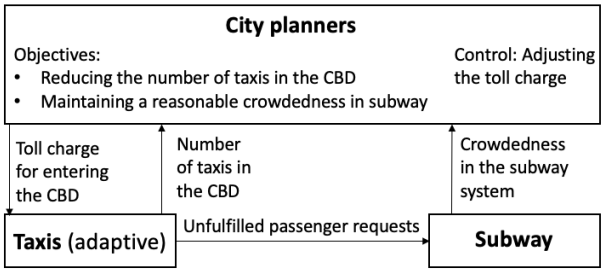


Figure 10: Objectives of city planners

5.2.1. Data preprocessing

The NYC taxi trip records are publicly available on the official website of NYC Taxi & Limousine Commission (https://www1.nyc.gov/site/tlc/about/tlc-trip-record-data.page). We use the data for both yellow and green taxis during May 2014 before the wide adoption of ridesharing service such as Uber and Lyft and after the business of green taxis gradually stabilizes. A data sample is listed in Table (3). Each entry in Table (3) collects the order information, including pickup and dropoff time and locations and fares. In total there are around 16 million taxi trips. We first remove the weekend data because trip patterns over weekends are obviously different from that on weekdays. We then restrict the time interval of interest as the evening peak, i.e., 4 PM to 8 PM. There are 2 million taxi trips in the weekday data after preprocessing.

Table 3: Taxi data sample

pickup datetime	dropoff datetime	pickup longitude	pickup latitude	dropoff longitude	dropoff latitude	fare amount
2014-05-01 16:59:00	2014-05-01 17:08:30	-73.978818	40.785048	-73.965570	40.800718	6.5
2014-05-01 16:59:00	2014-05-01 17:23:00	-73.960280	40.778892	-73.975542	40.751427	15.5

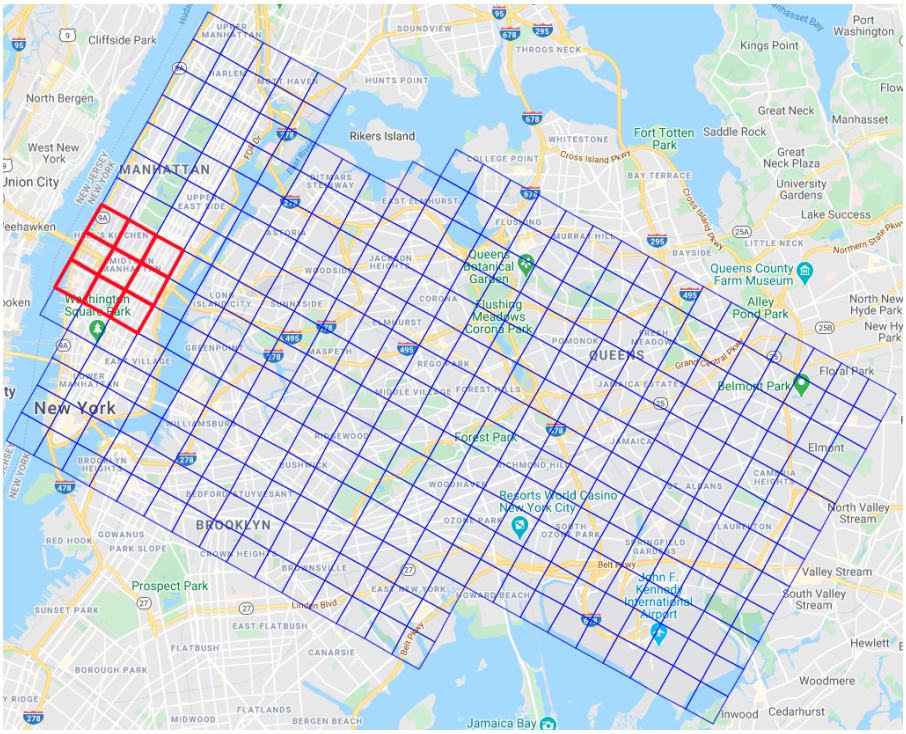


Figure 11: Spatial discretization of the area

Figure (11) presents the spatial discretization of the area of interest. There are in total 337 grids with a side length of 1 km covering the area from Manhattan to two airports located in Queens. Taxi orders outside grids consist of less than 2% of the overall taxi orders and are not considered. Each longitude and latitude coordinate is transformed into a grid index. As for the temporal discretization, the evening peak is divided into eighty 3-minute time intervals and the pickup time and dropoff time are transformed into time interval index. Grids shown as bold red squares cover the CBD of NYC, which is the area between 19th street and 59th street in Manhattan. The proposed congestion pricing is applied to vehicles cross the red square into the CBD.

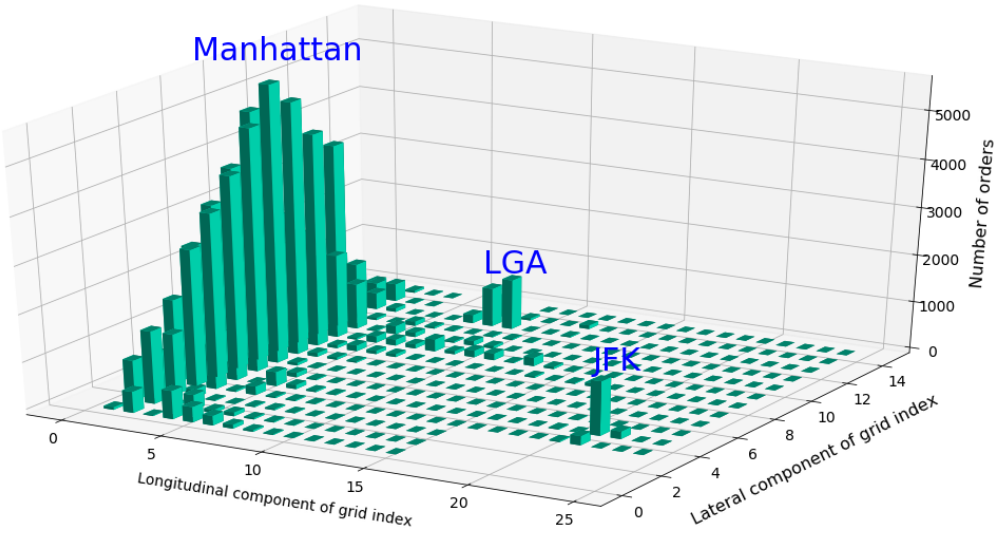


Figure 12: Spatial distribution of taxi orders during evening peak

Figure (12) presents the spatial distribution of taxi orders (pickup) during evening peak. It can be

seen that the majority of taxi orders emerge in Manhattan, especially in the CBD. There are two local hotspots near two airports.

NYC subway turnstile data is also publicly accessible (http://web.mta.info/developers/turnstile.html). A sample of the turnstile data is listed in Table (4). These two rows show that the reading of entries for turnstile ID (A002, R051, 02-00-00) is 4,593,637 at 4 PM and 4,594,523 at 8 PM on 05/01/2014. Taking the difference between two readings yields the net entries at this turnstile during the 4-hour time interval, i.e., 4,594,523 - 4,593,637 = 886. Similarly, we can calculate net entries and net exits for each turnstile. Net entries and net exits of a grid are then calculated by summing up the net entries and net exits of all turnstiles in that grid, respectively.

Table 4: Turnstile data sample

Turnstile ID	Date	Time	Entries	Exits
(A002, R051, 02-00-00)	05/01/2014	16:00:00	4,593,637	1,564,283
(A002, R051, 02-00-00)	05/01/2014	20:00:00	4,594,523	1,564,348

5.2.2. Objective function of city planners

As previously mentioned, the objective function of city planners consists of two components, namely, number of vehicles in the CBD and the crowdedness of the public transit. Now we formally define these two components based on NYC taxi data and subway turnstile data.

The first component is defined as the percentage of taxis in the CBD, i.e, the ratio of the number of taxis in the CBD to the total number of taxis. For each time step, we calculate one value of the percentage. We then take the average of the percentages across all time steps as the overall percentage of taxis in the CBD. Hereafter we call this PTC (percentage of taxis in the CBD). PTC decreases with toll charge because fewer vacant taxis enter the CBD with a higher toll charge.

The crowdedness of the subway system in each grid is further decomposed into two parts, namely, the entry crowdedness which is related with the net entries into the subway system within the grid, and the exit crowdedness which is related with the net exits from the subway system within the grid. After imposing a toll charge on taxis entering the CBD, the crowdedness of the subway system increases due to the unserviced taxi orders. Here we assume that travel demand stemming from the unserviced taxi orders goes to the subway system. For each grid, we count the unserviced taxi orders with its origin inside the grid and call this quantity the additional entry into the subway system. We then take the ratio of the additional entry to the net entries within the grid as the increase in the entry crowdedness. Similarly, we can calculate the increase in the exit crowdedness within the grid. Taking the average of the increase in the entry crowdedness and the increase in the exit crowdedness yields the increase in the crowdedness of the grid. Among the overall 337 grids, we focus on the top 20 grids in terms of crowdedness. Hereafter we call this ICS (increase in crowdedness in subway). ICS increases with the toll charge because more passengers are directed to the subway system with a higher toll charge.

From the perspective of city planners, both PTC and ICS are expected to be small. These two components, however, are competing against each other. With a small toll charge, ICS is small but PTC is large; while with a large toll charge, PTC becomes smaller but ICS gets larger. Therefore city planners need to maintain some balance between these two components. Here we use a weighted average of these two components as the objective of city planners. To ensure maximization, we add a minus sign:

$$f = -[w \times \text{PTC} + (1 - w) \times \text{ICS}] \tag{14}$$

where $w \in [0, 1]$ is the weight for PTC. In this case study, we set $w = \frac{1}{5}$ considering the difference in the magnitude of two components.

5.2.3. Multiclass MARL

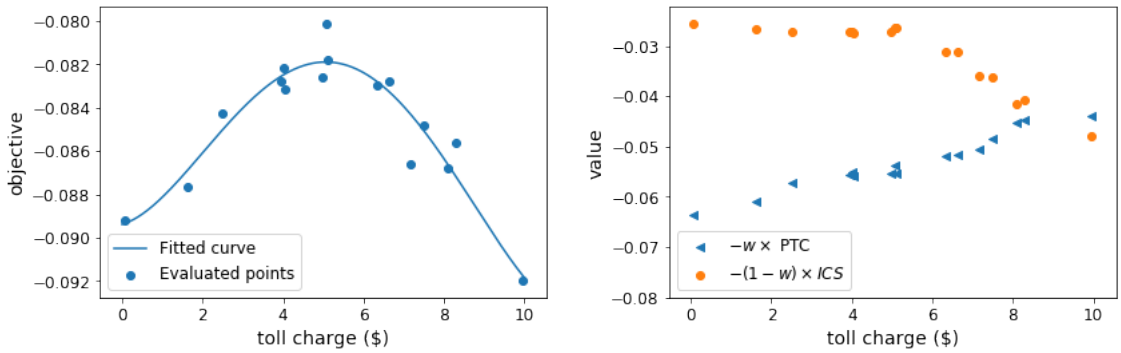
The NYC taxi market contains two types of taxicabs, namely yellow taxis and green taxis. They are different because yellow taxis can go and pick up passengers anywhere while green taxis are not allowed to pick up passengers in Manhattan below East 96th Street and West 110th Street and at two airports, namely LaGuardia airport (LGA) and John F. Kennedy airport (JFK). Therefore we need to model them differently in the lower level MARL. To incorporate these two classes of agents into MARL, we create two actors and critics. All the yellow taxis share one actor (i.e., a policy network) and one critic (i.e., value network), and green taxis share the other actor and the critic. In the actor-critic algorithm

demonstrated in Fig 4, both yellow and green agents interact with the same environment. They have the same observation space and action space. In other words, for both yellow and green taxis, its observation consists of the grid index and current time, and its action is to enter one of neighboring grids or to stay in the current grid. In addition, both of them aim to maximize their cumulative monetary return.

The key difference is, green taxis can drop off and search for passengers in those restricted areas (i.e., Manhattan below East 96th Street and West 110th Street and two airports), they can not pick up passengers there. From the modeling perspective, the environment will not assign orders to green taxis in restricted areas. This restriction discourages green taxis from searching for passengers or taking passengers to the restricted areas. Accordingly, the policy for green taxis is expected to be different from that of yellow taxis. Yellow taxis thus only compete among themselves in the restricted areas, while outside restricted areas, yellow and green taxis not only compete within the same type but also compete with the other taxi type.

5.2.4. Results

On weekdays, there are on average around 90,000 taxi orders during the evening peak. According to Wikipedia (https://en.wikipedia.org/wiki/Taxicabs_of_New_York_City), there are around 13,000 yellow "medallion" taxicabs in NYC. Considering that some drivers do not work during the evening peak and some drivers work outside the grid world, we thus set the number of yellow agents in MARL as 12,000. The number of green agents is set to 5,000 considering that there were in total around 6,000 green taxi drivers in 2015 and some of them may not work during the evening peak.



(a) Objective

(b) Two components

Figure 13: BO result

We then run the bilevel optimization with the objective function given in Equation (14) and derived the optimal toll charge as \$5.1. The result is presented in Figure (13). The objective is around -0.09 without any toll charge, as shown in Figure (13a) . The objective increases with toll charge before toll charge reaches \$5.1. With a \$5.1 toll charge, the objective is -0.08, which is 7.9% higher than -0.09. The objective decreases if toll charge is increased beyond \$5.1. The parabolic shape of the objective can be explained by Figure (13b). Before the toll charge reaches \$5.1, the steady increase in $-w \times PTC$ and the minor decrease in $-(1-w) \times ICS$ push the objective higher with a larger toll charge. After the toll charge is increased beyond \$5.1, $-(1-w) \times ICS$ declines faster and suppresses the effect of the increase in $w \times PTC$, resulting in a decrease in the objective.

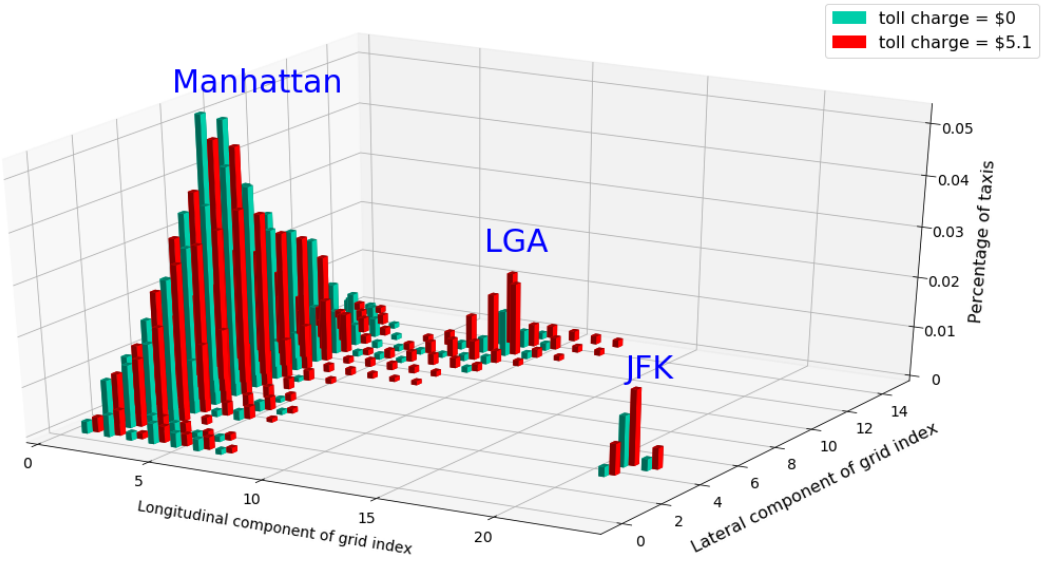


Figure 14: Percentage of taxis in each grid (grids with percentage lower than 0.001 are omitted)

Figure (14) presents the average percentage of taxis in each grid for two scenarios, namely without any toll charge and with the optimal toll charge. With the optimal toll charge, the percentage of taxis in Manhattan, especially in the CBD, is decreased, while that for two airports are increased. This is as expected because taxi drivers are penalized for entering the CBD vacantly, meaning that CBD now becomes less attractive to taxi drivers. According to the demand distribution shown in Figure (12), two airports become comparatively attractive.

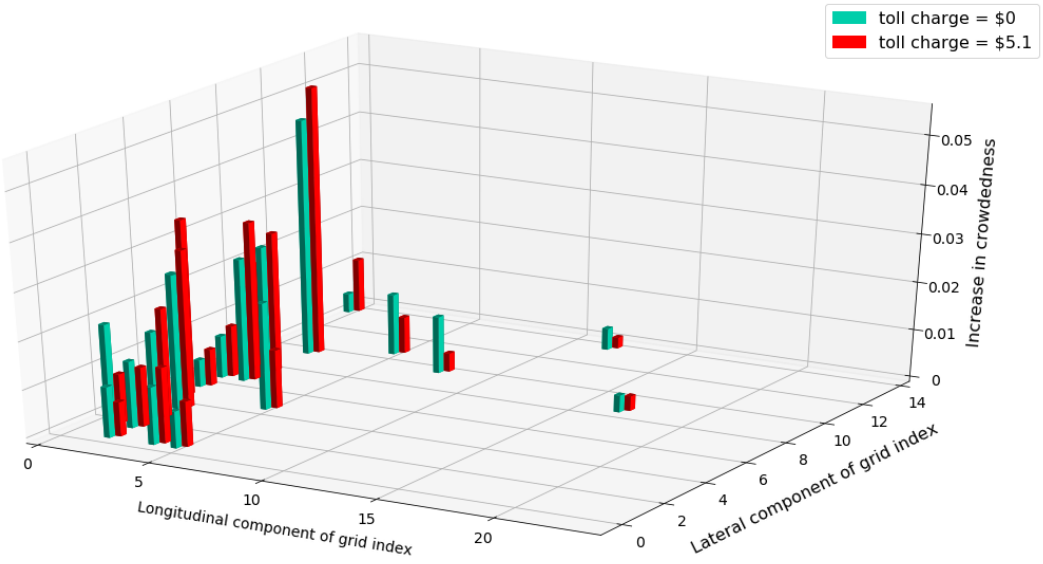


Figure 15: Increase in crowdedness across the busiest 20 grids

Figure (15) presents the increase in crowdedness across the busiest 20 grids. With the optimal toll charge, the increase in crowdedness in the subway is higher in the CBD, compared to that without any toll charge, because now there are fewer taxis in the CBD and therefore more passengers are directed to the subway system. For grids outside CBD, the increase in crowdedness can be either higher or lower because the crowdedness consists of two components, namely, the entry crowdedness and the exit crowdedness. The increase in entry crowdedness is expected to be lower for grids outside CBD because there are more vacant taxis outside the CBD who is willing to carry passengers into CBD. The increase

in exit crowdedness is higher in many grids because more people take subway to arrive in grids outside the CBD.

6. Conclusion

Noticing the underutilization of taxi resources due to idle taxi drivers' cruising behavior, this study aims to model the multi-driver repositioning task through a mean field multi-agent reinforcement learning approach. A mean field actor-critic algorithm is developed to solve the MARL with a given reward function. The direct application of the mean field actor-critic algorithm is, however, very likely to yield a suboptimal equilibrium from the standpoint of the system. Thus, this study proposes a bilevel optimization with the upper level as a reward design and the lower level as the MARL. The upper level interacts with the lower level by adjusting rewards. The bilevel optimization model is applied to two scenarios, namely, e-hailing driver repositioning under service charge and taxi driver repositioning under congestion pricing. In the case of e-hailing driver repositioning, the agreement between the derived optimal control from BO and that from an analytical solution validates the effectiveness of the model. It is also worth mentioning that the objective of the e-hailing platform is increased by 4.0% using a simple piecewise linear platform service charge. In the case of multiclass taxi driver repositioning, a \$5.1 toll charge increases the objective of city planners by 7.9%, compared to that without any toll charge. With the optimal toll charge, the number of taxis in the CBD is decreased, indicating a better traffic condition. The crowdedness is increased in the subway stations within the CBD due to fewer taxis. For subway stations outside the CBD, the crowdedness can be either higher or lower depending on the tradeoff between the entry crowdedness and the exit crowdedness.

The aforementioned two driver-repositioning applications validate the effectiveness of the proposed bilevel optimization model. We stress that the model is general and can be applied to various systems as long as there are two levels in the system and the upper level can affect the lower level through some control. With some optimal control, the performance of the system can be improved, which is beneficial for the urban economy.

There are some future work that can be done to overcome some limitations of this study.

1. Although the mean field approximation is effective to make MARL with a large number of agents tractable, it may oversimplify the interaction among agents. A theoretical or physics-informed approach can be developed to better capture the interaction among agents.

2. We will further explore the modeling of multiclass MARL model. In the second case study, although the difference between yellow taxis and green taxis is considered, the heterogeneity among yellow taxis (or green taxis) is neglected due to the homogeneity assumption. A more personalized model capturing the behavioral difference within the same type of agents is left in future research.

3. A predefined form of the control with a parameter α (e.g., the piecewise linear service charge and toll charge in previous case studies) was used in the upper level, meaning that the upper level aims to find the best control within the space defined the given form of the control. A more game-theoreical approach such as the leader-follower game may relax this restriction and find a globally optimal control.

References

Brown, N., Sandholm, T., Jan. 2018. Superhuman AI for heads-up no-limit poker: Libratus beats top professionals. Science 359 (6374), 418–424.

Brown, N., Sandholm, T., Aug. 2019. Superhuman AI for multiplayer poker. Science 365 (6456), 885–890.

Buoniu, L., Babuka, R., De Schutter, B., 2010. Multi-agent Reinforcement Learning: An Overview. In: Srinivasan, D., Jain, L. C. (Eds.), Innovations in Multi-Agent Systems and Applications - 1. Studies in Computational Intelligence. Springer Berlin Heidelberg, Berlin, Heidelberg, pp. 183–221.

de Palma, A., Lindsey, R., Dec. 2011. Traffic congestion pricing methodologies and technologies. Transportation Research Part C: Emerging Technologies 19 (6), 1377–1399.

Di, X., Ban, X. J., 2019. A unified equilibrium framework of new shared mobility systems. Transportation Research Part B: Methodological 129, 50–78.

Di, X., He, X., Guo, X., Liu, H. X., 2014. Braess paradox under the boundedly rational user equilibria. Transportation Research Part B: Methodological 67, 86–108.

Di, X., Liu, H. X., Ban, X. J., 2016. Second best toll pricing within the framework of bounded rationality. Transportation Research Part B 83, 74–90.

Di, X., Ma, R., Liu, H. X., Ban, X. J., 2018. A link-node reformulation of ridesharing user equilibrium with network design. Transportation Research Part B: Methodological 112, 230–255.

Foerster, J. N., Assael, Y. M., de Freitas, N., Whiteson, S., 2016. Learning to Communicate with Deep Multi-agent Reinforcement Learning. In: Proceedings of the 30th International Conference on Neural Information Processing Systems. NIPS'16. Curran Associates Inc., USA, pp. 2145–2153, event-place: Barcelona, Spain.

Frazier, P. I., Jul. 2018. A Tutorial on Bayesian Optimization. arXiv:1807.02811 [cs, math, stat]ArXiv: 1807.02811.

Gao, Y., Jiang, D., Xu, Y., Aug. 2018. Optimize taxi driving strategies based on reinforcement learning. International Journal of Geographical Information Science 32 (8), 1677–1696.

Ge, Y., Xiong, H., Tuzhilin, A., Xiao, K., Gruteser, M., Pazzani, M., 2010. An Energy-efficient Mobile Recommender System. In: Proceedings of the 16th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining. KDD '10. ACM, New York, NY, USA, pp. 899–908.

Goodfellow, I., Bengio, Y., Courville, A., 2016. Deep Learning. MIT Press, `http://www.deeplearningbook.org`.

Grondman, I., Busoniu, L., Lopes, G. A. D., Babuska, R., Nov. 2012. A Survey of Actor-Critic Reinforcement Learning: Standard and Natural Policy Gradients. IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews) 42 (6), 1291–1307.

Hwang, R.-H., Hsueh, Y.-L., Chen, Y.-T., Sep. 2015. An effective taxi recommender system based on a spatio-temporal factor analysis model. Information Sciences 314, 28–40.

Jin, J., Zhou, M., Zhang, W., Li, M., Guo, Z., Qin, Z., Jiao, Y., Tang, X., Wang, C., Wang, J., Wu, G., Ye, J., 2019. CoRide: Joint Order Dispatching and Fleet Management for Multi-Scale Ride-Hailing Platforms. In: Proceedings of the 28th ACM International Conference on Information and Knowledge Management. CIKM '19. ACM, New York, NY, USA, pp. 1983–1992, event-place: Beijing, China.

Konda, V. R., Tsitsiklis, J. N., Apr. 2003. On Actor-Critic Algorithms. SIAM J. Control Optim. 42 (4), 1143–1166.

Li, M., Qin, Z., Jiao, Y., Yang, Y., Wang, J., Wang, C., Wu, G., Ye, J., 2019. Efficient Ridesharing Order Dispatching with Mean Field Multi-Agent Reinforcement Learning. In: The World Wide Web Conference. WWW '19. ACM, New York, NY, USA, pp. 983–994, event-place: San Francisco, CA, USA.

Lin, K., Zhao, R., Xu, Z., Zhou, J., 2018. Efficient Large-Scale Fleet Management via Multi-Agent Deep Reinforcement Learning. In: Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining. KDD '18. ACM, New York, NY, USA, pp. 1774–1783.

Littman, M. L., 1994. Markov Games As a Framework for Multi-agent Reinforcement Learning. In: Proceedings of the Eleventh International Conference on International Conference on Machine Learning. ICML'94. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, pp. 157–163, event-place: New Brunswick, NJ, USA.

Lowe, R., Wu, Y., Tamar, A., Harb, J., Abbeel, P., Mordatch, I., 2017. Multi-agent Actor-critic for Mixed Cooperative-competitive Environments. In: Proceedings of the 31st International Conference on Neural Information Processing Systems. NIPS'17. Curran Associates Inc., USA, pp. 6382–6393, event-place: Long Beach, California, USA.

Matignon, L., Laurent, G. J., Fort-Piat, N. L., Feb. 2012. Independent reinforcement learners in cooperative Markov games: a survey regarding coordination problems. The Knowledge Engineering Review 27 (1), 1–31.

Meng, Q., Liu, Z., 2012. Impact analysis of cordon-based congestion pricing on mode-split for a bimodal transportation network. Transportation Research Part C: Emerging Technologies 21 (1), 134–147.

Mguni, D., Jennings, J., Sison, E., Valcarcel Macua, S., Ceppi, S., Munoz de Cote, E., May 2019. Coordinating the Crowd: Inducing Desirable Equilibria in Non-Cooperative Systems. In: Proceedings of the 18th International Conference on Autonomous Agents and MultiAgent Systems. AAMAS '19. International Foundation for Autonomous Agents and Multiagent Systems, Montreal QC, Canada, pp. 386–394.

Mnih, V., Kavukcuoglu, K., Silver, D., Rusu, A. A., Veness, J., Bellemare, M. G., Graves, A., Riedmiller, M., Fidjeland, A. K., Ostrovski, G., Petersen, S., Beattie, C., Sadik, A., Antonoglou, I., King, H., Kumaran, D., Wierstra, D., Legg, S., Hassabis, D., Feb. 2015. Human-level control through deep reinforcement learning. Nature 518 (7540), 529–533.

Nguyen, T. T., Nguyen, N. D., Nahavandi, S., Dec. 2018. Deep Reinforcement Learning for Multi-Agent Systems: A Review of Challenges, Solutions and Applications. arXiv:1812.11794 [cs, stat]ArXiv: 1812.11794.

OpenAI, 2018. Openai five. `https://blog.openai.com/openai-five/`.

Plitt, A., 2020. The new york city subway, explained.
URL `https://ny.curbed.com/2019/1/25/18195014/new-york-mta-subway-map-fare-history`

Powell, J. W., Huang, Y., Bastani, F., Ji, M., 2011. Towards Reducing Taxicab Cruising Time Using Spatio-temporal Profitability Maps. In: Proceedings of the 12th International Conference on Advances in Spatial and Temporal Databases. SSTD'11. Springer-Verlag, Berlin, Heidelberg, pp. 242–260.

Puterman, M. L., 1994. Markov Decision Processes: Discrete Stochastic Dynamic Programming, 1st Edition. John Wiley & Sons, Inc., New York, NY, USA.

Qu, M., Zhu, H., Liu, J., Liu, G., Xiong, H., 2014. A Cost-effective Recommender System for Taxi Drivers. In: Proceedings of the 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining. KDD '14. ACM, New York, NY, USA, pp. 45–54.

Rong, H., Zhou, X., Yang, C., Shafiq, Z., Liu, A., 2016. The Rich and the Poor: A Markov Decision Process Approach to Optimizing Taxi Driver Revenue Efficiency. In: Proceedings of the 25th ACM International on Conference on Information and Knowledge Management. CIKM '16. ACM, New York, NY, USA, pp. 2329–2334.

Schaller, B., Aug. 1999. Elasticities for taxicab fares and service availability. Transportation 26 (3), 283–297.

Shou, Z., Di, X., Ye, J., Zhu, H., Zhang, H., Hampshire, R., Feb. 2020. Optimal passenger-seeking policies on E-hailing platforms using Markov decision process and imitation learning. Transportation Research Part C: Emerging Technologies 111, 91–113.

Silver, D., Huang, A., Maddison, C. J., Guez, A., Sifre, L., van den Driessche, G., Schrittwieser, J., Antonoglou, I., Panneershelvam, V., Lanctot, M., Dieleman, S., Grewe, D., Nham, J., Kalchbrenner, N., Sutskever, I., Lillicrap, T., Leach, M., Kavukcuoglu, K., Graepel, T., Hassabis, D., Jan. 2016. Mastering the game of Go with deep neural networks and tree search. Nature 529 (7587), 484–489.

Silver, D., Schrittwieser, J., Simonyan, K., Antonoglou, I., Huang, A., Guez, A., Hubert, T., Baker, L., Lai, M., Bolton, A., Chen, Y., Lillicrap, T., Hui, F., Sifre, L., van den Driessche, G., Graepel, T., Hassabis, D., Oct. 2017. Mastering the

game of Go without human knowledge. Nature 550 (7676), 354–359.
Sutton, R. S., Barto, A. G., 1998. Introduction to Reinforcement Learning, 1st Edition. MIT Press, Cambridge, MA, USA.
Sutton, R. S., McAllester, D., Singh, S., Mansour, Y., 1999. Policy Gradient Methods for Reinforcement Learning with Function Approximation. In: Proceedings of the 12th International Conference on Neural Information Processing Systems. NIPS'99. MIT Press, Cambridge, MA, USA, pp. 1057–1063, event-place: Denver, CO.
Tampuu, A., Matiisen, T., Kodelja, D., Kuzovkin, I., Korjus, K., Aru, J., Aru, J., Vicente, R., Apr. 2017. Multiagent cooperation and competition with deep reinforcement learning. PLOS ONE 12 (4), e0172395.
Tan, M., 1993. Multi-Agent Reinforcement Learning: Independent vs. Cooperative Agents. In: In Proceedings of the Tenth International Conference on Machine Learning. Morgan Kaufmann, pp. 330–337.
Verma, T., Varakantham, P., Kraus, S., Lau, H. C., Jun. 2017. Augmenting decisions of taxi drivers through reinforcement learning for improving revenues. Proceedings of the Twenty-Seventh International Conference on Automated Planning and Scheduling ICAPS 2017: Pittsburgh, June 18-23, 409–417.
Vinyals, O., Babuschkin, I., Czarnecki, W. M., Mathieu, M., Dudzik, A., Chung, J., Choi, D. H., Powell, R., Ewalds, T., Georgiev, P., Oh, J., Horgan, D., Kroiss, M., Danihelka, I., Huang, A., Sifre, L., Cai, T., Agapiou, J. P., Jaderberg, M., Vezhnevets, A. S., Leblond, R., Pohlen, T., Dalibard, V., Budden, D., Sulsky, Y., Molloy, J., Paine, T. L., Gulcehre, C., Wang, Z., Pfaff, T., Wu, Y., Ring, R., Yogatama, D., Wnsch, D., McKinney, K., Smith, O., Schaul, T., Lillicrap, T., Kavukcuoglu, K., Hassabis, D., Apps, C., Silver, D., Nov. 2019. Grandmaster level in StarCraft II using multi-agent reinforcement learning. Nature 575 (7782), 350–354.
Yang, H., H. Bell, M. G., 1998. Models and algorithms for road network design: a review and some new developments. Transport Reviews 18 (3), 257–278.
Yang, Y., Luo, R., Li, M., Zhou, M., Zhang, W., Wang, J., Jul. 2018. Mean Field Multi-Agent Reinforcement Learning. In: International Conference on Machine Learning. pp. 5571–5580.
Yu, X., Gao, S., Hu, X., Park, H., Mar. 2019. A Markov decision process approach to vacant taxi routing with e-hailing. Transportation Research Part B: Methodological 121, 114–134.
Yuan, J., Zheng, Y., Zhang, L., Xie, X., Sun, G., 2011. Where to Find My Next Passenger. In: Proceedings of the 13th International Conference on Ubiquitous Computing. UbiComp '11. ACM, New York, NY, USA, pp. 109–118.
Zhang, K., Yang, Z., Baar, T., Nov. 2019. Multi-Agent Reinforcement Learning: A Selective Overview of Theories and Algorithms.
URL https://arxiv.org/abs/1911.10635v1
Zhang, X., Yang, H., 2004. The optimal cordon-based network congestion pricing problem. Transportation Research Part B: Methodological 38 (6), 517–537.
Zhou, M., Jin, J., Zhang, W., Qin, Z., Jiao, Y., Wang, C., Wu, G., Yu, Y., Ye, J., 2019. Multi-Agent Reinforcement Learning for Order-dispatching via Order-Vehicle Distribution Matching. In: Proceedings of the 28th ACM International Conference on Information and Knowledge Management. CIKM '19. ACM, New York, NY, USA, pp. 2645–2653, event-place: Beijing, China.
Zhou, X., Rong, H., Yang, C., Zhang, Q., Khezerlou, A. V., Zheng, H., Shafiq, M. Z., Liu, A. X., 2018. Optimizing Taxi Driver Profit Efficiency: A Spatial Network-based Markov Decision Process Approach. IEEE Transactions on Big Data, 1–1.