Harnessing Big Data for the Sharing Economy in Smart Cities

Zhenyu Shou

Submitted in partial fulfillment of the
requirements for the degree of
Doctor of Philosophy
under the Executive Committee
of the Graduate School of Arts and Sciences

COLUMBIA UNIVERSITY

2021

Abstract

Harnessing Big Data for the Sharing Economy in Smart Cities

Zhenyu Shou

Motivated by the imbalance between demand (i.e., passenger requests) and supply (i.e., available vehicles) in the ride-hailing market and severe traffic congestion faced by modern cities, this dissertation aims to improve the efficiency of the sharing economy by building an agent-based methodological framework for optimal decision-making of distributed agents (e.g., autonomous shared vehicles), including passenger-seeking and route choice. Furthermore, noticing that city planners can impact the behavior of agents via some operational measures such as congestion pricing and signal control, this dissertation investigates the overall bilevel problem that involves the decision-making process of both distributed agents (i.e., the lower level) and central city planners (i.e., the upper level).

First of all, for the task of passenger-seeking, this dissertation proposes a model-based Markov decision process (MDP) approach to incorporate distinct features of e-hailing drivers. The modified MDP approach is found to outperform the baseline (i.e., the local hotspot strategy) in terms of both the rate of return and the utilization rate. Although the modified MDP approach is set up in the single-agent setting, we extend its applicability to multi-agent scenarios by a dynamic adjustment strategy of the order matching probability which is able to partially capture the competition among agents. Furthermore, noticing that the reward function is commonly assumed as some prior knowledge, this dissertation unveils the underlying reward function of the overall e-hailing driver population (i.e., 44,000 Didi drivers in Beijing) through an inverse reinforcement learning method, which paves the way for future research on discovering the underlying reward mechanism in a complex and dynamic ride-hailing market.

To better incorporate the competition among agents, this dissertation develops a model-free mean-field multi-agent actor-critic algorithm for multi-driver passenger-seeking. A bilevel optimization model is then formulated with the upper level as a reward design mechanism and the

lower level as a multi-agent system. We use the developed mean field multi-agent actor-critic algorithm to solve for the optimal passenger-seeking policies of distributed agents in the lower level and Bayesian optimization to solve for the optimal control of upper-level city planners. The bilevel optimization model is applied to a real-world large-scale multi-class taxi driver repositioning task with congestion pricing as the upper-level control. It is disclosed that the derived optimal toll charge can efficiently improve the objective of city planners.

With agents knowing where to go (i.e., passenger-seeking), this dissertation then applies the bilevel optimization model to the research question of how to get there (i.e., route choice). Different from the task of passenger-seeking where the action space is always fixed-dimensional, the problem of variable action set emerges in the task of route choice. Therefore, a flow-dependent deep Q-learning algorithm is proposed to efficiently derive the optimal policies for multi-commodity multi-class agents. We demonstrate the effect of two countermeasures, namely tolling and signal control, on the behavior of travelers and show that the systematic objective of city planners can be optimized by a proper control.

Table of Contents

List of Figures v

List of Tables viii

Acknowledgments ix

Dedication x

Chapter 1: Introduction and Background 1

1.1 Motivation 1

1.2 Research objectives and scope 2

1.3 Research contributions 3

1.4 Dissertation organization 4

Chapter 2: Passenger-seeking for e-hailing drivers 5

2.1 Literature review 5

2.2 Markov decision process (MDP) for a single agent 10

2.2.1 Preliminaries 10

2.2.2 MDP for e-hailing drivers 12

2.2.3 Extracting parameters from data 19

2.2.4 Numerical example 25

2.3 Sequential MDPs for multiple agents 28
2.4 Case study 32
2.4.1 Policy 35
2.4.2 Model Evaluation 37
2.4.3 Supply-demand ratio 42
2.4.4 Different optimal policies for multiple agents 43
2.4.5 Heterogeneity in reward functions 45
2.5 Summary 48

Chapter 3: Bilevel optimization for multi-driver passenger-seeking 50
3.1 Introduction 50
3.1.1 Problem statement 50
3.1.2 Literature review 52
3.1.3 Research gaps 54
3.1.4 Contributions of this chapter 57
3.2 Single agent reinforcement learning 57
3.2.1 Problem definition 57
3.2.2 Actor-Critic method 61
3.3 Multi-agent reinforcement learning 64
3.3.1 Problem definition 64
3.3.2 Techniques to simplify the Q-value function 66
3.3.3 Mean field actor-critic algorithm 69
3.4 Reward design for multi-agent reinforcement learning 74

3.5 Case Study 79
3.5.1 E-hailing driver repositioning under service charge 79
3.5.2 Multiclass taxi driver repositioning under congestion pricing 89
3.6 Summary 105

Chapter 4: Bilevel optimization for multi-agent route choice 107
4.1 Introduction 107
4.1.1 Single-agent route choice 107
4.1.2 Multi-agent route choice 108
4.1.3 MARL and Multi-agent Markov game 109
4.1.4 Literature on reinforcement learning based route choices 110
4.1.5 Contributions of this chapter 112
4.2 Mean field multi-agent reinforcement learning 113
4.2.1 Single-agent reinforcement learning 113
4.2.2 Multi-agent reinforcement learning 118
4.3 Linkage between DTA and Markov games 130
4.3.1 Numerical example 133
4.4 Bilevel optimization for multi-agent route choice 142
4.4.1 Numerical example 143
4.5 Case Study 151
4.5.1 Setup of the bilevel problem 151
4.5.2 Results 154
4.6 Summary 156

Chapter 5: Conclusions and future research directions 158

5.1 Conclusions 158

5.2 Future research directions 160

5.2.1 Bounded rationality 160

5.2.2 Inverse reinforcement learning 160

5.2.3 Demand modeling 161

5.2.4 Upper-level control 161

References 176

List of Figures

1.1 Dissertation components 2

2.1 Sequential decision-making processes for traditional and e-hailing taxi drivers . . . 9
2.2 MDP state transition 14
2.3 Distribution of seeking time 23
2.4 Setup of the small grid world example and the corresponding state transition 26
2.5 Distribution of number of cases 31
2.6 Exponential fitting 32
2.7 Distribution of the rate of return and utilization rate from the field data 34
2.8 Optimal policy (at the beginning of the morning peak) 35
2.9 Distribution of the order matching probability P_{order_match} 36
2.10 Distribution of the rate of return and the utilization rate of real drivers and the agent 38
2.11 Distributions of the number of completed orders, number of idling cells, the service time per order, and the profit per unit time of orders 39
2.12 Distribution of service time of orders starting in all grids and in grids with a relatively high order matching probability 40
2.13 Average service time of orders starting in each subinterval 41
2.14 Two zoom-in views of the optimal policy and the distribution of the demand 42
2.15 Two zoom-in views of the optimal policy for three agents (red, yellow, and light blue denote the first, second, and third agent, respectively) 44
2.16 Distribution of radius of gyration 46

3.1 An illustrative example (Single agent) . . . 59
3.2 Actor-critic algorithm . . . 62
3.3 An illustrative example (Multi agent) . . . 65
3.4 Mean field actor-critic algorithm for multi-driver repositioning . . . 72
3.5 Derived Q values for four scenarios of interest . . . 74
3.6 Architecture of the reward design . . . 76
3.7 Layout . . . 80
3.8 Convergence of the lower level MARL . . . 83
3.9 Posterior probability distribution and acquisition function at iteration 0 . . . 84
3.10 Convergence of BO . . . 84
3.11 Posterior probability distribution of f after BO converges . . . 85
3.12 Stochastic analytical method . . . 88
3.13 Objectives of city planners . . . 90
3.14 Spatial discretization of the area of interest . . . 92
3.15 Spatial distribution of taxi orders during evening peak . . . 92
3.16 Convergence plot of pretraining actors . . . 98
3.17 Convergence plot of actors . . . 100
3.18 Convergence plot of the critic of yellow taxis . . . 100
3.19 Posterior probability distribution and acquisition function at iteration 0 . . . 101
3.20 Convergence of BO . . . 102
3.21 BO result . . . 102
3.22 Sensitivity analysis . . . 103
3.23 Spatial distribution of percentage of idle taxis in each grid and increase in crowdedness across the busiest 20 grids . . . 105

4.1 Braess network . . . 114
4.2 Q functions . . . 123
4.3 Connection between DTA and Markov game . . . 131

4.4 A two-node two-link network 133
4.5 Convergence plots for DSO 137
4.6 Convergence plots for DUE 140
4.7 Bilevel optimization model 143
4.8 Convergence of MF-MA-DQL algorithm in the case with single-batch demand . . . 145
4.9 Comparison between numerical solution and analytical solution with varying k_{13} . 146
4.10 Convergence of MF-MA-DQL algorithm in the case with multi-batch demand . . . 147
4.11 Posterior probability distribution and acquisition function at iteration 0 149
4.12 Posterior probability distribution of f 149
4.13 Road network in SUMO 152
4.14 Convergence of BO 154
4.15 Posterior probability distribution of f at the 8^{th} iteration 155
4.16 Decomposition of average travel time 156

List of Tables

2.1 Existing MDP based models on passenger-seeking strategy 8

2.2 Notations 12

2.3 Statistics of the comparison between the performance of the agent under different policies 39

2.4 Coefficients 47

3.1 Existing research of order dispatching and driver repositioning using MARL 55

3.2 Q-value bimatrix for drivers 74

3.3 Values of interest 86

3.4 Taxi data sample 91

3.5 Turnstile data sample 93

4.1 Existing research on dynamic routing using RL based approaches 111

4.2 Optimal values of interest 130

4.3 DSO versus DUE 141

Acknowledgements

First of all, I would like to express my everlasting gratitude to my parents and my love. This dissertation would not have been possible without their unconditional love and support.

In addition, I would like to give special thanks to my advisor, Professor Sharon Di, for all her guidance and help on my graduate research over the past four years. I would like to express my gratitude to the members of my dissertation committee, Professor George Deodatis, Professor Andrew Smyth, Professor Huiming Yin, and Professor Shipra Agrawal, for taking their valuable time to read and comment on this dissertation.

Also, I would like to thank my friends and fellow graduate students, Liang Wang, Gan Song, Xin He, Siyu Zhu, Qiliang Lin, Tengxiang Wang, Weiyi Li, Zengrong Hao, Ketson Roberto, Olga Brudastova, Yuki Miura, Mengyao Shen, Siyan Wang, and Maria Katsidoniotak, for making my life at Columbia more pleasurable. I would like to extend my gratitude to my lab mates, Rongye Shi, Zhaobin Mo, Xu Chen, Kuang Huang, Kangrui Ruan, and Yongjie Fu.

Last but no least, I would like to thank all the faculty and staff members in the department of Civil Engineering and Engineering Mechanics for providing me support throughout my doctoral studies at Columbia. I am so grateful to Columbia University.

To my family.

Chapter 1: Introduction and Background

1.1 Motivation

The emergence of transportation network companies (TNCs) or e-hailing platforms (such as Didi and Uber) has revolutionized 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 [1], partly due to the imbalance between demand (i.e., passenger requests) and supply (i.e., available drivers) that results in long cruising trips for taxi drivers to find the next passenger and long waiting time for passengers to be picked up [2, 3]. Such cruising behavior has negative impact on the sharing economy by not only decreasing drivers' income but also generating additional vehicle miles traveled. In addition, with an increasing number of vehicles competing for the limited road resources, traffic congestion has become a key challenge faced by modern cities. Traffic congestion not only affects drivers' mental and physical health [4] but also results in environmental and economic problems [5].

Fortunately, emerging sensing (such as connected vehicles, mobile phones, on-board cameras) and communication technologies (such as 5G or wireless), along with powerful computational resources (such as edge computing), provide an exceedingly large amount of data and thus hold the potential for optimal decision-making of fleet management, traffic operations, and smart city management. This dissertation aims to leverage big data and AI techniques to improve the efficiency of the urban economy with shared and autonomous mobility. In the near future, shared autonomous vehicles (SAVs) will serve as the main transportation mode to fulfill the urban travel demand. How to operate SAVs in terms of passenger-seeking, route choice, and pricing thus becomes a key challenge faced by the sharing economy.

1.2 Research objectives and scope

To improve the efficiency of the sharing economy, this dissertation aims to build an agent-based methodological framework for optimal decision-making of SAVs. An SAV typically has two tasks, namely passenger-seeking and route choice. These two tasks are not parallel; instead, they are sequential. First, an SAV decides where to go. This “where" is either where to find the next passenger when the SAV is idle (i.e., passenger seeking) or the destination of the passenger when the SAV is on duty. The SAV then decides how to get there (i.e., route choice). In addition to SAVs, city planners can achieve some systematic goal by imposing externalities on individual SAV’s cost via various operational measures such as toll pricing which can impact the behavior of SAVs. In summary, the objectives of this dissertation are:

1. building an agent-based methodological framework for optimal decision-making of SAVs, including passenger-seeking and route choice;

2. formulating a bilevel optimization model to derive optimal operational measures, with which city planners can achieve some systematic goal.

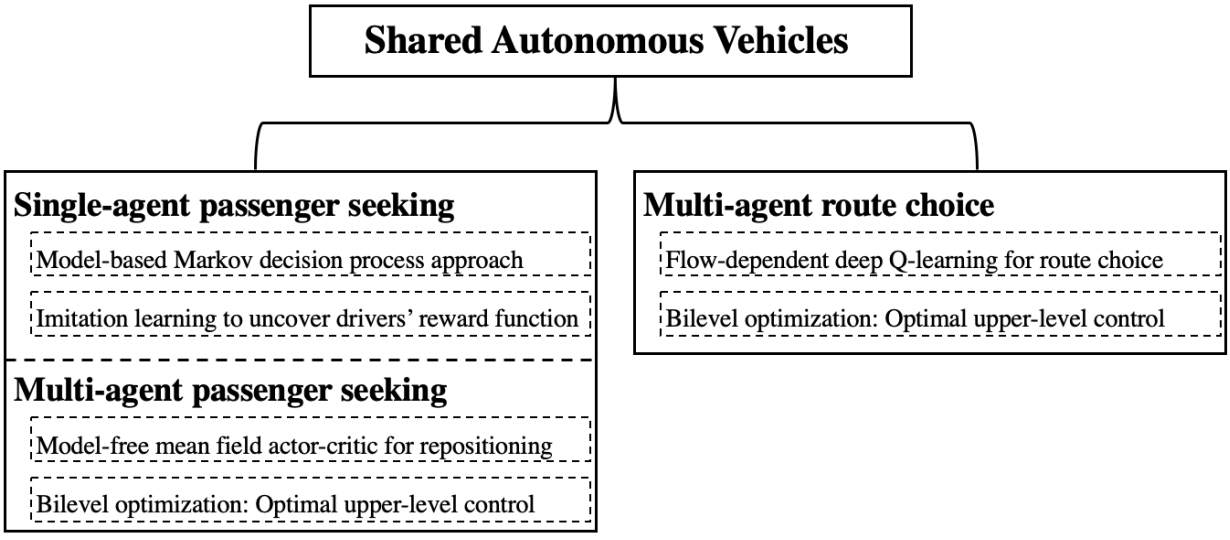


Figure 1.1: Dissertation components

Accordingly, the scope of this dissertation is schematically presented in Figure (1.1). SAVs have two major tasks, namely passenger-seeking and route choice. In the task of passenger-seeking, a model-based Markov decision process (MDP) approach is adopted in the single-agent setting, and a model-free multi-agent reinforcement learning (MARL) approach is developed in the multi-agent setting. A bilevel optimization model is formulated in the multi-agent setting to solve for the optimal upper-level control (i.e., service charge and congestion pricing) of city planners. In the task of route choice, a novel flow-dependent deep Q-learning approach is proposed to tackle the en-route dynamic routing of multiple agents. A bilevel optimization model is formulated to solve for the optimal upper-level control (i.e., toll pricing and traffic signal control) of city planners.

Note that in this research, each SAV is treated as a selfish and perfectly rational agent. Nevertheless, the developed methodological framework is versatile and can be applied to human drivers with bounded rationality. The notion of SAV is used to ensure behavioral compliance.

1.3 Research contributions

First of all, although passenger-seeking and route choice are not new research tasks, integrating decision-making of agents (i.e., lower-level optimization) and operational control of city planners (i.e., upper-level optimization) has been lacking in the literature. A bilevel optimization model is formulated for both the passenger-seeking and route choice tasks. In this dissertation, the bilevel optimization model is extensively applied to four cases, namely multi-driver passenger-seeking with service charge as the upper-level control, multi-driver passenger-seeking with congestion pricing as the upper-level control, multi-driver route choice with toll pricing as the upper-level control, and multi-driver route choice with traffic signal control, which demonstrate the versatility and the effectiveness of the bilevel optimization model.

Second, in the single-agent passenger-seeking, instead of following the literature where a known reward function is given based on some prior knowledge or assumptions [1, 6, 7, 8], this research unveils the underlying reward function of the overall e-hailing driver population and crafts a novel reward function which explains the behaviors of drivers with a relatively small radius of gy-

ration and thus paves the way for future research on discovering the underlying reward mechanism in a complex and dynamic ride-sharing market. In addition, this is the first research using large amounts data to devise and calibrate a dynamic adjustment strategy of the order matching probability to partly address the competition among multiple drivers. The strategy is further verified to be efficient in providing different recommendations for multiple drivers.

Third, with the single-agent passenger-seeking as a stepping stone, a multi-agent mean field actor-critic approach is developed to enable the learning of more than tens of thousands of agents and better capture the competition among them. A multi-class MARL is then proposed to account for the intrinsic behavioral difference between yellow taxis drivers and green taxi drivers in NYC.

Last but not least, a flow-dependent mean field deep Q-learning algorithm is developed to tackle the route choice task of multi-commodity multi-class agents. The flow-dependent mean action not only partially captures the competition among agents but also enables Q-value sharing and policy sharing. In addition, the linkage between the classical DUE paradigm and the proposed MARL paradigm is demonstrated. This research is the first-of-its-kind to unify the model-based (i.e., DUE) and data-driven (i.e., MARL) paradigms for dynamic routing games.

1.4 Dissertation organization

- Chapter 2 introduces a model-based MDP approach. Imitation learning is adopted to uncover the underlying reward function of the overall e-hailing driver population.
- Chapter 3 presents a model-free MARL approach for passenger-seeking of multi-class agents. A bilevel optimization model is formulated with service charge or congestion pricing as the upper-level control.
- Chapter 4 details a flow-dependent deep Q-learning approach for route choice of multi-commodity multi-class agents. A bilevel optimization model is formuated with toll pricing or traffic signal control as the operational measures of upper-level city planners.
- Chapter 5 concludes this dissertation and presents future research directions.

Chapter 2: Passenger-seeking for e-hailing drivers

2.1 Literature review

Taxis, complementary to massive transit systems such as bus and subway, provide flexible-route door-to-door mobility service. However, taxi drivers usually have to spend 35-60 percent of their time on cruising to find the next potential passenger [2]. Such passenger-seeking process not only decreases taxi drivers' income but also generates additional vehicle miles traveled, adding congestion and pollution into the increasingly saturated roads.

Cruising is primarily caused by an imbalance between travel demand and supply. Market regulation [9] or taxi fare structure design [10, 11, 12] were proposed respectively to balance taxi travel demand and supply. A network equilibrium model was developed [13, 14] to capture the spatial imbalance between travel demand and supply, where a logit-based probability was introduced to describe the meeting between a vacant taxi and a waiting passenger. This model, in which a taxi driver is supposed to minimize the individual search time for the next passenger, is further extended to incorporate congestion effects and customer demand elasticity [15], to include the fare structure and fleet size regulation [9], to consider multiple user classes, multiple taxi modes, and customer hierarchical modal choice [16], and to use a meeting function to describe the search frictions between vacant taxis and waiting passengers [17, 18, 19]. Recently, Di and Ban [20] proposed a unified equilibrium framework to model the shared mobility in congested road network.

As taxis GPS trajectories become increasingly available, qualitative analysis has been performed to uncover drivers' actual searching strategy. Liu et al. [21] found that drivers with higher profits prefer to choose routes with higher speed in both operational and idle states. Li et al. [22] discovered that hunting is a more efficient strategy than waiting by comparing profitable and non-profitable drivers. Several logit-based quantitative models were developed to capture idle drivers'

searching behavior [23, 24, 25, 26, 27, 28]. The bilateral searching behavior (i.e., taxi searching for customers and customers searching for taxis) was modeled through an absorbing Markov chain approach [29]. A probabilistic dynamic programming routing model was proposed to capture the taxi driver's routing decisions at intersections [30]. Furthermore, a two-layer approach, in which the first layer models the driver's pick-up location choice and the second layer accounts for the driver's detailed route choice behavior, was presented [31]. Recently, Zhang et al. [32] proposed an image-based representation of taxi drivers' passenger searching strategies and identified twenty four strategies using a dataset collected in Shenzhen, China.

Upon the understanding of drivers' searching behavior, recommendations can be provided to idle drivers on where to find the next passenger. An accurate prediction of both taxi supply [33] and demand [34, 35, 36, 37] as well as travel time [38, 39] are stepping stones to these recommendations. The objectives that recommendations aim to achieve include the minimization of waiting time at the recommended location [40] or of the distance between the current location and the recommended location [2, 40], and the maximization of the expected fare for the next trip [2, 40], of the probability of finding a passenger [41], or of the potential profit of a driver [42, 43].

The aforementioned studies mainly focused on the recommendation of the cruising routes or next cruising locations at the immediate next step without considering the optimization of long-run payoffs. A recommended customer searching strategy may help a driver to get an order as fast as possible but may not maximize this driver's overall profit in one day. Models which can capture drivers' long-term optimization strategy are needed. In recent years, Markov decision process (MDP) becomes increasingly popular in optimizing a single agent's sequential decision-making process given a period of time [44]. Several studies [6, 7, 8, 45, 46] have employed MDPs to model idle drivers' optimal searching strategy. In an MDP, an idle driver is an agent who makes sequential decisions of where to go to find the next passenger in a stochastic environment. The environment is characterized by a Markov process and transitions from one state to another once an action is specified by the idle driver. The driver aims to select an optimal policy which optimizes her long-term expected reward. Dynamic programming or Q-learning approaches are commonly

used to solve an MDP [47]. Table (2.1) summarizes the existing studies using MDPs for passenger-seeking optimization. Note that in e-hailing, there is actually no passenger-seeking because it is the e-hailing platform that matches an idle e-hailing driver to a passenger. However, e-hailing drivers still need to *reposition* themselves in order to get better chance of getting matched to a passenger request. In this dissertation, we will use the terminologies passenger seeking and repositioning interchangeably.

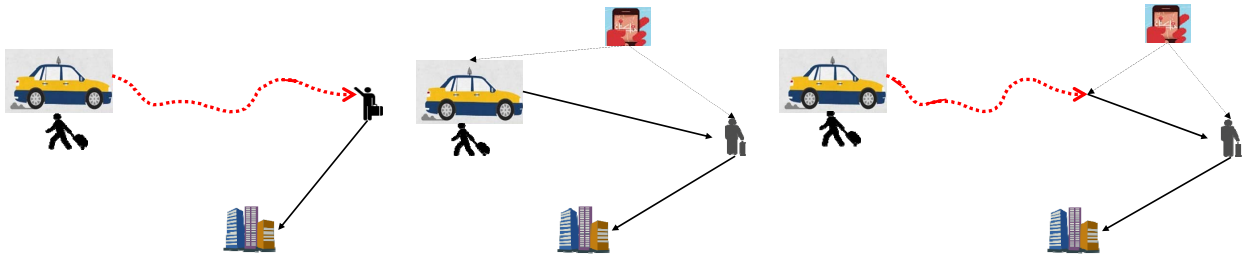
The existing MDP models were primarily developed for traditional taxi drivers' sequential decision-making where a driver has to see a passenger before a match happens. In other words, an idle driver's searching process ends only when this driver sees a passenger and the passenger accepts the ride (see Figure (2.1a)). E-hailing applications (such as Didi and Uber), on the other hand, offer an online platform to match a driver with a passenger even when they are not present in the same space at the same time [48, 49]. In other words, even when an idle driver sees a passenger waiting on the roadside, as long as the e-hailing platform does not match them, the driver cannot give a ride to the passenger. However, it does not mean e-hailing drivers always stay at the previous drop-off spot and wait for the platform to match. Drivers tend to *reposition* themselves so that the platform can find them a match sooner. As a result, the decision-making process of e-hailing drivers is quite different from the traditional taxi drivers in the following aspects:

1. An e-hailing driver may receive a matched order before she drops off the previous passenger, thus there is no passenger seeking (see Figure (2.1b)).

2. Different from traditional taxi that a driver has to see a passenger to find a match, e-hailing platforms very likely find a match even when the driver and the passenger are spatially far from each other. In other words, a driver's search process may end before a passenger is picked up (see Figure (2.1c)).

Table 2.1: Existing MDP based models on passenger-seeking strategy

Reference	Network representation	State space	Action space	Reward	Algorithm
[6]	Grid world	(grid id, time, incoming direction)	moving to a neighboring grid or staying in the current grid	Taxi fare	Dynamic programming
[7]	Grid world (static and dynamic zone structure)	(day-of-week, grid id, time-interval)	moving to any chosen grid (proposed an action detection algorithm)	Taxi fare - traveling distance cost - time cost	Q-learning (Monte Carlo)
[46]	Grid world	(grid id, operating status)	driving vacantly to neighboring grids to search, finding a passenger in the current grid, waiting static at the same spot	the ratio of the occupied taxi trip mileage to the previous empty mileage	Q-learning (Temporal Difference)
[8]	Link node	(node id, indicator of the current pick-up drop-off cycle)	outgoing links from the current node	taxi fare - operating cost	Value iteration
[1]	Grid world	(grid id, time interval, global state)	moving into a neighboring grid or staying in the current grid	taxi fare - operating cost	Reinforcement Learning (Deep Q learning)



(a) Traditional taxi driver

(b) E-hailing driver: no searching

(c) E-hailing driver: picking up in a different location

Figure 2.1: Sequential decision-making processes for traditional and e-hailing taxi drivers

Because of the inherent differences in drivers' decision-making, this chapter aims to develop an MDP to model e-hailing drivers' sequential decision-making in searching for the next passenger. 44,160 Didi drivers' 3-day GPS trajectories are used to calibrate and validate our model. Previously, there is research using Didi's data for the study of large-scale fleet management [1] and large-scale order dispatch [50] in e-hailing platforms.

The major contributions of this chapter are as follows: (1) Instead of following the literature where a known reward function is given based on some prior knowledge or assumptions [1, 6, 7, 8], this work unveils the underlying reward function of the overall e-hailing driver population and crafts a novel reward function which explains the behaviors of drivers with a relatively small radius of gyration and thus paves the way for future research on discovering the underlying reward mechanism in a complex and dynamic e-hailing market. With the incomplete and noisy observed policy, this work first extracts the underlying reward function and then solves an MDP to derive the optimal policy which completes and corrects the observed policy. (2) To the best of our knowledge, this is the first study using large amounts data to devise and calibrate a dynamic adjustment strategy of the order matching probability to address the competition among multiple drivers. The strategy essentially attenuates the order matching probability in an exponential manner for subsequent drivers to be guided into a grid when some drivers have already entered the grid. The strategy is further verified to be efficient in providing different recommendations for multiple drivers.

The remainder of the chapter is organized as follows. Section (2.2) introduces our modified

MDP model and details definitions of states, actions, and state transitions and the process of extracting parameters from the data. Section (2.3) presents the proposed dynamic adjustment strategy of the order matching probability and details the calibration process. Section (2.4) introduces the data we used in this research and presents the results, including the derived optimal policy and the Monte Carlo simulation. Section (2.5) summarizes this chapter.

2.2 Markov decision process (MDP) for a single agent

2.2.1 Preliminaries

An MDP is specified by a tuple (S, A, R, P, s_0), where S denotes the state space, A stands for the allowable actions, R collects rewards, P defines a state transition matrix, and s_0 is the starting state. Given a state $s_t = s \in S$ and a specified action $a_t = a \in A$ at time t, the probability of reaching state s' at time $t + 1$ is determined by the probability transition matrix $P(s, a, s')$, which is defined as

$$P(s, a, s') \equiv Pr(s_{t+1} = s' | s_t = s, a_t = a). \tag{2.1}$$

From the initial state s_0, the process proceeds repeatedly by following the dynamics of the environment defined by the Equation (2.1) until a terminal state (i.e., either the current time exceeds the terminal time or the current state is an absorbing state) is reached. An MDP satisfies the Markov property which essentially says that the future process is independent on the past given the present, i.e.,

$$Pr(s_{t+1} = s' | s_0, a_0, \cdots, s_{t-1}, a_{t-1}, s_t, a_t) = Pr(s_{t+1} = s' | s_t = s, a_t = a). \tag{2.2}$$

There are two types of value functions in MDPs, namely state value $V(s)$ and state-action value $Q(s, a)$. The actions that an agent will take form a policy π, which is a mapping from a state s and an action a to the probability $\pi(a|s)$ of taking action a at state s. Then the value function of a state s by following the policy π, denoted as $V_\pi(s)$, can be taken as the expectation of the future rewards,

i.e.,

$$V_\pi(s) = \mathbb{E}_\pi \left[\sum_{k=0}^{K} \gamma^k r_{t+k+1} | s_t = s \right], \tag{2.3}$$

where γ is a discount factor. The state-action value of taking action a at state s by following policy π is

$$Q_\pi(s, a) = \mathbb{E}_\pi \left[\sum_{k=0}^{K} \gamma^k r_{t+k+1} | s_t = s, a_t = a \right]. \tag{2.4}$$

The value function $V_\pi(s)$ is actually a weighted average of the state-action value $Q_\pi(s, a)$, i.e.,

$$V_\pi(s) = \sum_{j=1}^{J} \pi(a = a_j | s) Q_\pi(s, a = a_j), \tag{2.5}$$

where $\pi(a = a_j|s)$ is again the probability of taking action a_j at state s according to policy π, and J is the total number of actions that are allowed to be taken in state s.

Several algorithms have been developed to solve the MDP, i.e., to derive the optimal policy, and the corresponding value functions, such as the dynamic programming method and the Q-learning approach [47]. The dynamic programming algorithm is used in this work and will be explained later in Section (2.2.2.4). Furthermore, these two types of value functions at optimality are related by the following mechanism

$$V(s) = \max_a Q(s, a). \tag{2.6}$$

The rationale underlying the relationship at optimality is simply to choose a policy which maximizes the value function expressed in Equation (2.5). For example, when an agent is at state s, the optimal policy simply suggests the agent to take an action a with the largest state-action value, i.e., $\pi(\arg\max_a Q(s, a)|s) = 1$ (i.e., the probability of taking action $\arg\max_a Q(s, a)$ in state s is 1). Accordingly, the state-action value at optimality can be written as

$$Q(s, a) = \sum_{s'} Pr(s'|s, a)(V(s') + r(s, a, s')), \tag{2.7}$$

where $Pr(s'|s, a)$ is the probability of landing in state s' after taking action a in state s, and $r(s, a, s')$

is the reward for choosing action a at state s and landing in state s'.

2.2.2 MDP for e-hailing drivers

In this section, we will develop an MDP model for e-hailing drivers' stochastic passenger-seeking process. Notations which will be used in the subsequent analysis are listed in Table (2.2).

Table 2.2: Notations

Variable	Explanation
l	Index of the current grid
t	Current time
I	Indicator, denoting whether the driver has been matched to a request before the next drop-off
s	State, $s = (l, t, I)$
S	State space, a collection of all states
a	Action
A	Action space
$t_{seek}(l_a)$	Time spent on seeking for a passenger in grid l_a
$t_{drive}(l, k)$	Time spent on moving from grid l to grid k
$d_{seek}(l_a)$	Distance traveled when seeking for a passenger in grid l
$d_{drive}(l, k)$	Distance traveled for moving from grid l to grid k
$p_{order\_match}(l_a)$	The probability that the driver can be matched to a request during cruising in grid l_a
$p_{pickup}(l_a, l'')$	The probability of picking up a passenger in grid l'' when the request from the passenger was matched to the driver in grid l_a
$p_{dest}(l'', l''')$	The probability of dropping off a passenger in grid l''' when the passenger was picked up in grid l''
$p_{match}(l'', l''')$	The probability of receiving a new request before the driver finishing her current order at grid l'''
$f(l'', l''')$	The average taxi fare from grid l'' to grid l'''
α	Coefficient of fuel consumption and other operating costs per unit distance

2.2.2.1 States

In our MDP model, state $s = (l, t, I)$ consists of three components, namely grid index $l \in L$, current time $t \in T$, and an indicator $I \in \{0, 1\}$. Note that a hexagonal grid world setting with 6,421 grids is adopted in this chapter and will be explained later. Considering the fact that an e-hailing driver may receive a request before she drops off the previous passenger, we have thus

added an indicator into the state. The indicator denotes whether the driver has been matched to a request before she arrives at the current state. Accordingly, states with indicator 0 are decision-making states in which the driver needs to spend time on seeking the next passenger, and states with indicator 1 are non-decision-making states. For example, $(1, 2, 1)$ is a non-decision-making state which says that the driver is in grid 1 when $t = 2$ and the driver has already been matched to a request so she will not spend time on seeking at the current state.

2.2.2.2 Actions

In decision-making states, the driver has to choose one from eight allowable actions, denoted as A. In non-decision-making states, the driver will not take any action but drive to pick up the next passenger and transport the passenger to the destination. Among the allowable action space A, each of the first six actions is to transit from the current grid to one of the six neighbor grids. Note that some of the six neighboring grids may be non-reachable, we thus add a large penalty, i.e., a large distance, to the transition from a grid to a non-reachable neighboring grid to prevent the agent from taking the action which leads the agent to the non-reachable neighboring grid. The seventh action is to stay and cruise around within the current grid. The last action is to wait in the current grid. We stress that the last two actions are essentially different because from the data we have observed that some drivers will just wait near the previous drop-off spot, especially when they are around downtown or transportation terminals while some drivers usually cruise within the current grid after completing a ride. In addition, the fuel cost associated with waiting can be neglected while that of staying can be substantial because the driver keeps cruising around during his/her staying in the current grid. Furthermore, drivers can take a rest and refresh their minds during waiting and hence their driving strategy can be more efficient for future trips. These arguments, however, do not necessarily suggest that the driver should always choose waiting rather than staying. Actually, drivers have to cruise around to get closer to the potential requests under certain circumstances.

2.2.2.3 State transition

After completing a ride, there are two possible scenarios according to two different values of the indicator. If the indicator is 0, the driver needs to specify an action, i.e. where to find the next passenger, and then moves into the grid along the direction defined by the action and spends some amount of time seeking for the next passenger in the new grid. There are two possible outcomes associated with this passenger seeking process. Either the driver confirms a request and ends up arriving at a different grid by following the passenger's travel plan or the driver fails to find a request and stays in the current grid. The reward is usually positive for the former while negative for the latter due to the fuel consumption and other operating costs. If the indicator is 1, the driver drives to the pick-up spot and then transports the passenger to the destination without any passenger seeking involved.

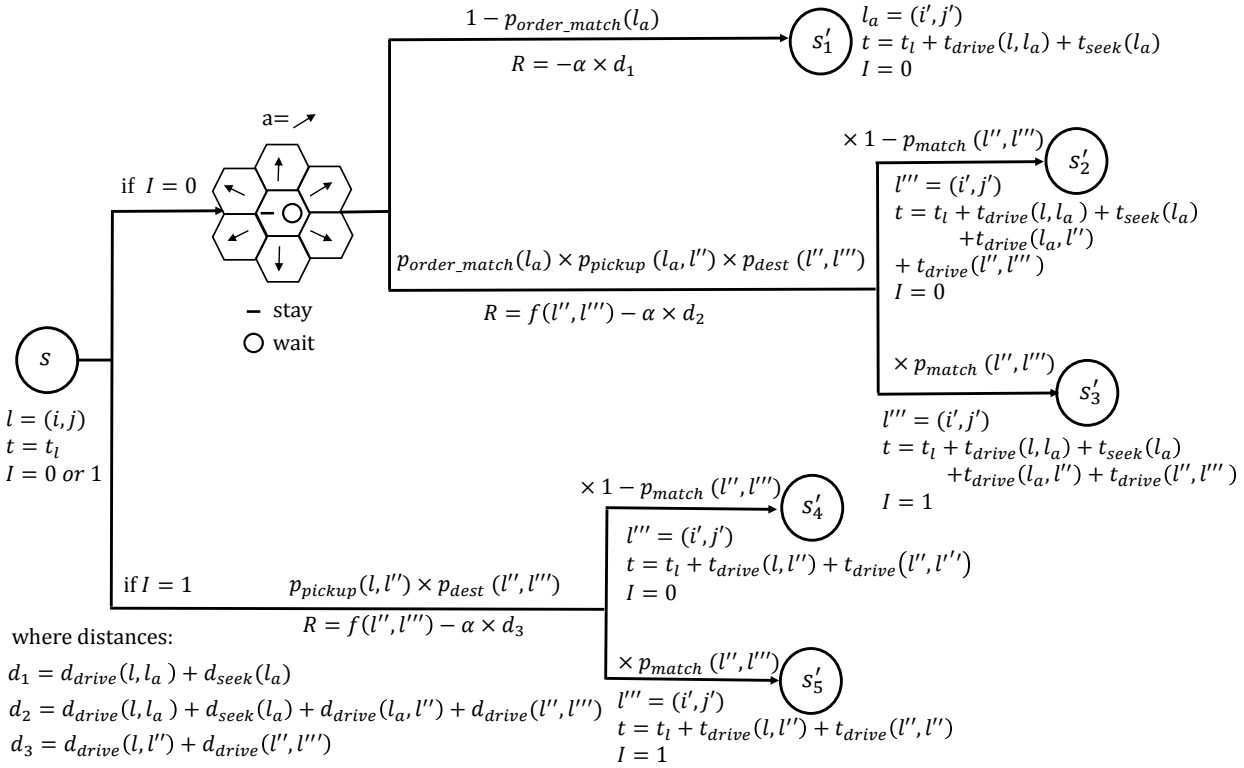


Figure 2.2: MDP state transition

Figure (2.2) illustrates the aforementioned state transition process. The driver currently stays at state $s = (l, t, I)$.

If $I = 0$, the driver specifies an action a, which is assumably taken as going northeast in the demonstration. Then the driver moves into the grid l_a along the direction defined by the action a and thereafter spends some amount of time $t_{seek}(l_a)$ on seeking the next passenger. There are two possible outcomes of this passenger seeking process.

The first possibility is that the driver fails to get any request in grid l_a after $t_{seek}(l_a)$. In this case, the state of the driver will be $s'_1 = (l', t + t_{drive}(l, l_a) + t_{seek}(l_a), 0)$. The reward for this passenger seeking process is $R = -\alpha(d_{drive}(l, l_a) + d_{seek}(l_a))$, which is negative. Let $P_{order\_match}(l_a)$ denote the probability that the driver will receive at least one request in grid l_a. Then the probability of the occurrence of this outcome is $1 - P_{order\_match}(l_a)$. In other words, with probability $1 - P_{order\_match}(l_a)$, the driver will end up in state $s'_1 = (l_a, t + t_{drive}(l, l_a) + t_{seek}(l_a), 0)$.

The second possibility is that the driver confirms one request during the cruising process in l_a. The probability of the occurrence of this outcome is $P_{order\_match}(l_a)$. We let $P_{pickup}(l_a, l'')$ denote the probability of confirming a request in grid l_a and picking up the passenger in grid l''. Once the passenger is on board, the driver will directly move to the destination l''', which only depends on the passenger's travel plan. We let $P_{dest}(l'', l''')$ denote the probability of picking up a passenger in grid l'' and dropping off the passenger in grid l'''. After dropping off the passenger, the driver will end up in grid l''' at time $t + t_{drive}(l, l_a) + t_{seek}(l_a) + t_{drive}(l_a, l'') + t_{drive}(l'', l''')$ and earn a reward of $r(l, l''') = f(l'', l''') - \alpha(d_{drive}(l, l_a) + d_{seek}(l_a) + d_{drive}(l_a, l'') + d_{drive}(l'', l'''))$. Hence, the probability of the driver receives a request in grid l_a, picks up the passenger in grid l'', and transports the passenger to grid l''' is $P_{order\_match}(l_a) \times P_{pickup}(l_a, l'') \times P_{dest}(l'', l''')$. Notice that for a driver, during her trip from the passenger's origin l'' to the passenger's destination l''', there is a probability at which the driver will confirm a request before she drops off the passenger. Let $P_{match}(l'', l''')$ denote the probability of receiving a request before the driver reaches grid l'''. Then we can conclude that with probability $P_{order\_match}(l_a) \times P_{pickup}(l_a, l'') \times P_{dest}(l'', l''') \times (1 - P_{match}(l'', l'''))$, the driver will end up in state $s'_2 = (l''', t + t_{drive}(l, l_a) + t_{seek}(l_a) + t_{drive}(l_a, l'') + t_{drive}(l'', l'''), 0)$, and with probability $P_{order\_match}(l_a) \times P_{pickup}(l_a, l'') \times P_{dest}(l'', l''') \times P_{match}(l'', l''')$, the driver will end up in state $s'_3 = (l''', t + t_{drive}(l, l_a) + t_{seek}(l_a) + t_{drive}(l_a, l'') + t_{drive}(l'', l'''), 1)$.

If $I = 1$, the driver will not need to specify any action and will directly drive to the pick-up spot of the next passenger and then transport the passenger to the destination. Again, during her trip to the passenger's destination, there is a probability, denoted as $P_{match}(l'', l''')$, at which the driver will receive a request before she drops off the passenger. As illustrated in Figure (2.2), with probability $P_{pickup}(l, l'') \times P_{dest}(l'', l''') \times (1 - P_{match}(l'', l'''))$, the driver will end up in state $s'_4 = (l''', t + t_{drive}(l, l'') + t_{drive}(l'', l'''), 0)$; with probability $P_{pickup}(l, l'') \times P_{dest}(l'', l''') \times P_{match}(l'', l''')$, the driver will end up in state $s'_5 = (l''', t + t_{drive}(l, l'') + t_{drive}(l'', l'''), 1)$.

In both scenarios, namely either $I = 0$ or $I = 1$, the driver will thereafter start the whole process from s' again until the current time exceeds the time interval, i.e., a terminal state has been reached.

2.2.2.4 Solving MDP

The objective of the MDP model is to maximize the total expected revenue of a driver. Considering the fact that in a time interval, a driver can finish a finite number of pick-up and drop-off cycles, indicating that the MDP model is finite-horizon. When current time of the driver has reached the end of the time interval, no more actions can be taken and no more rewards can be earned. Suppose a driver is currently at state $s = (l, t, I)$. If $I = 0$, meaning that the driver is at a decision-making state, the maximum expected revenue that a driver can earn by starting from s and specifying an action a is

$$\begin{aligned} Q(s,a) \quad = \quad & (1 - P_{order\_match}(l_a)) \times [-\alpha(d_{drive}(l, l_a) + d_{seek}(l_a)) + V^*(s'_1)] \\ & + \sum_{l'' \in L} \sum_{l''' \in L} P_{order\_match}(l_a) \times P_{pickup}(l_a, l'') \times P_{dest}(l'', l''') \\ & \quad \times [f(l'', l''') - \alpha(d_{drive}(l, l_a) + d_{seek}(l_a) + d_{drive}(l_a, l'') + d_{drive}(l'', l''')) \\ & \quad + (1 - P_{match}(l''')) \times V^*(s'_2) + P_{match}(l''') \times V^*(s'_3)], \end{aligned} \tag{2.8}$$

where $l_a = l_a(s, a)$, meaning that the grid l_a in which an e-hailing driver will be cruising is dependent on the current state s, actually through the grid index l of s, and the specified action a, $s'_1 = (l_a, t + t_{drive}(l, l_a) + t_{seek}(l_a), 0)$, $s'_2 = (l''', t + t_{drive}(l, l_a) + t_{seek}(l_a) + t_{drive}(l_a, l'') + t_{drive}(l'', l'''), 0)$,

$s'_3 = (l''', t + t_{drive}(l, l_a) + t_{seek}(l_a) + t_{drive}(l_a, l'') + t_{drive}(l'', l'''), 1)$, and $V^*(s'_1)$, $V^*(s'_2)$, and $V^*(s'_3)$ stand for the maximum expected revenue that a driver can earn by reaching state s'_1, s'_2, and s'_3, respectively. If $I = 1$, meaning that the driver is at a non-decision-making state, the driver will not specify any action, and the expected revenue that the driver can earn is

$$\begin{aligned} Q(s, .) \quad = \quad & \sum_{l'' \in L} \sum_{l''' \in L} P_{pickup}(l, l'') \times P_{dest}(l'', l''') \\ & \times [f(l'', l''') - \alpha(d_{drive}(l, l'') + d_{drive}(l'', l''')) \\ & + (1 - P_{match}(l'', l''')) \times V^*(s'_4) + P_{match}(l'', l''') \times V^*(s'_5)], \end{aligned} \tag{2.9}$$

where $s'_4 = (l''', t + t_{drive}(l, l'') + t_{drive}(l'', l'''), 0)$, $s'_5 = (l''', t + t_{drive}(l, l'') + t_{drive}(l'', l'''), 1)$, and $V^*(s'_4)$ and $V^*(s'_5)$ stand for the maximum expected revenue that a driver can earn by reaching state s'_4 and s'_5, respectively.

Then the optimal policy for a driver to follow at a decision-making state s is

$$\pi(s) = \arg\max_a [Q(s, a)], \tag{2.10}$$

and the maximum expected revenue that a driver can earn by reaching state s is

$$V^*(s) = \begin{cases} \max_a Q(s, a) & \text{if } s \text{ is a decision-making state;} \\ Q(s, .) & \text{if } s \text{ is a non-decision-making state.} \end{cases} \tag{2.11}$$

The policy in Equation (2.10) is deterministic, meaning that the driver can only take one action at the current decision-making state s if she follows the policy. Actually here we slightly abuse the notation. The policy is supposed to be $\pi(\arg\max_a [Q(s, a)]|s) = 1$, i.e., the probability of taking action $\arg\max_a [Q(s, a)]$ at state s is 1. It is equivalent to say that at state s, the action to take is $\arg\max_a [Q(s, a)]$, and thus we write the policy at state s as Equation (2.10). A deterministic policy defines a one-to-one mapping from a state to an action. The deterministic policy works when there is only one driver who learns the optimal policy and follows the policy. Otherwise, there might

be excess taxi supply at some areas, resulting in a localized competition among taxis. A circulating mechanism was employed to tackle this overload problem [41]. A multi-agent reinforcement learning approach [1] was proposed to consider the competition among drivers. In this research, we use a dynamic adjustment strategy to update the order matching probability when multiple idling e-hailing drivers are guided into the same grid. The proposed dynamic adjustment strategy will be introduced in Section (2.3).

To efficiently solve the MDP, i.e., to derive an optimal policy, a dynamic programming approach is employed [47, 51]. The basic idea of the dynamic programming algorithm is to divide the overall problem into subproblems and hence to make use of the results of the subproblems to solve the overall problem. An important advantage of the dynamic programming algorithm is that it caches results of all subproblems and thus it is guaranteed that the same subproblem is only solved once.

Now we elucidate how we apply the dynamic programming algorithm to solve the MDP. The goal is to solve the optimal value for all states $s = (l, t, I)$, where $l \in L$, $t \in \{0, 1, 2, \cdot, 180\}$, and $I \in \{0, 1\}$. There are in total $6,421 \times 181 \times 2 = 2,324,402$ states, and half of them are decision-making states. We define one subproblem as solving the optimal value for one state and thus we have in total $2,324,402$ subproblems. Noticing that at the final time step, i.e., $t = T = 180$, the maximum expected reward that a driver can earn is obviously zero, we thus have $V^*(s) = 0$ for all states s where $t = T$. For any state s with $t < T$ and a chosen action a, the calculation of the state-action value $Q(s, a)$ depends on the value of some future states, i.e., s'_1, s'_2, and s'_3 in Equation (2.8). In other words, the subproblem, i.e., solving the optimal value for state s, depends on some subproblems, i.e., solving the optimal value of some future states, e.g., s'_1, s'_2, and s'_3. For a future state s', there might be several states s from which the agent will reach the future state s', indicating the calculation of the optimal value of all these states s requires the calculation of the optimal value of the future state s', resulting in calculating the optimal value of the same state s' multiple times and thus wasting computation power. To avoid the repeated calculation of the optimal value for the same state, we adopt the dynamic programming algorithm. Since the optimal

values for all states with $t = T$ are known and the optimal value of a state s depends on the optimal value of some future states, we solve the optimal value of states backwards in time and simply store the solved optimal values in a hash table. Then for a state s and a chosen action a, we simply read the optimal values of future states s'_1, s'_2, and s'_3 from the hash table and use Equation (2.8) to calculate the state-action value $Q(s, a)$, based on which the optimal value of the state s can be derived from Equation (2.11). The pesudo code is in Algorithm (1).

Algorithm 1 Dynamic programming algorithm

Input: L, $T = 180$, A, P_{order_match}, P_{pickup}, P_{dest}, P_{match}, fare f, d_{drive}, d_{seek}, t_{drive}, t_{seek}
Initialize: a hash table V^* to store optimal values
$V^*(s) = 0$ for all states s where $t = T$
for $t = T - 1$ `to 1` **do**
    for $l \in L$ **do**
        Form a decision-making state $s = (l, t, I = 0)$
        for $a \in A$ **do**
            Calculate $Q(s, a)$ by Equation (2.8), the optimal values of the dependent future states are read from the hash table V^*
        end for
        Derive the optimal policy for decision-making state s, $\pi(s)$, by Equation (2.10)
        Calculate the optimal value for state s, $V^*(s)$, by Equation (2.11)
        Form a non-decision-making state $s = (l, t, I = 1)$
        Calculate $Q(s, .)$ by Equation (2.9), the optimal values of the dependent future states are read from the hash table V^*
        Calculate the optimal value for state s, $V^*(s)$, by Equation (2.11)
    end for
end for
return the V^* and π

2.2.3 Extracting parameters from data

In the dataset we used in this research, we have GPS trajectories for both the empty and occupied trips. We now extract the parameters used in the state transition from the dataset.

2.2.3.1 Order matching probability P_{order_match}

The order matching probability estimates the probability at which a vacant taxi can be matched to a passenger when the taxi is cruising, including staying, or waiting at grid l_a. As we have

mentioned before, the purposes for introducing waiting and staying in this work are different. In addition to six actions which allow an e-hailing driver to move into one of the six neighboring grids, the action staying gives the driver extra flexibility in choosing to stay and cruise within the current grid due to some potential benefits, such as a relatively high order matching probability in the current grid, a possibly high cost to move into neighboring grids vacantly, etc. Actually, as we have listed in Table (2.1), there are several studies in the literature that have already included the action staying into the action space [1, 6, 7]. Thus, the way to calculate the order matching probability for staying is the same as the way to calculate the order matching probability for cruising into one of the six neighboring grids. In other words, the order matching probability for a driver just entering the grid from one of the six neighboring grids is supposed to be the same as the order matching probability for a driver who was in the grid and chose to stay in the grid.

Waiting, different from staying and other six actions which allow the driver to move into one of the six neighboring grids, is included into the action space based on the observation that sometimes a driver will choose to stop cruising and simply to wait statically for passenger requests to come in, especially when the driver is around downtown or transportation terminals. The action waiting was previously included in the action space in [46].

We thus approximate the order matching probabilities for cruising and waiting separately. We say a driver is waiting for a passenger request whenever the driver's traveling distance is less than 200 meters for a 3-minute interval. To rule out some unrealistic waiting actions, such as a driver being stuck in traffic, we further limit the possible locations for waiting to be the places around subway stations, bus terminals, airports, and some famous tourism attractions. For cruising, the order matching probability can be approximated as the ratio of the number of times that a taxi is matched to a passenger in grid l_a while cruising, denoted as $n_{order\_match\_cruising}(l_a)$, to the number of times that the grid l_a is passed by an empty taxi while cruising, denoted as $n_{passby\_cruising}(l_a)$. For waiting, the order matching probability can be approximated as the ratio of the number of times that a taxi is matched to a passenger in grid l_a while waiting, denoted as $n_{order\_match\_waiting}(l_a)$, to the number of times that empty taxis have waited in the grid l_a, denoted as $n_{passby\_waiting}(l_a)$.

$$P_{order\_match}(l_a) = \begin{cases} \dfrac{n_{order\_match\_cruising}(l_a)}{n_{passby\_cruising}(l_a)} & \text{if cruising;} \\ \dfrac{n_{order\_match\_waiting}(l_a)}{n_{passby\_waiting}(l_a)} & \text{if waiting.} \end{cases} \tag{2.12}$$

2.2.3.2 Pick-up probability P_{pickup}

The pick-up probability $P_{pickup}(l_a, l'')$ measures the likelihood of picking up a passenger at grid l'' when the request sent from the passenger was matched to the driver at grid l_a. This parameter can be estimated as the ratio of the the number of passenger pick-ups in grid l'' which were matched to drivers in grid l_a, denoted as $n_{pickup}(l_a, l'')$, to $n_{order\_match}(l_a)$, which is the summation of $n_{order\_match\_cruising}(l_a)$ and $n_{order\_match\_waiting}(l_a)$.

$$P_{pickup}(l_a, l'') = \frac{n_{pickup}(l_a, l'')}{n_{order\_match}(l_a)}. \tag{2.13}$$

2.2.3.3 Destination probability P_{dest}

The destination probability $P_{dest}(l'', l''')$ measures the likelihood of the destination of the passenger being grid l''' when the passenger was picked up in grid l''. This parameter can be estimated by dividing the number of trips ending in grid l''' which originated from grid l'', denote as $n_{dest}(l'', l''')$, by the total number of pick-ups in grid l'', denoted as $n_{pickup}(l'')$.

$$P_{dest}(l'', l''') = \frac{n_{dest}(l'', l''')}{n_{pickup}(l'')}. \tag{2.14}$$

2.2.3.4 Order matching probability while on trip P_{match}

As we have mentioned before, there is a probability at which the driver will receive a request when she is on the trip to transport the current passenger to the destination. We denote this order matching probability while on trip as P_{match}. This probability can be estimated by dividing the number of occupied trips among which there is at least one request received by the driver before

the driver reaching the destination l''' while the origin is l'', denoted as $n_{match}(l'', l''')$, by the total number of occupied trips ending in grid l''' and originating in grid l'', denoted as $n_{trips}(l'', l''')$.

$$P_{match}(l'', l''') = \frac{n_{match}(l'', l''')}{n_{trips}(l'', l''')}. \tag{2.15}$$

2.2.3.5 Driving time t_{drive} and driving distance d_{drive}

The driving time $t_{drive}(l, k)$ and the driving distance $d_{drive}(l, k)$ denote the estimated driving time and driving distance from grid l to grid k, respectively. Here we simply take the average of all driving times from grid l to grid k as an approximation of the $t_{drive}(l, k)$. Similarly, the driving distance is calculated by taking the average of all driving distances between grid l and grid k.

2.2.3.6 Taxi fare f

The taxi fare $f(l'', l''')$ denotes the estimated gross revenue that a driver can earn by transporting a passenger from grid l'' to her destination grid l'''. Here we take the average of all the fares of the occupied trips which are from grid l'' to grid l''' as a proxy of the real taxi fare from grid l'' to l'''.

2.2.3.7 Seeking time t_{seek} and seeking distance d_{seek}

The seeking time $t_{seek}(l_a)$ and the seeking distance $d_{seek}(l_a)$ denote the estimated seeking time and seeking distance within grid l_a, respectively. From the field data, the distribution of the seeking time in each grid was extracted and is shown in Figure (2.3). The median of the distribution of the seeking time is approximately 45 seconds. Since the time step size is 1 minute in this work, thus we simply take the seeking time as 1 minute. Considering the average speed of seeking trips (around 300 meters/minute), the seeking distance is taken as 300 meters for each grid. Note that the seeking distance is zero when the driver chooses to wait.

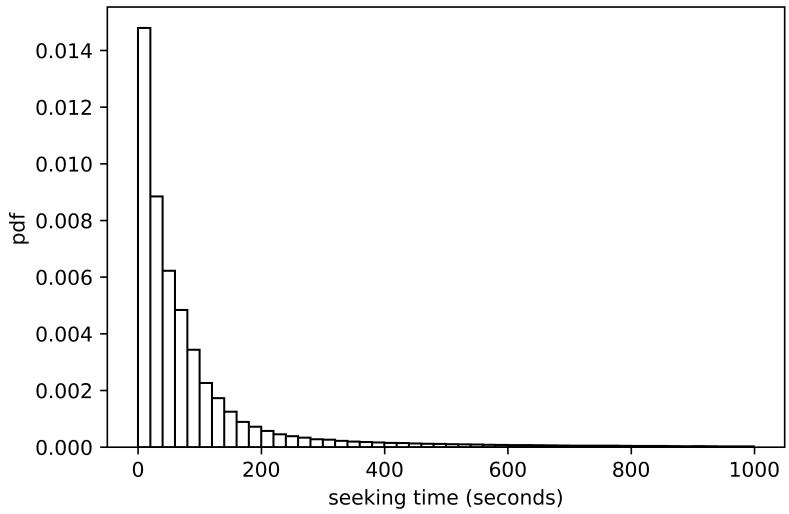


Figure 2.3: Distribution of seeking time

2.2.3.8 The fuel consumption coefficient α

α estimates the fuel consumption and other operating cost per unit distance during driving. In the literature, the value of α is typically assumed to be a known constant based on some common knowledge [1, 6, 7, 8]. This assumption, however, can easily result in a gap between the reward function in MDP models and the reward function of real drivers, meaning that drivers will not follow the optimal policy since the reward function used to derive the optimal policy is not the reward function of real drivers. To more appropriately determine α, we opt for the inverse reinforcement learning (IRL) approach. IRL is a powerful technique to disclose the underlying reward function based on the observed behaviors, especially in the field of imitation learning where an agent's behavior is observed by a learner who tries to imitate the agent [52].

A linear programming formulation of IRL in finite state spaces was first proposed in [53], where an extension to large state spaces was also made possible by adopting a linear function approximation. In the context of understanding the observed behaviors, Liu et al. [54] argued that a mixed integer linear programming formulation can be more revealing. Noticing the similarity between the two approaches, we adopt the mixed integer linear programming formulation for simplicity. For the purpose of self-explanatory, we provide a brief introduction of mixed integer linear

programming formulation. Interested readers are referred to [54] for detailed explanations.

In the linear programming formulation, the underlying reward function $R(s, s')$ from a state s to another state s' is expressed as a linear combination of some simple known reward functions ϕ_is, i.e.,

$$R(s, s') = \alpha_1\phi_1(s, s') + \alpha_2\phi_2(s, s') + \cdots + \alpha_n\phi_n(s, s'). \tag{2.16}$$

As an example, $\phi_i(s, s')$ can be either the fare a driver can collect from s to s' or the distance a driver traveled from s to s'. The former is considered to be positive while the latter is negative. For each simple reward function ϕ_i, the optimal value function V^{ϕ_i} can be derived by solving the MDP with ϕ_i as the reward function. Due to linearity, the optimal value function under the underlying reward function can be calculated as

$$V^R = \alpha_1 V^{\phi_1} + \alpha_2 V^{\phi_2} + \cdots + \alpha_n V^{\phi_n}. \tag{2.17}$$

The optimal policy π^R is also derived when solving the MDP. The objective of the mixed integer linear programming IRL is to minimize the difference between the optimal policy π^R and the observed policy π^O, i.e., minimize$\sum_{s\in S}[\pi^R(s) \neq \pi^O(s)]$. After some mathematical manipulations, a mixed integer linear programming formulation is formed as

$$\begin{aligned}
&\text{minimize} && \sum_{s\in S} C_s, && (2.18)\\
&\text{s.t.} \\
&&& \alpha_i \geq 0; && (2.19)\\
&&& \sum_{s'} P_{\pi^O}(s, s')(R(s, s') + \gamma V^{\pi^R}(s')) \\
&&& \quad - \sum_{s'} P_a(s, s')(R(s, s') + \gamma V^{\pi^R}(s')) + M \times C_s \geq 0, && (2.20)
\end{aligned}$$

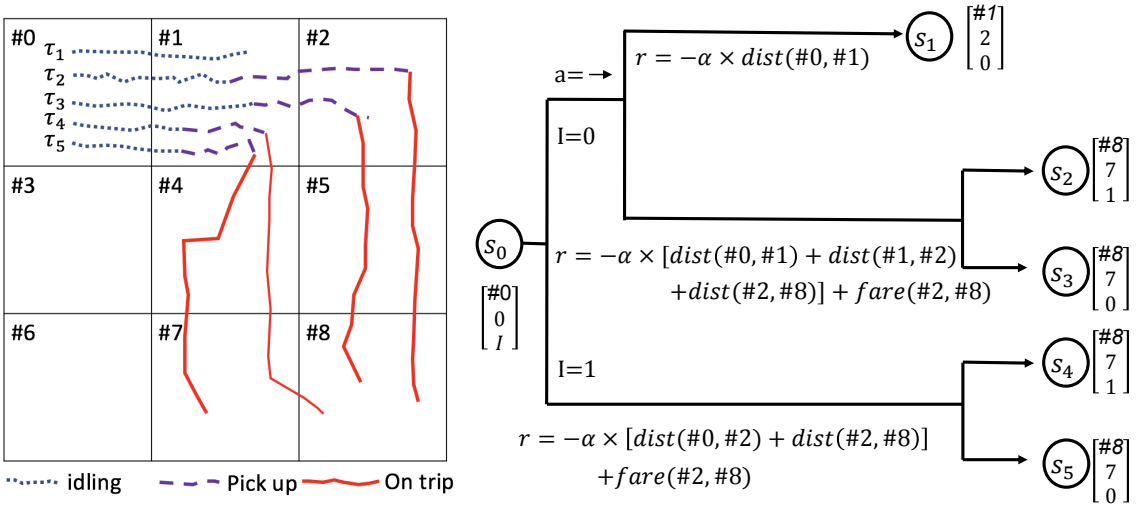
where C_s is a binary variable and takes value of 0 or 1 and M is an arbitrarily large number. $\sum_{s\in S} C_s$ denotes the number of states where the observed policy and the optimal policy differ. The objective is thus to minimize the difference between π^R and π^O. Constraint (2.19) restricts the weighting

parameter to be nonnegative. In the last constraint (2.20), the first term $\sum_{s'} P_{\pi^O}(s, s')(R(s, s') + \gamma V^{\pi^R}(s'))$ is the optimal value at state s following the observed policy, and the second term $\sum_{s'} P_a(s, s')(R(s, s') + \gamma V^{\pi^R}(s'))$ is the value following other policies. Therefore, it is expected that $\sum_{s'} P_{\pi^O}(s, s')(R(s, s') + \gamma V^{\pi^R}(s')) \geq \sum_{s'} P_a(s, s')(R(s, s') + \gamma V^{\pi^R}(s'))$ holds. In reality, however, this constraint can be violated, and thus we add a large positive number to the violated constraint to keep (2.20) hold. $P_\pi(s, s')$ is the probability of the transition $s \to s'$ following the policy π. In this work, $P_\pi(s, s')$ or $P_a(s, s')$ has been demonstrated in Figure (2.2).

Remark. To identify an overall reward function (i.e., determining α_is), we assume part of the observed policy is optimal or near optimal. Different from simply assuming a reward function based on common knowledge in the literature [1, 6, 7, 8], we argue that the assumption used here is relaxed to some degree. Real drivers, at least part of them, are deemed to be intelligent and experienced and exhibit optimal or near optimal strategies. For a state visited by multiple drivers for multiple times, we believe the most frequently taken action carries useful information about the optimal strategy in this state and reflects the crowd wisdom. After uncovering the underlying reward function, the purpose of solving the MDP and thus deriving the optimal policy is to complete and correct the optimal policy. The incompleteness and noise in the observed policy stem from the following aspects: (1) For a real-world problem with a huge number of states (in our case study in Section (2.4) the number of decision-making states is $6,421 \times 180 = 1,115,780$), a considerable portion of the states are not visited or at least not frequently visited; (2) Even in states with sufficient data (i.e., enough actions chosen by agents in this state), the most frequently chosen action which is taken as the observed policy in this state can still differ from the subsequent derived optimal policy, due to behavioral inconsistency; and (3) the observed policy, which is assumed to be deterministic, can be ambiguous when several actions share similar frequency in some states.

2.2.4 Numerical example

To illustrate the Markov decision process of e-hailing drivers, we use a 3 by 3 grid world numerical example, as shown in Figure (2.4a).



(a) Setup (b) State transition

Figure 2.4: Setup of the small grid world example and the corresponding state transition

Suppose now we have the following five trajectories.

1. $\tau_1 = (\#0, 0, 0) \xrightarrow{\text{idling}} (\#1, 2, 0)$
2. $\tau_2 = (\#0, 0, 0) \xrightarrow{\text{idling}} (\#1, 2, 0) \xrightarrow{\text{pickup}} (\#2, 3, 0) \xrightarrow[\text{no match}]{\text{ontrip}} (\#8, 7, 0)$
3. $\tau_3 = (\#0, 0, 0) \xrightarrow{\text{idling}} (\#1, 2, 0) \xrightarrow{\text{pickup}} (\#2, 3, 0) \xrightarrow[\text{match}]{\text{ontrip}} (\#8, 7, 1)$
4. $\tau_4 = (\#0, 0, 0) \xrightarrow{\text{idling}} (\#1, 2, 0) \xrightarrow{\text{pickup}} (\#1, 3, 0) \xrightarrow[\text{no match}]{\text{ontrip}} (\#8, 7, 0)$
5. $\tau_5 = (\#0, 0, 0) \xrightarrow{\text{idling}} (\#1, 2, 0) \xrightarrow{\text{pickup}} (\#1, 3, 0) \xrightarrow[\text{no match}]{\text{ontrip}} (\#7, 6, 0)$

Each element in the trajectory is a tuple consisting of three items, namely, the grid index, current time, and a status indicator showing if the driver has been matched to another order during the trip. For example, $(\#0, 0, 0)$ basically states that the driver is at grid #0 at time 0, and the driver has not been matched to any order before she finished the previous trip.

All five trajectories started in grid #0, and then the driver moved into grid #1 during idling. After the driver searched the grid #1, there are two possible outcomes: either the driver finds an order match or the driver fails to find any e-hailing order. If the driver fails to get an order match after searching, the driver will move into other grids to find another order or the driver will stop

working. To simplify the demonstration, we simply assume the trajectory τ_1 ends in grid #1. For other four trajectories, the driver managed to find an e-hailing order in grid #1. Based on this piece information, we can calculate the probability of finding an e-hailing order in grid #1 as $P_{order\_match}(\#1) = \frac{n_{order\_match}(\#1)}{n_{passby}(\#1)} = \frac{4}{5} = 80\%$.

After confirming an order match in grid #1, the driver drives into grid #2 to pick up the passenger in trajectories τ_2 and τ_3 and stays within grid #1 to pick up the passenger in trajectories τ_4 and τ_5 , respectively. Thus, the pick-up probability can be calculated as $P_{pickup}(\#1, \#1) = \frac{n_{pickup}(\#1, \#1)}{n_{order\_match}(\#1)} = \frac{2}{4} = 50\%$ and $P_{pickup}(\#1, \#2) = \frac{n_{pickup}(\#1, \#2)}{n_{order\_match}(\#1)} = \frac{2}{4} = 50\%$.

When the driver picks up the passenger in grid #1, as illustrated in trajectories τ_4 and τ_5, the passenger's destination is grid #8 in τ_4 and grid #7 in τ_5, respectively. Thus, the destination probability can be calculated as $P_{dest}(\#1, \#7) = \frac{n_{dest}(\#1, \#7)}{n_{pickup}(\#1)} = \frac{1}{2} = 50\%$ and $P_{dest}(\#1, \#8) = \frac{n_{dest}(\#1, \#8)}{n_{pickup}(\#1)} = \frac{1}{2} = 50\%$.

In trajectories τ_2 and τ_3, the driver drives to grid #2 to pick up the passenger, and the passenger goes to grid #8. During the trip, the driver has a P_{match} of receiving a new order before she arrives at the destination of the passenger. The order matching probability while on trip can thus be calculated as $P_{match}(\#8) = \frac{n_{match}(\#8)}{n_{trips}(\#8)} = \frac{1}{2} = 50\%$.

Based on the probabilities calculated above, an example of state transition is presented in Figure (2.4b). To make the state transition consistent with the five trajectories, we suppose the driver is initially in grid #1 with a status indicator I, i.e., the driver is in state $s_0 = (\#0, 0, I)$. If $I = 0$, meaning that the driver needs to seek for an e-hailing order, the driver drives into grid #1 and seeks for e-hailing orders in the grid. There are two possible outcomes associated with this case.

1. The driver fails to find any e-hailing order in grid #1. The driver will end up in state $s_1 = (\#1, 2, 0)$ and receive a negative reward $-\alpha \times dist(\#0, \#1)$, which is actually the fuel cost. This outcome happens with probability $p_1 = 1 - P_{order\_match}(\#1) = 1 - 80\% = 20\%$

2. The driver successfully finds an e-hailing order in grid #1. For the purpose of demonstration, we assume the driver goes to grid #2 to pick up the passenger, and the destination

of the passenger is grid #8. The probability of this outcome is $p_2 = P_{order\_match}(\#1) \times P_{pickup}(\#1, \#2) \times P_{dest}(\#2, \#8) = 80\% \times 50\% \times 100\% = 40\%$. The driver will receive a total reward of $r = fare(\#2, \#8) - \alpha \times [dist(\#0, \#1) + dist(\#1, \#2) + dist(\#2, \#8)]$ by completing this ride. During the trip, the driver may have a probability $P_{match}(\#8) = 50\%$ of getting a new request before she arrives at the destination of the previous passenger. Hence, there are two possible subbranches from this outcome.

(a) If the driver is matched to a new request before she drops off the previous passenger, then the driver will end up in state $s_2 = (\#8, 7, 1)$. The probability of the occurrence of this subbranch is $p_2 \times P_{match}(\#8) = 40\% \times 50\% = 20\%$.

(b) if the driver fails to be matched to another request while on trip, the driver will then end up in $s_3 = (\#8, 7, 0)$. This subbranch occurs with a probability $p_2 \times (1 - P_{match}(\#8)) = 40\% \times (1 - 50\%) = 20\%$

For the sake of the completeness of the state transition, the other two subbranches associated with $I = 1$ are also displayed in Figure (2.4b). These two subbranches are quite self-explanatory, and thus the detailed discussion will be omitted.

2.3 Sequential MDPs for multiple agents

Note that the deterministic policy derived is only applicable when there is one agent following the policy. Otherwise there can be local competition among e-hailing drivers since several drivers may be guided into the same grid. We thus need to address the competition among e-hailing drivers if there are multiple idling e-hailing drivers being present in the same region within a short time interval. Lin et al. [1] proposed a contextual multi-agent reinforcement learning approach in which the multi-agent effect is captured by attenuating the reward through an averaging fashion. Zhou et al. [45] employed a simple discounting factor $\frac{1}{n}$ to update the order matching probability when the $(n+1)^{th}$ taxi is being guided to a road if there are already n taxis going to that road. The discounting factor proposed $\frac{1}{n}$ is effective in the sense that it makes the order matching probability

smaller for subsequent taxis following the policy. However, the simple discounting factor may underestimate the order matching probability since the effect of the number of orders in each grid was neglected. In other words, except the effect of the number of drivers being guided into a grid, there is an underlying correlation between the decrease in the order matching probability and the number of orders in that grid. Here we use an example to illustrate the existence of the aforementioned correlation. We suppose an e-hailing driver is guided into grid l with an order matching probability 50%. We consider two extreme scenarios: (1) there was 1 order emerging in grid l and (2) there were infinite orders emerging in grid l. After one driver is guided into grid l, for a second driver, the order matching probability in grid l is supposed to decrease substantially in the first scenario while almost keeps the same in the second scenario. The rationale underlying this argument is that compared to a grid with a smaller number of orders, a grid with a larger number of orders is capable of accepting more cruising drivers while still maintain a relatively acceptable level of order matching probability.

To incorporate this correlation, we develop a dynamic adjustment strategy. Before formally providing the form of the strategy, we list four intuitive observations: (1) The order matching probability for the first driver being guided into grid l is simply $P_{order\_match}(l)$; (2) The order matching probability for the n^{th} driver being guided into grid l decreases with n, meaning that the order matching probability is getting smaller when there are more drivers cruising vacantly in grid l; (3) For the n^{th} driver, the order matching probability increases with the number of orders in grid l, meaning that a grid with a larger number of orders is able to accept more cruising drivers; (4) Under the extreme scenario where there are infinitely many orders in grid l, the order matching probability keeps its level at $P_{order\_match}(l)$ regardless of the number of drivers being guided into grid l, as long as it is finite. Based on these four observations, we postulate that the order matching probability of the n^{th} driver in grid l takes the exponential form, i.e.,

$$Pr(l,n) = P_{order\_match}(l) \times e^{-\frac{\beta}{\#orders(l)}\times(n-1)}, \tag{2.21}$$

where $\#orders(l)$ is the number of orders in grid l and β is a parameter to be determined.

To calibrate the strategy for the n^{th} driver, the order matching probability $Pr(l, n)$ is required. Note that $Pr(l, 1) = P_{order\_match}(l)$, then Equation (2.21) can be written as

$$\frac{Pr(l, n)}{Pr(l, 1)} = e^{-\frac{\beta}{\#orders(l)}\times(n-1)}. \tag{2.22}$$

To utilize linear regression, we further take the logarithm and then the reciprocal of both sides of Equation (2.22), and thus Equation (2.22) can be rewritten as

$$\frac{1}{log\frac{Pr(l, n)}{Pr(l, 1)}} = -\frac{1}{\beta \times (n-1)} \times \#orders(l). \tag{2.23}$$

Denoting the left hand side of Equation (2.23) as $prob_n(l)$, the purpose of the calibration is simply to verify the existence of the linear correlation between the two variables $\#orders$ and $prob_n$ and determine the parameter β. $\#orders(l)$ is simply the number of historical orders in grid l. The order matching probability for the n^{th} driver entering grid l can be calculated as follows. We use 18 10-minute intervals to split the 3-hour morning peak used in this study. For each interval, the number of orders within the interval is counted, and it is a success if the counted number is not less than n. The probability of success across 18 intervals is $Pr(l, n)$. Calculating $prob_n(l)$ and $\#orders(l)$ for all grids, samples of $prob_n(l)$ and $\#orders(l)$ are obtained.

Ideally, the calibration can be run for any integer value of n. However, due to the relatively small size of the grid and the finite number of idling drivers, n cannot be taken as a very large number. To get an appropriate upper bound of n in the calibration, we simply count the number of cases in which there are n drivers idling in a grid within a time interval. The distribution of the number of cases versus the number of idling drivers in a grid is presented in Figure (2.5). For example, there are more than 25,000 cases in which there are only one idling driver in a grid within a 10-minute time interval. The trend is decreasing, indicating that the number of cases for more drivers idling in a grid within a time interval is less, which is as expected. To obtain statistically

meaningful results in the calibration, we need as many samples as possible. Thus, here we choose to do the calibration for n up to 4.

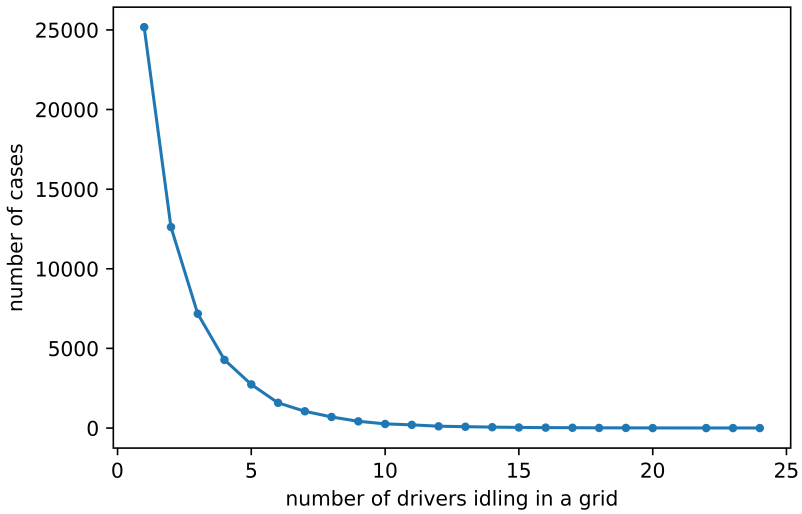


Figure 2.5: Distribution of number of cases

We obtain three lists of $prob_n$ and three lists of $\#orders$ for $n = 2, 3$, and 4, although the three lists $\#orders$ for different n are identical. Noticing that here we only have one parameter β, we concatenate the three lists of $prob_n$ simply into a long list $prob = [prob_2, prob_3, prob_4]$ and concatenate the three lists of $\#orders$ as a long list $\#orders\_long = [\#orders, \frac{\#orders}{2}, \frac{\#orders}{3}]$. Then we run a linear regression through origin of the list $prob$ over $orders\_long$. Here we choose the linear regression through origin because the model is naturally linear without intercept, as shown in Equation (2.23), stemming from the requirement that the order matching probability for the first driver in a grid l is $Pr(l, 1) = P_{order\_match(l)}$. The regression result suggests the slope $-\frac{1}{\beta} = -0.0842$ with p-value less than 0.001, indicating that the slope is significant. The R-square value is determined as 65%, indicating the fitted linear model is able to explain 65% of the variability and thus the adoption of a linear regression is reasonable. To visually show the goodness of the fitting, we substitute the determined value of β into Equation (2.21) and plot the fitted curve together with the data for $n = 2, 3$, and 4 in Figure (2.6).

Based on the calibrated dynamic adjustment strategy, we can now build sequential MDPs for

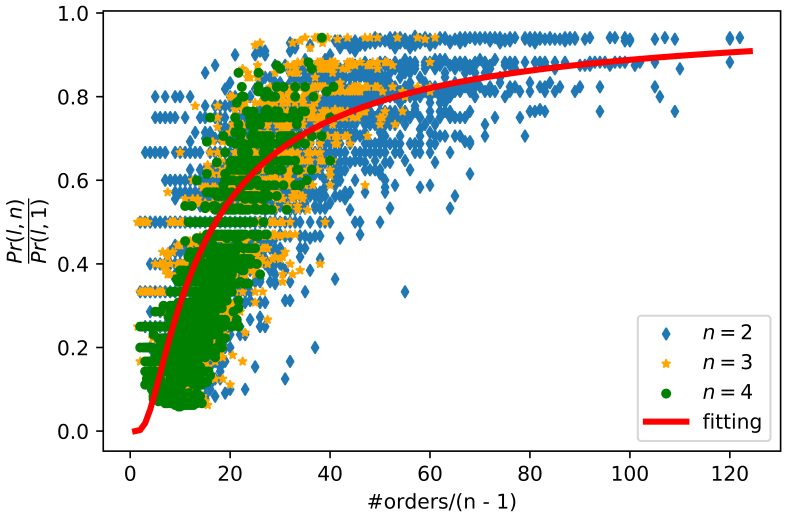


Figure 2.6: Exponential fitting

multiple agents and then derive one optimal policy for each agent sequentially. Recall that the dynamic adjustment strategy actually attenuates the order matching probability for the n^{th} driver to be guided into grid l when there are $n-1$ drivers idling in grid l. Thus, the general MDP framework proposed in the previous section can directly be applied to the n^{th} driver with a table of relatively lower order matching probabilities. In other words, there is one MDP model for each agent when multiple agents coexist, and the main distinction among these MDP models is the order matching probability (i.e., order matching probabilities of the n^{th} driver are smaller than that of the $(n-1)^{th}$ driver). The optimal policy for each agent can then be obtained by solving her MDP model.

2.4 Case study

Nowadays, GPS-enabled devices are ubiquitously used in different types of applications. Especially for drivers, GPS devices can not only help them navigate but also record the real-time location and speed of the vehicle. Hence, a large amount of GPS trajectories, representing sequences of time-stamped geographical points, have been collected and can be a valuable asset for people to understand and tackle real-world traffic problems [55, 56, 57].

In this research we use large-scale real-world historical GPS traces collected in Beijing in the morning peak, i.e., (7 AM, 10 AM), of the first 3 weekdays in November 2017 by Didi Chuxing, China's leading ride-hailing company. The dataset contains recorded GPS traces of 44,160 e-hailing vehicles and 158,784 e-hailing orders. Whenever an e-hailing driver is online, i.e., she has turned on the e-hailing application, one data point is sampled every 3 seconds. Each data point contains the vehicle's location (i.e., longitude and latitude), current timestamp, the fare of the current occupied trip, if applicable, and a status indicator to record the taxi's current operating state, which includes idle, after matching before pick-up, waiting at the pick-up location, and on trip.

Based on the characteristics of the spatial distribution of the passenger pick-up spots and passengers' destinations, we construct a bounding box within the six ring road to cover the city area in which 90% of the e-hailing orders are preserved. The e-hailing orders which fall outside the bounding box will be disregarded. A hexagonal grid world setup is then adopted, and the city area within the bounding box is split into 6,421 hexagonal grids with the length of the diagonal of a hexagon of approximately 700 meters.

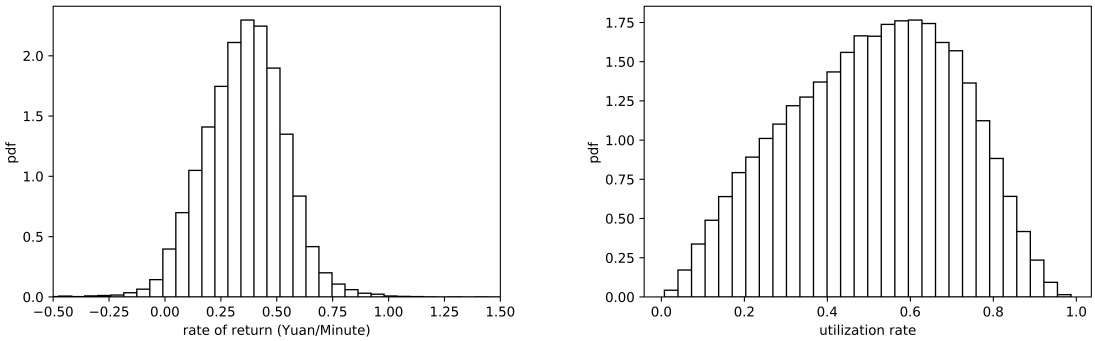
Now we use the IRL technique to uncover the general reward function for all drivers, i.e., to derive the parameter α. The observed policy $\pi^O(s)$ in each state s is simply the most frequently taken action by all drivers in that state. Applying the aforementioned IRL technique with two known reward functions (i.e., fare $\phi_1(s, s')$ and traveling distance $-\phi_2(s, s')$) and setting $\alpha_1 = 1$ (i.e., assuming the driver will earn all the fare), we obtain $\alpha = \alpha_2 = 0.64$. In other words, the coefficient of fuel consumption and other operating costs per unit distance is 0.64 Chinese Yuan. The general reward function applicable to all drivers is $R(s, s') = \phi_1(s, s') - 0.64 \times \phi_2(s, s')$.

Different from other public transportation modes, including buses and subways, which are operated according to a fixed schedule and a predefined route, e-hailing drivers are free to choose their own actions after completing a ride, resulting in a discrepancy in drivers' income. In general, an experienced driver is familiar with the city where she operates the vehicle and is usually aware of where to go after completing a ride in order to get the next request as soon as possible. For

example, an experienced driver knows where and when passenger demands will be high near some attractions, hotels, or transportation terminals. In practice, however, there is no guarantee that all drivers are experienced and have a good judgment of where to go next. In particular, the popularity of e-hailing applications has lowered the entry barrier of becoming a driver and brought a tons of rookie into the e-hailing market. To gain some basic understanding of the performance of e-hailing drivers, we adopt two metrics, namely, the rate of return and the utilization rate.

Definition 2.4.1. (Rate of return) An e-hailing driver's rate of return on one day is defined as the ratio of the driver's net income (i.e., gross income minus the operating cost) to the driver's working time.

Definition 2.4.2. (Utilization rate) The utilization rate of an e-hailing vehicle is defined as the ratio of the time spent on carrying a passenger to the total operating time of the vehicle.



(a) Distribution of the rate of return (b) Distribution of the utilization rate

Figure 2.7: Distribution of the rate of return and utilization rate from the field data

The rate of return measures a driver's earning ability per unit time. Thus, compared with the total income, which can be largely influenced by a driver's working time, the rate of return is a better metric to be used to measure the performance of drivers. Figure (2.7a) presents the probability density function (pdf) of the rate of return of all drivers across morning peaks on the first three weekdays in November 2017. The average rate of return is 0.36 (Yuan/minute). About 80% drivers have a rate of return fall within the range 0.11 to 0.60 (Yuan/minute). Top 10% drivers

can reach a rate of return of 0.60 (Yuan/minute) and higher, while the bottom 10% have a rate of return below 0.11 (Yuan/minute). In terms of the utilization rate, real drivers can on average reach 0.51.

2.4.1 Policy

All drivers' data was then used to train the MDP model. We will then qualitatively examine the optimal policy derived from our MDP model and conduct numerical experiments to evaluate the effectiveness of the policy.

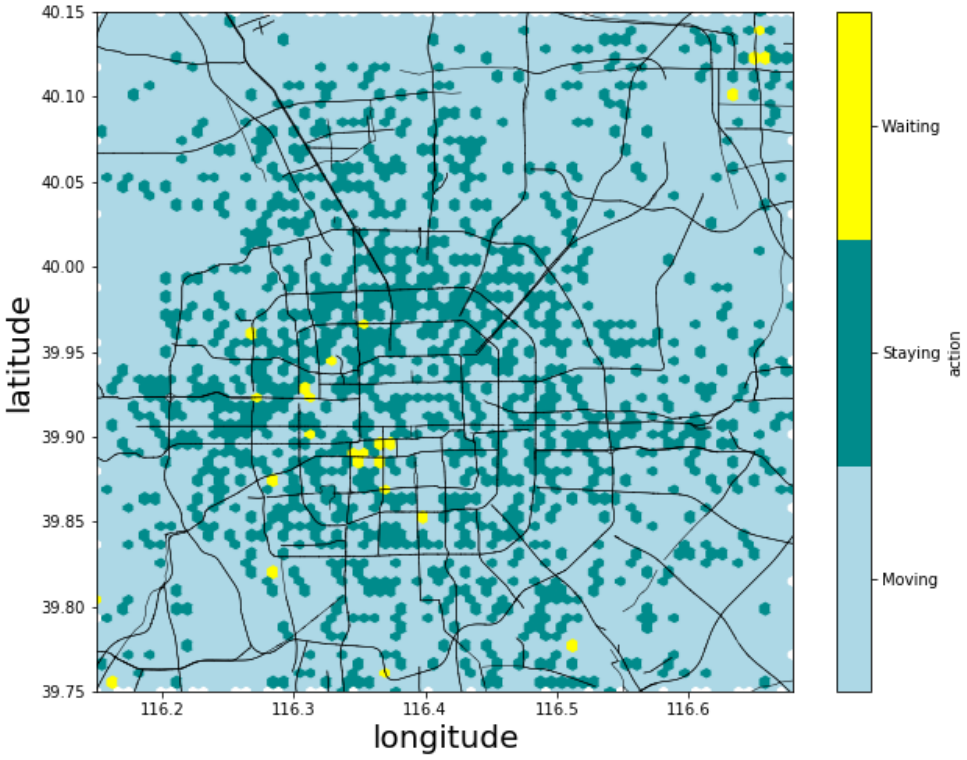


Figure 2.8: Optimal policy (at the beginning of the morning peak)

Figure (2.8) presents the optimal policy at the beginning of the morning peak. Light blue color suggests the driver in the current grid to move into one of its neighboring grids to seek for the next potential e-hailing order. The color dark green in a grid stands for staying, meaning that the

optimal policy is to stay and cruise within the current grid. The color yellow in a grid means waiting, indicating that the optimal policy for the driver to follow is simply to wait in the current grid. It can be seen that in many grids within the city area (i.e., around the center part of the figure), the optimal policy suggests a driver to stay after completing a ride. In suburban areas, optimal policy usually suggests drivers to move around to some grids with a high probability of receiving a request. Also, there are several places where the probability of receiving a request while waiting is quite high, and the optimal policy advises drivers to wait in these places.

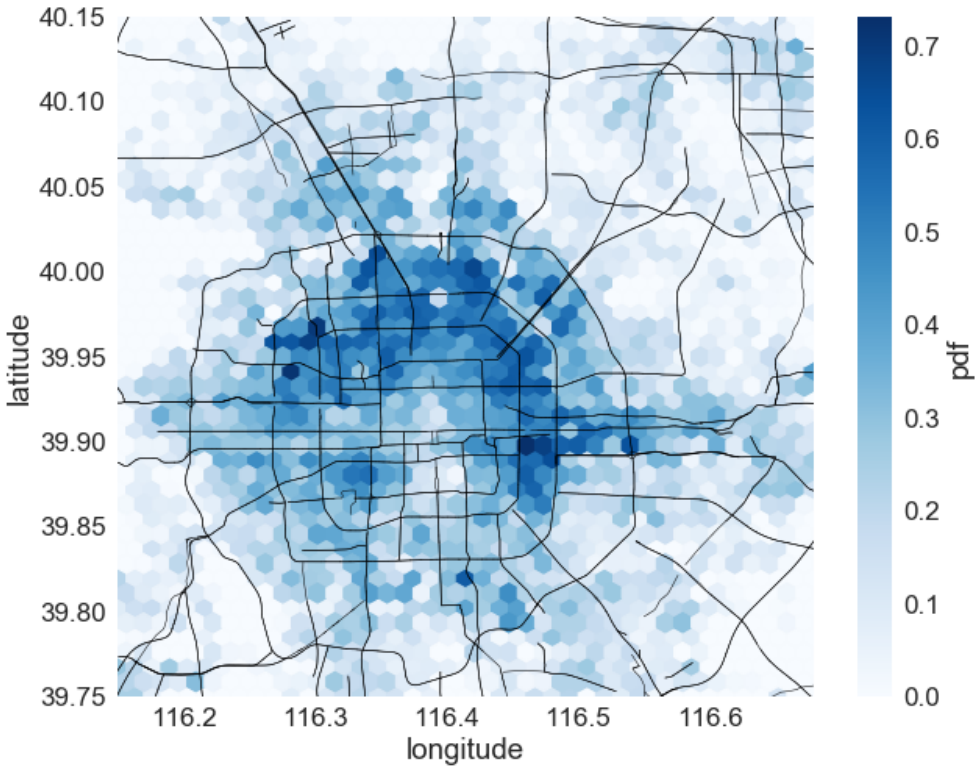


Figure 2.9: Distribution of the order matching probability P_{order_match}

Figure (2.9) plots the distribution of the order matching probability. It can be seen that the majority of the color dark blue, indicating a higher order matching probability, is distributed within the fourth ring road, meaning that the order matching probability is higher in the city area. Furthermore, the distribution of the order matching probability agrees well with the distribution of the

optimal policy if we compare Figure (2.9) with Figure (2.8). The dark green places, indicating staying, in Figure (2.8) generally overlap with blue or dark blue locations in Figure (2.9), indicating a relatively high order matching probability. Also, when a driver is in a grid with a low order matching probability (e.g., in the nearly white area in Figure (2.9)), the driver needs to move around (as shown by the light blue in Figure (2.8)) to enter a grid with a higher order matching probability. This overlapping essentially indicates that grids with a higher order matching probability is more preferable by the agent, compared with a grid with a lower order matching probability.

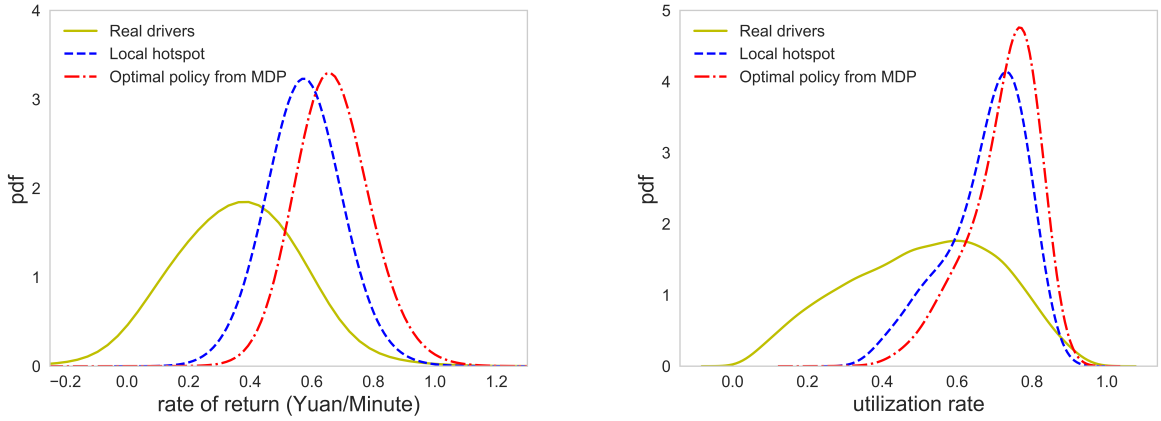
2.4.2 Model Evaluation

To evaluate the effectiveness of the optimal policy derived from the MDP model, the performance of an agent under the guidance of the optimal policy is compared with that of an agent following one baseline heuristic, i.e., the local hotspot strategy. The local hotspot strategy essentially suggests the agent to move into grids with a higher demand sequentially and is found to perform the best among three baseline heuristics, namely random walk, global hotspot, and local hotspot [8].

To obtain the performance of an agent according to different policies, a Monte Carlo simulation is conducted. The basic idea of the simulation is to randomly place an agent in one grid at the beginning, and let the agent move around according to the chosen policy. The environment is determined by the parameters extracted in Section (2.2.3). In particular, every time when the agent is in a grid l_a, we sample a probability of finding a request from a binomial distribution with a success probability $p_{order\_match}(l_a)$. We then sample the pick-up grid and drop-off grid from a multinomial distribution determined by the probabilities p_{pickup} and p_{dest}, respectively. The driving time and driving distance can be simply obtained from t_{drive} and d_{drive} after the pickup spot and destination have been determined.

The simulation is run for millions of times to obtain robust results. We first adopt two metrics, namely, rate of return and the utilization rate of the vehicle to compare the performance of the agent under the guidance of different policies. We then examine the distribution of the number of

completed orders, idling time, service time per order, and the profit per unit time of each order.



(a) rate of return (Yuan/Minute)

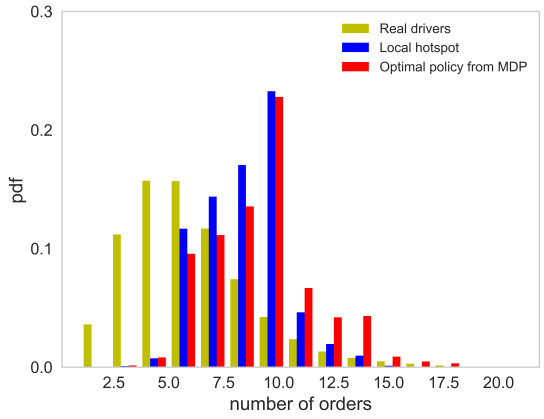
(b) utilization rate

Figure 2.10: Distribution of the rate of return and the utilization rate of real drivers and the agent

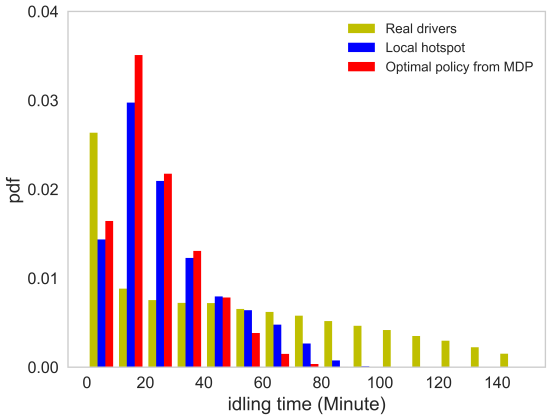
Figures (2.10) presents the distribution of the rate of return and the utilization rate of the agent following the optimal policy and the local hotspot strategy, respectively. It can be seen that the performance of the agent following the optimal policy is on average better than that of the agent following the local hotspot strategy. In terms of the rate of return, the average value that the agent can reach are 0.67 (Yuan/Minute) and 0.57 (Yuan/Minute) under the guidance of the optimal policy and the local hotspot strategy, respectively, meaning that the optimal policy is able to increase the average rate of return by 17.5% over the local hotspot strategy. The average rate of return of real drivers is 0.36 (Yuan/Minute). In terms of the utilization rate, the average value that the agent can reach are 0.72 and 0.67 by following the optimal policy and the local hotspot strategy, respectively, indicating a 7.5% improvement of the optimal policy over the local hotspot strategy. The average utilization rate of real drivers is around 0.51.

Table 2.3: Statistics of the comparison between the performance of the agent under different policies

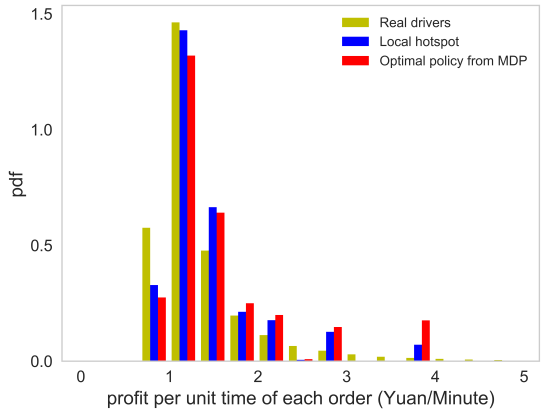
	real drivers	agent following the local hotspot strategy	agent following the optimal policy
average rate of return (Yuan/Minute)	0.36	0.57	0.67
average utilization rate	0.51	0.67	0.72
average number of orders	6.08	8.21	8.92
average idling time (Minute)	47.98	27.12	22.61
average profit per unit time of each order (Yuan/Minute)	1.35	1.51	1.68
average service time per order (Minute)	16.97	14.83	14.56



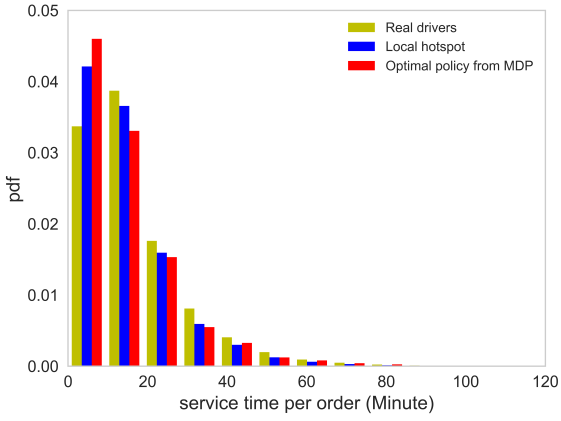
(a) Distribution of number of orders



(b) Distribution of idling time



(c) Distribution of profit per unit time of each order



(d) Distribution of service time per order

Figure 2.11: Distributions of the number of completed orders, number of idling cells, the service time per order, and the profit per unit time of orders

Figure (2.11) presents distributions of the number of completed orders, idling time, service time per order, and the profit per unit time of each order. The corresponding average value of each case is listed in Table (2.3). There are several observations worth mentioning: (1) On average the agent following the optimal policy can achieve a higher number of completed orders than the agent following the local hotspot strategy. (2) The agent following the optimal policy and the local hotspot strategy can achieve a significantly lower idling time, compared with that of real drivers. Thus, both the optimal policy and the local hotspot strategy are able to help agents find requests faster. (3) The distribution of the profit per unit time essentially says that under the optimal policy, the agent is able to find better orders. Here we say an e-hailing order is better when the profit per unit time of the order is higher. (4) The average service time of orders taken by the agent is shorter than that of real drivers.

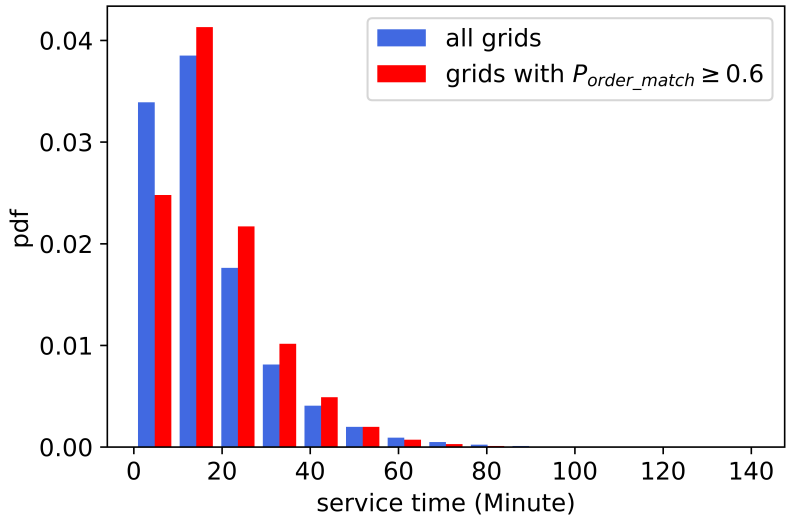


Figure 2.12: Distribution of service time of orders starting in all grids and in grids with a relatively high order matching probability

The aforementioned observation (4), however, does not necessarily indicate that on average a shorter order is more preferable compared with a longer order. The reasons are as follows. As previously stated, for an agent, grids with relatively higher order matching probability is more preferable, compared with grids with a moderate or low order matching probability. Figure (2.12) presents the distribution of service time of orders starting in all grids and in grids with a relatively

high order matching probability. The average service time of orders starting in all grids and in grids with a relatively high order matching probability are 17 minutes and 18.34 minutes, respectively. In other words, the passenger requests in grids with a relative higher order matching probability on average has a slightly greater service time. Hence, the average service time per order of the agent is supposed to be slightly higher than that of real drivers, which is not consistent with the observation (4).

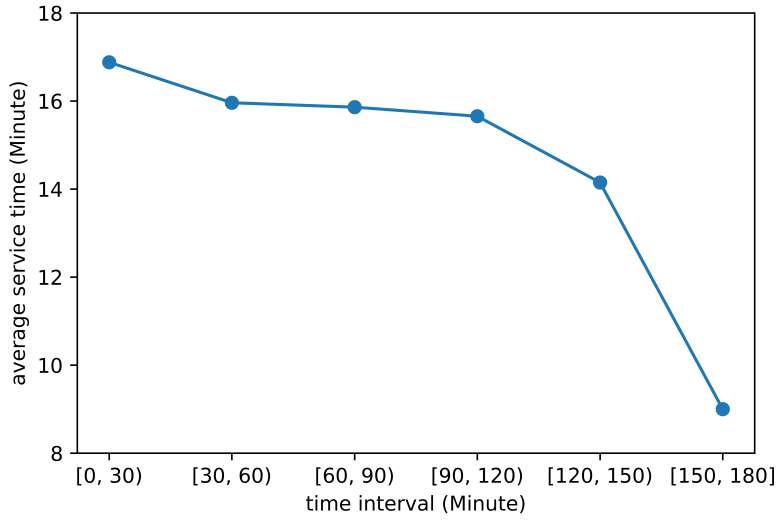


Figure 2.13: Average service time of orders starting in each subinterval

To explain the inconsistency, we split the 3-hour time interval, i.e., the morning peak, into 6 subintervals of even length, namely, $[0, 30)$, $[30, 60)$, $[60, 90)$, $[90, 120)$, $[120, 150)$, and $[150, 180]$, and then extract the average service time of orders starting in each subinterval, which is shown in Figure (2.13). The average service time of orders starting in each subinterval decreases as time elapses. In the first subinterval, i.e., around the beginning of the morning peak, the agent actually takes orders with an average service time of 16.88 minutes, which is very close to 16.97 minutes, i.e., the average service time of all orders. This is as expected because the initial position of the agent is randomly chosen, meaning that the distribution of the service time of orders taken by the agent at the beginning of the time interval is supposed to be similar to that of all orders. As time elapses, the average service time of orders taken by the agent decreases because in the

simulation we restrict the ending time of an order that an agent can take to be within the 3-hour time interval, and thus the agent to some degree prefers shorter orders, especially when the agent is in the last two subintervals, i.e., around the end of the 3-hour time interval. Low values of the service time of orders in the last two subintervals, i.e., 14.12 minutes and 9.01 minutes largely drag down the overall average service time of orders taken by the agent and caused the aforementioned inconsistency. Here we emphasize that in the simulation, the agent is set to complete the 3-hour time interval, and the restriction imposed by the 3-hour time interval implicitly forces the agent to take relatively shorter orders. All references listed in Table (2.1) except [46] used finite horizon time intervals [1, 6, 7] or finite pickup-dropoff cycles [8] in MDP modeling. Actually, in reality, an e-hailing driver is free to stop working at any time as long as she has achieved her preset goal, such as a nominal income, a personalized utility function, etc. Thus, an MDP with optimal stopping time modeling is a more realistic and efficient model and is left for future research.

2.4.3 Supply-demand ratio

(a) Around Xibeiwang (outside the 5th ring road)

(b) Around Chaoyang (inside the 3rd ring road)

Figure 2.14: Two zoom-in views of the optimal policy and the distribution of the demand

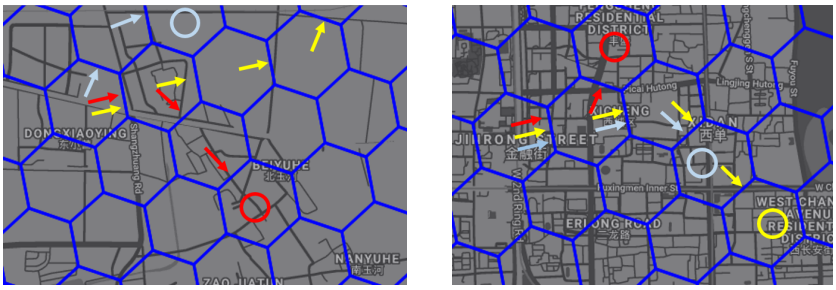
Figure (2.14a) presents a zoom-in view of the optimal policy (shown in arrows and circles where arrows suggest the driver to move along the direction denoted by the arrow and the circle denotes staying or waiting) and the distribution of the demand (shown in heatmap) near Xibeiwang, an area outside the 5th ring road. We can see that our optimal policy suggests drivers to stay around

the grid where the demand is high to take advantage of the locally high demand. Outside the 5th ring road, the overall demand is not very high and the number of idle drivers, indicating the supply, is also not large, resulting in a locally high order matching probability in high demand areas. Thus, a driver can simply take advantage of the high demand to make more profit.

Figure (2.14b) presents a zoom-in view of the optimal policy around Chaoyang district, which is located within the 3rd ring road. Different from the consistency between the policy and the distribution of the demand observed in Figure (2.14a), now it seems the derived optimal policy and the distribution of the demand is not very consistent. In other words, the optimal policy suggests drivers to move away from the areas with a high demand. The main reason is that the optimal policy is supposed to be consistent with the order matching probability, which captures the supply-demand ratio under the assumption that the number of unmatched order is negligible, instead of the real distribution of the demand. Although the order matching probability is calculated from the distribution of the demand and the number of pass-bys of idle drivers, there may exist some shift or even inconsistency between the order matching probability and the distribution of the demand. For instance, a cell with a very high demand, e.g. 100 order matches, may have a 1,000 pass-bys by idle drivers, resulting in a relatively low order matching probability in the cell, which is 0.1 in this example. Hence, it is not surprising that the derived optimal policy suggests the driver to move away from the cell with a high demand and a low order matching probability. A grid with a high demand may also have a high supply, resulting in a low order matching probability which is not preferable to the agent.

2.4.4 Different optimal policies for multiple agents

Now, we verify the effectiveness of the proposed dynamic adjustment strategy of the order matching probability by recommending the next several optimal actions to take for three agents who are currently in the same grid. We randomly selected two places, namely, one place around Dongxiaoying (outside the 5th ring road, thus in suburban area) and one place around Jinrongjie (inside the 2nd ring road, thus in city area).



(a) Around Dongxiaoying (outside the 5th ring road)

(b) Around Jinrongjie (inside the 2nd ring road)

Figure 2.15: Two zoom-in views of the optimal policy for three agents (red, yellow, and light blue denote the first, second, and third agent, respectively)

Figure (2.15) presents the recommended optimal actions to take for three agents. In both cases, three agents are being recommended at the same time in the same grid. The results shows that our proposed dynamic adjustment strategy is able to provide different recommended actions for different agents. When three agents are in the same grid around Dongxiaoying, which is located outside the 5th ring road and has a relatively lower number of orders, the strategy guides the first two agents into the same grid but refers the third agent into a different grid because after guiding two agents into the same grid, the order matching probability in that grid drops quickly. The strategy also guides the first two agents into two different directions afterwards due to the competition. When three agents are in the same grid around Jinrongjie, which is located inside the 2nd ring road and has a relatively higher number of orders, the strategy guides all three agents into the same grid. Although the strategy refers the first agent into a different direction soon, the strategy guides the following two agents in almost the same direction.

As we have mentioned before, the historical number of orders in one grid is supposed to have an impact on the decrease of the order matching probability of the grid. For a grid with a higher average number of orders, like the grid located around Jinrongjie, the decrease in the order matching probability is supposed to be slower, thus it is able to accept more cruising agents while still maintains a relatively decent order matching probability. For a grid with a lower number of average number of orders, like the grid located in Dongxiaoying, the decrease in the order matching probability is supposed to be faster because with a small historical number of orders, it is not able

to withstand too much competition among agents. In other words, for a grid with a small number of orders, the order matching probability is decreasing rapidly when some agents are being guided into the grid.

2.4.5 Heterogeneity in reward functions

Although the determined coefficient α is applicable to the overall driver population at the aggregate level, it may vary from individual to individual. In other words, each driver may have a specific reward function which can be quite different from other drivers', resulting in some discrepancies in driving patterns and strategies. To examine various driving patterns of taxis drivers, especially the change in the driving patterns from the time without e-hailing to the time when e-hailing is widely adopted, Ma et al. [58] carried out in-depth analysis using trajectories of taxi drivers in Shanghai, China, and unveiled that on average, taxi driving patterns which were previously concentrated in some central areas are now more spread out. Inspired by this, we examine the spread of a driver's trajectories using the radius of gyration R_g. Radius of gyration is defined as the standard deviation of the spatial distances between a driver's location and the centroid of the driver's visited locations, i.e.,

$$R_g = \sqrt{\frac{1}{n}\sum_{i=1}^{n} r_i^2} \tag{2.24}$$

where r_i is the distance between the driver's i^{th} location and the centroid of the driver's visited places, and n is the total number of the driver's visited locations.

Figure (2.16) plots the distribution of the radius of gyration across the e-hailing driver population. The radius of gyration of the majority (95%) of e-hailing drivers is distributed within the range of [5.5 km, 16 km]. The e-hailing drivers on the left tail (2.5%) of the distribution has a radius of gyration below 5.5 km, and the drivers on the right tail has a radius of gyration above 16 km. The substantial discrepancy in the radius of gyration across the driver population indicates that drivers exhibit different driving patterns, which may stem from different intrinsic reward functions. For example, the trajectory of a driver on the left tail with a small radius of gyration is more

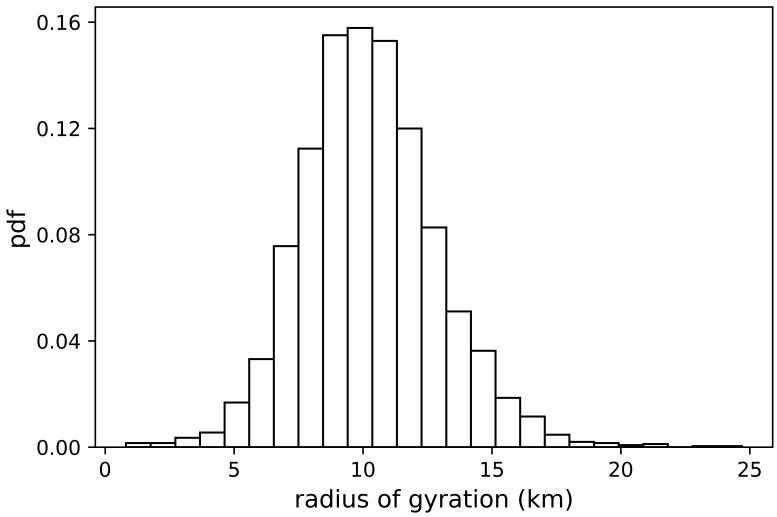


Figure 2.16: Distribution of radius of gyration

concentrated in a small region, indicating that the driver may have some incentives (such as being close to home or being more familiar with the region) to stay within the region.

To shed some light on the distinction among driving patterns of e-hailing drivers, we simply take one driver on the left tail with a small radius of gyration (1.2 km) and one driver on the right tail with a large radius of gyration (17.8 km) and employ the IRL technique to disclose some underlying information regarding the driver's intrinsic reward function. Here we emphasize that understanding drivers' intrinsic reward function can be very helpful from the perspective of the platform to appropriately assign e-hailing orders. For instance, for a driver who prefers to stay within a small region, the platform is supposed to assign relatively shorter e-hailing orders to this driver. Otherwise, the driver will have a lower utility and may even be unwilling to take the assignment since the driver cares more about staying within the region over a simple higher monetary return. Thus, appropriately assigning orders can help increase the drivers' response rate and therefore the utilization of vehicle resources, as well as the passenger satisfaction, which is beneficial for all players in the e-hailing market, including the platform, the drivers, and the passengers.

Without any knowledge of the driver's demographic information, we devise the third simple

Table 2.4: Coefficients

	$\phi_1(s, s')$	$\phi_2(s, s')$	$\phi_3(s, s')$
The driver with a small radius of gyration	1.00	0.21	1.42
The driver with a large radius of gyration	1.00	0.37	0.17

reward function $\phi_3(s, s')$ as the difference between the spatial distance between state s and the centroid c of the driver's visited locations and the spatial distance between state s' and the centroid c, i.e.,

$$\phi_3(s, s') = distance(s, c) - distance(s', c), \tag{2.25}$$

where $distance(s, c)$ denotes the spatial straight line distance between s and c. The rationale of this simple reward function can be explained as follows. For a driver with a small radius of gyration, the driver prefers to come back around the centroid after completing an order (otherwise the radius of gyration would be larger), indicating that coming back to the centroid may increase the driver's utility/intrinsic reward. ϕ_3 exactly does this. The driver gets a positive reward when $distance(s', c) < distance(s, c)$ and a negative reward when $distance(s', c) > distance(s, c)$, meaning that it is beneficial for the driver to go back to the centroid. While for a driver with a large radius of gyration, the driver may not care about his/her distance to the centroid, and thus ϕ_3 may not be important in the driver's intrinsic reward function. We apply the IRL technique to the observed policy of these two drivers with three simple reward functions, namely, ϕ_1 (fare), ϕ_2 (traveling distance), and ϕ_3, and the derived coefficients are listed in Table (2.4). Again, we fix $\alpha_1 = 1$ under the assumption that the driver gets all the fare.

The coefficient for ϕ_3 is quite large (1.42) for the driver with a small radius of gyration and relatively small (0.17) for the driver with a large radius of gyration. This validates the effectiveness of the devised reward function ϕ_3 in explaining the driver's intrinsic reward function when the driver's radius of gyration is small. When the driver's radius of gyration is large, the reward function ϕ_3 does not contribute enough to the driver's underlying reward, meaning that the driver with a large radius of gyration does not gain more utility/reward by coming back to the centroid.

We believe that there exist other types of simple reward functions which may be important in explaining a driver's underlying reward function when the driver has a large radius of gyration and will be left in future work.

2.5 Summary

Based on a large-scale real-world historical GPS traces, this chapter investigated how to improve the income and rate of return of e-hailing drivers through a modified MDP approach. We proposed an MDP model which incorporates the following distinct features of drivers with e-hailing: (1) An e-hailing driver may receive a matched order before she drops off the previous passenger, thus there is no passenger seeking; (2) Different from traditional taxi that a driver has to see a passenger to find a match, e-hailing platforms very likely find a match even when the driver and the passenger are spatially far from each other. In other words, a driver's search process may end before a passenger is picked up.

We used 44,160 Didi drivers 3-day trajectories to train the model with a reward function uncovered by IRL. We then examined the optimal policy learned from our model and found that with e-hailing, the optimal policy suggests drivers to stay when they are in the city area, to move to some local areas with a high probability of receiving a request if they are in suburban areas, and to wait when they are at some places with a very high likelihood of receiving a request. To validate the effectiveness of the derived policy, a Monte Carlo simulation is conducted, and two metrics, namely the rate of return and utilization rate, are employed to compare the performance of the agent following the derived optimal policy with that of the agent following one baseline heuristic, namely, the local hotspot strategy. The comparison validates the effectiveness of the proposed model and shows that our model is able to achieve a 17.5% improvement and a 7.5% improvement over the local hotspot strategy in terms of the rate of return and the utilization rate, respectively. Also, the results show that under the guidance of the optimal policy, the agent is able to complete more order, decrease idling time, and find better orders. In addition, we disclose the reason why the agent not necessarily prefers a shorter order.

In the modified MDP model, the order matching probability captures the supply-demand ratio by its definition, considering the fact that the number of drivers in this study is sufficiently large and thus the number of unmatched orders is assumed to be negligible. Results show that the optimal policy does not necessarily guide an e-hailing driver to a grid with a high demand. Instead, the optimal policy suggests a driver to a grid with a low supply-demand ratio, i.e., a high order matching probability. To accurately capture the competition among drivers, we have devised and calibrated a dynamic adjustment strategy of the order matching probability when there are multiple drivers need recommendations. Also, the heterogeneity in reward functions across the driver population has been investigated. The devised simple reward function ϕ_3 is validated to explain the driving patterns of drivers with a relatively small radius of gyration and paves the way for future research on the underlying reward function of e-hailing drivers.

Chapter 3: Bilevel optimization for multi-driver passenger-seeking

3.1 Introduction

The emergence of transportation network companies (TNCs) or e-hailing platforms (such as Didi and Uber) has revolutionized 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 [1], partly due to the imbalance between demand (i.e., passenger requests) and supply (i.e., available drivers) that results in long cruising trips for taxi drivers to find the next passenger and long waiting time for passengers to be picked up [2, 3]. 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.

3.1.1 Problem statement

This chapter aims to solve the multi-driver repositioning task, in which a large number of idle drivers make sequential decisions on where to reposition themselves over a planning horizon, so that they can seek for the next passengers and maximize their individual cumulative rewards. When one idle driver has to make a sequential decision of repositioning while others are doing so simultaneously, competition among drivers arises.

Markov decision process (MDP) and reinforcement learning (RL) have become popular tools for sequential decision-making. Single-agent RL has been used in several studies [46, 59] to solve the single-driver repositioning problem. However, single-agent RL assumes each agent is an independent learner and ignores the impact of competition on each individual agent's policy. To

capture competition among multiple agents in the process of repositioning, this chapter employs a multi-agent reinforcement learning (MARL) approach.

Nevertheless, this chapter goes beyond simply modeling multi-driver repositioning with MARL. We notice that every driver plays a non-cooperative game, meaning each driver's goal is to select a sequence of reposition strategies to maximize the individual net profit, given other drivers select their own optimal reposition strategies. Accordingly, the direct application of MARL to the multi-driver system under a given reward function will likely yield a suboptimal equilibrium for the system, thanks to the selfishness of drivers. Therefore, the ultimate goal of this chapter is to propose a reward design scheme with which a more desired equilibrium can be reached. We would like to emphasize the difference between "reward" and "reward design". In this chapter, the reward for an idle driver is simply the net monetary return one driver earns, while the reward design scheme is imposed by a higher-level planner (e.g., the TNC platform or city planners) to modify drivers' monetary return. Analogous to the role of congestion pricing, reward design aims to spatially redistribute drivers, driven by individual reward maximization, through an external cost imposed to those who tend to move towards already oversupplied spots. Under the external cost, drivers still behave selfish but their movement is now governed by a modified reward, which consequently balances demand and supply and drives the system towards a more desirable equilibrium.

To justify that reward design can be an effective way of balancing supply and demand, we would like to briefly introduce our problem set-up below. We assume that each driver, behaving like an intelligent agent, selects a sequence of optimal reposition policies to maximize its own cumulative reward, which is the monetary income. Usually a TNC platform charges a commission fee to drivers. Thus, the revenue a TNC driver receives is an order's fare minus the commission fee. The TNC platform can adjust the percentage of fares going to drivers, in order to balance demand and supply. For instance, if the platform notices that excessive drivers try to move to one hotspot to compete for a limited number of orders, the platform can increase the commission fee in that grid to prevent oversupply. Drivers are still free to reposition themselves toward that

grid, but adding a higher commission fee would reduce their long-run accumulative rewards, and accordingly, influence their reposition policies, which hopefully discourage some drivers from moving to that grid.

3.1.2 Literature review

In recent years, we have witnessed a growing body of literature on TNC using reinforcement learning, including pricing mechanism design [60, 61], vehicle routing [62, 63, 64], ride order dispatching [65, 66, 67, 68], and driver repositioning [1, 46, 69]. This study focuses on the multi-driver repositioning task.

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 [41, 40, 43, 42]. These studies extracted useful aggregated statistical quantities such as taxi demand and travel time from historical data and recommended a next cruising location [41], a sequence of potential pickup points [40], a driving route [42], or a route and a location [43]. 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 MDP based approaches have been proposed to model idle drivers' passenger searching process [6, 7, 8, 45, 46, 70]. In an MDP, a driver is the agent who makes decisions of where to go next and interacts with the environment. The agent aims to derive an optimal policy which maximizes her expected cumulative reward. When the environment is known to the agent, dynamic programming can be used to solve the MDP and derive an optimal policy. When the dynamic environment is unknown to the agent, RL algorithms such as Q-learning and temporal difference learning [47] can be employed to derive an optimal policy.

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. MARL [71] 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 [72, 73], Poker [74, 75], Dota 2 [76], and StarCraft II [77].

MARL tasks can be broadly grouped into three categories, namely fully cooperative, fully competitive, and a mix of the two, depending on different applications [78]: (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 [79]. A progression from fully competitive to fully cooperative behavior of agents was also presented in [79] 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 [80, 81]. To tackle this issue, one way is to exchange some information among agents. In some contexts, agents do exchange information with their peers through some coordination. For example, in the game of hunters capturing preys, Tan [82] 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 that 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 [65, 83, 84].

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. [85] 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. [65] 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.

3.1.3 Research gaps

Recent studies have successfully applied MARL to order dispatching and driver repositioning problems. Table (3.1) summarizes these papers in terms of MARL set-up and algorithms. Because this chapter is focused on repositioning, we will mainly highlight the research gaps that exist in the studies on multi-driver repositioning.

There are a few studies that have employed MARL for the task of driver repositioning [1, 69]. For each time period, based on a difference matrix between the supply and the predicted demand, Yang et al. [69] adopted a WoLF (Win or Learn First) policy hill-climbing algorithm to reposition cooperative drivers with the goal of balancing demand and supply. Cumulative reward of drivers over multiple time periods, however, is not optimized. Lin et al. [1] proposed a contextual actor-critic model to reposition thousands of drivers with each driver aims to maximize her cumulative reward. To make training tractable, a global state which includes the overall distribution of demand and supply is used. Global information, however, may not be accessible from the perspective of an agent, especially during execution. To fill the above gaps, this chapter aims to tackle the multi-driver repositioning task only using local observation of each driver.

Table 3.1: Existing research of order dispatching and driver repositioning using MARL

Research problem	Reference	Agent	State representation	Reward	Algorithm	Reward design	Gap
Order dispatching	[65]	Driver	(grid_id, time, on-trip flag)	Fare, destination potential, pickup distance	Mean field actor-critic	No	-
	[66]	Driver	(grid_id, time, dynamic features, static features)	Fare	Semi MDP and cerebellar value networks	No	-
	[67]	Passenger	(grid_id, cumulative waiting time, expected distance to matched driver, global distribution of supply and demand)	A constant reward for a match, cost	Spatial-temporal multi-agent actor-critic	No	-
	[68]	Driver	(grid_id, number of idle vehicles, number of valid orders, distribution of orders' destinations)	Fare (proportional to distance)	Action selection Q-learning	No	-
Driver repositioning and order dispatching	[86]	Grid	(number of vehicles, number of orders, entropy, number of vehicles to reposition, distribution of order features)	Accumulated driver income, order response rate	Deep deterministic policy gradient	No	-
Driver repositioning	[1]	Driver	(grid_id, time, global state)	Fare (averaged in each grid), fuel cost	Contextual actor-critic	No	Using global information (i.e. global state)
	[69]	Driver	(grid_id)	100 points if the chosen action leads to a balance between demand and supply otherwise -1	WoLF(Win or Learn First) policy hill-climbing	No	Not optimizing cumulative reward
	This work	Driver	(grid_id, time)	Fare	Mean field actor-critic	Yes	

Although the approaches listed in Table (3.1) are efficient under a given reward function, the reached equilibrium is very likely to be suboptimal from the overall perspective of the system. In other words, there may exist another reward function that yields a better equilibrium in terms of some metrics such as gross merchandise volume (GMV) and order response rate (ORR) in the context of taxi hailing. One possible approach to achieve a better equilibrium is to intentionally craft a reward function that is aligned with the goal of the overall system. For example, Li et al. [65] proposed the order destination potential as a component in the reward function to enable some cooperation among drivers and discourage drivers from being selfish. Human-drivers, however, are selfish in nature and focus on their own monetary return. Thus, the embedded goal of the system in the predefined reward function may not reflect the intrinsic interest of drivers, and therefore drivers may not follow the derived optimal policy. To address the above limitations of a predefined reward function, this research aims to achieve a more desirable equilibrium by adjusting drivers' monetary return through some realistic measures of the system. For example, e-hailing platforms can adjust the platform service charge (aka the commission fee) to achieve a better GMV or ORR. Likewise, city planners commonly use congestion pricing to drive the traffic system performance towards a system optimum in transportation network design problems [87, 88, 89, 90, 91, 92, 93]. We call these measures as "**reward design**" mechanism.

In this chapter, 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. [94] 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 designer's 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.

3.1.4 Contributions of this chapter

The major contributions of this chapter are as follows:

1. To achieve a better equilibrium for the overall system, a reward design scheme is proposed, formulated as the upper level optimization in a bilevel mathematical program, to adjust drivers' monetary return for social optima.

2. With the repositioning task as the lower level in the bilevel mathematical program, a mean field actor-critic algorithm is employed to enable the learning involving thousands of agents. A Bayesian optimization algorithm is then developed to solve the bi-level problem.

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 chapter is organized as follows. Section (3.2) introduces the single-agent actor-critic algorithm, which is a stepping stone for MARL. Section (3.3) presents the mean field multi-agent reinforcement learning algorithm. Section (3.4) presents a reward design mechanism and formulates a bilevel optimization problem. Section (3.5) presents the result and validates the effectiveness of the proposed reward design. Section (3.6) summarizes this chapter.

3.2 Single agent reinforcement learning

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

3.2.1 Problem definition

To make this chapter self-explanatory, we briefly repeat the introduction of MDPs. An MDP 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 [47] $V^\pi(s) = \mathbb{E}_{a\sim\pi(\cdot|s),s'\sim P(\cdot|s,a)}\left[r(s, a, s') + \gamma V^\pi(s')\right]$. 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 [47]:

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

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 formulate an MDP under the context of e-hailing driver reposition, we will use examples on a 2-by-2 grid world throughout the chapter every time when models are introduced.

Example 3.2.1. (Single-Agent 2×2). The single-agent driver reposition is presented in Fig-

ure (3.1). 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. In addition to the emergence of passenger requests (i.e., orders), the stochastic environment also has an order dispatching component. Considering that the scope of this study is driver repositioning instead of order dispatching, we assume passenger requests are randomly assigned to available drivers within the same grid.

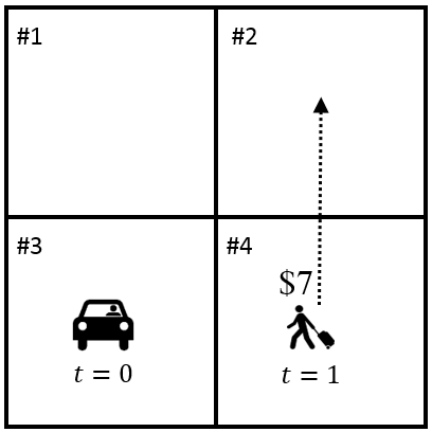


Figure 3.1: An illustrative example (Single agent)

Note that there are a few studies solving vehicle routing problems based on a physical roadway network [8, 54]. Considering that the research scope of this chapter is where to reposition drivers, we adopt an abstract network (i.e., a grid world setup). In other words, this chapter is more focused on the direction along which a driver positions herself after dropping of previous passengers and not on the specific route the driver takes. Using a real network representation for driver-repositioning will be left for future research.

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.

Note. At each time step, a taxi driver can only pick one of the surrounding grids or the current grid as her preferred spot to find a passenger. In other words, taxi drivers are only allowed local search at each time step. Although it seems to be restrictive, it is actually consistent with what real drivers do, and the reason is as follows. From the perspective of a real driver, she can pick a faraway grid as her destination. However, she can not jump directly from her current grid to that grid. Instead, the driver needs to specify a route (i.e., a sequence of grids) and repositions herself one grid each time along the chosen route. If the driver meets a passenger before reaching her chosen destination, she has to take the passenger because she is not allowed to refuse a fare by law. From the perspective of an agent who is conducting local search, although she can not choose a faraway grid, she may end up searching in a faraway grid by repositioning herself one grid at a time. Actually, this definition of action is widely used in driver repositioning problems in the literature [1, 6, 70].

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 0 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 $P(s'|s, a) = 20\%$.

Note. In the temporal component of state s', we assume searching time is 1, i.e., the time that one idle driver spent on cruising from her current grid to one of neighbor grids, and use $\Delta t_{\#4 \to \#2}$ to denote the time that one on-duty driver spent on transporting the passenger to the destination.

Actually, the travel time from passenger's origin to the destination is extracted from some historical data. In addition, the length of each time slice is 1 in this example.

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.

□

3.2.2 Actor-Critic method

To derive 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 [95, 96]. To overcome the shortcomings of these methods, actor-critic methods are developed to combine strengths of both methods [95].

Figure (3.2) 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 $s \rightarrow s'$ and receives a reward r from

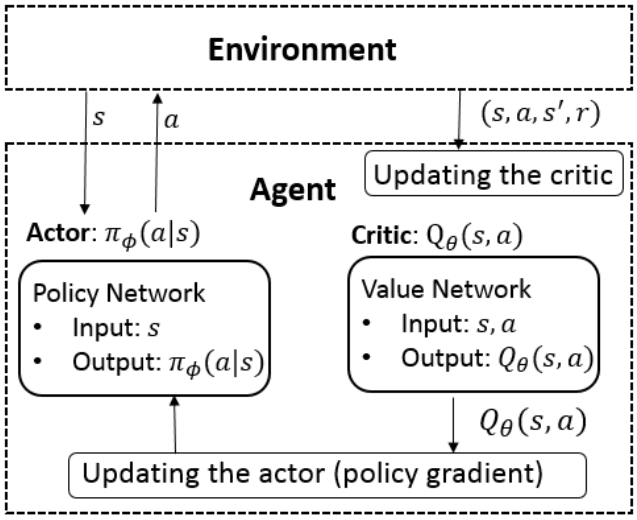


Figure 3.2: Actor-critic algorithm

the environment. Based on the one-step transition $s \rightarrow s'$ 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 $s \rightarrow s'$ with reward $r(s, a, s')$ and updates the Q-value by

$$Q(s, a) \leftarrow Q(s, a) + \eta[r(s, a, s') + \gamma \max_{a'} Q(s', a') - Q(s, a)], \tag{3.1}$$

where η is the learning rate and $0 < \eta \leq 1$. If η reduces over time properly, the Q-learning update converges [47]. Equation (3.1), 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 [97]. 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

$$\mathcal{L}(\theta) = \mathbb{E}_{s,a,s'}[\underbrace{(r(s,a,s') + \gamma \max_{a'} Q_{\theta^-}(s',a')}_{\text{target}} - Q_\theta(s,a))^2]. \tag{3.2}$$

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. Q_{θ^-} is a target network which is a copy of Q_θ. The target network Q_{θ^-} is not updated by the gradient descent and is copied from Q_θ after a certain number of steps to ensure training stability [97].

Actor. The actor takes as input state s and outputs a probability distribution on all allowable actions in this state. Similar 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 [98]

$$\nabla_\phi \rho(\pi_\phi) = \mathbb{E}_{s,a}[\underbrace{(Q_{\theta^-}(s,a) - b_{\theta^-}(s))}_{\text{advantage}} \nabla_\phi log(\pi_\phi(a|s))], \tag{3.3}$$

where b_{θ^-} is some baseline (e.g., $b_{\theta^-} = V_{\theta^-}$, i.e., the value function following the policy V_{θ^-}), and $Q_{\theta^-}(s,a) - b_{\theta^-}(s)$ 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 (3.3) 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., $Q_{\theta^-}(s,a) - b_{\theta^-}(s) > 0$, 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 (3.3) 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 (3.2)) and the actor (by the gradient given in Equation (3.3)) as more samples are fed in.

3.3 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 multi-agent reinforcement learning approach to model the multi-driver repositioning task. Drivers are assumed to be intelligent and perfectly rational, meaning they select the best repositioning strategies to maximize their cumulative rewards. Bounded rationality [99, 100] is left for future research.

3.3.1 Problem definition

The multi-agent problem is modeled as a partially observable Markov game[101], 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.,

$$Q_i = Q_i(\mathbf{s}, \mathbf{a}). \tag{3.4}$$

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 3.3.1. (Multi-Agent 2×2). The multi-agent driver reposition is presented in Figure (3.3). 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.

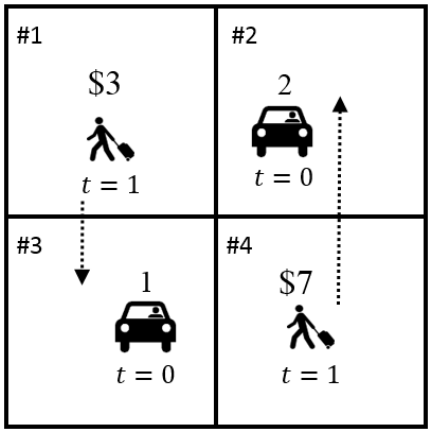


Figure 3.3: An illustrative example (Multi agent)

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 $\mathbf{s} \rightarrow \mathbf{s}'$ 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 $s_1' = (\#2, 1 + \Delta t_{\#4 \rightarrow \#2})$ where $\Delta t_{\#4 \rightarrow \#2}$ is the transition time from grid #4 to grid #2. Similarly, 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 $s_2 = (\#3, 1 + \Delta t_{\#1 \rightarrow \#3})$ where $\Delta t_{\#1 \rightarrow \#3}$ is the transition time from grid #1 to grid #3. $\mathbf{s}' = ((\#2, 1 + \Delta t_{\#4 \rightarrow \#2}), (\#3, 1 + \Delta t_{\#1 \rightarrow \#3}))$. 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.

3.3.2 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 (3.4), 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

$$Q_i = Q_i(o_i, a_i). \tag{3.5}$$

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 [80].

3.3.2.1 Centralized training and decentralized execution

To make the training more stable and ensure convergence, we employ the centralized training and decentralized execution paradigm [1, 83, 84, 65]. 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

$$Q_i = Q_i(o_i, o_{-i}, a_i, a_{-i}), \tag{3.6}$$

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

$$Q_i = Q_i(o_i, a_i, a_{-i}). \tag{3.7}$$

3.3.2.2 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 [85]. 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

$$Q_i \approx \frac{1}{N_i} \sum_{k \in N(i)} Q_i(o_i, a_i, a_k) \approx Q_i(o_i, a_i, \bar{a}_i). \tag{3.8}$$

Interested readers can refer to [85] for a detailed explanation and proof.

Example 3.3.2. (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 (3.3), 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. □

3.3.3 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, however, there are typically hundreds of or even thousands of agents, resulting in a prohibitively large number of

deep neural networks to maintain, which is not computationally tractable. To address this issue, we assume drivers are homogeneous and they share the same state space, action space, and reward function. We believe this assumption is reasonable when agents are anonymous and one's influence to the entire system performance is negligible. While heterogeneity across agents does exist when agents have different reward functions [70], this study aims to reveal some insights of multi-driver repositioning by learning a policy for the generic taxi population. A more personalized reposition scheme that captures the heterogeneity across the driver population is left for future research.

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 (3.2) for the single-agent setting, now becomes

$$\mathcal{L}(\theta) = \mathbb{E}_{o_i,a_i,o_i'}(r(o_i, a_i, o_i') + \gamma \sum_{a_i'} \pi_{\phi^-}(a_i'|o_i')\mathbb{E}_{\bar{a}_i'}[Q_{\theta^-}(o_i', a_i', \bar{a}_i')] - Q_\theta(o_i, a_i, \bar{a}_i))^2. \tag{3.9}$$

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 (3.3) for single-agent setting, is now

$$\nabla_\phi \rho(\pi_\phi) = \mathbb{E}_{o_i,a_i}[(\mathbb{E}_{\bar{a}}[Q_{\theta^-}(o_i, a_i, \bar{a}_i)] - V(o_i))\nabla_\phi log(\pi_\phi(a_i|o_i))], \tag{3.10}$$

where the baseline $V(o_i) = \sum_{a_i} \pi_{\phi^-}(a_i|o_i)\mathbb{E}_{\bar{a}_i}[Q_{\theta^-}(o_i, a_i, \bar{a}_i)]$.

The mean field actor-critic algorithm is presented in Algorithm (2). With the input of some parameters such as the exploration parameter ϵ, target network update period τ, and learning rate η, number of training batches K, and batch size b, a neural network Q_θ for the critic and a neural network π_ϕ for the actor are initialized. Target networks Q_{θ^-} and π_{ϕ^-} are copied from Q_θ and π_ϕ, respectively. An experience replay buffer B is also initialized to store experience tuples of all agents. Now we let agents repetitively interact with the environment and update the actor and the critic as follows. Agents are randomly placed in the environment initially. All agents choose

Algorithm 2 Mean field actor-critic algorithm

Input: exploration parameter $\epsilon = \epsilon_0$, target network update period τ, learning rate $\eta = \eta_0$, batch size b, number of training batches K
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 a target Q-value network Q_{θ^-} with parameter $\theta^- = \theta$ and a target policy network π_{ϕ^-} with $\phi^- = \phi$
Initialize a replay buffer B
Set $I = 0$
repeat
    Randomly initialize a starting grid for all agents, and each agent $i \in \{1, 2, \cdots, N\}$ draws a private observation o_i
    for $t = 0$ to T **do**
        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_{ϕ^-} for all available agents
        end if
        Each available agent takes its action a_i, observes a reward r_i and a mean action $\bar{a}_i$, and draws a new observation o_i'
        Store the experience tuple $\{o_i, a_i, r_i, \bar{a}_i, o_i'\}$ into the replay buffer B
    end for
    for $k = 1$ to K **do**
        Sample b experience tuples from the buffer B
        Update the critic Q by minimizing the loss defined Equation (3.9)
        Update the actor π using the gradient defined in Equation (3.10)
    end for
    Decrease the exploration parameter ϵ
    Decrease the learning rate η
    $I = I + 1$
    if I mod $\tau = 0$ **then**
        Update the parameter of target networks, i.e., $\theta^- \leftarrow \theta$ and $\phi^- \leftarrow \phi$
    end if
until the algorithm converges
Return the optimal policy π

actions according to the ϵ-greedy method. That is, agents have ϵ probability of choosing actions randomly and $1 - \epsilon$ probability of choosing actions according to the target policy network π_{ϕ^-}. Each agent i then executes the chosen action a_i in the environment, observes a reward r_i and a mean action $\bar{a}_i$, and draws a new observation o_i'. This experience tuple $\{o_i, a_i, r_i, \bar{a}_i, o_i'\}$ is stored

in the replay buffer B. All idle agents then choose actions again and repeat the aforementioned procedure until reaching the terminal state with $t = T$. After collecting experience tuples of all agents, b sample experience tuples are randomly sampled from the replay buffer B and used to train neural networks Q_θ and π_ϕ by Equations (3.9) and (3.10).

Example 3.3.3. (Multi-Agent 2×2). Now we apply the mean field actor-critic algorithm to the multi-driver example shown in Figure (3.3).

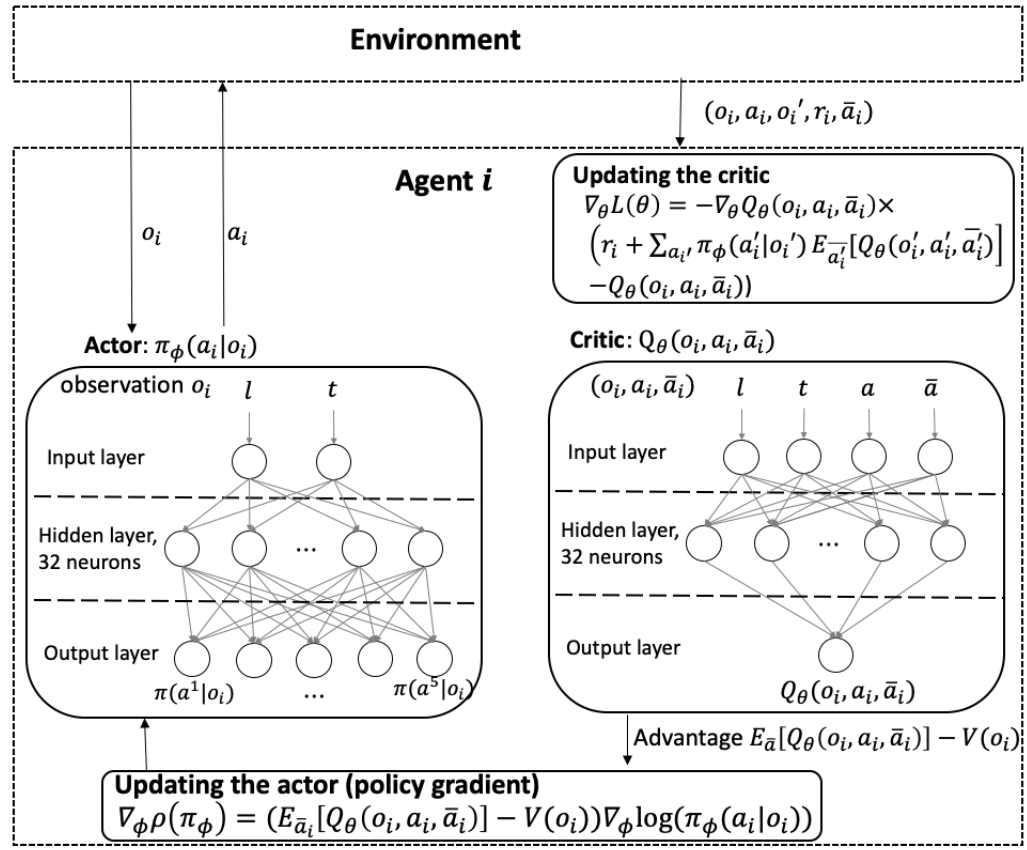


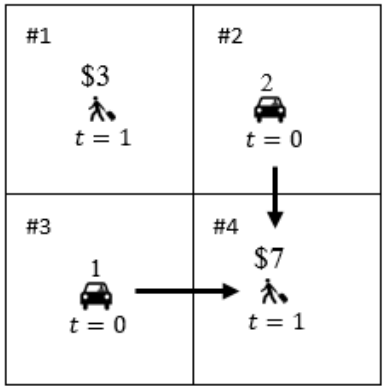
Figure 3.4: Mean field actor-critic algorithm for multi-driver repositioning

Figure (3.4) 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 $\mathbf{s} \rightarrow \mathbf{s}'$ 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 (3.9). Based on the advantage calculated from the critic, agent i updates the shared actor using the gradient presented in Equation (3.10).

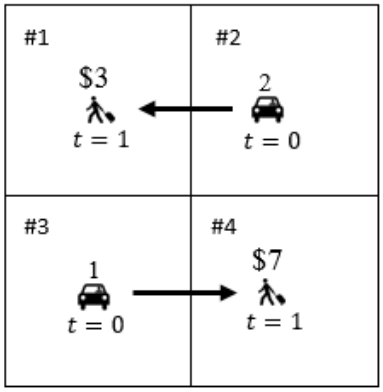
The aforementioned training process is centralized because the mean action used in the critic is actually some global information. During execution, agents only need to use the updated actor, which only takes as input the local information, i.e., the private observation. In other words, the shared critic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

The derived Q values corresponding to four scenarios of interest are presented in Figure (3.5). In Figure (3.5a), when both drivers choose action #4, the observed mean action for both of them is the ratio of demand to supply, i.e., $\bar{a}_1 = \bar{a}_2 = \dfrac{\text{1 passenger request}}{\text{2 drivers}} = 0.5$. 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 $\dfrac{\text{1 driver}}{\text{2 drivers}} = 50\%$ 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 (3.2) 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 (3.5d). Similarly, Q-values for both drivers can be read from Figure (3.5) 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.



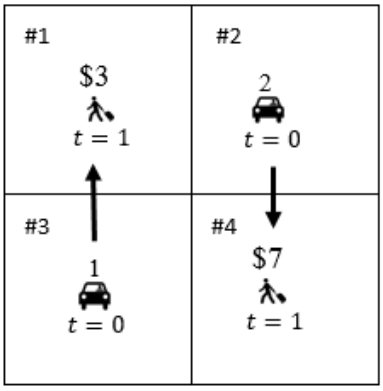
$o_1 = (\#3,0), a_1 = \#4, \bar{a}_1 = 0.5, Q(o_1, a_1, \bar{a}_1) = 3.5$
$o_2 = (\#2,0), a_2 = \#4, \bar{a}_2 = 0.5, Q(o_2, a_2, \bar{a}_2) = 3.5$

(a) Both drivers take action #4



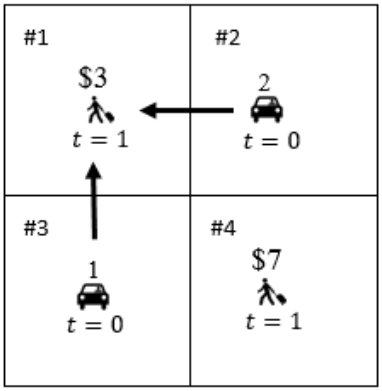
$o_1 = (\#3,0), a_1 = \#4, \bar{a}_1 = 1, Q(o_1, a_1, \bar{a}_1) = 7$
$o_2 = (\#2,0), a_2 = \#1, \bar{a}_2 = 1, Q(o_2, a_2, \bar{a}_2) = 3$

(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 3.5: Derived Q values for four scenarios of interest

Table 3.2: Q-value bimatrix for drivers

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

3.4 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 systematic objectives. To guide an MAS towards a desirable equilibrium, system planners could resort to reward design mechanisms by modifying the reward function of agents (e.g., congestion pricing). In this chapter, 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 optimize some system performance measure dependent of α, denoted as $f(\alpha)$. System planners first choose a value of α and input it to the MAS. With the given α that influences the reward received by distributed agents, the developed mean field actor-critic algorithm is employed to derive an optimal policy π, which is dependent on α, for all distributed agents in the system. The system performance measure f, which is calculated by executing the derived optimal policy π for all agents, is then fed into the reward design for updating the control parameter α. 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,

$$\max_{\alpha \in \mathcal{A}} f(\alpha) \tag{3.11}$$

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 (3.6).

The optimization problem presented in Equation (3.11), however, is not straightforward to solve due to the unknown complex structure of f over the parameter α. Traditional gradient based methods such as gradient descent are thus no longer applicable. In addition, the black-box function f is expensive to evaluate, meaning that evaluating one value of α can take a long period of time.

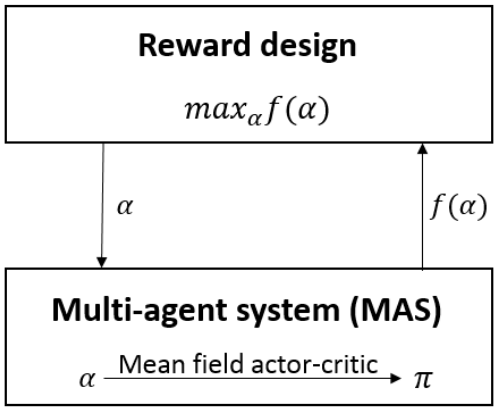


Figure 3.6: Architecture of the reward design

Therefore random or grid search based optimization techniques [102] are less effective because they typically require a large number of evaluations.

To find optima with a limited number of evaluations, we adopt Bayesian optimization [103] (hereafter we call it BO). Due to the unknown structure of the objective function f, BO places a prior on the objective function, for example a Gaussian process (GP). With some observed data (i.e., several evaluations of α's), BO derives a posterior, based on which the next α to be evaluated can then be determined.

We now detail the procedure of BO in our context. First, BO places a prior statistical model on the objective function f, for example a GP in this study. A GP is a stochastic process satisfying the following conditions: 1) For any $\alpha \in \mathcal{A}$, $f(\alpha)$ is a random variable; 2) For a collection of any finite number of α's (i.e., $\alpha_1, \alpha_2, \cdots$, and α_m $\forall m \in \mathcal{R}$), the joint distribution $\boldsymbol{f} = [f(\alpha_1), f(\alpha_2), \cdots, f(\alpha_m)]^T$ is a multivariate Gaussian. Mathematically,

$$\boldsymbol{f} \sim \mathcal{N}(\boldsymbol{\mu}, \boldsymbol{K}),$$

where $\boldsymbol{\mu}$ is a m by 1 mean vector and $\boldsymbol{K}$ is a m by m covariance matrix with each entry K_{ij} denoting the covariance between $f(\alpha_i)$ and $f(\alpha_j)$. A square exponential kernel $\kappa(\alpha_i, \alpha_j)$ is used to calculate

the covariance K_{ij} and is defined as

$$\kappa(\alpha_i, \alpha_j) = \sigma_f^2 e^{-\frac{1}{2l^2}(\alpha_i - \alpha_j)^2},$$

where l is the length parameter and σ_f the variance parameter. The square exponential kernel yields a large covariance when α_i and α_j are close (i.e., $(\alpha_i - \alpha_j)^2$ is small) and a small covariance when α_i and α_j are far away (i.e., $(\alpha_i - \alpha_j)^2$ is large), which is as expected because for a typical function f, $f(\alpha_i)$ and $f(\alpha_j)$ are similar when α_i and α_j are similar, meaning a large covariance. The prior used in this study is a GP with a zero mean (i.e., $\boldsymbol{\mu} = \mathbf{0}$) and a covariance matrix $\boldsymbol{K}$ calculated by the aforementioned square exponential kernel.

Second, with a given prior and some observed data, now our task is to derive a posterior probability distribution of $\boldsymbol{f}^* = [f_1^*, f_2^*, \cdots, f_m^*]^T$ at some $\boldsymbol{\alpha}^* = [\alpha_1^*, \alpha_2^*, \cdots, \alpha_m^*]^T$. We assume $\boldsymbol{f} = [f_1, f_2, \cdots, f_n]^T$ is the observed data at some $\boldsymbol{\alpha} = [\alpha_1, \alpha_2, \cdots, \alpha_n]^T$. By the definition of a GP, the joint distribution $(\boldsymbol{f}, \boldsymbol{f}^*)$ is a multivariate Gaussian. Mathematically, the prior probability distribution of $(\boldsymbol{f}, \boldsymbol{f}^*)$ with zero mean is

$$\begin{bmatrix} \boldsymbol{f} \\ \boldsymbol{f}^* \end{bmatrix} \sim \mathcal{N}\left(\begin{bmatrix} \mathbf{0}_n \\ \mathbf{0}_m \end{bmatrix}, \begin{bmatrix} \boldsymbol{K}_{n\times n} & \boldsymbol{K}^*_{n\times m} \\ \boldsymbol{K}^{*T}_{m\times n} & \boldsymbol{K}^{**}_{m\times m} \end{bmatrix} \right),$$

where $\mathbf{0}_n$ is n-dimensional vector with all entries as zeros, $\mathbf{0}_m$ m-dimensional vector with all entries as zeros, $\boldsymbol{K}$ a n by n covariance matrix with each entry $K_{ij} = \kappa(\alpha_i, \alpha_j) + \sigma_y^2 \delta_{ij}$ (σ_y is the noise in the observed data and $\delta_i j$ is the Kronecker delta), $\boldsymbol{K}^*$ a n by m matrix with each entry $K^*_{ij} = \kappa(\alpha_i, \alpha_j^*)$, and $\boldsymbol{K}^{**}$ a m by m matrix with each entry $K^{**}_{ij} = \kappa(\alpha_i^*, \alpha_j^*)$. According to the multivariate Gaussian theorem [104], the probability distribution of $\boldsymbol{f}^*$ conditioned on the observed $\boldsymbol{f}$ is

$$p(\boldsymbol{f}^* | \boldsymbol{f}) \sim \mathcal{N}(\boldsymbol{\mu}^*, \boldsymbol{\Sigma}^*), \tag{3.12}$$

where $\boldsymbol{\mu}^* = \boldsymbol{K}^{*T}\boldsymbol{K}^{-1}\boldsymbol{f}$ and $\boldsymbol{\Sigma}^* = \boldsymbol{K}^{**} - \boldsymbol{K}^{*T}\boldsymbol{K}^{-1}\boldsymbol{K}^*$.

Third, with the derived posterior probability distribution of $\boldsymbol{f}^*$ conditioned on the seen data

$\boldsymbol{f}$, we now aim to define an acquisition function, based on which the next α to be evaluated can be determined. The acquisition function used in this study is the upper confidence bound (UCB) [105], i.e.,

$$UCB(\alpha) = \mu^*(\alpha) + \sqrt{2log(t^{d/2+2}\pi^2/3\delta)}\sigma^*(\alpha), \tag{3.13}$$

where $\mu^*(\alpha)$ is the posterior mean at α, $\sigma^*(\alpha)$ the posterior standard deviation at α (derived from $\boldsymbol{\Sigma}^*$), d is the dimension of the search space of α, and δ is a parameter. The next α' to be evaluated is simply

$$\alpha' = \arg\max_{\alpha \in \mathcal{A}} UCB(\alpha). \tag{3.14}$$

Remark. With the acquisition function defined as UCB in Equation (3.13), it has been shown that BO is no regret with high probability and has a lower-bound on the convergence rate [106]. Interested readers are referred to [105, 106, 107] for more details on global optimality and convergence properties of BO.

The pseudo-code of BO used in this study is listed in Algorithm (3).

Algorithm 3 Bayesian Optimization

```
Initialize a GP prior with zero mean on the objective function f
Set computational budget B, a initial number of evaluations b_0, and b = b_0
Evaluate f at b_0 randomly chosen α's, i.e., f = [f(α_1), f(α_2), ··· , f(α_{b_0})]
while b ≤ B do
    Calculate the posterior probability distribution of f* conditioned on f by Equation (3.12)
    Calculate the acquisition function UCB by Equation (3.13) with t = b
    Locate the next α' to be evaluated by Equation (3.14)
    Evaluate f at α' and append f(α') to f
    b ← b + 1
end while
Return α that maximizes f
```

To be more concrete, now we use the multi-agent 2×2 example presented in Figure (3.3) to illustrate the potential of the reward design.

Example 3.4.1. (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 (3.3), 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, leading to a 100% ORR.

3.5 Case Study

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

3.5.1 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.

3.5.1.1 Lower level MARL setup

The dataset consists of seventy deterministic passenger requests in a 2-by-2 grid world setup, as shown in Figure (3.7). At $t = 0$, there are fifty idle drivers in grid #2 and fifty in grid #3. At time $t = 1$, fifty passenger requests with fare \$10 deterministically appear in grid #4 and twenty

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., $o_i = (l_i, t)$. In a grid world setup with square grids, the action space is five dimensional and is to enter one of neighboring grids or to stay at the current grid. If a driver chooses an action that will lead her out of the study area after taking the action, the driver receives a large penalty (i.e., a large negative reward) and is forced to stay in the current grid. Thus drivers will not choose an action that will lead them out of study area after training. The reward that a driver earns is her monetary return. That is,

$$r = \text{fare} \times (1 - SC_l),$$

where SC_l is the service charge in percentage if the driver takes the passenger request in grid l. SC_l is defined later in Equation (3.15).

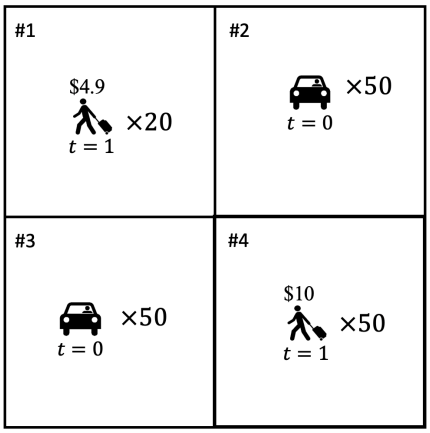


Figure 3.7: Layout

3.5.1.2 Upper level objective function

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., 100 drivers compete for 50 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 $50/70 = 71.43\%$, 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 that 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 formally provide two notations, namely demand to supply ratio (DS) and service charge (SC). We then construct an effective 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.,

$$SC_l = \begin{cases} \alpha \times (1 - DS_l) & \text{if } DS_l \leq 1, \\ 0 & \text{otherwise,} \end{cases} \tag{3.15}$$

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. Parameter α is the SC at $DS_l = 0$.

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

$$f = w \times \text{ORR} + (1 - w) \times (1 - \text{OSC}). \tag{3.16}$$

OSC is defined as

$$OSC = \frac{\sum_l \sum_{\text{order} \in \text{serviced\_orders}} \mathbb{1}_{\text{order}_{\text{origin}} \in l} \times SC_l \times \text{fare}_{\text{order}}}{\sum_{\text{order} \in \text{serviced orders}} \text{fare}_{\text{order}}},$$

where the denominator is the total amount of fare of all serviced orders and the numerator is the total amount of service charge.

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 − OSC) as the objective of the platform, i.e., 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.

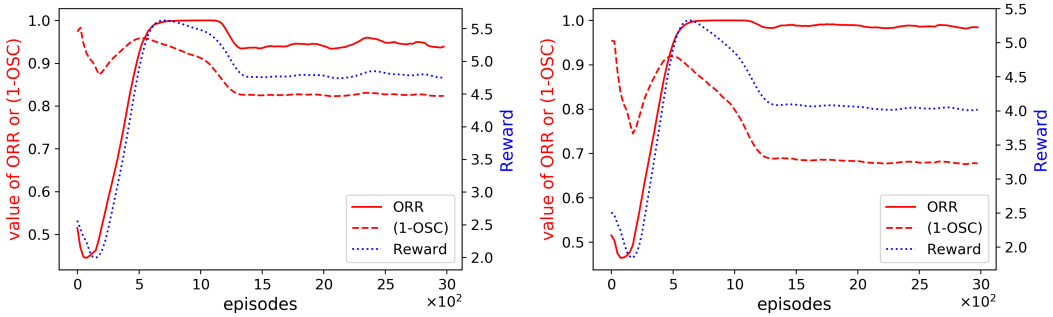
3.5.1.3 Results

We use two methods, namely BO and an analytical method, to derive the optimal value of α.

1. **BO**. We first employ BO with the objective function given in Equation (3.16). When we apply the mean field actor-critic algorithm to the lower level MAS, the critic and the actor are both parameterized by a multilayer perceptron (MLP) with three hidden layers (64, 32, 8). ReLU is used as the activation function between hidden layers, and a softmax function is used to transform the final output from the actor to be a probability distribution. Learning rates η for both the actor and the critic are 10^{-4}. Discount factor $\gamma = 1$ under the assumption that drivers typically do not depreciate their monetary income on the same day. Exploration

parameter is initially set as $\epsilon_0 = 0.5$ and linearly decreases to a minimum value of 0.01. The target network update period is $\tau = 10$, meaning that the parameters of target networks are updated every 10 episodes.

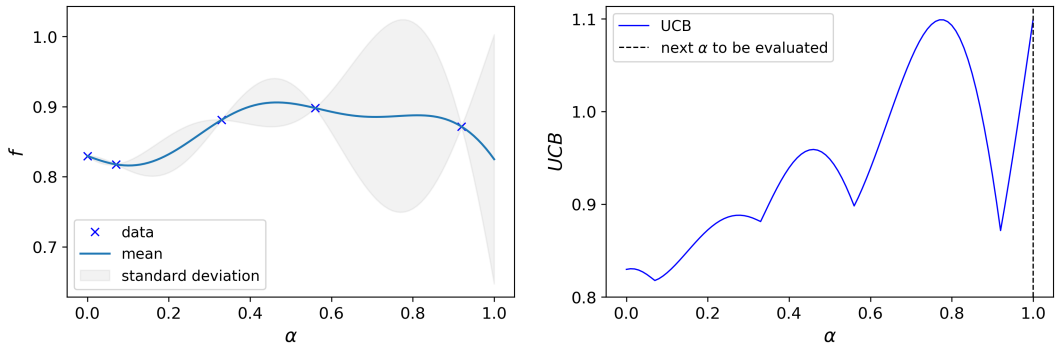
For a bilevel optimization problem, first we need to check the convergence of the lower level. As an example to validate the convergence, ORR, (1 - OSC), and the average reward of all drivers (i.e., Reward in Figure (3.8)) versus the index of episodes are presented in Figure (3.8) with $\alpha = 0.5$ and $\alpha = 1.0$. In both scenarios, despite an initial downturn, ORR and reward increase quite fast and gradually reach convergence after 1,300 episodes. The converged values of ORR are 0.94 for $\alpha = 0.5$ and 0.98 for $\alpha = 1.0$, and the converged values of reward are 4.76 and 4.01 for $\alpha = 0.5$ and $\alpha = 1.0$, respectively. After bouncing back and forth during the first 1,300 episodes, (1 - OSC) reaches its converged values of 0.82 and 0.68 for $\alpha = 0.5$ and $\alpha = 1.0$, respectively. During the first 1,300 episodes, agents explore the environment and learn the optimal policy. After 1,300 episodes, agents mainly exploit the knowledge they have gained through their previous explorations.



(a) $\alpha = 0.5$ (b) $\alpha = 1.0$

Figure 3.8: Convergence of the lower level MARL

With the validated convergence of the lower level MARL, we now run BO to solve the bilevel optimization problem. As for the starting point of BO, in total we evaluate five α's including $\alpha = 0$ (i.e., without any service charge). Figure (3.9a) plots the mean and standard deviation of the posterior probability distribution of f conditioned on these five evaluated points. As one can see, the standard deviation is small around the points that have been evaluated and



(a) Posterior probability distribution of f conditioned on the initial five evaluated points

(b) Acquisition function (UCB)

Figure 3.9: Posterior probability distribution and acquisition function at iteration 0

is large at locations where we do not have any data. The acquisition function shown in Figure (3.9b) reveals that the next α to be evaluated is 1.0.

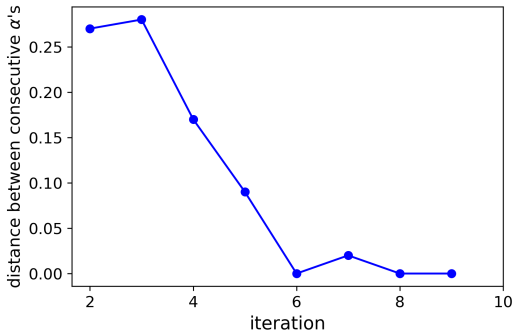


Figure 3.10: Convergence of BO

With the initial five evaluated points, we run BO until convergence. The convergence of BO in this dissertation is defined as choosing four consecutive α's with the distance between every two consecutive α's below a threshold value. In this case study, the threshold value is taken as 0.05. In other words, BO converges when it starts choosing similar α's to evaluate. The convergence is presented in Figure (3.10). Note that the horizontal axis starts with iteration index 2 because it is meaningless to calculate the distance between α in the first iteration and the initial five α's. We can see that BO initially chooses quite different α's and gradually converges from the 6^{th} iteration. In other words, BO chooses similar α's from the 6^{th} iteration.

The resulting posterior probability distribution of f from BO is presented in Figure (3.11). It is noticeable that the evaluation of the objective on α's seems a bit noisy. In other words, the evaluated objective may be slightly different at the same α. This is expected because there are multiple local optima when solving the lower level MARL. Actually, it is commonly impossible to find a global optimum using deep learning. Thus researchers usually settle for local optima [108]. 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. The optimal range of α is $[0.50, 0.58]$, yielding an optimal objective of 0.90 (0.896, to be precise). Nonetheless, the optimal value of α is 0.54, yielding an optimal mean objective $f = 0.90$, which is 8.4% higher than the objective $f = 0.83$ without any reward design. Also, the analytical solution is plotted and appears to agree well with the fitted curve. The derivation of the analytical solution will be explained in the next section.

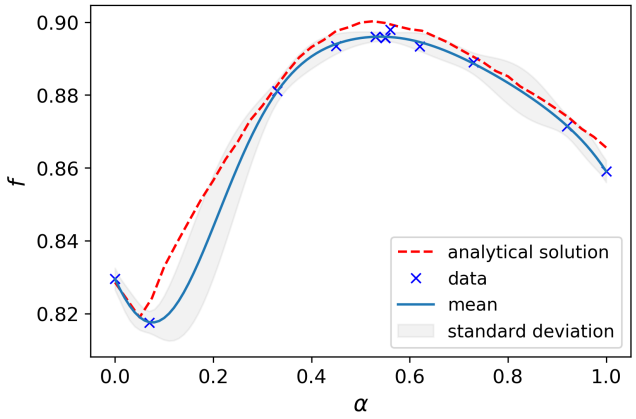


Figure 3.11: Posterior probability distribution of f after BO converges

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, some drivers will choose to enter grid #1 instead of grid #4 for a higher monetary return. We now present both a deterministic and a stochastic analytical method.

In the deterministic analytical method, we assume the number of drivers entering a grid is always an integer. 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 99 drivers entering grid #4, resulting a $50/99 = 0.51$ DS ratio in grid #4. $SC_{\#4} = \alpha \times (1 - 0.51) = 0.49\alpha$, meaning that the expected return for these 99 drivers is $\frac{\$10 \times (1 - 0.49\alpha) \times 50}{99} = 5.05 \times (1 - 0.49\alpha)$. The expected return for the driver entering grid #1 is \$4.9. We then have the critical condition $5.05 \times (1 - 0.49\alpha) = 4.9$, yielding $\alpha = 0.06$. Similarly, we calculate the critical value of α below which there are 19 drivers choosing to enter grid #1 while above which there are 20 drivers attracted by grid #1, and the critical value is $\alpha = 0.58$.

Table 3.3: Values of interest

α	0	0.06	0.58 (optimal)
DS ratio in grid #1 ($t = 1$)	N.A. (supply is zero)	$20/1 = 2000\%$	$20/20 = 100\%$
DS ratio in grid #4 ($t = 1$)	$50/100 = 50.0\%$	$50/99 = 50.5\%$	$50/80 = 62.5\%$
ORR	$50/70 = 71.4\%$	$51/70 = 72.9\%$	100%
OSC	0	0.03	0.18
f	0.83	0.83	0.93 (optimum)

Values of interest are presented in Table (3.3). 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 0.06, there is one driver attracted by grid #1, resulting in a 72.9% ORR. The OSC is calculated as follows. The DS ratio in grid #4 is now 0.51, resulting in $SC_{\#4} = 0.06 \times (1 - 0.51) = 0.03$. Thus, $\text{OSC} = \frac{\$10 \times 50 \times 0.03}{\$10 \times 50 + \$4.9} = 0.03$. The objective is $\frac{3}{5} \times \text{ORR} + \frac{2}{5}(1 - \text{OSC}) = 0.83$. 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.

In the stochastic analytical method, agents in the same state share the same stochastic policy. We now derive this optimal stochastic policy at equilibrium. First of all, agents will only choose from two actions, namely either entering grid #4 or entering grid #1 for a positive expected reward. Other actions such as staying in the current grid or going out of the boundary will yield either a zero or a negative reward. We therefore denote the probability of agents currently in grid #2 choosing to enter grid #1 as p_{21} and to enter grid #4 as $p_{24} = 1 - p_{21}$. Similarly, we have p_{31} and $p_{34} = 1 - p_{31}$. The Nash equilibrium for this problem is not unique under a given α. In other words, there are multiple combinations of p_{21} and p_{31} with which no agents can get better off by a unilateral deviation. To ease the analysis, we choose to derive the equilibrium with $p_{31} = 0$, meaning that all 50 agents in grid #3 choose to enter grid #4. To further simplify the notation, $p_{21} = p$. At equilibrium, the expected reward of agents entering grid #1 is the same as the expected reward of agents entering grid #4. Mathematically, the former is

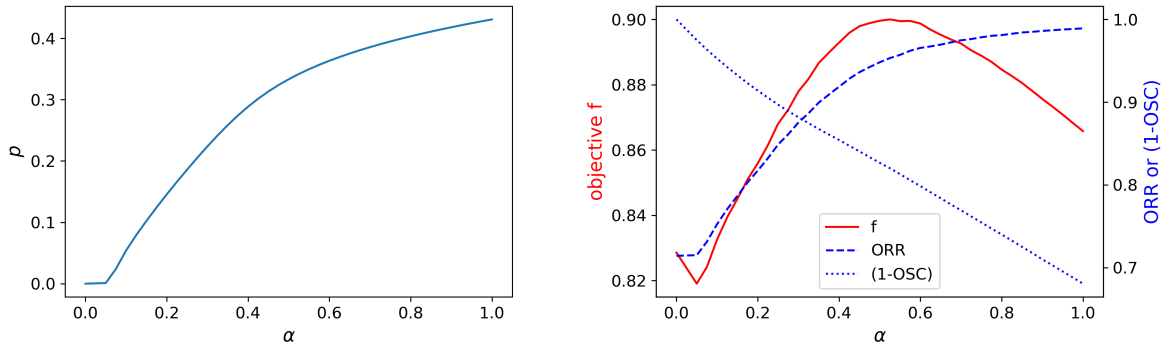
$$\begin{aligned}(1-p)^{50}\left[1-\alpha(1-\frac{50}{100})\right]\frac{50\times 10}{100} \\ +\sum_{k=1}^{20}\binom{50}{k}p^k(1-p)^{50-k}\times 4.9 \\ +\sum_{k=21}^{50}\binom{50}{k}p^k(1-p)^{50-k}\left[1-\alpha(1-\frac{20}{k})\right]\frac{4.9\times 20}{k},\end{aligned} \tag{3.17}$$

and the latter is

$$\sum_{k=0}^{50}\binom{50}{k}p^k(1-p)^{50-k}\left[1-\alpha(1-\frac{50}{100-k})\right]\frac{50\times 10}{100-k}. \tag{3.18}$$

Using an efficient numerical algorithm such as binary search, one could find p for any chosen α, as plotted in Figure (3.12a). It could been seen that p increases with α, which is consistent with our previous discussion. Thereafter, one could evaluate the objective f by executing the

derived optimal policy (i.e., p). The derived objective f and its two constituent components are plotted in Figure (3.12b). The objective f first increases with α due to a large increase in ORR and then decreases with α because of the linear decrease in (1-OSC). The objective f peaks at 0.90 around $\alpha = 0.53$.



(a) p versus α (b) The objective f and its two components

Figure 3.12: Stochastic analytical method

The derived optimal value of α from BO, i.e., 0.54, or the optimal range of $[0.50, 0.58]$, agrees well with the analytically derived optimal values of α, i.e., 0.58 from the deterministic analytical method and 0.53 from the stochastic analytical method. The optimum from BO, i.e., 0.90 is quite close to its deterministic analytical counterpart, i.e., 0.93, and is the same as its stochastic analytical counterpart, i.e., 0.90. The small discrepancy between the numerical solution and the deterministic analytical one is explained as follows. In the deterministic analytical solution, the policy for agents is deterministic and exactly twenty 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 20 and the expected number of agents in grid #4 is 80, the probability of 21 agents choosing grid #1 is $\binom{100}{21} \cdot 0.8^{79} \cdot 0.2^{21} = 9.5\%$. This variance reduces both ORR and (1 - OSC), resulting in a lower objective from BO, compared with the objective from the deterministic analytical solution. With the small discrepancy between the numerical solution and the deterministic analytical solution

explained, we note that the numerical solution agrees well with its stochastic analytical counterpart, indicating the effectiveness of the numerical method.

3.5.2 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 [109]. 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 could 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 [110], 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 demand falls by 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 congestion 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 [111]. 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.

Note. Considering that the goal of this case study is to demonstrate the effectiveness of the proposed bi-level optimization model on a real-world large-scale problem, we simplify the modeling of demand to a linear model in which demand falls when fare increases. We acknowledge that

a more realistic econometric model such as a logit model could be adopted to model passengers' willingness to pay and demand elasticity under congestion pricing. However, we may not be able to calibrate the econometric model due to data accessibility. In addition, the calibration process may introduce more uncertainty into our model. Therefore, a linear model is used as a demonstration in this case study, and a more realistic model to capture the coupling effect between the demand and supply is left for future research.

The objective of city planners thus consists of two components, namely, the percentage of idle 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 (3.13) 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 percentage of idle 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.

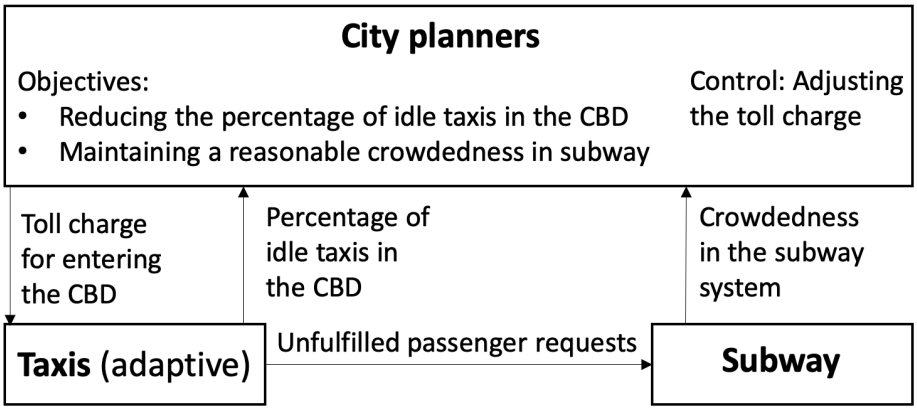


Figure 3.13: Objectives of city planners

3.5.2.1 Data preprocessing

The NYC taxi trip records are publicly available on the official website of NYC Taxi & Limousine Commission (TLC) (https://www1.nyc.gov/site/tlc/about/tlc-trip-record-data.page). We use the data for both yellow and green taxis during June 2016 after the wide adoption of ridesharing service such as Uber and Lyft and before the TLC modifying the format of the taxi trip data (i.e., longitude and latitude of the pickup and dropoff are no longer available after June 2016). A data sample is listed in Table (3.4). Each entry in Table (3.4) collects the order information, including pickup and dropoff time and locations and fares (including tips). In total there are around 12.5 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 3.4: Taxi data sample

pickup date-time	dropoff date-time	pickup longitude	pickup latitude	dropoff longitude	dropoff latitude	fare
2016-06-01 16:00:00	2016-06-01 16:23:23	-73.980446	40.759228	-73.948486	40.777637	17.5
2016-06-01 16:00:01	2016-06-01 16:16:09	-73.993614	40.749981	-73.983543	40.738266	11.0

Figure (3.14) 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 10% 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 (3.15) 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

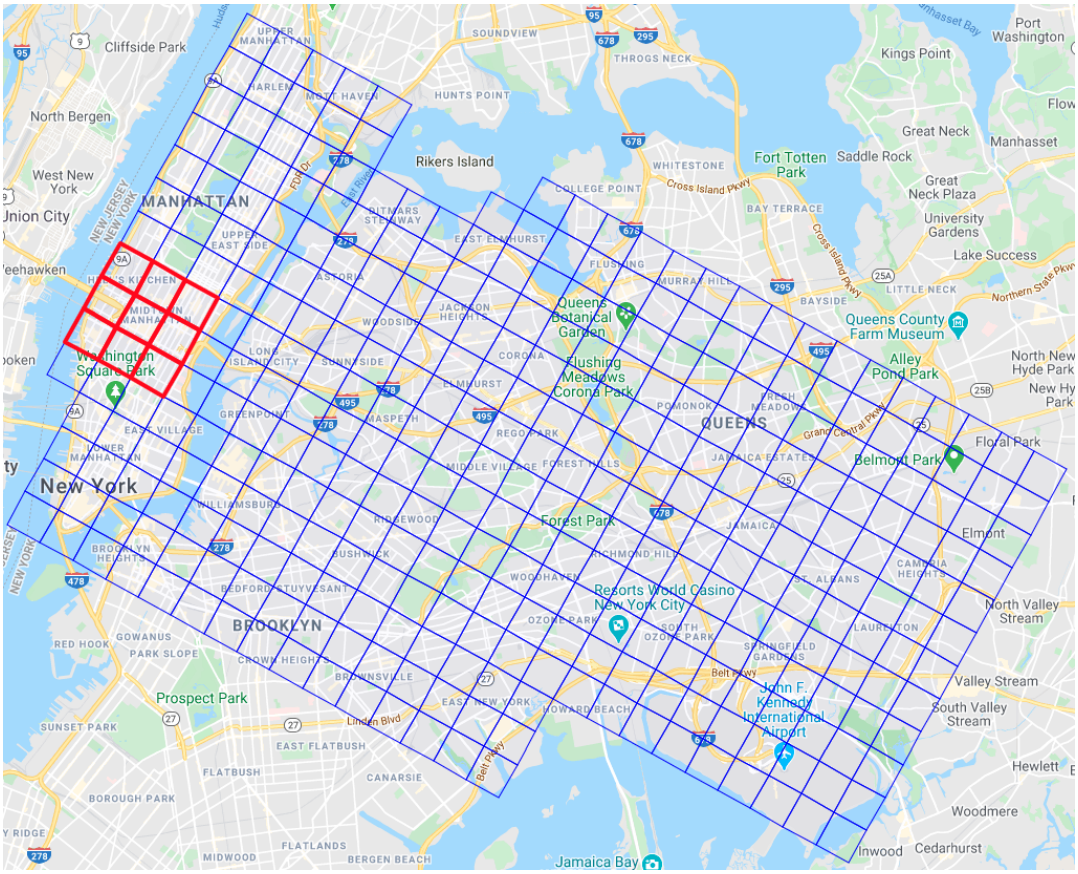


Figure 3.14: Spatial discretization of the area of interest

Figure 3.15: Spatial distribution of taxi orders during evening peak

two local hotspots near two airports.

NYC subway turnstile data is also publicly accessible via the official website of Metropolitan Transportation Authority (http://web.mta.info/developers/turnstile.html). A sample of the turnstile data is listed in Table (3.5). These two rows show that the reading of entries for turnstile ID (A002,R051,02-03-05) is 9,341,345 at 4 PM and 9,342,509 at 8 PM on 06/01/2016. Taking the difference between two readings yields the net entries at this turnstile during the 4-hour time interval, i.e., 9,342,509 - 9,341,345 = 1,164. 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 3.5: Turnstile data sample

Turnstile ID	Date	Time	Entries	Exits
(A002,R051,02-03-05)	06/01/2016	16:00:00	9,341,345	1,261,806
(A002,R051,02-03-05)	06/01/2016	20:00:00	9,342,509	1,261,817

3.5.2.2 Lower level MARL setup

To be concrete, we now provide the setup of the lower level MARL.

- Observation. The observation space for driver i consists of the grid index and current time, i.e., $o_i = (l_i, t)$. One-hot encoding is used for both grid index and time. Yellow and green taxis share the same observation space.

- Action. The action space for driver i is to enter one of neighboring grids or to stay at the current grid. Due to a grid world setup with square grids, the action space is thus 5-dimensional.

- Reward. The reward of each state transition for driver i is her monetary return, i.e.,

$$r = \text{fare} - \text{toll charge},$$

where

$$\text{toll charge} = \begin{cases} \alpha & \text{if driver goes from outside the CBD into the CBD,} \\ 0 & \text{otherwise.} \end{cases}$$

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 Figure (3.4), 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.

3.5.2.3 Upper level objective function of city planners

As previously mentioned, the objective function of city planners consists of two components, namely, percentage of idle 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 idle taxis in the CBD, i.e, the ratio of the number of idle taxis in the CBD to the total number of idle 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 idle taxis in the CBD. Hereafter we call this PTC (**p**ercentage of idle **t**axis in the **C**BD). PTC decreases with toll charge because fewer vacant taxis enter the CBD with a higher toll charge.

$$PTC = \frac{\sum_{t\in\{1,2,\cdots,T\}} \frac{\sum_{l\in CBD} \text{num\_idle\_taxis}_{tl}}{\text{num\_idle\_taxis}_t}}{T},$$

where $\text{num\_idle\_taxis}_{tl}$ is the number of idle taxis in grid l at time step t and num_idle_taxis_t is the number of total idle taxis at time step t. Considering that green taxis do not seek for passengers in the CBD, only yellow taxis are counted when calculating PTC.

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 a grid l, we count the number of passengers of 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 in grid l, denoted as ICS_l^{entry}

where ICS stands for "an *increase in* ***c****rowdedness of the* ***s****ubway*". Mathematically,

$$ICS_l^{entry} = \frac{\sum_{\text{order}\in\text{unserviced orders}} \mathbb{1}_{\text{origin}_{\text{order}}\in l} \times \text{num\_passengers}_{\text{order}}}{\text{net\_enteries}_l}.$$

where $\text{origin}_{\text{order}}$ and $\text{num\_passengers}_{\text{order}}$ denote the origin and number of passengers of the order, respectively. Similarly, we can calculate the increase in the exit crowdedness in grid l as

$$ICS_l^{exit} = \frac{\sum_{\text{order}\in\text{unserviced orders}} \mathbb{1}_{\text{destination}_{\text{order}}\in l} \times \text{num\_passengers}_{\text{order}}}{\text{net\_enteries}_l}.$$

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, namely,

$$ICS_l = \frac{ICS_l^{entry} + ICS_l^{exit}}{2}.$$

Among the overall 337 grids, we focus on the top m grids in terms of crowdedness. Thus the overall ICS is calculated as

$$ICS = \frac{\sum_{l\in\text{top m grids}} ICS_l}{m},$$

where $m = 20$ in this case study. 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:

$$f = -[w \times \text{PTC} + (1 - w) \times \text{ICS}] \tag{3.19}$$

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. We also conduct sensitivity analysis to investigate the impact of w on the final result.

3.5.2.4 Pretraining of actors and critics

In the mean field actor-critic algorithm for this multi-driver repositioning task, both critics (i.e., one for yellow taxis and one for green taxis) are parameterized by an MLP with four hidden layers $(256, 128, 64, 32)$, and both actors are parameterized by an MLP with four hidden layers $(128, 64, 32, 16)$. As is well known, training deep neural networks (i.e., actors and critics) in such a large-scale multi-agent system may be challenging and could suffer from issues such as training instability. In addition, it is commonly impossible to find a global optimum using deep learning, and thus researchers usually settle for local optima [108]. Therefore, we opt for pretraining of actors and critics in this section. Compared with random initialization of neural networks, the neural networks from pretraining could not only save computational resources but also lead to a better optimum. Actually, in the task of repositioning, taxi drivers are already aware of the historical distribution of passenger requests. It is thus reasonable to pre-train actors and critics using historical order information.

First of all, it is a good strategy for taxi drivers to seek for passengers in areas with a larger number of historical taxi orders. As plotted in Figure (3.15), the majority of taxi orders concentrated in Manhattan, especially in the CBD. Therefore, cruising to the CBD, or cruising within Manhattan is a good baseline strategy for drivers. Inspired by this, we pre-train the actor (i.e., the policy network) using the number of orders in each grid and time interval.

The pesudo code for this pretraining is listed in Algorithm (4). In each possible observation $o = (l, t)$, we count the number of average taxi orders. The value of observation o is defined as $V(o) = \min_{l' \in \text{nei}(l)} n_{l',t+1}$, where $\text{nei}(l)$ denotes the neighbor grids of grid l, including grid l itself. Then the advantage of choosing action a from observation o is calculated as $\text{ADS}(o, a) = \dfrac{n_{l_a,t+1} - V(o)}{c_1} + c_2$, where c_1 and c_2 are two scaling parameters and are taken as 10 and 0.1, respectively. Then actors

Algorithm 4 Pretraining of actors

Initialize a replay buffer B
for $o \in O$ **do**
    Count the number of average taxi orders at observation $o = (l, t)$, denoted by $n_{l,t}$
    Calculate baseline value at observation o as $V(o) = \min_{l' \in \text{nei}(l)} n_{l',t+1}$
    Calculate advantage of choosing action a at observation o as $\text{ADS}(o, a) = \frac{n_{l_a,t+1} - V(o)}{c_1} + c_2$
    Store o, a, and $\text{ADS}(o, a)$ into replay buffer B
end for
for $i = 1$ to I **do**
    Sample a batch of size b from the replay buffer B
    Train the actor using the gradient in Eqaution (3.10)
end for
Return pretrained actors

could be trained by using the gradient in Equation (3.10). The rationale of such definition of advantage is as follows. If action a leads to an observation with a larger number of taxi orders, the advantage is large and thus the probability of taking action a will be increased; if action a leads to an observation with a lower number of taxi orders, the advantage is low and thus the probability of taking this action will be decreased. In addition, the use of two scaling parameters c_1 and c_2 smooths advantage values among different actions and prevents actors from outputting one-hot probability distributions.

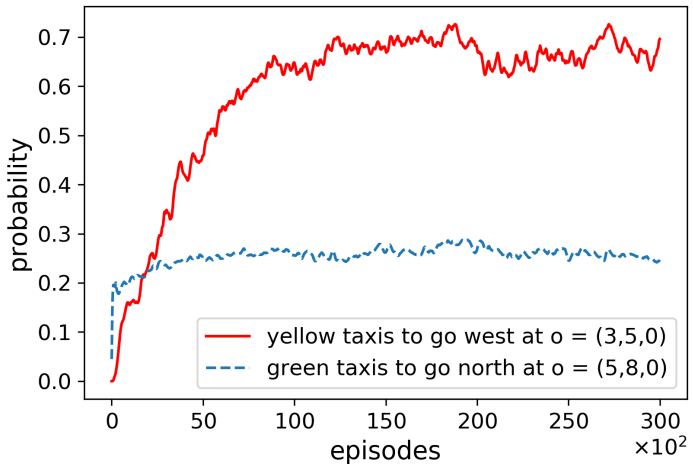


Figure 3.16: Convergence plot of pretraining actors

Figure (3.16) presents the convergence plot of actors for yellow and green taxis. Noticing that actors are neural networks that are hard to be visualized, we simply choose to plot the probability of choosing to go west (i.e., to enter the CBD) at observation $o = (3, 5, 0)$ for yellow taxis and the probability of choosing to go north (i.e., a local hotspot outside restricted areas) at observation $o = (5, 8, 0)$ for green taxis. During the first 12,000 training episodes, the probability of choosing to go west for yellow taxis keeps increasing until reaches its converged value around 0.7. This 70% probability of choosing to enter the CBD indicates that yellow taxis are attracted by the larger number of orders in the CBD, which is as expected. For green taxis, the probability of choosing to go north increases fast during the initial stage and stabilizes within the range of $[0.25, 0.30]$. In a 5-dimensional action space, the average probability of choosing each action is 20%, i.e., a random policy. A probability within the range of $[0.25, 0.30]$ means that on average around 25 to 30 percent green taxis are willing to go north for the local hotspot.

After the pretraining of actors, the pretraig of critics is straightforward. We simply let agents interact with the environment following the pretrained actors to collect experience tuples $(o, a, \bar{a}, r, o')$. We then train critics by minimizing the loss defined in Equation (3.9). Learning rates η in pretraining for both actors and critics are 10^{-4}.

3.5.2.5 Results

On weekdays, there are on average around 80,000 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 as well as the competition between taxis and e-hailing vehicles (e.g., Uber and Lyft), we thus set the number of yellow agents in MARL as 10,000. The number of green agents is set to 3,000 considering that there were fewer taxi orders outside restricted areas for green taxis.

For a bilevel optimization problem, first we need to check the convergence of both actors and critics in the lower level. Note that we no longer use random initialization and start from the

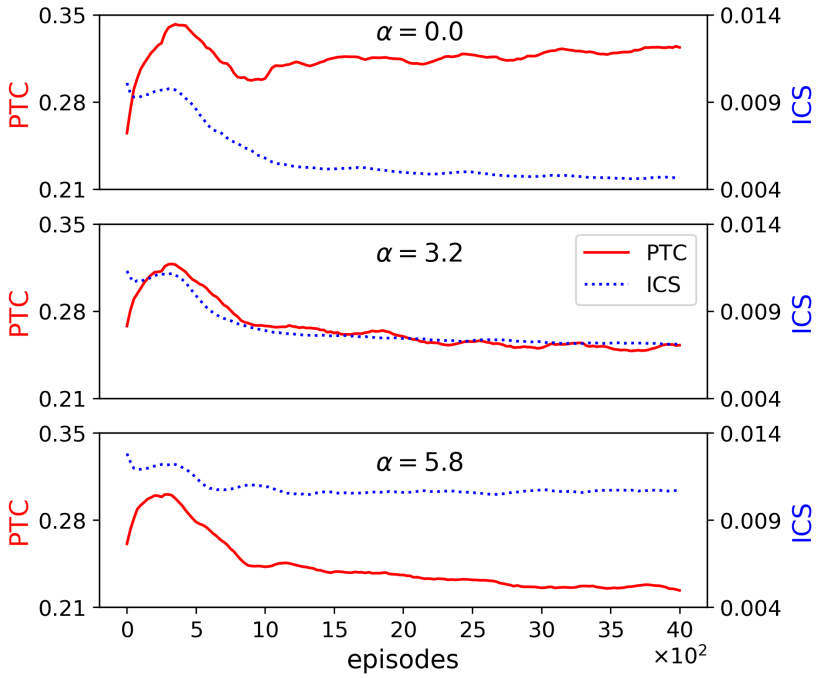


Figure 3.17: Convergence plot of actors

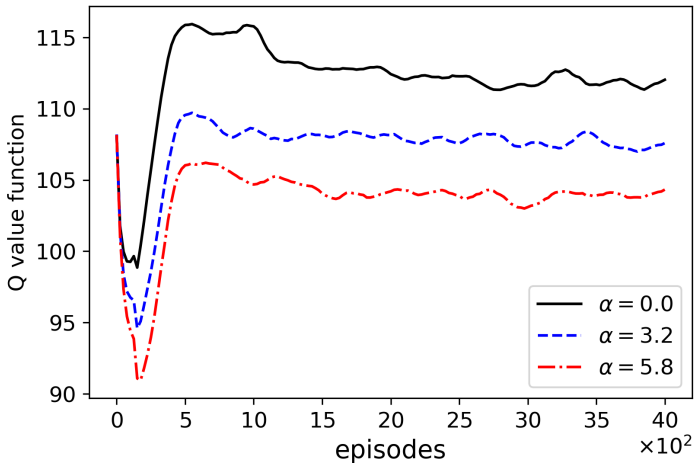
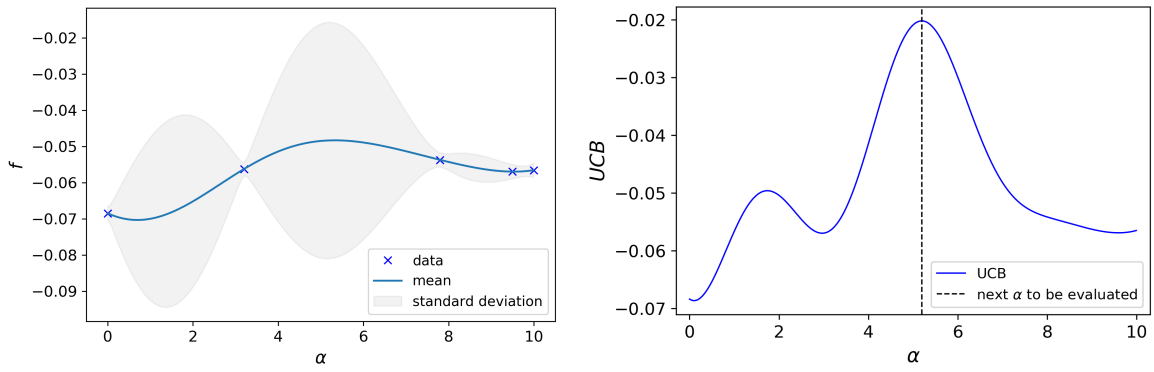


Figure 3.18: Convergence plot of the critic of yellow taxis

pretrained actors and critics. The exploration parameter is initially set as $\epsilon_0 = 0.1$ and linearly decreases to 0.01. Learning rates for both actors and critics are $\eta = 10^{-5}$. The target network update period is $\tau = 10$. As an example to validate the convergence of actors, PTC and ICS of all agents versus number of episodes are presented in Figure (3.17) with $\alpha = 0.0$ (i.e., no toll charge), $\alpha = 3.2$, and $\alpha = 5.8$. PTC and iCS are directly correlated with actors, because PTC and ICS

are obtained through the evaluation of actors. For the purpose of comparison, we use the same scale for PTC or ICS across all scenarios. In all scenarios, despite some bouncing back and forth during the first 1,000 episodes due to the initial exploration of agents, both PTC and ICS gradually reach convergence after 3,000 episodes. The converged values of PTC are 0.32, 0.25, and 0.22 for $\alpha = 0.0$, $\alpha = 3.2$, and $\alpha = 5.8$, respectively. This is as expected, because as α increases, there are fewer idle taxis entering the CBD due to the toll charge, leading to a lower value of PTC. The converged values of ICS are 0.005, 0.007, and 0.011 for $\alpha = 0.0$, $\alpha = 3.2$, and $\alpha = 5.8$, respectively. Increase of ICS in α is also expected due to more unserviced passenger requests when there are fewer taxis entering the CBD. After validating the convergence of actors, we also validate the convergence of critics. As an example, we plot the Q value function of yellow taxis to enter grid $(3, 5)$ at time $t = 0$ with an observed mean action $\bar{a} = 1$. Starting from the same value, the Q values decrease during the first 200 episodes and then increase to their highest values around 500 episodes in all scenarios. The Q values then gradually decrease to and stabilize around their converged values at 112, 107, and 104 for $\alpha = 0$, $\alpha = 3.2$, and $\alpha = 5.8$, respectively.



(a) Posterior probability distribution of f conditioned on the initial five evaluated points

(b) Acquisition function (UCB)

Figure 3.19: Posterior probability distribution and acquisition function at iteration 0

With the validated convergence of the lower level MARL, we use BO to solve the bilevel problem. In total we evaluate five α's as the starting point of BO. Figure (3.19a) plots the mean and standard deviation of the posterior probability distribution of f conditioned on these five evaluated points. As one can see, the standard deviation is small around the points that have been evaluated

and is large at locations where we do not have any data. The acquisition function shown in Figure (3.19b) reveals that the next α to be evaluated is 5.2. According to Figure (3.19a), the mean is high and the standard deviation is high around 5.2, indicating a large acquisition.

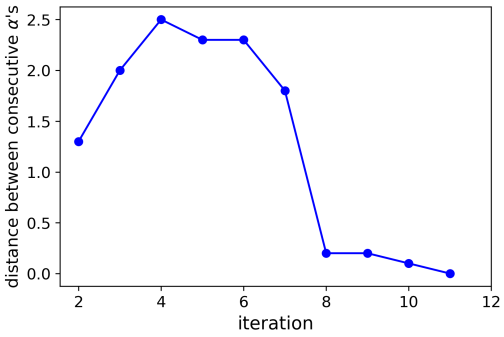
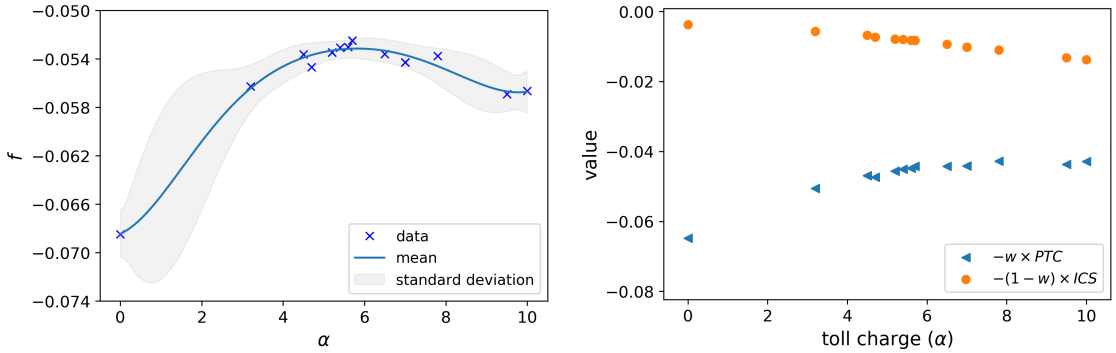


Figure 3.20: Convergence of BO

The convergence of BO in this case is defined as choosing four consecutive α's with the distance between every two consecutive α's below a threshold of 0.5. The convergence of BO is plotted in Figure (3.20). We can see that BO initially chooses quite different α's and gradually converges from the 8^{th} iteration.



(a) Posterior probability distribution of f after BO converges

(b) Two components

Figure 3.21: BO result

The resulting posterior probability distribution of f from BO is presented in Figure (3.21a). The optimal range of toll charge is $[5.5, 6.0]$, yielding an optimal objective of -0.05 (-0.0531, to be precise). Nevertheless, the optimal toll charge is \$5.8 with an optimal objective around -0.05. The objective is -0.07 without any toll charge. The objective increases with the toll charge when

smaller than \$5.8. With a \$5.8 toll charge, the objective is −0.05, which is 22% higher than −0.07. The objective decreases if toll charge is increased beyond \$5.8. The standard deviation within the range of $[3.2, 7.8]$ is quite small because we have enough evaluations within the range so the uncertainty is low. As for the range of $[0, 3.2]$ and $[7.8, 10]$, the standard deviation is larger, indicating a larger uncertainty in these ranges. BO does not choose to evaluate α in these ranges with a large uncertainty because of a low mean value and thus a small acquisition in these ranges. The parabolic shape of the objective can be explained by Figure (3.21b). Before the toll charge reaches \$5.8, 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 \$5.8, $-(1-w) \times ICS$ keeps decreasing and suppresses the effect of the slight increase in $-w \times PTC$, resulting in a decrease in the objective.

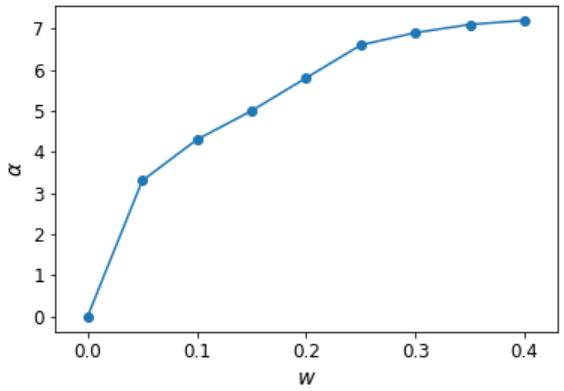
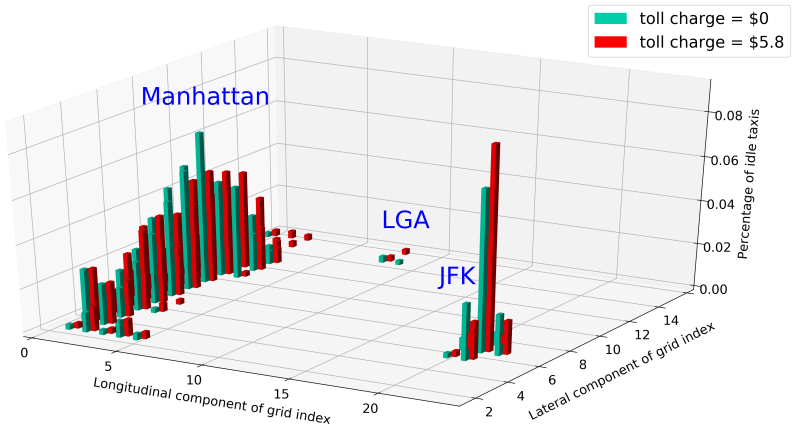


Figure 3.22: Sensitivity analysis

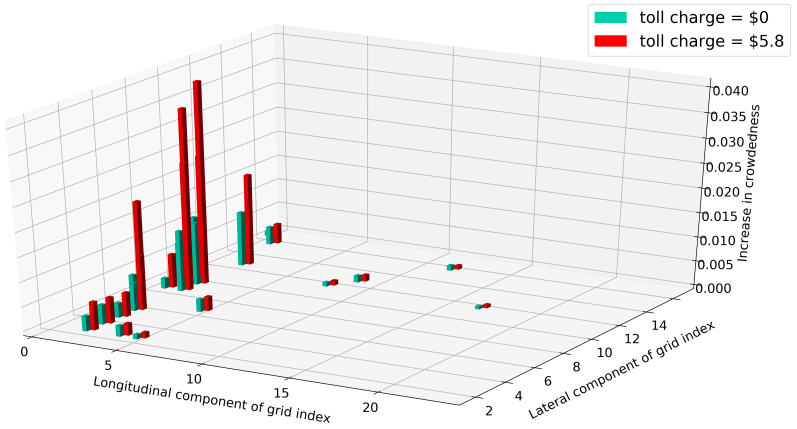
We now conduct sensitivity analysis on the weighting parameter w. Considering the magnitude of PTC and ICS (i.e., PTC around 0.2 to 0.3 and ICS around 0.01), a relatively large weighting parameter indicates that city planners mainly focus on PTC, because the impact of ICS on the objective (i.e., $-w \times PTC - (1-w) \times ICS$) is minor or even negligible. Therefore, the sensitivity analysis is performed on w within the range of $[0.0, 0.4]$. The result is presented in Figure (3.22). With $w = 0$, city planners put all weights on ICS, and the optimal toll charge is simply zero. As w increases, city planners gradually put more weights on PTC, resulting in an increase in the optimal

toll charge. With $w = \frac{1}{20}$, $-w \times PTC$ and $-(1-w) \times ICS$ are in the same magnitude of -0.01, and the optimal toll charge is \$3.3. With $w = \frac{1}{10}$ and $w = \frac{3}{20}$, optimal toll charges are \$4.3 and \$5.0, respectively. As plotted in Figure (3.21b), the magnitude of $-w \times PTC$ is already larger than that of $-(1-w) \times ICS$ with $w = \frac{1}{5}$. Increasing w further increases the discrepancy between the magnitude of $-w \times PTC$ and that of $-(1-w) \times ICS$. In other words, the magnitude of $-w \times PTC$ is much larger than that of $-(1-w) \times ICS$ when $w > \frac{1}{5}$, especially when $w > \frac{2}{5}$. In practice, the selection of the weight depends on how city planners' weighs in subway crowdedness and traffic congestion arising from taxis in CBD. Figure (3.22) also provides a guidance to planners if they also need to maintain a reasonable value of the optimal toll.

Figure (3.23a) presents the average percentage of idle 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 idle 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 (3.15), two airports become comparatively attractive. Figure (3.23b) 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 available 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.



(a) Percentage of idle taxis in each grid (grids with percentage lower than 0.002 are omitted)



(b) Increase in crowdedness across the busiest 20 grids

Figure 3.23: Spatial distribution of percentage of idle taxis in each grid and increase in crowdedness across the busiest 20 grids

3.6 Summary

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 an MAS. 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 8.4% using a simple piecewise linear platform service charge. In the case of multiclass taxi driver repositioning, a \$5.8 toll charge increases the objective of city planners by 22%, compared to that without any toll charge. With the optimal toll charge, the percentage of idle 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.

Chapter 4: Bilevel optimization for multi-agent route choice

4.1 Introduction

With a growing number of drivers relying on GoogleMaps or other navigation tools for dynamic routing, one question of interest is that, when everybody follows the advisory of shortest paths provided by one central platform, will these drivers still be able to gain from taking these paths? This research aims to tackle this question by accounting for competition among drivers. When one chooses a path dynamically while navigating a road network, others may also do so to compete for limited road resources. Therefore it is crucial to model the dynamic routing choices of many drivers simultaneously. To address the above problem, we will first review the literature on route choice of single-agent and then move to that of multi-agent.

4.1.1 Single-agent route choice

While traveling from an origin to a destination, one needs to select a sequence of road segments. Thus, en-route path choice is inherently a sequential decision-making process, in which at each junction one decides what the next link to take. Markov decision processes (MDPs) [44] and reinforcement learning (RL) [47] have become popular tools to model such a sequential process, in which travelers are rational agents to optimize a prescribed objective function (or reward) [112, 113]. The assumption of rationality has been challenged by various behavioral and economic theories, including, to name a few, bounded rationality [99, 100], prospect theory [114, 115, 116, 117], and regret theory [118]. In this chapter, we assume travelers are perfectly rational agents in the context of selfish routing. In other words, every traveler is a self-interested agent who aims to maximize individual accumulative rewards or minimize individual cost or maximize payoffs.

4.1.2 Multi-agent route choice

One's route choice contributes marginally to traffic congestion on its picked route. Accordingly, travelers on a road network interact among one another while selecting routes collectively. When everybody optimizes her own rewards while others do so simultaneously, a routing (or congestion) game forms. Multi-agent selfish routing games have been studied using both prescriptive (i.e., what one ought to behave in route choice) and descriptive (i.e., how one actually select routes) approaches. Prescriptive approaches can be categorized as **model-based** because of its high reliance on models that prescribe route choice behaviors, while descriptive ones are categorized as **data-driven** being accountable for real-world routing data.

The most popular prescriptive models belong to the traffic assignment problem. The traffic assignment problem models one's routing behavior while interacting with others in order to predict network-wide traffic congestion. Static traffic assignment is proposed for the long-term planning purpose, and dynamic traffic assignment (DTA) predicts dynamic traffic flow evolution in the short term by modeling one's within-day route choice or/and departure-time choice [119, 120, 121]. DTA integrates both notions of travel demand (consistent with static traffic assignment) and traffic flow (i.e., dynamic traffic evolution) and is thus appropriate for modeling dynamic movement of travelers across a network [122]. DTA typically consists of two components, namely a route choice (and/or departure time choice) model determining inflows to each link and a dynamic network loading model propagating traffic flows through the network [123, 124]. However, such a normative framework could be challenging to accommodate the substantive amount of trajectories of individual vehicles thanks to emerging technologies (e.g., GPS, mobile phones, and DSRC). It thus warranties a paradigm shift from model-based to data-based routing to leverage high-resolution trajectories, in order to better model one's en-route adaptive behavior in a dynamic and competitive environment.

This motivates us to believe that multi-agent reinforcement learning (MARL) [101, 125] could be a promising direction for dynamic routing games with a large amount of interacting travelers on networks. Different from DUE that evolves traffic dynamics based on mathematical models (with

which traffic state transitions and travel costs are computed), MARL based approaches simply let agents (i.e., drivers) interact with the road network and learn towards optimal policies, thus enjoying the flexibility of being model-free. Furthermore, thanks to the advancement of deep Q networks [97], deep reinforcement learning (DRL) based approaches can now tackle a large-sized problem with a very large or even continuous state and action space. In summary, MARL is advantageous over the classical model-based DTA framework for its computational efficiency with a large amount of agents on large-sized road networks, modeling of en-route adaptive behaviors of individual agents, and taking one's online travel experiences as input to dynamically update routing behaviors.

4.1.3 MARL and Multi-agent Markov game

As a game-theoretic framework for MARL, Markov game [101] is a generalization of MDPs to multiple interacting agents with competing goals, in which the environment makes transitions probabilistically in response to the agents' actions. Markov game is typically defined as a non-cooperative game where self-interested agents aiming to maximize their own payoffs. In a Markov game, the solution concept of Nash equilibrium is generally adopted [125]. At a Nash equilibrium, one agent's strategy (i.e., policy) is the best response to other agents' strategy. MARL is an efficient and versatile tool to solve for the optimal policy of each agent. Among the pioneering studies, Littman [101] proposes a minimax Q-learning algorithm to solve for the optimal policies of two agents with opposed goals in a zero-sum game, and Hu and Wellman [125] extends it to a general-sum game and provided a multi-agent Q-learning algorithm.

Recent literature has witnessed various applications of MARL to high-dimensional and complicated tasks such as playing the game of Go [72, 73], Poker [74, 75], Dota 2 [76], and StarCraft II [77]. Outside the computer game domain, MARL has also attracted significant attention and been used in energy sharing [126], federated control [127], and sequential social dilemma [128], to name a few. Interested readers are referred to [129] for a comprehensive review of applications and challenges of MARL. In the transportation domain, we have also seen a growing trend of

applying MARL to vehicle routing [62, 130], traffic signal control [131, 132], fleet management [1, 133], order dispatching [65], and autonomous driving [134, 135]. However, its application to dynamic routing game is still at its nascent stage.

4.1.4 Literature on reinforcement learning based route choices

The existing literature on dynamic routing games using RL-based approaches is listed in Table (4.1). Based on how to define the action of an agent, we broadly categorize RL-based route choice models into two groups. In the first group, the action of an agent is a route [136, 137, 138]. For example, in [136, 137], for an agent aiming to go to her destination node from her origin node, the action space for her is the k shortest paths from her origin to her destination. These studies assume that every driver does follow her chosen route until she reaches her destination. In reality, however, drivers could deviate from their chosen route when they realize that traffic condition on some alternative routes might be better, especially when they get stuck in traffic on the current route. To capture this en-route adaptive behavior of drivers, the second group of studies focus on en-route traffic assignment, or route choice from the perspective of a driver. In these studies, the action space of an agent currently at a node is the outbound links from the node [62, 139, 140]. In other words, every agent needs to decide which outbound link to choose whenever the agent arrives at a node, until the agent reaches some terminal node. This en-route decision-making process captures the adaptive behavior of real drivers.

To study the en-route decision-making process of drivers, both single-agent RL and multi-agent RL are used in the literature. A single-agent Q-learning approach is developed in [140] with the use of some global information such as congestion status in the definition of state. The authors bootstrapped two traffic profiles from the PeMS dataset and derived an optimal policy for the agent. Unfortunately, single-agent RL fails to capture the competition among adaptive agents. Multi-agent RL is recently used to tackle the multi-driver route choice task [62, 139]. Despite of modeling multi-driver interactions, these studies use independent multi-agent tabular Q-learning where every agent is treated as an independent learner who has no information of other agents.

Table 4.1: Existing research on dynamic routing using RL based approaches

Research type	Reference	Setting	State	Action set	Reward	Algorithm	Bilevel optimization	Gap
Route-based	[136]	Multi-agent	-	k shortest routes from origin to destination	Negative travel time	Independent tabular Q-learning	No	-
	[137]	Multi-agent	-	k shortest routes from origin to destination	App-based regret (anticipated disutility)	Independent tabular Q-learning	No	-
	[138]	Multi-agent	-	Feasible routes from origin to destination	Negative travel time	Bush-Mosteller RL scheme	No	-
En-route	[139]	Multi-agent	(node)	Outbound links	Negative travel time (Individual and systematic)	Independent tabular Q-learning	No	Independent learning suffers from non-stationary and non-Markovian issues; tabular Q-learning could not handle continuous or large state/action space
	[62]	Multi-agent	(node)	Outbound links	Negative travel time	Independent tabular Q-learning	No	
	[140]	Single-agent	(time, node, congestion status)	Successor nodes	Negative travel time	Tabular Q-learning	No	Using global information, i.e., congestion status in both training and execution
	This study	Multi-agent	(time, node)	Outbound links	Negative travel cost	Flow-dependent deep Q-learning	Yes	

A key issue arises when independent learners treat other agents as part of the stochastic environment, that is, theoretical convergence guarantee for Q-learning does not hold, because the environment is no longer Markovian and stationary [80, 81]. To fill this research gap, we develop a flow-dependent multi-agent deep Q-learning approach which captures the interaction among adaptive agents via a flow-dependent mean action. The flow-dependent mean action is defined as the traffic flow on the chosen link right after an agent enters the link. The use of the flow-dependent mean action not only captures the competition among agents, but also enables Q-value sharing and policy sharing, which is computationally favorable, especially in a large-sized problem with multi-commodity multi-class agents.

Noticing that in a road network, in addition to the infrastructure supply (i.e., road resources) and travelers (i.e., traffic demand), there is another player, that is, city planner, who can impact the behavior of travelers.With selfish and rational travelers aiming to minimize their travel cost, city planners can achieve some systematic goal by imposing externalities on individuals' cost via various operational measures such as congestion pricing or traffic signal control. Therefore, we formulate such interactions between travelers and city planners as a bilevel optimization task. We employ the developed flow-dependent multi-agent deep Q learning approach on the lower level to solve for optimal route choices of travelers and a Bayesian optimization scheme on the upper level to solve for optimal controls by city planners.

4.1.5 Contributions of this chapter

Here we will highlight the main contributions of this chapter as follows. First, a flow-dependent mean field deep Q-learning algorithm is developed to tackle the route choice task of multi-commodity multi-class agents. The flow-dependent mean action not only partially captures the competition among agents but also enables Q-value sharing and policy sharing. Second, we demonstrate the linkage between the classical DUE paradigm and our proposed MARL paradigm. This research is the first-of-its-kind to unify the model-based (i.e., DUE) and data-driven (i.e., MARL) paradigms for dynamic routing games. Third, we formulate the overall task including travelers and city plan-

ners as a bilevel optimization task. We demonstrate the effect of two administrative measures, namely tolling and signal control, on the behavior of travelers and show that the systematic objective of city planners can be optimized by a proper control.

The remainder of the chapter is organized as follows. Section (4.2) introduces the developed flow-dependent mean field multi-agent deep Q-learning algorithm. We detail the necessity and benefits of using the flow-dependent mean action which carries not full but partial information of nearby agents. Section (4.3) demonstrates the linkage between the classical DUE paradigm and our proposed MARL paradigm. We illustrate that actually DUE is a special case of multi-agent Markov games with a perfect information structure and a deterministic environment described by mathematical models. Section (4.4) introduces the bilevel optimization model where the upper level city planners can impact the behavior of lower level travelers through operational measures. A case study of the application of the bilevel optimization model to the Braess network with tolling is presented. Section (4.5) presents the application of the bilevel optimization model to solve for optimal traffic signal control over a real-world large-scale road network near Columbia University's campus in the City of New York. Section (4.6) summarizes this chapter.

4.2 Mean field multi-agent reinforcement learning

In this section, we introduce a flow-dependent mean field multi-agent deep Q-learning approach to tackle the multi-driver route choice task.

4.2.1 Single-agent reinforcement learning

As a stepping stone, we first introduce a single-agent RL approach in the context of route choice. Within the single-agent scope, there is only one agent interacting with the stochastic environment. The goal of the deep Q-learning approach is to derive an optimal policy so that the agent could get the maximum expected cumulative reward by following the policy in the environment.

To be specific to the route choice task, we introduce the single-agent deep Q learning approach on a Braess network, as presented in Figure (4.1). In the Braess network, there are four nodes,

namely $\{n_0, n_1, n_2, n_3\}$. n_0 is the origin node and n_3 the destination/terminal node. There are five directed links connecting these nodes. We denote the link connecting n_i and n_j as l_{ij}. Link travel time on l_{ij} is denoted by Δt_{ij}. In this study, $\Delta t_{01} = 45$, $\Delta_{23} = 45$, $\Delta t_{02} = k_{02} \cdot x$, $\Delta t_{13} = k_{13} \cdot x$, and $\Delta t_{21} = \alpha$, where x denotes the travel flow (i.e., number of vehicles) on the link, k_{02} and k_{13} are two multipliers, and α is a control parameter.

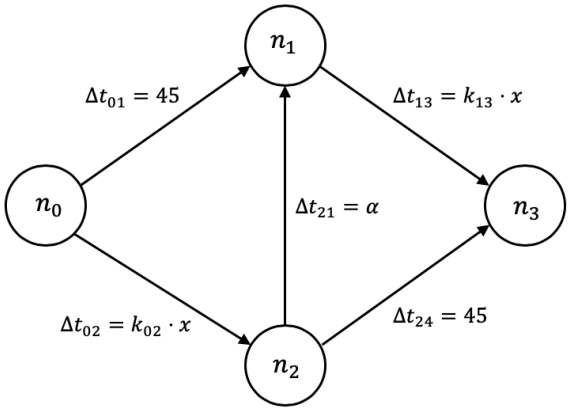


Figure 4.1: Braess network

First and foremost, the interaction between the agent and the environment is typically characterized by a Markov decision process or an MDP for short [44]. An MDP is specified by a tuple (S, A, R, P, γ), where S denotes the state space, A the action space, R the reward function, P the state transition probability matrix, and $\gamma \in [0, 1]$ the discount factor. An MDP goes as follows. At any state $s \in S$ but some terminal state, the agent chooses action $a \in A$ and executes the action in the environment. It observes a state transition from s to a new state $s' \in S$ with probability $P(s'|s, a)$ and receives a reward $r(s, a, s') \in R$. In the context of route choice,

- $s \in S$. State s consists of two components, namely node and time, i.e., $s = (n, t)$.
- $a \in A$. Action a for the agent currently in state $s = (n, t)$ is simply one of the outbound links from node n. For example, $a = l_{n,n'}$, where $l_{n,n'}$ is the link connecting node n and node n'. In this study, we assume action is deterministic, meaning that the agent will enter the chosen link. Note that the number of outbound links from a node varies from node to node in the

network, indicating that the number of allowable actions is dependent on the node where the agent is currently located.

- P. After taking action a in state s, the agent arrives at a new state $s' = (n', t')$ with probability $P(s'|s, a)$, where n' is the end node of the chosen link (i.e., $l_{n,n'}$), and t' is the time right after the state transition. P is typically unknown to the agent. Therefore, the agent needs to repeatedly interact with the environment to gain state transition experiences, i.e., (s, a, s').

- $r \in R$. In addition to the observed state transition $s \rightarrow s'$, the agent receives a reward $r_{t'}(s, a, s')$ after executing action a, where the subscript t' explicitly denotes that the reward is received by the agent at time t'. In the context of route choice, the reward function $r_{t'}(s, a, s')$ is typically chosen as some negative travel cost related to the state transition $s \rightarrow s'$. For example, $r_{t'}(s, a, s') = -(t' - t)$, i.e., negative travel time, and $r_{t'}(s, a, s') = -\text{dist}(n, n')$, where $\text{dist}(n, n')$ measures the distance between node n and n', i.e., negative travel distance.

- γ. The discount factor γ is used to discount the future reward. When $\gamma = 1$, the agent does not differentiate future rewards from immediate rewards. As γ get smaller, the agent cares less about rewards received in the distant future, therefore her decision-making gets more myopic. In this study, we take $\gamma = 1$ because usually drivers aim to minimize their cumulative travel cost for a trip. In other words, they value a future travel cost and an immediate cost similarly.

- $\rho = \sum_{t=0}^{T} \gamma^t r_t$. ρ is the discounted cumulative reward. The agent aims to maximize ρ by deriving an optimal policy.

The agent uses the collected experience tuples, i.e., (s, a, s', r), to derive an optimal policy. A policy μ is a mapping from state s to action a, i.e., $\mu(s) = a$. Following policy μ, value function $V^\mu(s)$ is defined as the expected discounted cumulative reward from state s. Mathematically, it is given in a recursive form, namely $V^\mu(s) = \mathbb{E}_{a\sim\mu(s),s'\sim P(\cdot|s,a)}[r(s, a, s') + \gamma V^\mu(s')]$. The optimal policy, denoted as μ^*, is the policy that maximizes the value function, i.e., $\mu^* = \arg\max_\mu V^\mu(s),\ \forall s \in S$. The optimal value function, denoted as $V(s)$, is given as [47]

$$V(s) = \max_a \mathbb{E}_{s' \sim P(\cdot|s,a)}[r(s, a, s') + \gamma V(s')]. \tag{4.1}$$

While the value function $V(s)$ captures the optimal expected cumulative reward that an agent can earn from state s, the state-action value $Q(s, a)$ forces the agent to take action a and therefore captures the optimal expected cumulative reward from state s and action a. Mathematically, $V(s) = \max_a Q(s, a)$. Substituting the relation between V and Q into Equation (4.1) yields

$$Q(s, a) = \mathbb{E}_{s' \sim P(\cdot|s,a)}[r(s, a, s') + \gamma \max_{a'} Q(s', a')]. \tag{4.2}$$

With the optimal action-value function Q, optimal policy can be explicitly derived as $\mu^*(s) = \arg\max_a Q(s, a)$.

With the collected experience tuple (s, a, s', r), the agent can update the Q value by the widely used tabular Q-learning algorithm

$$Q(s, a) \leftarrow Q(s, a) + \eta[r + \gamma \max_{a'} Q(s', a') - Q(s, a)], \tag{4.3}$$

where $\eta \in (0, 1]$ is the learning rate. Convergence is guaranteed for the Q-learning algorithm when the learning rate is decayed properly over time [47]. Unfortunately, the tabular Q-learning algorithm in Equation (4.3) is only applicable to finite and discrete state and action spaces. Tabular Q-learning maintains a Q-table for all possible combinations of state and action. It is thus not feasible to be used in a problem with large or continuous state and action spaces. Thanks to the emergence of deep Q networks (DQNs) [97], we could resort to neural networks as a functional approximator of the Q function. Denoting the functional approximator parameterized by θ as Q_θ, DQN updates its parameter θ by minimizing the following loss

$$\mathcal{L}(\theta) = \mathbb{E}_{s,a,s'}\left[\left(r(s, a, s') + \gamma \max_{a'} Q_{\theta^-}(s', a') - Q_\theta(s, a)\right)^2\right], \tag{4.4}$$

where Q_{θ^-} is called a target network copied from Q_θ every τ episodes to ensure training stability,

where τ is a hyperparameter.

Example 4.2.1. (Single-agent route choice).

To be more concrete, we demonstrate the aforementioned notations in the Braess network. Supposing an agent is initially placed at n_0 in the Braess network, the initial state of the agent is $s = (n_0, 0)$. There are two allowable actions for the agent, namely l_{01} and l_{02}. Assuming the agent chooses action l_{02} in the initial state, the agent arrives at $s' = (n_2, 1)$ with probability $P(s' = (n_2, 1)|s = (n_), 0), a = l_{02}) = 100\%$. The time component in s' is 1 because the link travel time $\Delta t_{02} = x = 1$ (i.e., there is one agent on this link). The state transition probability is 100% because both the action and the state transition are deterministic in this case. Assuming the reward function is the negative travel time, the agent receives a reward of $-\Delta t_{02} = -1$ along with the state transition $s \to s'$.

We now apply the single-agent Q-learning to the Braess network with one agent initially at n_0. For the purpose of demonstration, we assume $\alpha = 1$ and $k_{02} = k_{13} = 40$ in this example. Due to the simplicity of this case, it could be solved analytically. Noticing the static nature of this example, we neglect the time component in state. n_3 is the terminal node, meaning that $V(n_3) = 0$. At node n_1, the only allowable action is l_{13} leading the agent to terminal node n_3, yielding $Q(n_1, l_{13}) = -\Delta t_{13} + \gamma V(n_3) = -40$ and $V(n_1) = -40$. At node n_2, there are two actions, namely l_{21} and l_{24}. Action l_{21} leads the agent to n_1, yielding $Q(n_2, l_{21}) = -\Delta t_{21} + \gamma V(n_1) = -(\alpha + 40) = -41$. Action l_{24} leads the agent to the terminal node n_3, yielding $Q(n_2, l_{24}) = -\Delta t_{24} + \gamma V(n_3) = -45$. Therefore, $V(n_2) = \max(Q(n_2, l_{21}), Q(n_2, l_{24})) = -41$ and $\mu^*(n_2) = l_{21}$. At node n_0, there are two actions, namely l_{01} and l_{02}. Action l_{01} leads the agent to n_1, yielding $Q(n_0, l_{01}) = -\Delta t_{01} + \gamma V(n_1) = -85$. Similarly, $Q(n_0, l_{02}) = -\Delta t_{02} + \gamma V(n_2) = -81$. Therefore, $V(n_0) = \max(Q(n_0, l_{01}), Q(n_0, l_{02})) = -81$ and $\mu^*(n_0) = l_{02}$. Following optimal policy μ^*, the agent takes route $n_0 \to n_2 \to n_1 \to n_3$, which is the shortest path in this example.

4.2.2 Multi-agent reinforcement learning

Although the single-agent Q learning efficiently finds the shortest path, it fails to capture the competition among agents in a multi-agent system (MAS). In a single-agent RL problem, only one agent is placed in the environment to learn the optimal policy, which maximizes the expected cumulative reward of the agent. Unfortunately, when multiple agents follow this optimal policy, their expected cumulative reward might be low. For example, if there are 50 agents at node n_0 initially in the Braess network, their expected cumulative reward is -110 by following the optimal policy (i.e., the shortest path $n_0 \rightarrow n_2 \rightarrow n_1 \rightarrow n_3$). This reward (i.e., -110) is lower than that of following another route (e.g., $n_0 \rightarrow n_1 \rightarrow n_3$ yields an expected cumulative reward of -95). Therefore, to tackle a multi-driver route choice task, we develop a multi-agent deep Q-learning approach in this section to capture the competition among agents.

4.2.2.1 Problem formulation

Similar to the previous single-agent case, we introduce the multi-agent RL approach on the Braess network shown in Figure (4.1).

Due to the existence of multiple agents, the multi-agent RL problem is formulated as a Markov game [101], which is a generalization of Markov decision processes to multiple interacting agents with competing goals, in which the environment makes transitions probabilistically in response to the agents' actions. We denote the Markov game by a tuple $(N, S, O, A, P, R, \gamma)$, where N, S, O, A, P, R, γ are the number of agents, environmental state space, joint private observation space, joint action space, state transition probability functions, reward functions, and the discount factor. In the context of route choice,

- N. There are N controllable adaptive agents, denoted by $\{1, 2, \cdots, N\}$. Note that in a traffic network, there are also other traffic participants who are not controlled by the learning algorithm developed in this study. We regard those uncontrollable traffic participants as background traffic. The background traffic affects the behavior of controllable agents by its

impact on link travel cost. For example, if the background traffic has caused a traffic jam on a link, it is very likely that controllable agents will avoid this jammed link.

- $\mathbf{s} \in S$. Environmental state $\mathbf{s}$ consists of some global information such as distribution of agents and traffic condition on each link. $\mathbf{s}$ is not fully accessible to agents.

- $\mathbf{o} \in O$. Although the environmental state $\mathbf{s}$ is not observable by agent i, the agent is able to draw a private observation $o_i \in O_i$, which is correlated with $\mathbf{s}$. The Cartesian product of private observation spaces of all agents forms the joint observation space, i.e., $O = O_1 \times O_2 \times \cdots \times O_N$. In this research, o_i consists of two components, namely node and time, i.e., $o_i = (n, t)$. Joint observation $\mathbf{o} = (o_1, o_2, \cdots, o_N)$.

- $\mathbf{a} \in A$. Based on $o_i = (n, t)$, the allowable action set for agent $i \in \{1, 2, \cdots, N\}$ consists of all outbound links from node n. For example, $a_i = l_{n,n'}$, where $l_{n,n'}$ is the link connecting n and n'. Joint action $\mathbf{a} = (a_1, a_2, \cdots, a_N)$.

- P. Joint action $\mathbf{a}$ triggers a state transition $\mathbf{s} \rightarrow \mathbf{s}'$ with probability $P(\mathbf{s}'|\mathbf{s}, \mathbf{a})$. Agent $i \in \{1, 2, , \cdots, N\}$ draws a new private observations, namely $o_i' = (n', t')$, where n' is the end node of the chosen link $l_{n,n'}$ and t' is the time when agent i arrives at n'. The private observation o_i' is correlated with $\mathbf{s}'$.

- $r \in R$. In addition to o_i', agent $i \in \{1, 2, \cdots, N\}$ receives a reward $r_{i,t'}(\mathbf{s}, \mathbf{a}, \mathbf{s}')$, where the subscript i, t' explicitly denotes that the reward is received by agent i at time t'. Similar to the single-agent RL, reward is typically some negative travel cost such as travel time and travel distance in the context of route choice.

- γ. Similar to the single-agent RL, $\gamma = 1$.

- $\rho_i = \sum_{t=0}^{T} \gamma^t r_{i,t}$. ρ_i is the discounted cumulative reward received by agent i. If the goal of agent i is to maximize ρ_i, agents are non-cooperative and selfish; if the goal of agent i is to maximize the average discounted cumulative rewards of all agents, i.e., $\frac{1}{N} \sum_{i=1}^{N} \rho_i$, agents are cooperative.

Due to the coexistence of other agents, Q function of agent i, i.e., Q_i is now a function of environmental state $\mathbf{s}$ and joint action $\mathbf{a}$, namely

$$Q_i = Q_i(\mathbf{s}, \mathbf{a}). \tag{4.5}$$

Similarly, value function of agent i, i.e., V_i is a function of environmental state $\mathbf{s}$, namely $V_i = V_i(\mathbf{s})$. Note that each agent i has her own Q function, i.e., Q_i, which may be different from Q functions of other agents, and therefore has an optimal policy μ_i^* different from policies of other agents. This distinguishes multi-agent RL from its single-agent counterpart, because agents may behave differently even in the same state with multi-agent RL while they choose the same action in the same state with single-agent RL.

Considering that environmental state $\mathbf{s}$ is not fully observable by agents, Q function shown in Equation (4.5) is not tractable from the perspective of agent i. Actually, it is private observation o_i based on which agent i chooses next action. We therefore rewrite Q function for agent i as

$$Q_i = Q_i(o_i, o_{-i}, a_i, a_{-i}), \tag{4.6}$$

where o_{-i} and a_{-i} denote the joint observation and joint action of all agents except agent i, respectively. In a non-coordinated environment (i.e., agents are not sharing their private observations or actions), o_i and a_i are but o_{-i} and a_{-i} are not accessible to agent i. Unfortunately, removing the dependency of Q_i on a_{-i} or o_{-i} may introduce large instability in Q-learning. In addition, convergence guarantee in single-agent Q-learning is no longer valid due to the adaptive behavior of other agents if o_{-i} and a_{-i} are not included [80].

4.2.2.2 Multi-commodity multi-class agents

For a real-world multi-agent problem, there are commonly more than tens of thousands of agents. Maintaining one Q function for each agent is thus computationally infeasible. Although heterogeneity does exist among agents [70], some agents do share some homogeneous aspects

(e.g., utility function). We thus aim to reduce the number of Q functions by leveraging this homogeneity.

In the context of route choice, we focus on four aspects, namely the observation space, the action space, the reward function, and the destination. First, considering that all agents are intelligent and have the same visibility, agents share the same observation space. In other words, agents have the same observation when they arrive at the same node at the same time. Second, considering that all agents have the same freedom of choosing actions, agents share the same action space. In other words, the allowable action set is the same for agents arriving at the same node. Third, agents may have different reward functions. For example, some agents may aim to minimize their travel time while other agents may focus more on the travel distance, which is related to the fuel consumption. Therefore, agents are *multi-class*. Finally, agents may have different destinations. The reason for considering the destination as a concerned aspect is as follows. The Q function in Equation (4.6) essentially records the negative optimal cumulative travel cost starting from observation o_i to the destination of agent i, conditioning on action a_i, joint observation o_{-i}, and joint action a_{-i}. Therefore, agents with different destinations have different Q functions. Agents are thus said to be *multi-commodity*. In summary, we consider *multi-commodity multi-class* agents with the same observation space and the same action space in this study.

Based on the reward function and the destination of agents, we broadly categorize agents into C groups. Within each group, agents share the same state space, action space, reward function, and destination. Therefore, agents are homogeneous within each group. The multi-agent problem is then largely simplified by sharing the same Q function among agents within the same group. We denote the shared Q function for group $c \in \{1, 2, , \cdots, C\}$ by

$$Q^c(o_i, o_{-i}, a_i, a_{-i}). \tag{4.7}$$

Note that although Q^c is shared among agents in group c, each agent has her own optimal policy $\mu_i^*(o_i) = \arg\max_{a_i} \mathbb{E}_{o_{-i} \sim D_{-i}, a_{-i} \sim \mu_{-i}^*}[Q^c(o_i, o_{-i}, a_i, a_{-i})]$, $\forall o_i \in O_i$, where D_{-i} denotes the dis-

tribution of o_{-i} and μ^*_{-i} is the joint optimal policy of all agents except agent i. For example, assuming agents i and j intend to choose the same action, i.e., $a_i = a_j$, at the same observation, i.e, $o_i = o_j$, their expected Q values might be different, i.e., $\mathbb{E}_{o_{-i}\sim D_{-i}, a_{-i}\sim\mu^*_{-i}}[Q^c(o_i, o_{-i}, a_i, a_{-i})] \neq \mathbb{E}_{o_{-j}\sim D_{-j}, a_{-j}\sim\mu^*_{-j}}[Q^c(o_j, o_{-j}, a_j, a_{-j})]$. In other words, agents have different expected Q values with respect to o_{-i} and a_{-i} even for the same observation and action pair.

Remark. When agents have different departure times, there are typically two ways to tackle the route choice task. The first is to have a shared Q function among all agents who have the same reward function and the same destination, regardless of their departure times. The second is to have a shared Q function among agents who have the same reward function, the same destination, and the same departure time. There is some tradeoff between these two approaches. The former maintains fewer neural networks and have more experience tuples to share among agents. The latter may have better performance but requires more neural networks and more training episodes to get enough experience tuples for training. As an approach in between, one could categorize departure times into several groups and allow Q function sharing in each group.

4.2.2.3 Flow dependent Q function

Although the number of Q functions to be maintained has been largely reduced, Q^c in Equation (4.7) easily becomes intractable when the number of agents in the environment gets large. Specifically, the joint observation space and the joint action space easily become prohibitively large as the number of agents increases. For example, the joint action space for 1,000 agents is $1,000 \times d$ dimensional if each action is a d dimensional vector.

To address the above issue, we first notice that joint observation o_{-i} and joint action a_{-i} record full information of other agents, which may contain redundant information from the perspective of agent i. Actually, in the task of route choice, observations and actions of agents who are currently far away from agent i have a very limited influence on agent i. It is nearby agents who exert an impact on the traffic condition around agent i. For example, after agent i chooses a link at a node, an immediate influential quantity is the traffic flow on the chosen link. If the flow is high, agent

i may experience a high travel cost on the link; if the flow is low, agent i may have a smooth transition to the end of the link. Therefore, high-dimensional o_{-i} and a_{-i} in Equation (4.7) could be approximated by some low-dimensional aggregate information of nearby agents. Similar to the mean field approximation in [85], we call this aggregate information as the mean action of nearby agents and denote it by $\bar{a}_i$. Q function in Equation (4.7) thus becomes

$$Q^c(o_i, o_{-i}, a_i, a_{-i}) \approx Q^c(o_i, a_i, \bar{a}_i). \tag{4.8}$$

We now formally define the mean action in the context of route choice.

Definition 4.2.1. (*Mean action*). Mean action $\bar{a}_i$ is defined as the traffic flow on the link that is chosen by agent i. Note that the traffic flow is calculated right after agent i enters the link.

With the above definition of the mean action, we interpret the right hand side of Equation (4.8), i.e., $Q^c(o_i, a_i, \bar{a}_i)$, as the flow-dependent Q function.

Definition 4.2.2. (*Flow-dependent Q function*). With agent i choosing action a_i at observation o_i, the explicit inclusion of mean action $\bar{a}_i$, i.e., the traffic flow on the chosen link, makes $Q^c(o_i, a_i, \bar{a}_i)$ a flow-dependent Q function.

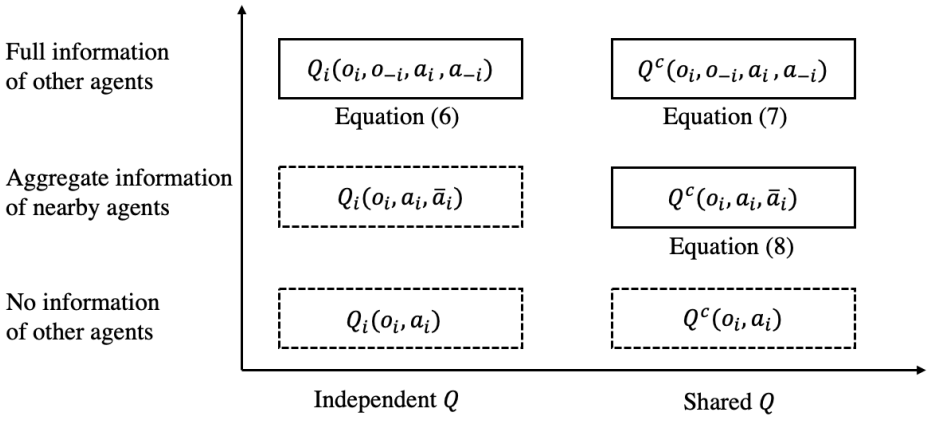


Figure 4.2: Q functions

We summarize the aforementioned Q functions in Figure (4.2). The horizontal axis indicates whether the Q function is independent (i.e., Q_i for agent i) or shared (i.e., homogeneous agents

in group c share Q^c). The vertical axis denotes the type of information that is used by the Q function. There are three types of information, namely full information of other agents, aggregate information of nearby agents, and no information of other agents. Consequently, there are six kinds of Q functions, as shown in Figure (4.2). Although Q functions in the dashed rectangle are not previously discussed, we present them in the figure for the purpose of completeness. Here we stress the advantage of the flow-dependent Q function, i.e., $Q^c(o_i, a_i, \bar{a}_i)$, in an MAS with more than tens of thousands of agents. As aforementioned, independent Q functions result in maintaining tens of thousands of different Q functions and are thus computationally infeasible. Therefore, we resort to shared Q functions by considering the heterogeneity and homogeneity among agents. To be precise, now we present the advantage of the flow-dependent Q function over other two shared Q functions from two perspectives, namely a top-down perspective and a bottom-up perspective.

- The top-down perspective. $Q^c(o_i, o_{-i}, a_i, a_{-i})$ uses full information of other agents ("top" view). Unfortunately, it is not tractable due to the prohibitively large dimension of the joint observation o_{-i} and joint action a_{-i}. Therefore, noticing the fact that the major impact on the traffic condition surrounding agent i comes from nearby agents, we approximate $Q^c(o_i, o_{-i}, a_i, a_{-i})$ by $Q^c(o_i, a_i, \bar{a}_i)$.

- The bottom-up perspective. $Q^c(o_i, a_i)$ uses no information of other agents ("bottom" view). The optimal policy for agent i in group c is $\mu_i^*(o_i) = \text{argmax}_{a_i} Q^c(o_i, a_i)$, $\forall o_i \in O_i$. Consequently, all agents in group c share the same deterministic optimal policy. This is single-agent RL instead of multi-agent RL, and we have illustrated the problem of applying single-agent RL to a multi-agent problem at the beginning of Section (4.2.2). To derive different policies for agents in the same group, we augment the input to the Q function by including $\bar{a}_i$, resulting in the flow-dependent Q function. With $Q^c(o_i, a_i, \bar{a}_i)$, the optimal policy for agent i in group c is $\mu_i^*(o_i) = \text{argmax}_{a_i} \mathbb{E}_{\bar{a}_i \sim \mu_{-i}^*}[Q^c(o_i, a_i, \bar{a}_i)]$. Agent i and agent j in group c could have different policies because $\mathbb{E}_{\bar{a}_i \sim \mu_{-i}^*}[Q^c(o_i, a_i, \bar{a}_i)]$ may be different from $\mathbb{E}_{\bar{a}_j \sim \mu_{-j}^*}[Q^c(o_j, a_j, \bar{a}_j)]$ when $o_i = o_j$ and $a - i = a_j$. In other words, agents have different expected Q values with respect to $\bar{a}_i$ even for the same observation and action pair.

4.2.2.4 Mean field multi-agent deep Q-learning

Different from single-agent RL, agent i in group $c \in \{1, 2, \cdots, C\}$ in an MAS interacts with not only the environment but also other agents and collects experience tuples in the form of $(o_i, a_i, o_i', r_i, \bar{a}_i)$. Considering that mean action $\bar{a}_i$ is typically in a large discrete or even continuous space, a DQN parameterized by θ_c, denoted by $Q^c(o_i, a_i, \bar{a}_i|\theta_c)$, is used to approximate $Q^c(o_i, a_i, \bar{a}_i)$. Similar to Equation (4.4), DQN updates its parameter θ_c by minimizing the following loss

$$\mathcal{L}(\theta_c) = \mathbb{E}_{o_i, a_i, \bar{a}_i, o_i'} \left[(r_i + \gamma \max_{a_i'} \mathbb{E}_{\bar{a}_i' \sim \mu_{-i}^*} [Q^c(o_i', a_i', \bar{a}_i'|\theta_c^-)] - Q^c(o_i, a_i, \bar{a}_i|\theta_c))^2 \right], \tag{4.9}$$

where $Q^c(\cdot|\theta_c^-)$ is a target network copied from $Q^c(\cdot|\theta_c)$ every τ episodes to stabilize training. After updating Q function, optimal policy

$$\mu_i^*(o_i) = \arg\max_{a_i} \mathbb{E}_{\bar{a}_i \sim \mu_{-i}^*} [Q^c(o_i, a_i, \bar{a}_i|\theta_c)], \ \forall o_i \in O_i. \tag{4.10}$$

Remark. Although the mean action $\bar{a}_i$ may not be fully observable by agent i, we include it in the flow-dependent Q function, i.e., $Q^c(o_i, a_i, \bar{a}_i)$ during training. In execution, only the derived optimal policy μ_i^* which is computationally a dictionary mapping from o_i to a_i is used, without involving mean action $\bar{a}_i$. This is essentially the idea of the centralized training and decentralized execution paradigm where some global information is used in training but not in execution [84].

The mean field multi-agent deep Q-learning (MF-MA-DQL) algorithm is summarized in Algorithm (5). In an MAS, state transition probability functions are typically unknown due to the adaptive behavior of other agents. We therefore resort to a model-free RL approach where agents repeatedly interact with the stochastic environment. We first initialize a centralized deep Q network, parameterized by θ_c, and a target Q network, parameterized by θ_c^- for each group $c \in \{1, 2, \cdots, C\}$. Although Q function is shared among agents in the same group, each agent i in group c has her own optimal policy μ_i^*, which is a dictionary mapping from observation o_i to action a_i. From initial

Algorithm 5 Mean field multi-agent deep Q-learning (MF-MA-DQL)

Input: exploration parameter $\epsilon = \epsilon_0$, learning rate $\eta = \eta_0$, target network update period τ
Initialize one DQN $Q^c(o, a, \bar{a}|\theta)$, parameterized by θ_c, and one target network $Q^c(o, a, \bar{a}|\theta_c^-)$ for each group $c \in \{1, 2, \cdots, C\}$
Initialize N dictionaries to store the optimal policy for agents, i.e., $\mu_i^*, \forall i \in \{1, 2, \cdots, N\}$
Initialize one experience replay buffer B_c for each group $c \in \{1, 2, \cdots, C\}$
Set $episode = 0$
repeat
    From the initial environmental state $\mathbf{s}_0$, each agent i draws observation o_i
    Set $t = 0$
    repeat
        For each agent i, select action a_i according to the ϵ-greedy method, i.e., randomly select an allowable action with probability ϵ and greedily according to policy μ_i^* with probability $1-\epsilon$
        Execute joint action $\mathbf{a} = (a_1, a_2, \cdots, a_N)$ in the environment to trigger state transition $\mathbf{s} \rightarrow \mathbf{s}'$
        Each agent i draws new observation o_i', receives reward r_i, and observes mean action $\bar{a}_i$
        Store experience tuple $(o_i, a_i, o_i', r_i, \bar{a}_i)$ into replay buffer B_c if agent i belongs to group c
        $t \leftarrow t + 1$
    until $t = T$
    for $c = 1$ to C **do**
        for $j = 1$ to J_c **do**
            Sample a batch of size K_c from replay buffer B_c
            Update parameter θ_c of Q^c by minimizing the loss defined in Equation (4.9)
            Update optimal policy μ_i^* of agent i by Equation (4.10)
        end for
    end for
    $episode = episode + 1$
    Decrease the exploration parameter ϵ
    Decrease the learning rate η
    if $episode$ mod $\tau = 0$ **then**
        for $c = 1$ to C **do**
            Update parameter θ_c^- of the target network, i.e., $\theta_c^- \leftarrow \theta_c$
        end for
    end if
until the algorithm converges
Return optimal policies $\mu_i^*, \forall i \in \{1, 2, \cdots, N\}$

environmental state $\mathbf{s}_0$, each agent $i \in \{1, 2, , \cdots, N\}$ keeps drawing private observation o_i and taking action a_i according to the widely used ϵ-greedy method (i.e., agent i chooses action randomly with probability ϵ and greedily from optimal policy μ_i^* with probability $1 - \epsilon$), until reaching some

terminal state. Joint action $\mathbf{a} = (a_1, a_2, \cdots, a_N)$ is executed in the environment to trigger the state transition $\mathbf{s} \to \mathbf{s}'$. Each agent i draws new private observation o_i', receives reward r_i, and observes mean action $\bar{a}_i$. An experience replay buffer B_c for each group c is used to store experience tuple $(o_i, a_i, r_i, \bar{a}_i, o_i')$ generated by agent i in group c. Batch training is then used to update the centralized Q function by sampling a batch of size K_c from buffer B_c and minimizing the loss shown in Equation (4.9) over this batch. With the updated centralized Q function, each agent updates her optimal policy by Equation (4.10).

Remark. As a value based approach, the developed MF-MA-DQL algorithm does not suffer the problem of variable action set. A variable action set means that the number of allowable actions might vary from state to state. For example, there are two outbound links from node n_0 and one outbound link from node n_1 in the Braess network shown in Figure (4.1), causing the problem of variable action set. A policy based approach, e.g., the widely used actor-critic method, does not fit naturally well into a RL problem with variable action set. The main reason is that the policy neural network typically takes as input a state and outputs a probability distribution on all allowable actions (i.e., the action set). Consequently, a variable action set might result in some challenges to maintain a good structure of the policy network. Fortunately, value based approaches automatically handle the problem of variable action set, because a value based approach calculates the expected Q value of all possible actions at a state and selects the action with the highest Q value as the optimal policy.

Example 4.2.2. (Multi-agent route choice). Now we apply the MF-MA-DQL algorithm to a multi-agent route choice problem. In the Braess network shown in Figure (4.1), now two agents are initially placed at node n_0. The goal of both agents is to travel to the destination node n_3 as soon as possible. Consequently, these two agents are considered to be homogeneous, because they share the same reward function (i.e., the negative travel time) and go to the same destination. We first demonstrate the notations of Markov game in this example.

There are $N = 2$ agents in the Braess network. Environmental state $\mathbf{s}$ is not observable, Correlated with $\mathbf{s}$, the joint observation of agents $\mathbf{o} = ((n_0, 0), (n_0, 0))$. There are two allowable actions

for both agents, namely l_{01} and l_{02}. The joint action $\mathbf{a} = (a_1, a_2)$. $\mathbf{a}$ triggers state transition $\mathbf{s} \rightarrow \mathbf{s}'$ with probability $P(\mathbf{s}'|\mathbf{s}, \mathbf{a})$. Each agent $i \in \{1, 2\}$ then draws new private observation o_i', receives reward r_i, and observes mean action $\bar{a}_i$. To be precise, we illustrate o_i', r_i, and $\bar{a}_i$ under two joint actions.

- With $\mathbf{a} = (l_{01}, l_{02})$, $o_1' = (n_1, \Delta t_{01} = 45)$ and $o_2' = (n_2, \Delta t_{02} = k_{02})$. $r_1 = -\Delta t_{01} = -45$ and $r_2 = -\Delta t_{02} = -k_{02}$. $\bar{a}_1 = 1$ and $\bar{a}_2 = 1$.

- With $\mathbf{a} = (l_{02}, l_{02})$, $o_1' = (n_2, \Delta t_{02} = 2 \cdot k_{02})$ and $o_2' = (n_2, \Delta t_{02} = 2 \cdot k_{02})$. $r_1 = -\Delta t_{02} = -2 \cdot k_{02}$ and $r_2 = -\Delta t_{02} = -2 \cdot k_{02}$. $\bar{a}_1 = \bar{a}_2 = 2$.

Note that although agent 2 takes action l_{02} in both cases, the agent arrives at different observations, i.e., (n_2, k_{02}) and $(n_2, 2 \cdot k_{02})$ due to the changing behavior of agent 1 (i.e., taking action l_{01} in the first case and l_{02} in the second).

Due to the simplicity of this case, we now analyze optimal value at each node in the network. For the purpose of demonstration, we assume $\alpha = 10$ and $k_{02} = k_{13} = 40$ in this example. Similar to Example (4.2.1), we neglect the time component in private observation due to the static nature of this problem.

1. Noticing that node n_3 is the terminal node, $V(n_3) = 0$.

2. At node n_1, the only action that agents can take is l_{13}. After one or two agents, depending on whether both of them reach node n_1, taking action l_{13}, there are two scenarios: 1) $\bar{a} = 1$, i.e., traffic flow on link l_{13} is 1, meaning currently there is only one agent on link l_{13} and it takes the agent $\Delta t_{13} = k_{13} \cdot 1 = 40$ to reach node n_3, therefore $Q(o = n_1, a = l_{13}, \bar{a} = 1) = -40$; 2) $\bar{a} = 2$, i.e., traffic flow on link l_{13} is 2, meaning that there are two agents on link l_{13} and it takes the agent $\Delta t_{13} = k_{13} \cdot 2 = 80$ to reach node n_3, therefore $Q(o = n_1, a = l_{13}, \bar{a} = 2) = -80$. Note that in scenario 2), it could be either both agents choose action l_{13} at node n_1 simultaneously or one agent chooses action l_{13} first because she arrives at n_1 first and the other agent chooses action l_{13} later due to her later arrival at node n_1 (e.g., one agent chooses

route $n_0 \rightarrow n_1$ and arrives at n_1 at $t = 45$, and the other agent chooses route $n_0 \rightarrow n_2 \rightarrow n_1$ and arrives at n_1 at $t = k_{02} + \alpha = 50$). In the former case, both agents observe $\bar{a} = 2$ and spend time 80 on their way to n_3; in the latter case, the agent who arrives at n_1 first observes $\bar{a} = 1$ and spends time 40 on her way to n_3 and the other agent observes $\bar{a} = 2$ and spends time 80. With $Q(n_1, l_{13}, 1) = -40$ and $Q(n_1, l_{13}, 2) = -80$, optimal value at n_1, i.e., $V(n_1)$, is within the range of $[-40, -80]$ and depends on the distribution of $\bar{a}$.

3. At node n_2, there are two outbound links, namely l_{21} and l_{23}. If one or two agents at n_2 choose action l_{21}, $Q(n_2, l_{21}, \bar{a}) = -\alpha + \gamma \times V(n_1) = -10 + V(n_1)$, which is lower than -50, regardless of $\bar{a}$. If one or two agents at n_2 choose action l_{23}, $Q(n_2, l_{23}, \bar{a}) = -\Delta t_{23} = -45$. Therefore, action l_{21} is strictly dominated by action l_{23}, indicating that optimal policy at n_2 for both agents is to choose action l_{23} and $V(n_2) = -45$.

4. Finally, at node n_0, there are two possible actions, namely l_{01} and l_{02}, and both agents choose action simultaneously. There are 3 possible scenarios: 1) when both agents choose action l_{01}, $Q(n_0, l_{01}, \bar{a} = 2) = -\Delta t_{01} + \gamma V(n_1) = -45 + (-80) = -125$. Note that $V(n_1) = -80$ because both agents arrive at node n_1 and choose action l_13 at the same time. 2) when both agents choose action l_{02}, $Q(n_0, l_{02}, \bar{a} = 2) = -\Delta t_{02} + \gamma V(n_2) = -80 + (-45) = -125$. 3) when one agent chooses action l_{01} and the other chooses action l_{02}, $Q(n_0, l_{01}, \bar{a} = 1) = -\Delta t_{01} + \gamma V(n_1) = -45 + (-40) = -85$ and $Q(n_0, l_{02}, \bar{a} = 1) = -\Delta t_{02} + \gamma V(n_2) = -40 + (-45) = -85$. Note that $V(n_1) = -40$ because only one agent uses link l_{13}. Therefore, the optimal policy of one agent is to choose action l_{01} and that of the other agent is to choose action l_{02} at n_0.

In summary, optimal values of interest are listed in Table (4.2). Link l_{21} is not used by agents because it is strictly dominated by l_{23}. From n_0, the payoff for agents are $(-85, -85)$ if they choose actions differently and $(-125, -125)$ if they choose the same action (i.e., either l_{01} or l_{02}). Therefore, optimal policy of one agent is to choose action l_{01} and that of the other is to choose l_{02}, and this is a Nash equilibrium because no agent can get better payoff by a unilateral deviation.

Table 4.2: Optimal values of interest

observation o	action a	mean action $\bar{a}$	$Q(o, a, \bar{a})$	$V(o)$
n_1	l_{13}	1	-40	$V(n_1) = \mathbb{E}_{\bar{a}}[Q(n_1, l_{13}, \bar{a})]$ is within the range of $[-80, -40]$.
n_1	l_{13}	2	-80	
n_2	l_{21}	1 or 2	≤ -50	Action l_{23} strictly dominates l_{21}. $V(n_2) = Q(n_2, l_{23}, \bar{a}) = -45$.
n_2	l_{23}	1 or 2	-45	
n_0	l_{01}	1	-85	$V(n_0) = -85$ if agents choose actions differently at n_0 and $V(n_0) = -125$ if agents choose the same action.
n_0	l_{01}	2	-125	
n_0	l_{02}	1	-85	
n_0	l_{02}	2	-125	

4.3 Linkage between DTA and Markov games

Dynamic traffic assignment has been widely used to model route choice behavior of travelers. A general DTA problem typically consists of two parts, namely a dynamic network loading (DNL) module and a route choice model.

DNL. Given traffic demand and route choices, the DNL module propagates traffic flow dynamics and/or traffic congestion through the network. There are two components in the DNL module, namely a link model and a junction model. A link model is used to propagate inflow traffic demand to exit and is usually described by either a delay/latency function or an exit-flow function [121]. As for the former, both linear delay functions and nonlinear functions (e.g., the widely used BPR function) are used in the literature. As for the latter, there are a large body of literature proposing or using various exit-flow functions such as the M-N model [141, 142], the point-queue model [143, 144], and the cell transmission model [145, 146]. A junction model is used to determine the flow distribution at an intersection.

Route choice. Route choice models guide travelers to properly choose next go-to links right before they arrive at a node/intersection. According to the objective of travelers, there are typically two types of research problems, namely dynamic system optimum (DSO) and dynamic user equilibrium (DUE). While DSO aims to minimize total or average travel cost of all travelers [147], DUE describes the equilibrium state in which no individual traveler could decrease her travel cost

by unilaterally deviating from her current route choices [122, 148].

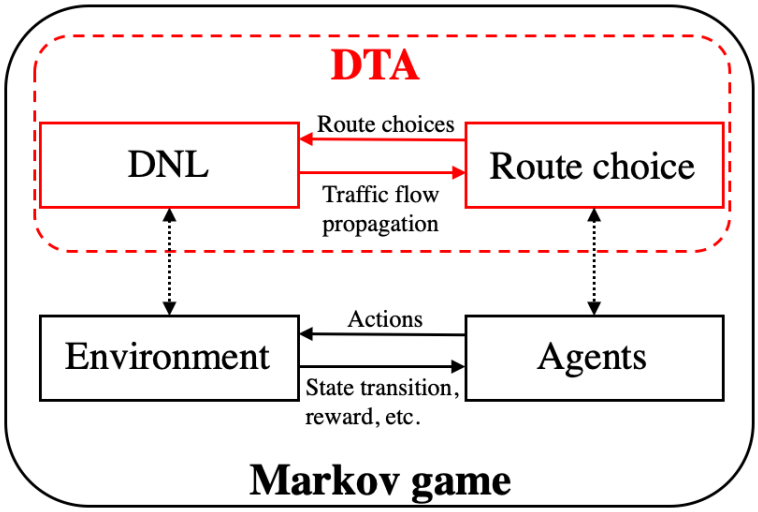


Figure 4.3: Connection between DTA and Markov game

With the aforementioned Markov game and DTA, we now discuss the connection between DTA and the corresponding Markov game (i.e., in the context of route choice). Recall that there are two components in a DTA, namely a DNL module and a route choice model, and two components in a Markov game, namely an environment and agents. The connection between DTA and Markov game is componentwise, as schematically presented in Figure (4.3). The DNL in DTA is regarded as the environment in Markov game, and the route choices in DTA are made by agents in Markov game. Further, the interaction between the DNL and route choice in DTA is similar to that between the environment and agents in Markov game. In DTA, a route choice model outputs route choices of travelers whenever they arrive at a node or an intersection, and the DNL module propagates traffic flow dynamics. In Markov game, agents (i.e., travelers) take actions (i.e., route choices) whenever they arrive at a node, and the environment experiences a state transition which is determined by the DNL module in DTA. To be precise, the actions taken by agents in Markov game are route choices in DTA, and the state transition, rewards, etc. in Markov game are determined by the DNL module in DTA.

In summary, DUE is essentially a special case of multi-agent Markov games with a perfect information structure and a deterministic environment described by mathematical models. Com-

pared to generic Markov games, the DTA paradigm might suffer from several shortcomings:

1. Computational efficiency of algorithms of solving a dynamic user equilibrium (DUE) might be highly compromised in large-sized road networks.

2. Stochasticity arising from traffic environments is challenging to model in the DTA paradigm. Such randomness comes from various sources, including demand side (such as random numbers of travelers) and supply side (such as weather, accidents, or events). The existing DTA approaches typically require an explicit functional form of the stochastic sources [149, 150, 151]. In addition, adaptive route choice behavior of other agents may contribute to the inherent stochasticity in traffic evolution, which has not been generally accounted for in the existing DTA paradigm.

3. A majority of DUE models cannot depict the adaptive behavior of travelers along a route, in other words, individuals' next-go-to link choice at a node based on cumulative travel costs to destinations. This is because either path-based choice representation is adopted that disables en-route choice at intermediate nodes [122, 148], or because future traffic conditions along a route are not accounted for even when next-go-to link selection is modeled [152]. Although we acknowledge that some specially design DUE methods might be able to capture en-route adaptive behavior of drivers, it could be hard to solve.

4. Rather than modeling individual agents' route choice behavior, DUE models are non-atomic games focused on aggregate traffic flow evolution, which renders them not flexible to accommodate heterogeneity in travelers. Such aggregation could also render these models fail to accommodate individual observations such as travel trajectories or mobile traces.

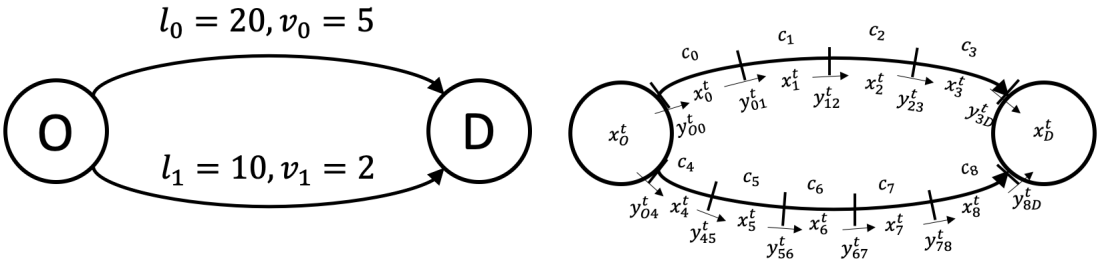
5. A fundamental issue in prescriptive models is their inability to adapt to real-time observations. In other words, its calibration and validation process is open-loop even when presented with real-world data (such as vehicle trajectories or aggregate flows). For instance, the common practice is to calibrate parameters of a DTA model offline and then run the calibrated

model for prediction. The calibration and prediction processes are thus separated. With the emergence of sensing and communication technologies, an explosive amount of vehicle driving trajectories and mobile traces can be obtained, which warranty adaptive models in which agents are capable of learning prevailing traffic environment while adjusting their en-route path choice dynamically.

Given these shortcomings, we will demonstrate how Markov games can solve dynamic routing games with imperfect information structure and a stochastic model-free environment. Before delving into it, we will first illustrate on numerical examples the equivalence between DUE or DSO and the equilibria computed from our developed MARL algorithm.

4.3.1 Numerical example

We first apply the developed MF-MA-DQL algorithm to a two-node two-link network, as presented in Figure (4.4a). Two nodes are denoted by O (origin) and D (destination), respectively. There are two links, namely link 0 with length $l_0 = 20$ and link 1 with length $l_1 = 10$. Further, we take free flow speed on link 0 as 5, i.e., $v_0 = 5$ and that on link 1 as 2, i.e., $v_1 = 2$. In other words, during light traffic, it takes a vehicle $l_0/v_0 = 4$ time steps to travel from node O to node D via link 0 and $l_1/v_1 = 5$ time steps via link 1. We denote the time-varying travel demand from node O to node D by $d(t)$.



(a) Original network (b) Cell discretized network

Figure 4.4: A two-node two-link network

After vehicle drivers making route choices, the traffic propagation (i.e., dynamic network loading) is described by the widely used cell transmission model (CTM) [145, 146]. The first thing

in a CTM is to discretize each link into several cells. The length of each cell is supposed to be the distance traveled by a vehicle in light traffic. Therefore, in the two-node two-link network, the length of a cell on a link is essentially the free flow velocity on this link. Consequently, there are four cells of length 5 on link 0 and five cells of length 2 on link 1, respectively. The resulting cell discretization is presented in Figure (4.4b). Each cell is denoted by c_i, where $i \in \{0, 1, \cdots, 8\}$. x_i^t records the number of vehicles in cell i at time t, and y_{ij}^t is the flow from cell i to cell j at time t. Noticing that the demand goes from node O to node D, there are x_O^t and x_D^t recording the number of vehicles on nodes and y_{O0}^t, y_{O4}^t, y_{3D}^t, and y_{8D}^t denoting the flow from and to nodes. For notation simplicity, we denote the upstream cells of cell i as $\Gamma^{-1}(i)$ and downstream cells of cell i as $\Gamma(i)$. According to the flow conservation, the number of vehicles in cell $i \in \{0, 1, \cdots, 8\}$ at time $t + 1$ equals to the number of vehicles in the cell at time t plus the inflow and minus the outflow, namely

$$x_i^{t+1} = x_i^t + \sum_{k \in \Gamma^{-1}(i)} y_{ki}^t - \sum_{j \in \Gamma(i)} y_{ij}^t. \tag{4.11}$$

While Equation (4.11) applies to the destination node D, the time-dependent demand term, i.e., $d(t + 1)$ needs to be added to the right hand side of the equation for the origin node O. The flow between cells on either link is governed by the following equation

$$y_{ij}^t = \min(x_i^t, Q_{ij}^t, N_j^t - x_j^t), \tag{4.12}$$

where Q_{ij}^t is the maximum allowed flow rate between cells i and j at time t, and N_j^t denotes the capacity of cell j at time t. Equation (4.12) indicates that the flow between two cells is restricted by (1) the number of vehicles in the upstream cell (i.e., x_i^t), (2) the maximum flow rate between these two cells, and (3) the remaining capacity of the downstream cell. As suggested in [145], Equation (4.12) implicitly assumes that the density wave backward propagation speed is equal to the free flow speed. Otherwise, there is supposed to be a coefficient δ, which is the ratio of the free flow speed to the wave backward propagation speed, in front of the remaining capacity of the downstream cell. In this case study, we follow the explanations in [145] and assume that the

density wave propagates backwards at the speed of free flow. Similarly, for the flow into or out of nodes, we have the following inequality conditions

$$y^t_{O0} + y^t_{O4} \leq x^t_O,\ y^t_{O0} \leq Q^t_{O0},\ y^t_{O0} \leq N^t_0 - x^t_0,\ y^t_{O4} \leq Q^t_{O4},\ y^t_{O4} \leq N^t_4 - x^t_4,$$
$$y^t_{3D} + y^t_{8D} \leq N^t_D - x^t_D,\ y^t_{3D} \leq Q^t_{3D},\ y^t_{3D} \leq x^t_3,\ y^t_{8D} \leq Q^t_{8D},\ y^t_{8D} \leq x^t_8.$$

With the aforementioned CTM propagating traffic, now we apply the developed MF-MA-DQL algorithm to two scenarios, namely a fully cooperative scenario where all selfless vehicles aim to minimize the average travel cost of all vehicles and a competitive scenario where each selfish vehicle aims to minimize her own travel cost. The optimal route choice from the former yields a dynamic system optimum (DSO), and that from the latter yields a dynamic user equilibrium (DUE). Due to the simplicity of the two-node two-link network, we employ an analytical solution, i.e., a linear programming formulation [147], for the DSO scenario and a numerical solution, i.e., an iterative method [153], for the DUE scenario as baselines. We then compare the result from the MF-MA-DQL method to that from the corresponding baseline in both scenarios.

To complete the setup, without loss of generality, we use the discretized two-node two-link network presented in Figure (4.4b) with cell capacities $N_i = 4,\ \forall i \in \{0, 1, 3, 4, 5, 6, 8\}$ and $N_i = 2,\ \forall i \in \{2, 7\}$, maximum allowed flow rate between cells $Q_{ij} = 2,\ \forall ij \in \{01, 12, 23, 45, 56, 67, 78\}$, and maximum flow rate between cell and node $Q_{O0} = Q_{O4} = Q_{3D} = Q_{8D} = 4$. In addition, the time-dependent demand is given as

$$d(t) = \begin{cases} 20, & t = 0 \\ 40, & t = 5 \\ 10, & t = 10. \end{cases}$$

4.3.1.1 Dynamic system optimum (DSO)

To achieve a DSO, all vehicles using the network behave cooperatively and aim to minimize their average travel cost, or equivalently their total travel cost.

4.3.1.1.1 MF-MA-DQL for DSO As aforementioned, the local observation of agent i contains two components, namely a node component n and a time component t. Recall that vehicles only make route choices when they are at a node. In the two-node two-link network, agents only choose their actions at the origin node O, indicating that the node component could be removed from the local observation. Therefore, $o_i = t$ in this case study. When agents are at node O, there are two allowable actions, namely either link 0 or link 1. Therefore, the action space for agent i is $a_i \in \{\text{link 0, link 1}\}$. After agent i executing action a_i with observation o_i, the agent observes a mean action $\bar{a}_i$, i.e., the number of agents on the link chosen by agent i, and travels to the destination node D following the traffic propagated by the CTM. To achieve DSO, the reward for agent i is defined as the negative average travel time of all agents, mathematically,

$$r_i = -\frac{1}{N}\sum_{i=1}^{N} \Delta t_i^{OD},$$

where Δt_i^{OD} stands for the travel time from node O to node D of agent i. In other words, in the DSO scenario, all agents receive the same amount of reward after executing their actions. With such definition of reward function, agents behave cooperatively to minimize the average travel time and achieve DSO.

4.3.1.1.2 A linear programming formulation Noticing that there is exactly one destination node in the two-node two-link network, we employ the linear programming (LP) approach developed in [147] and [154] to solve the single-destination DSO problem. Mathematically, the LP formulation is

$$\text{Minimize} \sum_{t=0}^{T}(x_O^t + \sum_{i=0}^{8} x_i^t). \tag{4.13}$$

subject to

$$x_i^{t+1} = x_i^t + \sum_{k \in \Gamma^{-1}(i)} y_{ki}^t - \sum_{j \in \Gamma(i)} y_{ij}^t, \forall i \in \{0, 1, \cdots, 8, D\}, \tag{4.14}$$

$$x_i^{t+1} = x_i^t - \sum_{j \in \Gamma(i)} y_{ij}^t + d(t+1), \text{when } i = O, \tag{4.15}$$

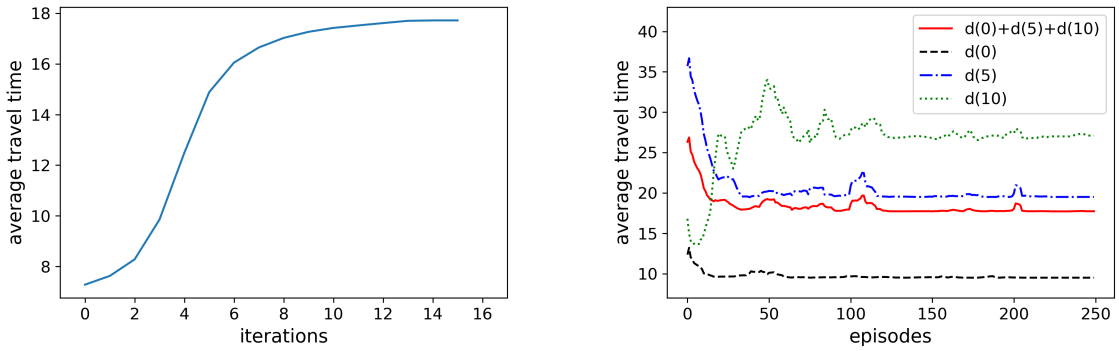
$$y_{ij}^t = min(x_i^t, Q_{ij}^t, N_j^t - x_j^t), \forall i \in \{0, 1, \cdots, 8\}, \tag{4.16}$$

$$y_{O0}^t + y_{O4}^t \leq x_O^t,\ y_{O0}^t \leq Q_{O0}^t,\ y_{O0}^t \leq N_0^t - x_0^t,\ y_{O4}^t \leq Q_{O4}^t,\ y_{O4}^t \leq N_4^t - x_4^t, \tag{4.17}$$

$$y_{3D}^t + y_{8D}^t \leq N_D^t - x_D^t,\ y_{3D}^t \leq Q_{3D}^t,\ y_{3D}^t \leq x_3^t,\ y_{8D}^t \leq Q_{8D}^t,\ y_{8D}^t \leq x_8^t, \tag{4.18}$$

$$x_i^0 = 0, \forall i \in \{0, 1, \cdots, 8, D\}, x_O^0 = d(0), \tag{4.19}$$

where the objective is to minimize the total travel cost of all vehicles. Equations (4.14) and (4.15) are the flow conservation, Equations (4.16), (4.17), and (4.18) state the restrictions on the flow between cells, Equation (4.19) is the initial condition.



(a) The LP formulation

(b) MF-MA-DQL algorithm for DSO

Figure 4.5: Convergence plots for DSO

4.3.1.1.3 Comparison between the LP solution and MF-MA-DQL solution Convergence plots of both methods for the DSO scenario are presented in Figure (4.5). For the LP formulation, it takes 15 iterations to reach convergence, as shown in Figure (4.5a). The converged value of the average travel time of all vehicles is 17.7. For the MF-MA-DQL algorithm, after bouncing back and forth during the first 150 episodes in Figure (4.5b), the average travel time of all vehicles

(i.e., $d(0) + d(5) + d(10)$) gradually reaches its converged value at 17.7, which agrees well with its analytical counterpart from the LP formulation. In addition, the converged average travel times of $d(0)$, $d(5)$, and $d(10)$ are 9.5, 19.5, and 26.9, respectively. It is worthwhile to mention that the average travel time of $d(0)$ converges faster than that of $d(5)$ and $d(10)$. This is as expected because the later demand has no effect on the earlier demand, under the assumption that overtaking does not occur. In other words, it takes some extra time for the average travel time of the later demand to achieve convergence, after that of the earlier demand has reached convergence. As for the optimal policy, 10 vehicles from $d(0)$, 21 vehicles from $d(5)$, and 5 vehicles from $d(10)$ choose link 0, and the remaining vehicles choose link 1.

4.3.1.2 Dynamic user equilibrium (DUE)

For a DUE problem, Wardrop's first principle, i.e., travel cost on all used routes for any given origin-destination pair is supposed to be the same, is a widely accepted guideline to derive DUE solutions. In other words, at equilibrium, no individual selfish vehicle could decrease her travel cost by unilaterally deviating from her current route choice strategy.

4.3.1.2.1 MF-MA-DQL for DUE All MF-MA-DQL settings such as the definition of local observation and action remain the same as those in MF-MA-DQL for DSO, except for the definition of the reward function. Recall that in DSO, the reward for an cooperative agent is the negative average travel time of all agents using the network. While in DUE, the reward for a non-cooperative agent is simply the negative of her own travel cost. In other words, instead of aiming to minimize the average travel cost of all cooperative agents in DSO, each non-cooperative agent aims to minimize her own travel cost in DUE.

4.3.1.2.2 An iterative method Noticing that the simple form of the objective in Equation (4.13) is no longer applicable to DUE problems, we employ the iterative method proposed in [153]. The basic idea of the iterative method is to let vehicles choose routes according to some policy and then calculate the network loading and travel cost by simulation so that vehicles gradually learn towards

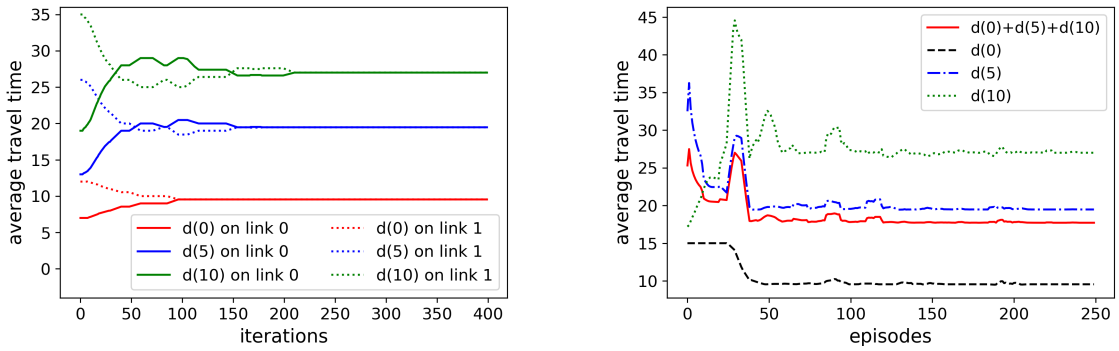
better route choices. By iterating this process, a stationary fixed point solution is supposed to be derived so that no vehicle could get better off by switching her route choices. To make this chapter self-explanatory, we now briefly introduce the iterative method adapted to the two-node two-link network. Interested readers could refer to [153] for more details.

Algorithm 6 An iterative method to solve DUE

1: Initialize a proportion $p_t(0)$ indicating the proportion of the demand at t choosing link 0. Consequently, $p_t(1) = 1 - p_t(0)$
2: Initialize a learning step size $\eta \in (0, 1]$
3: Initialize function $g(x) = exp(\frac{ax}{1-x^2})$, where a is a parameter, and function $f(x) = \frac{p_t(0)g(x)}{p_t(0)g(x) + p_t(1)}$
4: **repeat**
5: Calculate the time-dependent travel cost of both links, denoted by $\tau_t(0)$ and $\tau_t(1)$, from simulation
6: Calculate the relative cost difference $\delta_{01} = \frac{\tau_t(1) - \tau_t(0)}{\tau_t(1) + \tau_t(0)}$
7: Update the proportion by $p_t(0) = (1 - \eta)p_t(0) + \eta f(\delta_{01})$
8: Calculate $p_t(1) = 1 - p_t(0)$
9: Update function $f(x) = \frac{p_t(0)g(x)}{p_t(0)g(x) + p_t(1)}$
10: **until** the iterative method reaches a stationary fixed point
11: Return $p_t(0)$ and $p_t(1)$

The iterative method is presented in Algorithm (6). With demand $d(t)$ emerging from node O at time t, we randomly initialize $p_t(0)$ and $p_t(1)$ to denote the proportion of the demand choosing link 0 and link 1, respectively. We also initialize functions $g(x)$ and $f(x)$ which will be used to update $p_t(0)$ and $p_t(1)$. Note that $f : [-1, 1] \rightarrow [0, 1]$ is a monotonic increasing function. With $p_t(0)$ and $p_t(1)$, we run simulation to calculate the average travel cost on both links for the demand starting at time t and denote these time-dependent travel cost by $\tau_t(0)$ and $\tau_t(1)$. We then calculate the relative cost difference $\delta_{01} = \frac{\tau_t(1) - \tau_t(0)}{\tau_t(1) + \tau_t(0)}$ and use this cost difference to update the proportion $p_t(0)$ towards $f(\delta_{01})$ by step size η. The rationale of this updating rule is as follows. When δ_{01} is large (e.g., close to 1), meaning that the cost on link 0 is much less than that on link 1, we have $f(\delta_{01}) \approx 1$, indicating that $p_t(0)$ is updated towards a larger value; when δ_{01} is small (e.g., close to -1), meaning that the cost on link 0 is much larger than that on link 1, we have $f(\delta_{01}) \approx 0$,

indicating that $p_t(0)$ is updated towards a smaller value. After updating $p_t(0)$ and $p_t(1)$, we repeat the iterative process until reaching a stationary fixed point.



(a) The iterative method

(b) MF-MA-DQL algorithm for DUE

Figure 4.6: Convergence plots for DUE

4.3.1.2.3 Comparison between the solution of the iterative method and MF-MA-DQL solution Convergence plots of both methods for the DUE scenario are presented in Figure (4.6). In Figure (4.6a), the average travel time for each batch of demand (i.e., $d(0)$, $d(5)$, and $d(10)$) on each link (i.e., link 0 and link 1) is plotted. For example, for $d(0)$, there is a certain proportion of $d(0)$ choosing link 0 (we call this part $d(0)$ on link 0) and the remaining proportion choosing link 1 (we call this part $d(0)$ on link 1). Initially, $d(0)$ on link 1 experiences a higher average travel time than $d(0)$ on link 0. The iterative method then updates $p_0(0)$ (i.e., the proportion of $d(0)$ choosing link 0) to be larger and $p_1(0)$ to be smaller. The average travel time will thus become larger on link 0 and lower on link 1. Recall that the iterative method reaches convergence when the average travel times on both links become the same. After 220 iterations, the average travel times on both links for all batches of demand converge. The converged average travel times for $d(0)$, $d(5)$, and $d(10)$ are 9.6, 19.5, and 27.0, respectively.

For the MF-MA-DQL algorithm, after bouncing back and forth during the first 120 episodes in Figure (4.6b), average travel travel times of $d(0)$, $d(5)$, $d(10)$, and $d(0) + d(5) + d(10)$ gradually reach their converged values 9.6, 19.6, 27.0, and 17.7, respectively, which agree well with the

values derived from the iterative method. As for the optimal policy, 11 vehicles from $d(0)$, 19 vehicles from $d(5)$, and 6 vehicles from $d(10)$ choose link 0, and other vehicles choose link 1.

4.3.1.3 DSO versus DUE

The agreement between the LP solution and the MF-MA-DQL solution in the DSO scenario and that between the solution from the iterative method and MF-MA-DQL solution in the DUE scenario validate the effectiveness of the developed MF-MA-DQL algorithm. Here we stress that the MF-MA-DQL algorithm is more versatile and could be applied to cooperative, competitive, and even mixed scenarios.

It is worth mentioning the difference between DSO and DUE in this two-node two-link network. Table (4.3) lists the average travel time and optimal policy for each batch of demand in both DSO and DUE scenarios. Although one could see that both the average travel time and the optimal policy seem quite similar for these two scenarios, DSO and DUE are intrinsically different. For the purpose of demonstration, we analyze the first batch of demand, i.e., $d(0)$. In the DSO scenario, the optimal policy suggests 10 vehicles to choose link 0 and the other 10 vehicles to choose link 1. The average travel time for the 10 vehicles on link 0 is 9 and that for the remaining 10 vehicles on link 1 is 10, indicating that average travel times on two used links could be different in the DSO scenario. The average travel time of these 20 vehicles is 9.5. In the DUE scenario, the optimal policy suggests 11 vehicles to choose link 0 and the other 9 vehicles to choose link 1. The average travel time for both the 11 vehicles on link 0 and the remaining 9 vehicles on link 1 is 9.6. The average travel time of these 20 vehicles is 9.6. In summary, the average travel time is lower in the DSO scenario where the average travel times on used links could be different; while in the DUE scenario, the Wardrop first principle is respected.

Table 4.3: DSO versus DUE

	average travel time				optimal policy (choosing link 0)		
	$d(0)$	$d(5)$	$d(10)$	$d(0)+d(5)+d(10)$	$d(0)$	$d(5)$	$d(10)$
DSO	9.5	19.5	26.9	17.7	10	21	5
DUE	9.6	19.5	27.0	17.7	11	19	6

4.4 Bilevel optimization for multi-agent route choice

Traffic congestion has become one of the major issues faced by modern cities. With the limited transportation network capacity, traffic congestion occurs when there are too many vehicles using the road network, especially on some critical links. To mitigate traffic congestion, in addition to increasing the infrastructure supply which could be costly, city planners can actually resort to tolling to redistribute traffic demand on the network and improve the overall traffic condition. While imposing a toll charge on a critical link may make the link less attractive to travelers, similar to the effect of decreasing the capacity of the link, no toll may attract too many travelers to the link and result in traffic congestion or even jam on this link, as the Braess paradox suggests [155, 156]. In other words, compared with no tolling, a proper toll on a critical link could redistribute some traffic to other links and consequently improve the overall traffic condition.

In this section, we aim to investigate the effect of tolling a critical link on the overall traffic network. Note that from the perspective of travelers, the overall travel cost is the summation of their travel time and the toll charge paid out of their pocket. Without loss of generality, we assume that the travel time and toll charge could directly be added without unit conversion. In other words, we assume travelers value their time and monetary cost similarly. We study the effect of tolling a link on the Braess network, as presented in Figure (4.1). There are some travelers from origin node n_0 to terminal node n_3, and the travel time on link l_{21}, i.e., $\Delta t_{21} = \alpha$. We regard α as the toll charge on link l_{21}. A larger α means that the toll is higher, indicating that there might be fewer travelers choosing the link; while a smaller α means that the toll is lower, indicating that more travelers may choose this link. We aim to find an optimal value of α with which some overall systematic performance such as the average travel time of all vehicle drivers is optimized. We denote this systematic performance by f.

As aforementioned, behavior of travelers change with the adjustment in α. Recall that our goal is to find the optimal α, we thus formulate this as a bilevel optimization problem, as presented in Figure (4.7). The upper level is city planners aiming to optimize the systematic performance f and

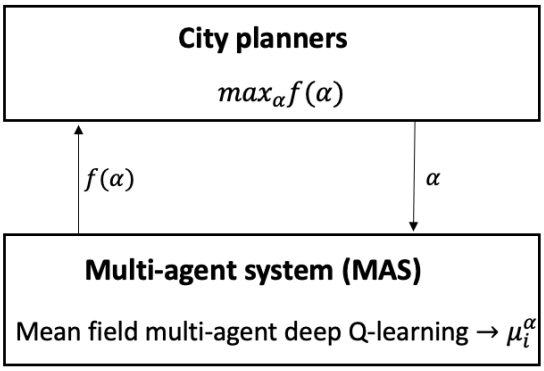


Figure 4.7: Bilevel optimization model

has some control, parameterized by α, over the lower level MAS. In the lower level MAS, each agent $i \in \{1, 2, \cdots, N\}$ derives her optimal policy μ_i^α by the developed MF-MA-DQL algorithm presented in the previous section. Note that we use superscript α in the notation of optimal policy, i.e., μ_i^α, to signify the dependency of optimal policy on the control parameter α. With optimal behavior of all agents, city planners observe the systematic performance $f(\alpha)$. City planners then adjust control parameter α until reaching optimum. Due to the unknown complex structure of f over α, we resort to Bayesian optimization, as presented in Section (3.4) of Chapter 3.

4.4.1 Numerical example

We now apply the bilevel optimization model to the Braess network presented in Figure (4.1). The rationale of adopting the Braess network in this section is as follows. First, the Braess network is simple, and usually an analytical solution could be derived, meaning that we could compare our numerical solution to its analytical counterpart. Second, recall that our goal is to find an optimal α with which the overall systematic performance (i.e., average travel time of all drivers in the Braess network) is optimized. The existence of Braess paradox in the Braess network makes our goal nontrivial.

4.4.1.1 Lower level: validation of the MF-MA-DQL on Braess network

We first validate the developed MF-MA-DQL algorithm in two cases, namely a case with single-batch demand and a case with multi-batch demand. Details of the setup of each case will be discussed later. In both cases, centralized Q functions are approximated by a multilayer perceptron (MLP) with three hidden layers. ReLU is the activation function used between layers. Learning rate $\eta = 10^{-3}$. Exploration parameter ϵ is initially set as 0.1, i.e., $\epsilon_0 = 0.1$, and then decreases to 0.01 during training. The target network update period $\tau = 5$, meaning that target network $Q(o, a, \bar{a}|\theta^-)$ is copied from the principal network, i.e., $Q(o, a, \bar{a}|\theta)$ every 5 episodes.

In the case with single-batch demand, number of drivers initially at origin node n_0 is set as 40, and $k_{02} = k_{13} = 1$. We further assume $\alpha = 100$, i.e., a very large number, indicating that link l_{21} is not supposed to be used by any agent after training in this case. The convergence of the MF-MA-DQL algorithm for this case is presented in Figure (4.8). The y axis is the average travel time of all 40 agents from origin node n_0 to terminal node n_3. The x axis shows the number of episodes. Initially, because agents are moving around randomly to explore the environment, the average travel time is as high as 130. During the first 10 episodes, the average travel time decreases very fast to around 70, indicating agents are learning towards a better policy. After some bouncing back and forth, the average travel time reaches its converged value at 65 after 100 episodes. The derived optimal policy shows that 20 agents choose route $n_0 \rightarrow n_1 \rightarrow n_3$ and the other 20 choose route $n_0 \rightarrow n_2 \rightarrow n_3$.

The analytical solution for this case could be derived as follows. First, notice that route $n_0 \rightarrow n_1$ strictly dominates route $n_0 \rightarrow n_2 \rightarrow n_1$, because $\Delta t_{02} + \Delta t_{21} > \Delta t_{01}$ always holds. Similarly, route $n_2 \rightarrow n_3$ strictly dominates route $n_2 \rightarrow n_1 \rightarrow n_3$. Therefore, rational agents will not use l_{21} under any circumstances. Consequently, there are only two reasonable routes from n_0 to n_3, namely $n_0 \rightarrow n_1 \rightarrow n_3$ and $n_0 \rightarrow n_2 \rightarrow n_3$. Assuming there are y agents choosing the former and $40 - y$ agents choosing the latter, at equilibrium, if both routes are used, the travel time on both routes is supposed to be the same. Otherwise, some agents currently on the route with a larger travel time

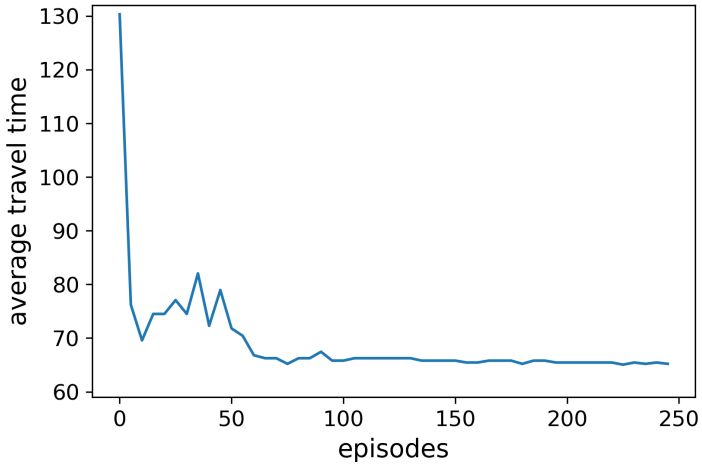


Figure 4.8: Convergence of MF-MA-DQL algorithm in the case with single-batch demand

could deviate and choose the other route to decrease their travel time. Mathematically,

$$45 + k_{13} \cdot y = k_{02} \cdot (40 - y) + 45,$$

where the left hand side is travel time on the former route and the right hand side is travel time on the latter route. By some simple linear algebra,

$$y = \frac{40 \cdot k_{02}}{k_{02} + k_{13}}. \tag{4.20}$$

Plugging $k_{02} = k_{13} = 1$, $y = 20$, meaning that at equilibrium, 20 agents choose the former route and 20 choose the latter. Travel time for all agents at equilibrium is 65. Both travel time and optimal policy from previous numerical solution agree well with their analytical counterparts from the analytical solution. Here we stress that this case validates the effectiveness of the developed MF-MA-DQL algorithm. Although all agents share one centralized Q function, agents find different optimal policies. In comparison, if a single-agent deep Q-learning was adopted to tackle this multi-agent problem, agents share a centralized Q function and one optimal policy. Unfortunately, a shared optimal policy guides all agents to use the same route, which is definitely not optimal for agents. For example, the travel time for an agent would be $45 + 40 = 85$ if all 40 agents choose

either route $n_0 \rightarrow n_1 \rightarrow n_3$ or route $n_0 \rightarrow n_2 \rightarrow n_3$.

Actually, the aforementioned case is symmetric, because travel time on both routes is $45 + x$, where x is the flow (i.e., number of agents) choosing the route. To further test the effectiveness of the MF-MA-DQL algorithm in asymmetric cases, we break the symmetry by keeping $k_{02} = 1$ and varying k_{13}. Specifically, we test the MF-MA-DQL algorithm with 10 different values of k_{13}, namely $k_{13} \in \{1, 2, \cdots, 10\}$. Other parameters such as α and number of agents remain unchanged. With respective to the analytical solution, it follows Equation (4.20) with $k_{02} = 1$, yielding $y = \frac{40}{1+k_{13}}$ and travel time for all agents as $45 + \frac{40 \cdot k_{13}}{1+k_{13}}$, at equilibrium. Note that the analytical solution applies to both integer and fractional k_{13}. The comparison of average travel time between the numerical solution (i.e., using MF-MA-DQL) and analytical solution is presented in Figure (4.9). The y axis is the average travel time of all agents after convergence under a given k_{13}. All red dots (i.e., numerical solution) are on the blue curve (i.e., analytical solution), indicating a very good agreement between the numerical and analytical solution. This validates the effectiveness of the developed MF-MA-DQL algorithm in asymmetric cases.

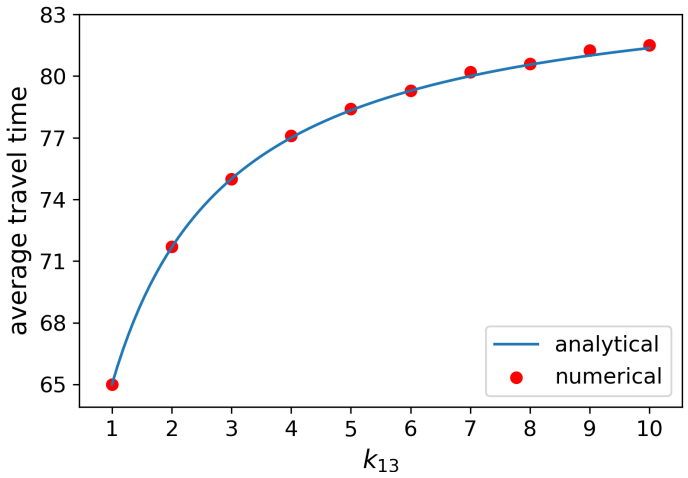


Figure 4.9: Comparison between numerical solution and analytical solution with varying k_{13}

After validating the MF-MA-DQL algorithm in the case with single-batch demand, we now aim to further validate it in a case with multi-batch demand. Similar to the case with single-batch demand, number of drivers initially at origin node n_0 is set as 40. In addition, another 20 drivers

will depart from n_0 at time $t = 10$. For notation simplicity, we call 40 agents who depart from n_0 at $t = 0$ the first 40 agents and 20 agents who depart from n_0 the latter 20 agents. In this case, route choice of the first 40 agents has impact on behavior of the latter 20 agents, because the first 40 agents are on some links and thus travel time on those links could be larger. We further assume $\alpha = 1$ in this case.

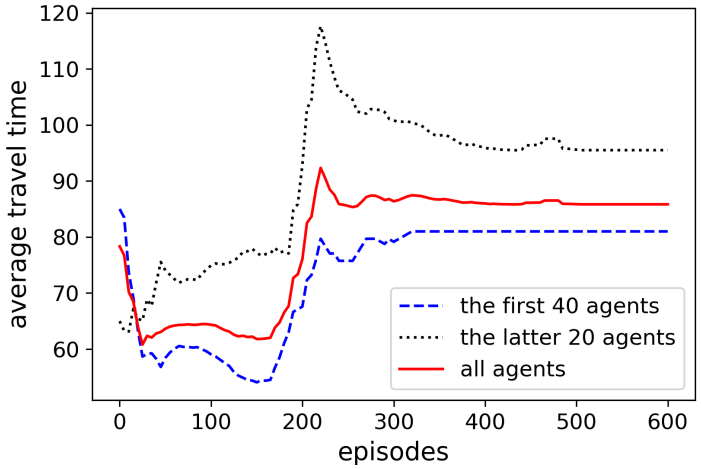


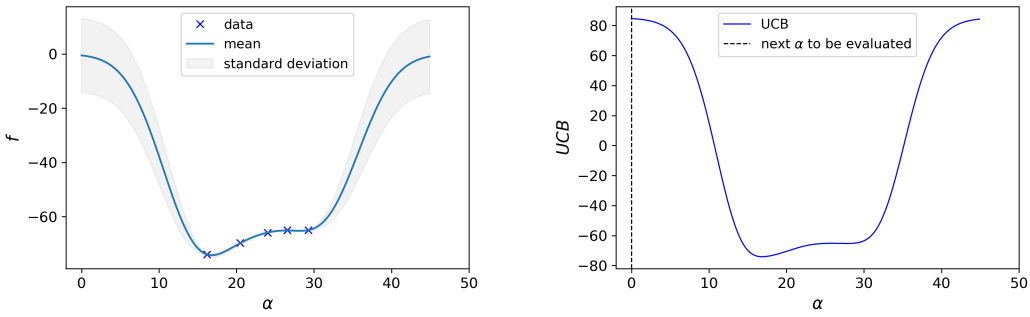
Figure 4.10: Convergence of MF-MA-DQL algorithm in the case with multi-batch demand

Figure (4.10) presents the average travel time of the first 40 agents, the latter 20 agents, and all 60 agents, respectively, versus the number of training episodes. The first 40 agents explore the environment and learn towards their optimal policies during the first 300 episodes. It is noteworthy that the average travel time of the first 40 agents decreases fast, then bounces back and forth below 60, and finally increases fast to around 80. The reason is as follows. Initially the first 40 agents act according to the randomly initialized centralized Q function and there is no experience accumulated, indicating that agents choose the same route (e.g., $n_0 \rightarrow n_1 \rightarrow n_3$), resulting in a large travel time of 85. After some exploration, some agents learn that the average travel time of other routes are lower (e.g., $n_0 \rightarrow n_2 \rightarrow n_1 \rightarrow n_3$). Therefore, these agents are now choosing the route with a lower travel time, leading to a lower average travel time (e.g., below 60). As more and more agents learn this better route with lower travel time, the average travel time increases, because the traffic flow (i.e., number of agents) on this route increases. The average travel time of

the first 40 agents reaches its converged value at 81 after 300 episodes, and the optimal policy is to choose route $n_0 \rightarrow n_2 \rightarrow n_1 \rightarrow n_3$. Actually, with $\alpha = 1$, route $n_0 \rightarrow n_2 \rightarrow n_1$ strictly dominates route $n_0 \rightarrow n_2$, because $\Delta t_{02} > \Delta t_{01} + \Delta t_{12}$ always holds. Similarly, route $n_2 \rightarrow n_1 \rightarrow n_3$ strictly dominates route $n_2 \rightarrow n_3$. Therefore, the analytical optimal policy for the first 40 agents is to choose route $n_0 \rightarrow n_2 \rightarrow n_1 \rightarrow n_3$ with average travel time $\Delta t_{02} + \Delta t_{21} + \Delta t_{13} = 40 + 1 + 40 = 81$. As for the latter 20 agents, their average travel time and optimal policies are strongly affected by the first 40 agents. Actually, after the average travel time of the first 40 agents converges at 300 episodes, it takes another 100 episodes for the average travel time of the latter 20 agents to converge. It is as expected because behavior of the first 40 agents has impact on that of the latter 20 agents, while the behavior of 20 agents has no effect on that of the first 40 agents, under the assumption that overtaking does not occur. In other words, after the first 40 agents find their optimal policy, the latter 20 agents need to keep exploring the environment to derive their own optimal policy. After 400 episodes, the average travel time of the latter 20 agents reaches its converged value at 95, which is consistent with the value from analytical solution. Note that due to similarity, we omit the derivation of analytical solution here.

4.4.1.2 Bilevel: emergence of Braess paradox and the optimal toll pricing

After validating the MF-MA-DQL algorithm in aforementioned cases, we now run bilevel optimization on the Braess network with $k_{02} = k_{13} = 1$ and α as the control variable of upper level city planners. Similar to case with single-batch demand, initially there are 40 travelers at node n_0. In other words, with the demand as 40 agents from n_0 to n_3, we aim to find the optimal α with which the average travel time of these 40 agents could be optimized. To be precise, the negative average travel time of these 40 agents is taken as the objective f of the upper level city planners. The range of α is set as $[0, 45]$. A case with $\alpha > 45$ is trivial because $\Delta t_{02} + \Delta t_{21} > \Delta t_{01}$ and $\Delta t_{21} + \Delta t_{13} > \Delta t_{23}$ always hold. Therefore, $n_0 \rightarrow n_1$ dominates $n_0 \rightarrow n_2 \rightarrow n_1$ and $n_2 \rightarrow n_3$ dominates $n_2 \rightarrow n_1 \rightarrow n_3$, indicating that link l_{21} will not be taken by any agent.



(a) Posterior probability distribution of f conditioned on the initial evaluated five α's

(b) Acquisition function

Figure 4.11: Posterior probability distribution and acquisition function at iteration 0

With the problem well defined, we use BO to solve the bilevel problem. As for the starting point of BO, we evaluate objective f at five randomly sampled α's. Figure (4.11a) presents the posterior probability distribution of f conditioned on these five initially evaluated α's. As one can see, standard deviation is small near evaluated α's while large around locations with no evaluated data. Acquisition funcition used in this study is UCB and is plotted in Figure (4.11b). Based on the acquisition function, the next α to be evaluated is 0.

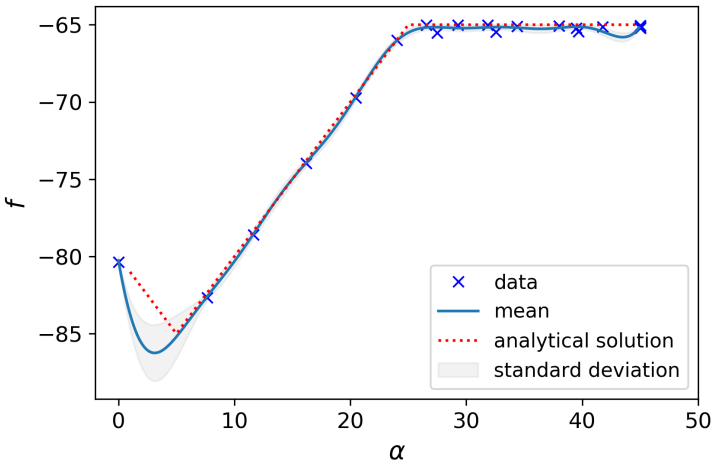


Figure 4.12: Posterior probability distribution of f

We then run BO for 15 iterations. In other words, we assume there is some computational

budget of 15 evaluations of f. The posterior probability distribution of f along with the analytical solution is plotted in Figure (4.12). The derivation of analytical solution is detailed as follows. From node n_0 to node n_3, there are in total three possible routes, namely route $n_0 \rightarrow n_1 \rightarrow n_3$, route $n_0 \rightarrow n_2 \rightarrow n_1 \rightarrow n_3$, and route $n_0 \rightarrow n_2 \rightarrow n_3$. Assuming there are y_1 agents choosing the first route, y_2 agents choosing the seconds route, and $40 - y_1 - y_2$ choosing the last route, at equilibrium, travel time on these routes is supposed to be the same, if all of them are used. Therefore, we have the following equation

$$45 + y_1 + y_2 = y_2 + (40 - y_1 - y_2) + (y_1 + y_2) = y_2 + (40 - y_1 - y_2) + 45.$$

By some simple linear algebra, we solve for y_1 and y_2 as follows

$$y_1 = \alpha - 5,$$
$$y_2 = 50 - 2\alpha.$$

In addition, there are constraints on number of agents choosing a route, i.e., $0 \leq y_1 \leq 40$ and $0 \leq y_2 \leq 40$. These constraints yield $5 \leq \alpha \leq 25$. Actually, with $\alpha < 5$, it can be easily seen that only route $n_0 \rightarrow n_2 \rightarrow n_1 \rightarrow n_3$ is used by agents; while with $\alpha > 25$, route $n_0 \rightarrow n_2 \rightarrow n_1 \rightarrow n_3$ is not used by any agent. Therefore, the objective f, i.e., negative average travel time, versus α could be solved as

$$f(\alpha) = \begin{cases} -(80 + \alpha), & \alpha < 5 \\ -(90 - \alpha), & 5 \leq \alpha \leq 25 \\ -65, & \alpha > 25. \end{cases}$$

The analytical is plotted as the red dotted line in Figure (4.12). The overall good agreement between the analytical solution and numerical solution validates the bilevel optimization model.

With $\alpha = 0$ (i.e., no toll), the optimal route choice, from both the numerical and analytical solution, for all agents is to use route $n_0 \rightarrow n_1 \rightarrow n_2 \rightarrow n_3$, and the average travel time of all agents is 80; with $\alpha > 25$ (i.e., a large toll), the optimal route choice is half of agents using route

$n_0 \rightarrow n_1 \rightarrow n_3$ and the other half using route $n_0 \rightarrow n_2 \rightarrow n_3$, and the average travel time of all agents is 65. This indicates that a small toll (i.e., a shorter travel time) on link l_{12} actually increases the overall travel time of all agents, resulting in the emergence of Braess paradox. In other words, decreasing the travel cost on a link by either expanding the capacity of the link or reducing the toll charge on the link may not benefit the overall traffic condition. On the contrary, it (i.e., decreasing the travel cost on a link) may deteriorate the overall traffic condition by attracting too many travelers to the link. In this case study, the optimal toll pricing on link l_{12} is a travel time greater than or equal to 25, which yields the optimal systematic objective.

4.5 Case Study

In this section, we apply the developed bilevel optimization model to a real-world road network with 69 nodes and 166 links, as presented in Figure (4.13). This road network covers the area from 133th Street (south) to 146th Street (north) and from Riverside Drive (west) to Convent Avenue (east) in upper Manhattan. In addition to Riverside Drive and Convent Avenue, there are Broadway and Amsterdam Avenue in the north-south direction. The road network is imported into SUMO.

4.5.1 Setup of the bilevel problem

Now we detail the setup of the bilevel problem.

4.5.1.1 Lower level: controllable agents and background traffic

There are two types of vehicles in the road network, namely controllable agents and background traffic. The former actively adapts her route choices while the latter constantly follows some prescribed travel pattern. In this case study, there are four groups of controllable agents: 1) three agents travel from node 14 to node 60, 2) three agents from node 15 to node 60, 3) three agents from node 48 to node 1, and 4) three agents from node 69 to node 1. With respect to the background traffic, there are in total around 1,600 vehicles in the south-north direction and 500 vehicles in the east-west direction within the simulation time period (i.e., 1000 seconds).

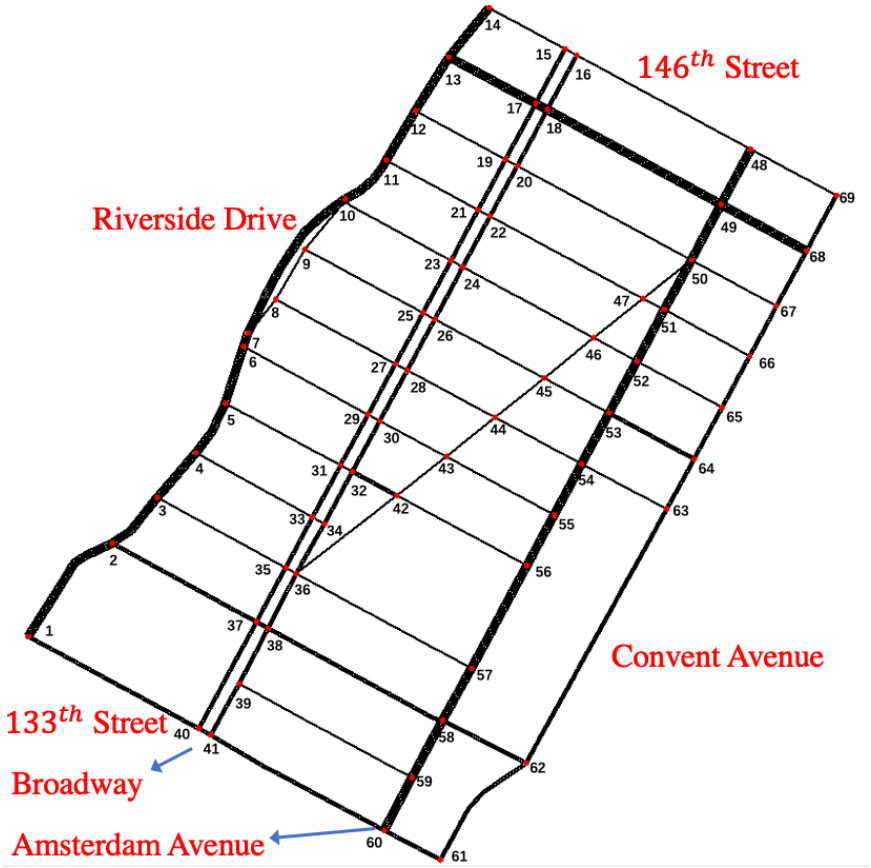


Figure 4.13: Road network in SUMO

The goal of each controllable agent is to minimize her travel cost from origin to destination. Noticing that the first two groups of controllable agents share the same destination, i.e., node 60, these agents thus share the same Q function, regardless of the difference in their departure times. We denote this Q function as Q^{60}, where the superscript 60 indicates that this Q function is used by agents whose destination is node 60. Obviously, Q value at destination node is 0, i.e., $Q^{60}(o_i = (60, t), a_i, \bar{a}_i) = 0$, regardless of time t, action a_i, and mean action $\bar{a}_i$. Similarly, the remaining controllable agents whose destination is node 1 share the same Q function, which is denoted by Q^1. $Q^1(o_j = (1, t), a_j, \bar{a}_j) = 0$.

4.5.1.2 Upper level: signal control

With the controllable agents aiming to minimize their travel time and background traffic following a prescribed traffic profile, city planners can affect the route choice behavior of adaptive

controllable agents by adjusting the traffic signals at intersections, i.e., signal control. The goal of city planners is to develop a proper signal control scheme so that the average travel time of all controllable agents is minimized.

In this study, we assume that city planners only adjust traffic signals on Broadway. In addition, we assume that there are green light with fixed duration of 60 seconds and red light with fixed duration of 30 seconds on Broadway. The only controllable variable is the offset between green lights of two consecutive intersections on Broadway. To be precise, we formally define the offset below.

Definition 4.5.1. (*Offset* [157]). The offset between green lights of two consecutive intersections is the difference between starting time of green light at one intersection and that at the previous upstream intersection.

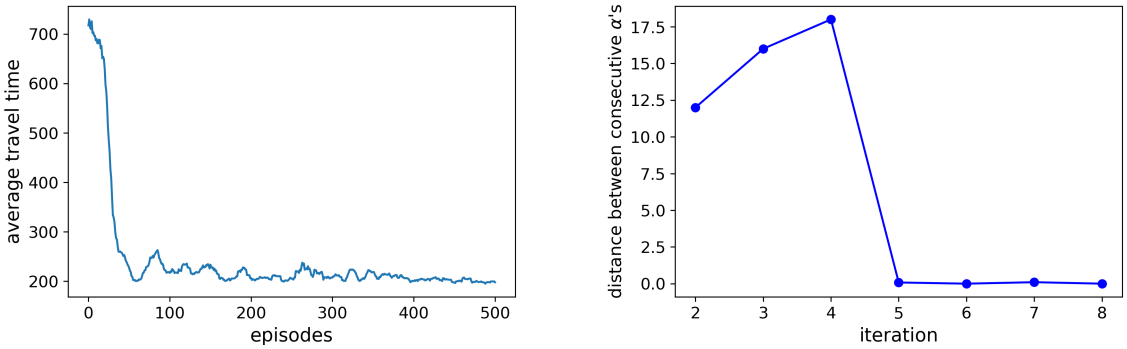
For example, along the north-south direction, the difference between the starting time of green light at node 17 and that at node 15 is the offset. Note that in this study, we assume that the same offset applies to every two consecutive intersections on Broadway. For example, the offset of node 17 and node 15 is the same as that of node 19 and node 17.

Denoting the negative average travel time of all controllable agents as f and the offset as α, we have $f(\alpha)$, meaning that f is dependent on α. The rationale underlying such dependency is partially explained as follows. With different values of α (i.e., the offset), controllable agents may experience different travel times. For example, with a proper α, controllable agents may take advantage of “green wave" on Broadway and thus spend less time reaching their destination, leading to a larger f. On the contrary, with a poorly chosen α, controllable agents may need to stop frequently and spend more time on waiting at the intersection, resulting in a smaller f. The goal is to find the optimal α^* which maximizes f, i.e.,

$$\alpha^* = \arg\max_{\alpha} f(\alpha).$$

4.5.2 Results

With the aforementioned setup, we run the bilevel optimization model until convergence.



(a) Lower level with offset as 4

(b) Upper level

Figure 4.14: Convergence of BO

Convergence plots of both the lower level flow-dependent deep Q-learning and the upper level BO are presented in Figure (4.14). Figure (4.14a) presents the variation of the average travel time of all controllable agents versus the number of episodes. Initially, agents spend a very long time (i.e., more than 700 seconds) on traveling to their destinations. Actually, some agents may not be able to reach their destinations and could record a total travel time of 1,000 seconds (i.e., the maximum allowed time in one episode). During the first 60 episodes, agents explore the road network and learn towards a better policy pretty fast. Therefore, their average travel time is substantially decreased from above 700 seconds to around 200 seconds. Despite some bouncing back and forth between 60 episodes and 300 episodes, the average travel time of all controllable agents stabilizes around 200 seconds after 300 episodes. Especially after 400 episodes, the average travel time almost barely changes. Figure (4.14b) presents the distance between two consecutive evaluated α's in the BO algorithm. A smaller distance means that BO chooses to evaluate similar α's, indicating that the algorithm approaches convergence. In this study, we stop the BO algorithm when the distance is smaller than a threshold value (i.e., 2.5) for four times in a row. With five randomly selected α's as starting point, BO reaches convergence after 9 additional iterations. As one could see, BO selects very different α's during the first 4 iterations and starts selecting similar α's from

the 5^{th} iteration.

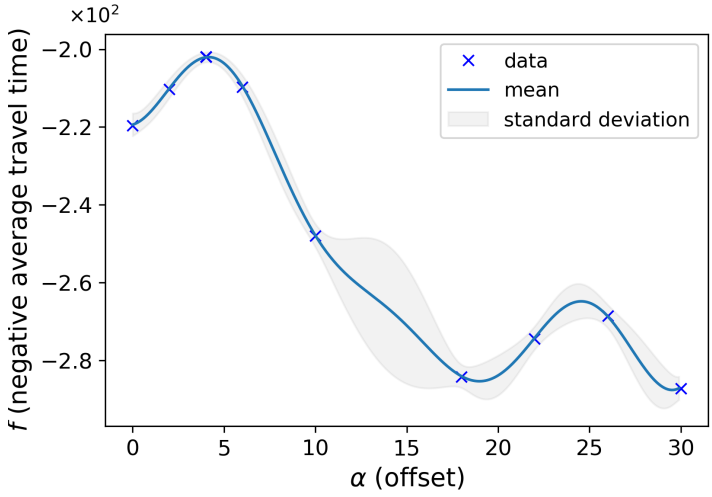


Figure 4.15: Posterior probability distribution of f at the 8^{th} iteration

With the illustrated convergence of both levels, the final result of the posterior probability distribution of the objective f (i.e., the negative average travel time) is shown in Figure (4.15). The x axis is the controllable variable α, i.e., the offset in this study. The y axis is the objective f. The data records all evaluations, i.e., the evaluated f at a chosen α. The mean and the standard deviation of the Gaussian process fitting based on the data are also plotted. As one could see, the standard deviation is small around evaluated α's while large when there is no nearby evaluated α's. The final result suggests that the optimal α is 4, i.e., $\alpha^* = 4$. With the optimal α, the optimal value of the objective is around -200, meaning that with offset as 4, the average travel time of all controllable agents is 200 seconds. Compared with $\alpha = 0$ and the corresponding average travel time as 220 seconds, the optimal α could reduce the average travel time by 9%. A much larger α (i.e., $\alpha > 7$) may deteriorate the traffic condition and results in a long average travel time.

To provide more insights, we decompose the average travel time of all controllable agents into two components, namely average waiting time (at intersection) and average cruising time. Both components are presented in Figure (4.16). In general, a smaller α (e.g., $\alpha < 10$) yields a smaller average waiting time and a smaller average cruising time while a larger α results in a higher average waiting time and a higher cruising time. This could be partially explained as follows. Although

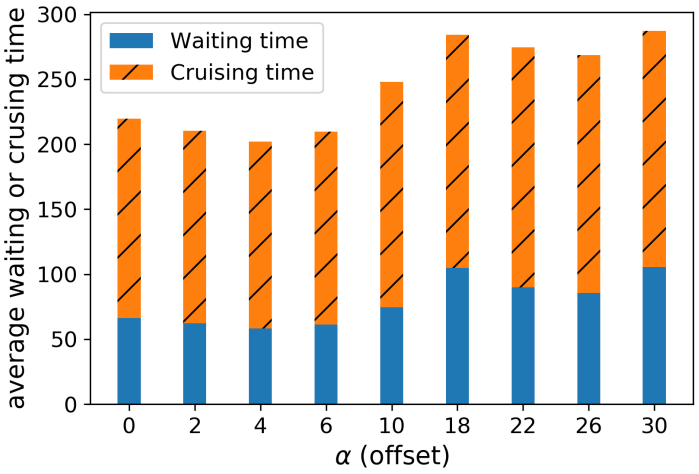


Figure 4.16: Decomposition of average travel time

the road network is large in the sense that it has a large number of nodes and links, it covers a relatively small area in Manhattan. The average cruising time from one intersection to an adjacent intersection along the south-north direction is typically around 5 seconds, which is significantly smaller than 10 seconds. With an offset larger than 10 seconds, vehicles have a higher probability of stopping at the next intersection after they stop and wait for green light at the current intersection, resulting in not only a longer accumulative waiting time but also a deteriorated traffic condition. While with a smaller α, vehicles could take advantage of the “green wave" and enjoy a smaller waiting time and better traffic condition. The lowest waiting time and the lowest cruising time are achieved when the offset is set as 4 seconds, which is exactly the previously derived optimal α^*.

4.6 Summary

This chapter develops a unified paradigm that models one’s learning behavior and the system’s equilibrating processes in a routing game among atomic selfish agents. Such a paradigm can assist policymakers in devising optimal operational and planning countermeasures under both normal and abnormal circumstances. A flow-dependent mean field deep Q-learning algorithm is developed to tackle the route choice task of multi-commodity multi-class agents. The flow dependent mean action, which is defined as the traffic flow on the chosen link, carries partial but not full

information of nearby agents and is thus able to partially capture the competition among agents. In addition, the use of flow-dependent mean action enables Q-value sharing and policy sharing, which is computationally efficient. The developed algorithm is extensively applied to three road networks, namely a two-node two-link network with CTM as its traffic propagation mechanism (see Section (4.3)), a Braess network with a linear volume delay function (see Section (4.4)), and a large-sized real-world road network with 69 nodes and 166 links implemented in SUMO (see Section (4.5)). The thorough testing unveils the versatility and effectiveness of the developed model-free MARL algorithm.

The linkage between the classic DUE paradigm and our proposed MARL paradigm is then demonstrated in Section (4.3). On a two-node two link network with CTM as the traffic propagation mechanism, we show that the numerical solution from the developed MARL algorithm agrees well with the solution from an analytical LP formulation in the DSO scenario and the solution from an iterative method in the DUE scenario. To our knowledge, this research is the first-of-its-kind to unify the mode-based (i.e., DUE) and data-driven (i.e., MARL) paradigms for dynamic routing games.

We demonstrate the effect of two countermeasures, namely tolling (see Section (4.4)) and signal control (see Section (4.5)), on the behavior of travelers and show that the systematic objective of city planners can be optimized by a proper control. The results show that on the Braess network in the study, the optimal toll charge on link l_{12} is greater or equal to 25, with which the average travel time of selfish agents is minimized and the emergence of Braess paradox could be avoided. In the large-sized real-world road network, the optimal offset for signal control on Broadway is derived as 4 seconds, with which the average travel time of all controllable agents is minimized. The decomposition of travel time into waiting time (at intersections) and cruising time (on the link) further reveals that travelers could take advantage of potential “green wave" when the offset is relatively small (i.e., less than 10 seconds). We note that both the waiting time and cruising time are minimized at the optimal offset, i.e., 4 seconds.

Chapter 5: Conclusions and future research directions

In this chapter, we will first conclude this dissertation and reiterate significant research findings and contributions. As agent-based models are gaining more momentum in MASs, we point out some future research directions in the transportation domain.

5.1 Conclusions

With the goal to improve the efficiency of the sharing economy, this dissertation develops an agent-based methodological framework for optimal decision-making of SAVs, including passenger-seeking and route choice. Noticing that each agent (i.e., SAV) is assumed to be perfectly rational and selfish, city planners can impact the behavior of agents by resorting to some operational measures such as congestion pricing and signal control. We thus formulate the overall problem that involves the decision-making process of both distributed agents and central city planners as a bilevel optimization problem, which is efficiently solved by a Bayesian optimization method.

As a stepping stone, we first propose a model-based modified MDP approach to tackle the e-hailing drivers' repositioning (i.e., passenger-seeking) problem in a single-agent setting. The modified MDP approach incorporates distinct features of e-hailing drivers and is found to achieve a 17.5% improvement and a 7.5% improvement over a baseline (i.e., the local hotspot strategy) in terms of the rate of return and the utilization rate, respectively. Although the modified MDP approach is set up in the single-agent setting, we extend its applicability to multi-agent scenarios by a dynamic adjustment strategy of the order matching probability which is able to partially capture the competition among agents. Furthermore, noticing that the reward function is commonly assumed as some prior knowledge, this research unveils the underlying reward function of the overall e-hailing driver population (i.e., 44,000 Didi drivers in Beijing) through an inverse reinforcement

learning method, which paves the way for future research on discovering the underlying reward mechanism in a complex and dynamic ride-hailing market.

To better incorporate the competition among agents, we then develop a model-free mean-field multi-agent actor-critic algorithm. A bilevel optimization model is then formulated with the upper level as a reward design mechanism and the lower level as an MAS. We use the developed mean field multi-agent actor-critic algorithm to solve for the optimal passenger-seeking policies of distributed agents in the lower level and Bayesian optimization to solve for the optimal control of upper-level city planners. The bilevel optimization model is applied to a multi-class taxi driver repositioning task with congestion pricing as the upper-level control. It is disclosed that a \$5.8 toll charge increases the objective of city planners by 22%, compared to that without any toll charge. With the optimal toll charge, the percentage of idle taxis in the CBD is decreased, indicating a better traffic condition. This real-world large-scale case study discloses the effectiveness of the proposed bilevel optimization model.

With agents knowing where to go (i.e., passenger-seeking), we now apply the bilevel optimization model to the research question of how to get there (i.e., route choice). Different from the task of passenger-seeking where the action space is always 5-dimensional, the problem of variable action set emerges in the task of route choice. Therefore, a flow-dependent deep Q-learning algorithm is proposed to efficiently derive the optimal policies for multi-commodity multi-class agents. We demonstrate the effect of two countermeasures, namely tolling and signal control, on the behavior of travelers and show that the systematic objective of city planners can be optimized by a proper control. The results show that on the Braess network, the optimal toll charge on link l_{12} is greater or equal to 25, with which the average travel time of selfish agents is minimized and the emergence of Braess paradox could be avoided. In the large-sized real-world road network with 69 nodes and 166 links, the optimal offset for signal control on Broadway is derived as 4 seconds, with which the average travel time of all controllable agents is minimized. The decomposition of travel time into waiting time (at intersections) and cruising time (on the link) further reveals that travelers could take advantage of potential “green wave" when the offset is relatively small (i.e.,

less than 10 seconds). We note that both the waiting time and cruising time are minimized at the optimal offset, i.e., 4 seconds.

In summary, this dissertation develops various RL-based approaches for the optimal decision-making of distributed agents, including a modified MDP approach, a mean field multi-agent actor-critic algorithm, and a flow-dependent deep Q-learning approach. A bilevel optimization model is then formulated for both the task of passenger-seeking and route choice. The bilevel model is extensively applied to real-world large-scale cases and its effectiveness is disclosed. In addition, this dissertation unveils the underlying reward function of the e-hailing driver population by an inverse reinforcement learning algorithm, which paves the way for future research on discovering the underlying reward mechanism in a complex and dynamic ride-hailing market.

5.2 Future research directions

In this section, we point out several future research directions which can be explored to overcome some limitations of this research.

5.2.1 Bounded rationality

In this dissertation, agents are assumed to be intelligent and perfectly rational, meaning that agents would always choose the action with the best possible reward. However, bounded rationality [99, 158] suggests agents may not switch from their current policy to an alternative policy with a higher reward if the difference in reward is not large enough. Therefore, bounded rationality could be studied to more realistically reflect the boundedly rational behavior of agents/travelers.

5.2.2 Inverse reinforcement learning

A more personalized reward function can be investigated and is expected to incorporate more factors such as safety. Coupled with some prior knowledge, a more powerful IRL approach (such as the maximum entropy IRL [52]) can be a promising way to uncover more information of the underlying reward function that results in the demonstrated behavior of travelers.

5.2.3 Demand modeling

In this dissertation, the demand, i.e., passenger requests in passenger-seeking and travelers in route choice, is either modeled separately from the supply or assumed to be invariable. The demand elasticity is thus not captured. For example, passenger requests may vary dynamically with the change in the taxi fare, and travelers may use other travel modes instead of driving when congestion pricing or some traffic signal control mechanism is in place. Therefore, demand elasticity can be studied to reveal its impact on the performance of the derived optimal operational measures. In addition, passengers may be willing to wait for a few time steps to get a ride. Such behavior could be modeled by some queuing theory.

5.2.4 Upper-level control

In this dissertation, a predefined form of the control with a parameter α (e.g., toll charge in passenger-seeking and signal control in route choice) was used in the upper level, meaning that the upper level aims to find the optimal control within the space defined by the given form of the control. A more game-theoretical approach such as the leader-follower game may relax this restriction and find a globally optimal control.

References

[1] K. Lin, R. Zhao, Z. Xu, and J. Zhou, "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*, ser. KDD '18, New York, NY, USA: ACM, 2018, pp. 1774–1783.

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

[3] X. Di and X. J. Ban, "A unified equilibrium framework of new shared mobility systems," *Transportation Research Part B: Methodological*, vol. 129, pp. 50–78, 2019.

[4] D. A. Hennessy, D. L. Wiesenthal, and P. M. Kohn, "The Influence of Traffic Congestion, Daily Hassles, and Trait Stress Susceptibility on State Driver Stress: An Interactive Perspective1," *Journal of Applied Biobehavioral Research*, vol. 5, no. 2, pp. 162–179, 2000.

[5] T. Cheng, G. Tanaksaranond, C. Brunsdon, and J. Haworth, "Exploratory visualisation of congestion evolutions on urban transport networks," *Transportation Research Part C: Emerging Technologies*, vol. 36, pp. 296–306, Nov. 2013.

[6] H. Rong, X. Zhou, C. Yang, Z. Shafiq, and A. Liu, "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*, ser. CIKM '16, New York, NY, USA: ACM, 2016, pp. 2329–2334.

[7] T. Verma, P. Varakantham, S. Kraus, and H. C. Lau, "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*, pp. 409–417, Jun. 2017.

[8] X. Yu, S. Gao, X. Hu, and H. Park, "A Markov decision process approach to vacant taxi routing with e-hailing," *Transportation Research Part B: Methodological*, vol. 121, pp. 114–134, Mar. 2019.

[9] H. Yang, S. C. Wong, and K. I. Wong, "Demand-supply equilibrium of taxi services in a network under competition and regulation," *Transportation Research Part B: Methodological*, vol. 36, no. 9, pp. 799–819, Nov. 2002.

[10] H. Yang, C. S. Fung, K. I. Wong, and S. C. Wong, "Nonlinear pricing of taxi services," *Transportation Research Part A: Policy and Practice*, vol. 44, no. 5, pp. 337–348, Jun. 2010.

[11] F. He, X. Wang, X. Lin, and X. Tang, "Pricing and penalty/compensation strategies of a taxi-hailing platform," *Transportation Research Part C: Emerging Technologies*, vol. 86, pp. 263–279, Jan. 2018.

[12] M. Battifarano and Z. S. Qian, "Predicting real-time surge pricing of ride-sourcing companies," *Transportation Research Part C: Emerging Technologies*, vol. 107, pp. 444–462, Oct. 2019.

[13] H. Yang and S. C Wong, "A network model of urban taxi services," *Transportation Research Part B: Methodological*, vol. 32, no. 4, pp. 235–246, May 1998.

[14] S. Wong and H. Yang, "Network Model of Urban Taxi Services: Improved Algorithm," *Transportation Research Record: Journal of the Transportation Research Board*, vol. 1623, pp. 27–30, Jan. 1998.

[15] K. I. Wong, S. C. Wong, and H. Yang, "Modeling urban taxi services in congested road networks with elastic demand," *Transportation Research Part B: Methodological*, vol. 35, no. 9, pp. 819–842, Nov. 2001.

[16] K. I. Wong, S. C. Wong, H. Yang, and J. H. Wu, "Modeling urban taxi services with multiple user classes and vehicle modes," *Transportation Research Part B: Methodological*, vol. 42, no. 10, pp. 985–1007, Dec. 2008.

[17] H. Yang, C. W. Y. Leung, S. C. Wong, and M. G. H. Bell, "Equilibria of bilateral taxi–customer searching and meeting on networks," *Transportation Research Part B: Methodological*, vol. 44, no. 8, pp. 1067–1083, Sep. 2010.

[18] H. Yang and T. Yang, "Equilibrium properties of taxi markets with search frictions," *Transportation Research Part B: Methodological*, vol. 45, no. 4, pp. 696–713, May 2011.

[19] T. Yang, H. Yang, and S. C. Wong, "Taxi services with search frictions and congestion externalities," *Journal of Advanced Transportation*, vol. 48, no. 6, pp. 575–587, Oct. 2014.

[20] X. Di and X. J. Ban, "A unified equilibrium framework of new shared mobility systems," *Transportation Research Part B: Methodological*, vol. 129, pp. 50–78, Nov. 2019.

[21] L. Liu, C. Andris, and C. Ratti, "Uncovering cabdrivers' behavior patterns from their digital traces," *Computers, Environment and Urban Systems*, GeoVisualization and the Digital City, vol. 34, no. 6, pp. 541–548, Nov. 2010.

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

[23] W. Y. Szeto, R. C. P. Wong, S. C. Wong, and H. Yang, "A time-dependent logit-based taxi customer-search model," *International Journal of Urban Sciences*, vol. 17, no. 2, pp. 184–198, Jul. 2013.

[24] R. Sirisoma, S. C. Wong, W. H. K. Lam, D. Wang, H. Yang, and P. Zhang, "Empirical evidence for taxi customer-search model," *Proceedings Of The Institution Of Civil Engineers: Transport*, vol. 163, no. 4, pp. 203–210, 2010.

[25] R. C. P. Wong, W. Y. Szeto, and S. C. Wong, "A cell-based logit-opportunity taxi customer-search model," *Transportation Research Part C: Emerging Technologies*, vol. 48, pp. 84–96, Nov. 2014.

[26] R. C. P. Wong, W. Y. Szeto, S. C. Wong, and H. Yang, "Modelling multi-period customer-searching behaviour of taxi drivers," *Transportmetrica B: Transport Dynamics*, vol. 2, no. 1, pp. 40–59, Jan. 2014.

[27] R. C. P. Wong, W. Y. Szeto, and S. C. Wong, "Sequential Logit Approach to Modeling the Customer-Search Decisions of Taxi Drivers," *Asian Transport Studies*, vol. 3, no. 4, pp. 398–415, 2015.

[28] R. C. P. Wong, W. Y. Szeto, and S. C. Wong, "A two-stage approach to modeling vacant taxi movements," *Transportation Research Part C: Emerging Technologies*, Special Issue on International Symposium on Transportation and Traffic Theory, vol. 59, pp. 147–163, Oct. 2015.

[29] K. I. Wong, S. C. Wong, M. G. H. Bell, and H. Yang, "Modeling the bilateral micro-searching behavior for urban taxi services using the absorbing markov chain approach," *Journal of Advanced Transportation*, vol. 39, no. 1, pp. 81–104, 2005.

[30] X. Hu, S. Gao, Y.-C. Chiu, and D.-Y. Lin, "Modeling routing behavior for vacant taxicabs in urban traffic networks," *Transportation Research Record*, vol. 2284, no. 1, pp. 81–88, Jan. 2012.

[31] J. Tang, H. Jiang, Z. Li, M. Li, F. Liu, and Y. Wang, "A two-layer model for taxi customer searching behaviors using gps trajectory data," *IEEE Transactions on Intelligent Transportation Systems*, vol. 17, no. 11, pp. 3318–3324, 2016.

[32] K. Zhang, Y. Chen, and Y. M. Nie, "Hunting image: Taxi search strategy recognition using Sparse Subspace Clustering," *Transportation Research Part C: Emerging Technologies*, vol. 109, pp. 250–266, Dec. 2019.

[33] S. Phithakkitnukoon, M. Veloso, C. Bento, A. Biderman, and C. Ratti, "Taxi-Aware Map: Identifying and Predicting Vacant Taxis in the City," in *Ambient Intelligence*, ser. Lecture Notes in Computer Science, Springer, Berlin, Heidelberg, Nov. 2010, pp. 86–95.

[34] L. Moreira-Matias, J. Gama, M. Ferreira, and L. Damas, "A predictive model for the passenger demand on a taxi network," in *2012 15th International IEEE Conference on Intelligent Transportation Systems*, Sep. 2012, pp. 1014–1019.

[35] I. Markou, K. Kaiser, and F. C. Pereira, "Predicting taxi demand hotspots using automated Internet Search Queries," *Transportation Research Part C: Emerging Technologies*, vol. 102, pp. 73–86, May 2019.

[36] Y. Wang, J. Gu, S. Wang, and J. Wang, "Understanding consumers' willingness to use ridesharing services: The roles of perceived value and perceived risk," *Transportation Research Part C: Emerging Technologies*, vol. 105, pp. 504–519, Aug. 2019.

[37] F. Alemi, G. Circella, P. Mokhtarian, and S. Handy, "What drives the use of ridehailing in California? Ordered probit models of the usage frequency of Uber and Lyft," *Transportation Research Part C: Emerging Technologies*, vol. 102, pp. 233–248, May 2019.

[38] X. Tan, J. Zhang, W. Cao, J. Li, and Y. Zheng, "When will you arrive? estimating travel time based on deep neural networks," in *AAAI*, 2018.

[39] K. Zhang, N. Jia, L. Zheng, and Z. Liu, "A novel generative adversarial network for estimation of trip travel time distribution with trajectory data," *Transportation Research Part C: Emerging Technologies*, vol. 108, pp. 223–244, Nov. 2019.

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

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

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

[43] J. Yuan, Y. Zheng, L. Zhang, X. Xie, and G. Sun, "Where to Find My Next Passenger," in *Proceedings of the 13th International Conference on Ubiquitous Computing*, ser. UbiComp '11, New York, NY, USA: ACM, 2011, pp. 109–118.

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

[45] X. Zhou, H. Rong, C. Yang, Q. Zhang, A. V. Khezerlou, H. Zheng, M. Z. Shafiq, and A. X. Liu, "Optimizing Taxi Driver Profit Efficiency: A Spatial Network-based Markov Decision Process Approach," *IEEE Transactions on Big Data*, pp. 1–1, 2018.

[46] Y. Gao, D. Jiang, and Y. Xu, "Optimize taxi driving strategies based on reinforcement learning," *International Journal of Geographical Information Science*, vol. 32, no. 8, pp. 1677–1696, Aug. 2018.

[47] R. S. Sutton and A. G. Barto, *Introduction to Reinforcement Learning*, 1st. Cambridge, MA, USA: MIT Press, 1998.

[48] F. He and Z.-J. M. Shen, "Modeling taxi services with smartphone-based e-hailing applications," *Transportation Research Part C: Emerging Technologies*, vol. 58, pp. 93–106, Sep. 2015.

[49] X. Qian and S. V. Ukkusuri, "Taxi market equilibrium with third-party hailing service," *Transportation Research Part B: Methodological*, vol. 100, pp. 43–63, Jun. 2017.

[50] Z. Xu, Z. Li, Q. Guan, D. Zhang, Q. Li, J. Nan, C. Liu, W. Bian, and J. Ye, "Large-Scale Order Dispatch in On-Demand Ride-Hailing Platforms: A Learning and Planning Approach," in *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, ser. KDD '18, New York, NY, USA: ACM, 2018, pp. 905–913.

[51] D. P. Bertsekas, *Dynamic Programming and Optimal Control*, 2nd. Athena Scientific, 2000, ISBN: 1886529094.

[52] B. D. Ziebart, A. Maas, J. A. Bagnell, and A. K. Dey, "Maximum Entropy Inverse Reinforcement Learning," in *Proceedings of the 23rd National Conference on Artificial Intelligence - Volume 3*, ser. AAAI'08, Chicago, Illinois: AAAI Press, 2008, pp. 1433–1438.

[53] A. Y. Ng and S. J. Russell, "Algorithms for Inverse Reinforcement Learning," in *Proceedings of the Seventeenth International Conference on Machine Learning*, ser. ICML '00, San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 2000, pp. 663–670.

[54] S. Liu, M. Araujo, E. Brunskill, R. Rossetti, J. Barros, and R. Krishnan, "Understanding Sequential Decisions via Inverse Reinforcement Learning," in *2013 IEEE 14th International Conference on Mobile Data Management*, vol. 1, Jun. 2013, pp. 177–186.

[55] X. Di, H. X. Liu, and G. A. Davis, "Hybrid extended kalman filtering approach for traffic density estimation along signalized arterials: Use of global positioning system data," *Transportation Research Record*, vol. 2188, no. 1, pp. 165–173, 2010.

[56] X. Di, H. X. Liu, S. Zhu, and D. M. Levinson, "Indifference bands for boundedly rational route switching," *Transportation*, vol. 44, 1169–1194, 2017.

[57] Z. Shou and X. Di, "Similarity analysis of frequent sequential activity pattern mining," *Transportation Research Part C: Emerging Technologies*, vol. 96, pp. 122–143, Nov. 2018.

[58] Q. Ma, H. Yang, H. Zhang, K. Xie, and Z. Wang, "Modeling and Analysis of Daily Driving Patterns of Taxis in Reshuffled Ride-Hailing Service Market," *Journal of Transportation Engineering, Part A: Systems*, vol. 145, no. 10, p. 04 019 045, Oct. 2019.

[59] K. O'Keeffe, S. Anklesaria, P. Santo, and C. Ratti, "Using reinforcement learning to minimize taxi idle times," *arXiv:1910.11918 [physics]*, Oct. 2019.

[60] V. Pandey and S. D. Boyles, "Multiagent Reinforcement Learning Algorithm for Distributed Dynamic Pricing of Managed Lanes," in *2018 21st International Conference on Intelligent Transportation Systems (ITSC)*, Nov. 2018, pp. 2346–2351.

[61] V. Pandey, E. Wang, and S. D. Boyles, "Deep Reinforcement Learning Algorithm for Dynamic Pricing of Express Lanes with Multiple Access Locations," Sep. 2019.

[62] A. L. C. Bazzan and R. Grunitzki, "A multiagent reinforcement learning approach to en-route trip building," in *2016 International Joint Conference on Neural Networks (IJCNN)*, Jul. 2016, pp. 5288–5295.

[63] M. Nazari, A. Oroojlooy, L. V. Snyder, and M. Takáč, "Reinforcement Learning for Solving the Vehicle Routing Problem," Feb. 2018.

[64] B. Peng, J. Wang, and Z. Zhang, "A Deep Reinforcement Learning Algorithm Using Dynamic Attention Model for Vehicle Routing Problems," Feb. 2020.

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

[66] X. Tang, Z. T. Qin, F. Zhang, Z. Wang, Z. Xu, Y. Ma, H. Zhu, and J. Ye, "A Deep Value-network Based Approach for Multi-Driver Order Dispatching," in *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, ser. KDD '19, Anchorage, AK, USA: Association for Computing Machinery, Jul. 2019, pp. 1780–1790.

[67] J. Ke, F. Xiao, H. Yang, and J. Ye, "Optimizing Online Matching for Ride-Sourcing Services with Multi-Agent Deep Reinforcement Learning," *arXiv:1902.06228 [cs]*, Feb. 2019.

[68] M. Zhou, J. Jin, W. Zhang, Z. Qin, Y. Jiao, C. Wang, G. Wu, Y. Yu, and J. Ye, "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*, ser. CIKM '19, event-place: Beijing, China, New York, NY, USA: ACM, 2019, pp. 2645–2653, ISBN: 978-1-4503-6976-3.

[69] Y. Yang, X. Wang, Y. Xu, and Q. Huang, *Multiagent Reinforcement Learning-Based Taxi Predispatching Model to Balance Taxi Supply and Demand*, Research Article, ISSN: 0197-6729 Library Catalog: www.hindawi.com Pages: e8674512 Publisher: Hindawi Volume: 2020, 2020.

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

[71] L. Buşoniu, R. Babuška, and B. De Schutter, "Multi-agent Reinforcement Learning: An Overview," in *Innovations in Multi-Agent Systems and Applications - 1*, ser. Studies in Computational Intelligence, D. Srinivasan and L. C. Jain, Eds., Berlin, Heidelberg: Springer Berlin Heidelberg, 2010, pp. 183–221.

[72] D. Silver, A. Huang, C. J. Maddison, A. Guez, L. Sifre, G. van den Driessche, J. Schrittwieser, I. Antonoglou, V. Panneershelvam, M. Lanctot, S. Dieleman, D. Grewe, J. Nham, N. Kalchbrenner, I. Sutskever, T. Lillicrap, M. Leach, K. Kavukcuoglu, T. Graepel, and D. Hassabis, "Mastering the game of Go with deep neural networks and tree search," *Nature*, vol. 529, no. 7587, pp. 484–489, Jan. 2016.

[73] D. Silver, J. Schrittwieser, K. Simonyan, I. Antonoglou, A. Huang, A. Guez, T. Hubert, L. Baker, M. Lai, A. Bolton, Y. Chen, T. Lillicrap, F. Hui, L. Sifre, G. van den Driessche, T. Graepel, and D. Hassabis, "Mastering the game of Go without human knowledge," *Nature*, vol. 550, no. 7676, pp. 354–359, Oct. 2017.

[74] N. Brown and T. Sandholm, "Superhuman AI for heads-up no-limit poker: Libratus beats top professionals," *Science*, vol. 359, no. 6374, pp. 418–424, Jan. 2018.

[75] N. Brown and T. Sandholm, "Superhuman AI for multiplayer poker," *Science*, vol. 365, no. 6456, pp. 885–890, Aug. 2019.

[76] OpenAI, *Openai five*, `https://blog.openai.com/openai-five/`, 2018.

[77] O. Vinyals, I. Babuschkin, W. M. Czarnecki, M. Mathieu, A. Dudzik, J. Chung, D. H. Choi, R. Powell, T. Ewalds, P. Georgiev, J. Oh, D. Horgan, M. Kroiss, I. Danihelka, A. Huang, L. Sifre, T. Cai, J. P. Agapiou, M. Jaderberg, A. S. Vezhnevets, R. Leblond, T. Pohlen, V. Dalibard, D. Budden, Y. Sulsky, J. Molloy, T. L. Paine, C. Gulcehre, Z. Wang, T. Pfaff, Y. Wu, R. Ring, D. Yogatama, D. Wünsch, K. McKinney, O. Smith, T. Schaul, T. Lillicrap, K. Kavukcuoglu, D. Hassabis, C. Apps, and D. Silver, "Grandmaster level in StarCraft II

using multi-agent reinforcement learning," *Nature*, vol. 575, no. 7782, pp. 350–354, Nov. 2019.

[78] K. Zhang, Z. Yang, and T. Başar, "Multi-Agent Reinforcement Learning: A Selective Overview of Theories and Algorithms," Nov. 2019.

[79] A. Tampuu, T. Matiisen, D. Kodelja, I. Kuzovkin, K. Korjus, J. Aru, J. Aru, and R. Vicente, "Multiagent cooperation and competition with deep reinforcement learning," *PLOS ONE*, vol. 12, no. 4, e0172395, Apr. 2017.

[80] L. Matignon, G. J. Laurent, and N. L. Fort-Piat, "Independent reinforcement learners in cooperative Markov games: A survey regarding coordination problems," *The Knowledge Engineering Review*, vol. 27, no. 1, pp. 1–31, Feb. 2012.

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

[82] M. Tan, "Multi-Agent Reinforcement Learning: Independent vs. Cooperative Agents," in *In Proceedings of the Tenth International Conference on Machine Learning*, Morgan Kaufmann, 1993, pp. 330–337.

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

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

[85] Y. Yang, R. Luo, M. Li, M. Zhou, W. Zhang, and J. Wang, "Mean Field Multi-Agent Reinforcement Learning," in *International Conference on Machine Learning*, Jul. 2018, pp. 5571–5580.

[86] J. Jin, M. Zhou, W. Zhang, M. Li, Z. Guo, Z. Qin, Y. Jiao, X. Tang, C. Wang, J. Wang, G. Wu, and J. Ye, "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*, ser. CIKM '19, event-place: Beijing, China, New York, NY, USA: ACM, 2019, pp. 1983–1992, ISBN: 978-1-4503-6976-3.

[87] H. Yang and M. G. H. Bell, "Models and algorithms for road network design: A review and some new developments," *Transport Reviews*, vol. 18, no. 3, pp. 257–278, 1998.

[88] X. Zhang and H. Yang, "The optimal cordon-based network congestion pricing problem," *Transportation Research Part B: Methodological*, vol. 38, no. 6, pp. 517–537, 2004.

[89] Q. Meng and Z. Liu, "Impact analysis of cordon-based congestion pricing on mode-split for a bimodal transportation network," *Transportation Research Part C: Emerging Technologies*, vol. 21, no. 1, pp. 134–147, 2012.

[90] X. Di, X. He, X. Guo, and H. X. Liu, "Braess paradox under the boundedly rational user equilibria," *Transportation Research Part B: Methodological*, vol. 67, pp. 86–108, 2014.

[91] X. Di, H. X. Liu, and X. J. Ban, "Second best toll pricing within the framework of bounded rationality," *Transportation Research Part B*, vol. 83, pp. 74–90, 2016.

[92] X. Di, H. X. Liu, X. Ban, and H. Yang, "Ridesharing user equilibrium and its implications for high-occupancy toll lane pricing," *Transportation Research Record*, vol. 2667, no. 1, pp. 39–50, 2017.

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

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

[95] V. R. Konda and J. N. Tsitsiklis, "On Actor-Critic Algorithms," *SIAM J. Control Optim.*, vol. 42, no. 4, pp. 1143–1166, Apr. 2003.

[96] I. Grondman, L. Busoniu, G. A. D. Lopes, and R. Babuska, "A Survey of Actor-Critic Reinforcement Learning: Standard and Natural Policy Gradients," *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)*, vol. 42, no. 6, pp. 1291–1307, Nov. 2012.

[97] V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. Riedmiller, A. K. Fidjeland, G. Ostrovski, S. Petersen, C. Beattie, A. Sadik, I. Antonoglou, H. King, D. Kumaran, D. Wierstra, S. Legg, and D. Hassabis, "Human-level control through deep reinforcement learning," *Nature*, vol. 518, no. 7540, pp. 529–533, Feb. 2015.

[98] R. S. Sutton, D. McAllester, S. Singh, and Y. Mansour, "Policy Gradient Methods for Reinforcement Learning with Function Approximation," in *Proceedings of the 12th International Conference on Neural Information Processing Systems*, ser. NIPS'99, event-place: Denver, CO, Cambridge, MA, USA: MIT Press, 1999, pp. 1057–1063.

[99] X. Di, H. X. Liu, J.-S. Pang, and X. J. Ban, "Boundedly rational user equilibria (BRUE): Mathematical formulation and solution sets," *Transportation Research Part B: Methodological*, vol. 57, pp. 300–313, Nov. 2013.

[100] X. Di and H. X. Liu, "Boundedly rational route choice behavior: A review of models and methodologies," *Transportation Research Part B: Methodological*, vol. 85, pp. 142–179, Mar. 2016.

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

[102] J. Bergstra and Y. Bengio, "Random search for hyper-parameter optimization," *The Journal of Machine Learning Research*, vol. 13, no. null, pp. 281–305, Feb. 2012.

[103] J. Mockus, *Bayesian Approach to Global Optimization: Theory and Applications*, ser. Mathematics and its Applications. Springer Netherlands, 1989.

[104] K. P. Murphy, *Machine Learning: A Probabilistic Perspective*. MIT Press, Aug. 2012.

[105] N. Srinivas, A. Krause, S. Kakade, and M. Seeger, "Gaussian process optimization in the bandit setting: No regret and experimental design," in *Proceedings of the 27th International Conference on International Conference on Machine Learning*, ser. ICML'10, Haifa, Israel: Omnipress, Jun. 2010, pp. 1015–1022.

[106] E. Brochu, V. M. Cora, and N. de Freitas, "A Tutorial on Bayesian Optimization of Expensive Cost Functions, with Application to Active User Modeling and Hierarchical Reinforcement Learning," *arXiv:1012.2599 [cs]*, Dec. 2010.

[107] F. Berkenkamp, A. P. Schoellig, and A. Krause, "No-Regret Bayesian Optimization with Unknown Hyperparameters," *arXiv:1901.03357 [cs, stat]*, Apr. 2019.

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

[109] A. de Palma and R. Lindsey, "Traffic congestion pricing methodologies and technologies," *Transportation Research Part C: Emerging Technologies*, vol. 19, no. 6, pp. 1377–1399, Dec. 2011.

[110] B. Schaller, "Elasticities for taxicab fares and service availability," *Transportation*, vol. 26, no. 3, pp. 283–297, Aug. 1999.

[111] A. Plitt. (2020). The new york city subway, explained, (visited on 02/26/2020).

[112] Seongmoon Kim, M. E. Lewis, and C. C. White, "Optimal vehicle routing with real-time traffic information," *IEEE Transactions on Intelligent Transportation Systems*, vol. 6, no. 2, pp. 178–188, Jun. 2005.

[113] G. Kim, Y. S. Ong, T. Cheong, and P. S. Tan, "Solving the Dynamic Vehicle Routing Problem Under Traffic Congestion," *IEEE Transactions on Intelligent Transportation Systems*, vol. 17, no. 8, pp. 2367–2380, Aug. 2016.

[114] S. Gao, E. Frejinger, and M. Ben-Akiva, "Adaptive route choices in risky traffic networks: A prospect theory approach," *Transportation research part C: emerging technologies*, vol. 18, no. 5, pp. 727–740, 2010.

[115] H. Xu, Y. Lou, Y. Yin, and J. Zhou, "A prospect-based user equilibrium model with endogenous reference points and its application in congestion pricing," *Transportation Research Part B: Methodological*, vol. 45, no. 2, pp. 311–328, 2011.

[116] J. Yang and G. Jiang, "Development of an enhanced route choice model based on cumulative prospect theory," *Transportation Research Part C: Emerging Technologies*, vol. 47, pp. 168–178, 2014.

[117] C. Zhang, T.-L. Liu, H.-J. Huang, and J. Chen, "A cumulative prospect theory approach to commuters' day-to-day route-choice modeling with friends' travel information," *Transportation Research Part C: Emerging Technologies*, vol. 86, pp. 527–548, 2018.

[118] G. de Moraes Ramos, W. Daamen, and S. Hoogendoorn, "Expected utility theory, prospect theory, and regret theory compared for prediction of route choice behavior," *Transportation Research Record*, vol. 2230, no. 1, pp. 19–28, 2011.

[119] T. L. Friesz, J. Luque, R. L. Tobin, and B.-W. Wie, "Dynamic network traffic assignment considered as a continuous time optimal control problem," *Operations Research*, vol. 37, no. 6, pp. 893–901, 1989.

[120] S. Peeta and A. K. Ziliaskopoulos, "Foundations of dynamic traffic assignment: The past, the present and the future," *Networks and spatial economics*, vol. 1, no. 3-4, pp. 233–265, 2001.

[121] X. Nie and H. M. Zhang, "A comparative study of some macroscopic link models used in dynamic traffic assignment," *Networks and Spatial Economics*, vol. 5, no. 1, pp. 89–115, 2005.

[122] T. L. Friesz, K. Han, P. A. Neto, A. Meimand, and T. Yao, "Dynamic user equilibrium based on a hydrodynamic model," *Transportation Research Part B: Methodological*, vol. 47, pp. 102–126, 2013.

[123] M. C. J. Bliemer, M. P. H. Raadsen, L. J. N. Brederode, M. G. H. Bell, L. J. J. Wismans, and M. J. Smith, "Genetics of traffic assignment models for strategic transport planning," *Transport Reviews*, vol. 37, no. 1, pp. 56–78, Jan. 2017.

[124] Y. Yu, K. Han, and W. Ochieng, "Day-to-day dynamic traffic assignment with imperfect information, bounded rationality and information sharing," *Transportation Research Part C: Emerging Technologies*, vol. 114, pp. 59–83, May 2020.

[125] J. Hu and M. P. Wellman, "Multiagent Reinforcement Learning: Theoretical Framework and an Algorithm," in *Proceedings of the Fifteenth International Conference on Machine Learning*, ser. ICML '98, San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., Jul. 1998, pp. 242–250.

[126] A. Prasad and I. Dusparic, "Multi-agent Deep Reinforcement Learning for Zero Energy Communities," in *2019 IEEE PES Innovative Smart Grid Technologies Europe (ISGT-Europe)*, Sep. 2019, pp. 1–5.

[127] S. Kumar, P. Shah, D. Hakkani-Tur, and L. Heck, "Federated Control with Hierarchical Multi-Agent Deep Reinforcement Learning," *arXiv:1712.08266 [cs]*, Dec. 2017.

[128] J. Z. Leibo, V. Zambaldi, M. Lanctot, J. Marecki, and T. Graepel, "Multi-agent Reinforcement Learning in Sequential Social Dilemmas," *arXiv:1702.03037 [cs]*, Feb. 2017.

[129] T. T. Nguyen, N. D. Nguyen, and S. Nahavandi, "Deep Reinforcement Learning for Multi-agent Systems: A Review of Challenges, Solutions, and Applications," *IEEE Transactions on Cybernetics*, vol. 50, no. 9, pp. 3826–3839, Sep. 2020.

[130] A. L. C. Bazzan and F. Klügl, "Re-routing Agents in an Abstract Traffic Scenario," in *Advances in Artificial Intelligence - SBIA 2008*, G. Zaverucha and A. L. da Costa, Eds., ser. Lecture Notes in Computer Science, Berlin, Heidelberg: Springer, 2008, pp. 63–72.

[131] B. Bakker, S. Whiteson, L. Kester, and F. C. A. Groen, "Traffic Light Control by Multiagent Reinforcement Learning Systems," in *Interactive Collaborative Information Systems*, ser. Studies in Computational Intelligence, R. Babuška and F. C. A. Groen, Eds., Berlin, Heidelberg: Springer, 2010, pp. 475–510, ISBN: 978-3-642-11688-9.

[132] C. Chen, H. Wei, N. Xu, G. Zheng, M. Yang, Y. Xiong, K. Xu, and Z. Li, "Toward A Thousand Lights: Decentralized Deep Reinforcement Learning for Large-Scale Traffic Signal Control," *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 34, no. 04, pp. 3414–3421, Apr. 2020, Number: 04.

[133] Z. Shou and X. Di, "Reward Design for Driver Repositioning Using Multi-Agent Reinforcement Learning," *arXiv:2002.06723 [cs, stat]*, Feb. 2020, arXiv: 2002.06723.

[134] P. Palanisamy, "Multi-Agent Connected Autonomous Driving using Deep Reinforcement Learning," *arXiv:1911.04175 [cs, stat]*, Nov. 2019, arXiv: 1911.04175.

[135] S. Bhalla, S. Ganapathi Subramanian, and M. Crowley, "Deep Multi Agent Reinforcement Learning for Autonomous Driving," in *Advances in Artificial Intelligence*, C. Goutte and X. Zhu, Eds., ser. Lecture Notes in Computer Science, Cham: Springer International Publishing, 2020, pp. 67–78.

[136] F. Stefanello, B. C. d. Silva, and A. L. C. Bazzan, "Using Topological Statistics to Bias and Accelerate Route Choice: Preliminary Findings in Synthetic and Real-World Road Networks," in *ATT@IJCAI*, 2016.

[137] G. d. O. Ramos, A. L. C. Bazzan, and B. C. da Silva, "Analysing the impact of travel information for minimising the regret of route choice," *Transportation Research Part C: Emerging Technologies*, vol. 88, pp. 257–271, Mar. 2018.

[138] B. Zhou, Q. Song, Z. Zhao, and T. Liu, "A reinforcement learning scheme for the equilibrium of the in-vehicle route choice problem based on congestion game," *Applied Mathematics and Computation*, vol. 371, p. 124 895, Apr. 2020.

[139] R. Grunitzki, G. d. O. Ramos, and A. L. C. Bazzan, "Individual versus Difference Rewards on Reinforcement Learning for Route Choice," in *2014 Brazilian Conference on Intelligent Systems*, Oct. 2014, pp. 253–258.

[140] C. Mao and Z. Shen, "A reinforcement learning framework for the adaptive routing problem in stochastic time-dependent network," *Transportation Research Part C: Emerging Technologies*, vol. 93, pp. 179–197, Aug. 2018.

[141] D. K. Merchant and G. L. Nemhauser, "A Model and an Algorithm for the Dynamic Traffic Assignment Problems," *Transportation Science*, vol. 12, no. 3, pp. 183–199, 1978.

[142] D. K. Merchant and G. L. Nemhauser, "Optimality Conditions for a Dynamic Traffic Assignment Model," *Transportation Science*, vol. 12, no. 3, pp. 200–207, Aug. 1978.

[143] M. Kuwahara and T. Akamatsu, "Decomposition of the reactive dynamic assignments with queues for a many-to-many origin-destination pattern," *Transportation Research Part B: Methodological*, vol. 31, no. 1, pp. 1–10, Feb. 1997.

[144] X. J. Ban, J.-S. Pang, H. X. Liu, and R. Ma, "Continuous-time point-queue models in dynamic network loading," *Transportation Research Part B: Methodological*, vol. 46, no. 3, pp. 360–380, Mar. 2012.

[145] C. F. Daganzo, "The cell transmission model: A dynamic representation of highway traffic consistent with the hydrodynamic theory," *Transportation Research Part B: Methodological*, vol. 28, no. 4, pp. 269–287, Aug. 1994.

[146] C. F. Daganzo, "The cell transmission model, part II: Network traffic," *Transportation Research Part B: Methodological*, vol. 29, no. 2, pp. 79–93, Apr. 1995.

[147] A. K. Ziliaskopoulos, "A Linear Programming Model for the Single Destination System Optimum Dynamic Traffic Assignment Problem," *Transportation Science*, vol. 34, no. 1, pp. 37–49, Feb. 2000, Publisher: INFORMS.

[148] T. L. Friesz and K. Han, "The mathematical foundations of dynamic user equilibrium," *Transportation research part B: methodological*, vol. 126, pp. 309–328, 2019.

[149] S. Peeta and C. Zhou, "Robustness of the off-line a priori stochastic dynamic traffic assignment solution for on-line operations," *Transportation Research Part C: Emerging Technologies*, vol. 7, no. 5, pp. 281–303, Oct. 1999.

[150] S. T. Waller and A. K. Ziliaskopoulos, "Stochastic Dynamic Network Design Problem," *Transportation Research Record*, vol. 1771, no. 1, pp. 106–113, Jan. 2001, Publisher: SAGE Publications Inc.

[151] A. Sumalee, R. X. Zhong, T. L. Pan, and W. Y. Szeto, "Stochastic cell transmission model (SCTM): A stochastic dynamic traffic model for traffic state surveillance and assignment," *Transportation Research Part B: Methodological*, vol. 45, no. 3, pp. 507–533, Mar. 2011.

[152] X. J. Ban, J.-S. Pang, H. X. Liu, and R. Ma, "Modeling and solving continuous-time instantaneous dynamic user equilibria: A differential complementarity systems approach," *Transportation Research Part B: Methodological*, vol. 46, no. 3, pp. 389–408, 2012.

[153] C. Gawron, "An Iterative Algorithm to Determine the Dynamic User Equilibrium in a Traffic Simulation Model," *International Journal of Modern Physics C*, vol. 09, no. 03, pp. 393–407, May 1998, Publisher: World Scientific Publishing Co.

[154] H. X. Liu, X. He, and X. Ban, "A Cell-Based Many-to-One Dynamic System Optimal Model and Its Heuristic Solution Method for Emergency Evacuation," Number: 07-2261, 2007.

[155] R. Steinberg and W. I. Zangwill, "The Prevalence of Braess' Paradox," *Transportation Science*, vol. 17, no. 3, pp. 301–318, Aug. 1983, Publisher: INFORMS.

[156] E. I. Pas and S. L. Principio, "Braess' paradox: Some new insights," *Transportation Research Part B: Methodological*, vol. 31, no. 3, pp. 265–276, Jun. 1997.

[157] H. Hu and H. X. Liu, "Arterial offset optimization using archived high-resolution traffic signal data," *Transportation Research Part C: Emerging Technologies*, vol. 37, pp. 131–144, Dec. 2013.

[158] R. Jayakrishnan, H. S. Mahmassani, and T.-Y. Hu, "An evaluation tool for advanced traffic information and management systems in urban networks," *Transportation Research Part C: Emerging Technologies*, vol. 2, no. 3, pp. 129–147, Sep. 1994.