Learn to Earn: Enabling Coordination Within a Ride-Hailing Fleet

Harshal A. Chaudhari
Dept. of Computer Science
Boston University
Boston, U.S.A
harshal@bu.edu

John W. Byers
Dept. of Computer Science
Boston University
Boston, U.S.A
byers@bu.edu

Evimaria Terzi
Dept. of Computer Science
Boston University
Boston, U.S.A
evimaria@bu.edu

2020 IEEE International Conference on Big Data (Big Data) | 978-1-7281-6251-5/20/$31.00 ©2020 IEEE | DOI: 10.1109/BigData50022.2020.9378416

***Abstract*—The problem of optimizing social welfare objectives on multi-sided ride-hailing platforms such as Uber, Lyft, etc., is challenging, due to misalignment of objectives between drivers, passengers, and the platform itself. An ideal solution aims to minimize the response time for each hyperlocal passenger ride request, while simultaneously maintaining high demand satisfaction and supply utilization across the entire city. Economists tend to rely on dynamic pricing mechanisms that stifle price-sensitive excess demand and resolve supply-demand imbalances that emerge in specific neighborhoods. In contrast, computer scientists primarily view it as a demand prediction problem with the goal of preemptively repositioning supply to such neighborhoods using *black-box* coordinated multi-agent deep reinforcement learning-based approaches. Here, we introduce explainability in the existing supply-repositioning approaches by establishing the need for coordination between the drivers at specific locations and times. Explicit need-based coordination allows our framework to use a simpler *non-deep* reinforcement learning-based approach, thereby enabling it to explain its recommendations *ex-post*. Moreover, it provides *envy-free* recommendations i.e., drivers at the same location and time do not envy one another's expected future earnings. Our experimental evaluation demonstrates the effectiveness, robustness, and generalizability of our framework. Finally, in contrast to previous works, we make available a reinforcement learning environment for *end-to-end* reproducibility of our work and to encourage future comparative studies.**

I. Introduction

Popular ride-hailing platforms, e.g., Uber, Lyft, Didi Chuxing, and Ola, have revolutionized the daily commute in cities across the world. Globally valued at over $61 billion and expected to grow to over $200 billion by 2025 these platforms operate as multi-sided marketplaces, seamlessly connecting drivers with riders through their smartphone applications [1]. The explosive growth of these ride-hailing platforms has motivated a wide array of questions for academic research at the intersection of computer science and economics, as we discuss in the related work section.

A large segment of these works aims to improve the performance of the platforms, ensuring high-reliability service for the passengers alongside high utilization and earnings for the drivers. The two main thrusts are *dynamic pricing* and *capacity repositioning*. Dynamic pricing [2]–[6] aims to balance demand and supply by increasing prices in certain neighborhoods. Intuitively, temporary increases in prices curtail price-sensitive demand and assist the platform in ensuring a high-reliability service. On the flip side, the potential for higher earnings also encourages more drivers to join the platform during such "price surges". The dynamic pricing literature uses game-theoretic analyses of the ride-hailing markets to show its effectiveness as a platform control mechanism.

The capacity repositioning approach aims to improve the platforms' performance by assisting drivers with recommendations for relocations inside a city. Although the initial work in this domain has focused on modeling the driver-repositioning problems as combinatorial optimization problems [7]–[11], the need for optimizing large driver fleets and the availability of high-dimensional historical data has recently led to the development of machine learning methods for the same problem [12]–[16]. Such approaches predominantly use multi-agent coordination reinforcement learning to solve a global capacity repositioning problem, using various forms of *attention mechanisms* in neural networks to achieve coordination. By design, they make a key assumption that there is *always* a need for coordination in the market. This assumption necessitates them to leverage recent breakthroughs in the scalability of deep learning models to exploit the high-dimensional historical data. Deep-learning based methods have a large number of hyperparameters required in their training, making their performance susceptible to external perturbations. Moreover, the *black-box* coordination issue, as yet unresolved, is a potential liability when using deep-learning based systems in domains such as this, where platform controllers would like to understand how the choices of recommendations to human drivers were made.

We aim to revise the capacity-repositioning approach by relaxing the assumption that coordination is always necessary. Specifically, our approach leverages the observation that driver actions are independent at most times of the day, with coordination required only during periodic times of peak demand, such as rush hours. Furthermore, the instances of supply-demand imbalances in a city are usually restricted to distinct neighborhoods. We exploit this loose spatio-temporal coupling of supply and demand to learn *when* and *where* the drivers need to coordinate, and otherwise act independently for the rest of the time. This observation allows us to combine vanilla reinforcement learning (i.e., not deep learning) algorithms with

simple combinatorial techniques for solving the repositioning problem. Moreover, our framework is scalable because the sizes of the combinatorial problems we need to solve in order to achieve capacity repositioning are constrained by the number of imbalanced neighborhoods. Broadly, our model is a combination of the combinatorial and machine-learning approaches to capacity repositioning.

As our framework does not rely on deep learning, we are able to explain *ex-post* all the recommendations given to the drivers, taking a step in direction of *transparent* AI proposed in the recent *General Data Protection Regulations* (GDPR) guidelines [17]. Moreover, our approach is *envy-free* in the sense that drivers at the same location and time do not envy one another's future earnings. The resulting model is relatively parameter-free and hence generalizes well in presence of daily variations in supply and demand. Finally, our framework is amenable to integration with any dynamic-pricing models by easily augmenting our data with the effects of such a model.

Contributions: To summarize, the contributions of our paper are the following:

- We consider the problem of capacity relocation on a ride-hailing platform in order to maximize welfare and propose a robust, explainable, and scalable framework that combines simple combinatorial techniques with vanilla reinforcement learning algorithms.
- We perform a thorough experimental evaluation of the dynamics of the fleet-management system, its effectiveness, robustness to imperfect hyperparameter tunings, and generalizability in the presence of external perturbations.
- We also make available an OpenAI gym environment[1] named `nyc-yellow-taxi-v0` at [19] so that any future multiagent reinforcement learning algorithm can be easily applied to this problem. To the best of our knowledge, this is the first environment based on real-world datasets.

II. Related Work

In this section, we discuss existing work in comparison to ours.

Driver recommender systems: The problem of spatio-temporal demand prediction to inform an idle taxi driver of favorable locations for passenger pickups had been studied extensively even before the advent of ride-hailing services. These works focus on the case of a self-interested individual acting in isolation. For instance, Li et al. [20] use a large-scale taxi GPS trace dataset to identify salient features associated with successful passenger pickup locations, while in two separate studies, Yuan et al. [21, 22] develop a recommender system to guide both idle taxi drivers and waiting passengers to convenient locations in order to optimize social welfare. More recently, Chaudhari et al. [23] devise driver-oriented strategies to recommend favorable driving schedules and pickup locations to optimize the earnings of an individual on-demand ride-sharing driver. In contrast to our approach, the recommender systems in these works are agnostic to driver interactions and may result in unfavorable supply excesses in certain locations when adopted by many drivers simultaneously.

Capacity repositioning systems: Traditional works in driver dispatch systems [7]–[9] typically rely on queuing-theoretical models to asymptotically optimize supply-demand matching while reducing congestion-related issues. Aided by the precise estimates of supply and demand in real-time, recent approaches [10, 11] leverage demand volume and ride destination forecasting models to use in combinatorial optimization techniques. These approaches do not scale well to use cases on contemporary platforms, where fleets of as many as 10,000 drivers serve a single city.

More recent work [12]–[16] addresses the scalability issue by using deep reinforcement learning to learn control policies in high-dimensional input space. While effective in this high-volume data domain, these methods rely on external proprietary models to generate inputs for the driver dispatch systems. For example, the approach of [14] heavily relies on a proprietary simulator built by Didi Chuxing in order to generate inputs required during model training, making it impossible to reproduce their results for comparison purposes. To the best of our knowledge, building such a simulator is itself an active research problem. Moreover, deep-learning based techniques suffer from a lack of explainability. Cognizant of these issues, our approach does not rely on any proprietary models but rather learns high-quality solutions *from scratch* based solely upon historically observed data. Moreover, we achieve that without sacrificing the explainability of the model. In the absence of the need for coordination, our model assumes homogeneity of drivers in the same location and provides *envy-free* recommendations while also making it scalable. Furthermore, we have publicly made available our entire codebase and the reinforcement learning environment required to reproduce every result presented in this work, thus enabling future comparative studies [19].

From the learning point of view, our approach not only learns *how* to coordinate but also *when* it is required to do so. This is achieved by augmenting vanilla reinforcement learning (in the form of tabular Q-learning) with combinatorial techniques to aid the rebalancing of driver distribution.

Platform economics: Studies of ride-hailing services as multi-sided economic marketplaces have investigated the impacts of the platform's pricing policies on the platform profits, the consumer surplus, and the driver wages [3, 5, 24, 25]. Sühr et al. [26] investigate fairness in driver earnings distribution using driver-passenger matchings optimized to attain income equality goals. Recently, Chen et al. [27] combines platform economics with the capacity repositioning problem using a contextual bandit framework. There is a growing body of literature studying the interplay between platform pricing and strategic driver behaviors, for which we refer the readers to [2]. Our work contributes to this domain by developing a scalable framework that can be used to verify the results of asymptotic dynamic pricing models via realistic simulations.

[1]OpenAI Gym is a toolkit for developing and comparing reinforcement learning algorithms [18].

III. Problem Setup

In this section, we describe the basics of our problem setup and provide the necessary notation.

A. City attributes

Throughout the paper, we assume that the city is divided into a set of m non-overlapping hexagonal zones denoted by $\mathcal{H}$. We also assume that time t advances in discrete time steps i.e., $\mathcal{T} = \{1, \cdots, T\}$, a standard industry practice [28]. Finally, we assume a total of n homogeneous drivers traveling between hexagon zones picking up and dropping off the passengers.

Our model uses the following city matrices and vectors that are time-varying; i.e., their entries change at every time step. However, for notational convenience, we do not introduce the time step t subscript unless required for context.

Demand matrix (D): A matrix $\mathbf{D} \in \mathbb{R}^{m \times m}$ such that each entry $d(h, h')$ denotes the number of passengers requesting a ride from zone h to zone h' at time t. With appropriately sized hexagonal city zones, we find that $\forall h \in \mathcal{H}, d(h, h) = 0$.

Travel time matrix (T): A matrix $\mathbf{T} \in \mathbb{R}^{m \times m}$ such that each entry $\tau(h, h')$ denotes the number of discrete time steps required for transiting from zone h to zone h'.

Reward matrix (R): A matrix $\mathbf{R} \in \mathbb{R}^{m \times m}$ such that each entry $r(h, h')$ denotes the net reward for a taxi driver carrying a passenger from zone h to zone h'. The net rewards include driver's earnings for delivering the passenger at the destination minus the sundries such as gas cost, vehicle depreciation, etc. Hence, each entry of the matrix is of form $r(h, h') = \text{earnings}(h, h') - \text{cost}(h, h')$.

Driver actions (A): At each time step t, a driver in zone h who is not currently on a trip can choose one of the two actions.

- **Wait:** A wait action $a(h, h)$ involves waiting for a passenger in the current zone i for the current time step. If successful, it can lead to a trip to some other zone h' with the driver earning a reward of $r(h, h')$. When the number of drivers choosing to wait in a zone exceeds the demand of the zone at the particular time, an unsuccessful wait may occur, and the driver earns a net reward of zero while staying in the same zone h for the next time step.
- **Relocate:** A relocate action $a(h, h')$ involves relocation *without a passenger* from zone h to zone h'. Undertaking a relocate action costs a driver a value denoted by $\text{cost}(h, h')$.

Thus, we consider a total of $|\mathbf{A}| = m^2$ actions. In case of a relocate action or a successful passenger pickup to zone h', the driver is busy traveling for next $\tau(h, h')$ time steps and is presented with the next action choice at time $t + \tau(h, h')$, while in case of an unsuccessful wait, the driver chooses the next action at time $t + 1$.

B. Model attributes

Using the city attributes from the previous section, we now define the attributes of our model:

Policy (π): A policy function $\pi : \mathcal{H} \times \mathcal{T} \rightarrow \mathbf{A}$ recommends the best action to drivers in every zone of the city at each time step, to maximize the model's objective function. We impose a constraint that all drivers in the same zone at the same time be recommended the same action unless driver coordination is required to resolve a supply-demand imbalance in the zone.

A driver i following a policy π performs location and time-dependent actions represented by a 3-tuple $\phi_t^\pi(i) = (t, h, a)$, where h and a are the location of the driver and the action chosen at time t respectively. We assume that if a driver is busy at time t, the corresponding 3-tuple is $(t, \varnothing, \varnothing)$.

Driver earnings ($\mathcal{E}$): Let function $E(t, h, a)$ denote the net earnings of a driver on taking action a at time t while located in zone h. If the action leads a driver to zone h', $E(t, h, a) = r(h, h')$. In the case of the relocate action, net earnings simply constitute the cost of relocation i.e., $r(h, h') = -\text{cost}(h, h')$. We can denote the gross earnings of n drivers following a policy π by: $\mathcal{E}^\pi(n, \mathbf{D}, \mathbf{T}, \mathbf{R}) = \sum_{t=1}^{T} \sum_{i=1}^{n} E\big(\phi_t^\pi(i)\big)$, where $E(t, \varnothing, \varnothing) = 0$.

Supply (S): A policy π induces the movement of drivers between different city zones through action choices. The supply, i.e., the number of drivers at zone h at time t induced by a policy π, is denoted using the supply function $S^\pi(t, h)$.

Demand fulfillment ($\mathcal{F}$): A driver in zone h choosing the wait action $a(h, h)$ at time t is randomly matched with any of the $\sum_{h'} d_t(h, h')$ passengers requesting a ride in zone h at the same time. Hence, a policy π, via its supply function, induces a demand fulfillment function. Demand fulfilled in zone h at time t when drivers follow a policy π is denoted using the demand satisfaction function $F^\pi(t, h)$. Obviously, $\forall_{\pi \in \Pi} F^\pi(t, h) \leq \sum_{h'} d_t(h, h')$. Hence, total demand fulfilled over the course of time steps $t \in \mathcal{T}$ by n drivers following the policy π can be given by: $\mathcal{F}^\pi(n, \mathbf{D}, \mathbf{T}, \mathbf{R}) = \sum_{t=1}^{T} \sum_{h=1}^{m} F^\pi(t, h)$.

C. Problem statement

Based on the above definitions, we now formulate the problem that we solve.

Problem 1(MaxEarnings): Given time-varying matrices $\mathbf{D}, \mathbf{T}, \mathbf{R}$ and the number of homogeneous drivers n, devise a policy π^* such that

$$\pi^* = \underset{\pi \in \Pi}{\arg\max}\, \mathcal{E}^\pi(n, \mathbf{D}, \mathbf{T}, \mathbf{R}). \tag{1}$$

Replacing the driver earnings ($\mathcal{E}$) by demand fulfillment ($\mathcal{F}$) in the Equation (1) above results in a variant of MaxEarnings problem, in which the goal is to maximize fulfilled rides, referred to henceforth as the MaxFulfillment problem.

IV. Learning Framework

In this section, we describe our approach for solving the MaxEarnings problem. Our method is a *model-based reinforcement learning* approach, and its description is provided in Algorithm 1.

As with any reinforcement learning approach, we train our model by allowing the drivers to repeatedly interact with an environment in form of the city's ride demand data from a representative day. Each interaction, which is T timesteps long, constitutes an *episode* of the training process. Each episode constitutes of 3 phases described below.

ALGORITHM 1: General learning approach

```
1  Initialization
     Q_I(t,h,a) ← 0, Q_C(t,h,a) ← 0, ξ(t,h) ← 0;
2  for each episode e = 1, ··· , E
3      for each time step t = 1, ··· , T
4          for each driver i = 1, ··· , n
5              Generate two random numbers
                 η_0, η_1 ∈ [0,1];
6              if η_0 ≤ ε
7                  Choose exploratory action;
8              else
9                  if η_1 ≤ ξ(t,h_i)
10                     a = Independent action a* from Q_I;
11                 else
12                     a = Coordinated action a^c from Q_C;
13             Receive reward E(t,h_i,a);
14             Compute rebalance matrix Z;
15     for each zone h ∈ H
16         ∀t,a update Q_I(t,h,a) ;
17         ∀t update degree of coordination ξ(t,h) ;
18         ∀t,a update Q_C(t,h,a) ;
```

Exploratory phase (lines 5-7): During this phase of the algorithm, drivers exhibit an *exploratory behavior* by choosing a pseudo-random action with a probability ϵ. These randomly chosen actions allow the model to explore a larger portion of the policy space, preventing its policy from converging to a local minimum. This is similar to the ϵ-greedy behavior of Q-learning [29].

Exploitative phase (lines 9-12): During this phase of the algorithm, the rest of the drivers exhibit an *exploitative behavior* using the policy learned up until the previous episode of training. The policy recommends exploitative actions to individual drivers based upon the time of the day and their locations, independently of each other, henceforth referred to as *independent actions*. However, certain recommended actions may result in supply-demand imbalances when a large number of drivers relocate to the same city zone with an insufficient demand, or too few of them relocate to a zone with excess demand. We postulate that explicit coordination is essential to prevent such supply-demand imbalances from occurring. Hence, we introduce the *degree of coordination (ξ)* - a probabilistic value that signifies the extent to which drivers located in the same city zone need to coordinate their actions. Whenever a zone has a positive degree of coordination, the exploitative actions recommended to a ξ fraction of drivers in the zone are derived from solving a reward-maximizing linear program, henceforth referred to as *coordinated actions*.

It should be noted that it is the explicit criterion for recommending a coordinated action that sets our approach apart from recent works in the field of deep reinforcement learning across different applications and domains.

Learning phase (lines 15-18): Actions recommended in the exploratory and exploitative phases of the episode result in drivers picking up passengers or relocating themselves to different city zones, thereby observing rewards of their actions (line 13). The learning phase of the algorithm computes a rebalancing matrix (line 14) to use in conjunction with the observed rewards to further improve upon the policy.

Having developed an intuition for the major building blocks of Algorithm 1, we now explain these phases in greater detail.

A. Exploratory phase

Over the course of training, when a driver located in zone h chooses to explore, we model the probability of driver's exploratory ride distance using a Gaussian function with a random variable $K_{\geq 0}$. Specifically, the probability that a driver relocates to a zone at distance $k \geq 0$ is given by: $Pr[K = k] = be^{-\frac{k^2}{2c^2}}$.

After sampling an exploration distance k, the driver chooses the actual destination by sampling uniformly at random from all zones at distance k. When $k = 0$, the driver chooses to wait in the current zone, while for $k > 0$, the driver chooses a relocate action. The experiments in this paper were all conducted using $b = 0.7$ and $c = 1$ (chosen via grid-search), allowing explorations up to 3 hexagonal zones away. In contrast, [14] restricts drivers to single zone distance relocations, reducing their ability to learn policies that mitigate supply-demand imbalances by relocating supply from zones further away in a single timestep. Over the course of training, ϵ is annealed exponentially from 1 to 0, thereby outputting an entirely exploitative model at the end of the training.

B. Exploitative phase

Exploitative behavior is manifested in the form of independent actions (line 10) and coordinated actions (line 12) when the degree of coordination is positive. We detail these next.

1) Choice of independent action: For each independent action chosen by a driver, we record the reward earned. This reward is then used to update the *value* of the action for the next episode, based on the learning rate (α) and the discount factor (γ). These values are stored in a Q-table denoted by $Q_I \in \mathbb{R}^{T \times m \times |\mathbf{A}|}$. For each zone h, at time t, the best independent action (a^*) is chosen by (line 10, Algorithm 1)

$$a^*(t,h) = \arg\max_{a \in \mathbf{A}_h} Q_I(t,h,a),$$

where $\mathbf{A}_h$ refers to the h-th row of $\mathbf{A}$.

2) Independent learning: Based upon the observations of drivers undertaking independent actions (both exploratory and exploitative), we update the independent learning matrix (Q_I) as described below.

Updating Q_I for wait actions: Let $\mathcal{W}_{(h,h')}$ denote the number of drivers choosing to wait in zone h at time t, and ending up in zone h'. A successful wait generates net earnings $E\big(t,h,a(h,h)\big) = r(h,h')$ and consumes a travel time $\tau(h,h')$, while an unsuccessful wait i.e., $h' = h$, generates zero net earnings and consumes one timestep. The utility of the wait action is therefore

$$\mathcal{U}_{(t,h,h)} = \sum_{h'} \mathcal{W}_{(h,h')} \Big[E\big(t,h,a(h,h)\big) + \gamma Q_I(t',h',a^*(t',h')) \Big]$$

where $t' = t + \tau(h,h')$ and we discount the future rewards with a factor γ. We use the utility of the wait action to update the entry $Q_I(t,h,h)$ as follows:

$$Q_I(t,h,h) \leftarrow (1-\alpha) Q_I(t,h,h) + \frac{\alpha}{\sum_{h'} \mathcal{W}_{(h,h')}} \mathcal{U}_{(t,h,h)}. \quad (2)$$

Normalizing the update term by the number of drivers choosing the wait action captures the average utility of the wait action. The term $Q_I(t,h,h)$ on the right hand side of the equation denotes the values learned upto the previous episode of the training, and α is the learning rate.

Updating Q_I for relocate actions: Let $\mathcal{R}_{(h,h')}$ denote the number of drivers relocating from zone h to zone h'. The utility of such relocation is given by

$$\mathcal{U}_{(t,h,h')} = \mathcal{R}_{(h,h')} \Big[E(t,h,a(h,h')) + \gamma Q_I(t',h',a^*(t',h')) \Big],$$

where $t' = t + \tau(h,h')$. We use the utility of the relocate actions to update the entry $Q_I(t,h,h')$ of the independent table as follows:

$$Q_I(t,h,h') \leftarrow (1-\alpha) Q_I(t,h,h') + \frac{\alpha}{\mathcal{R}_{(h,h')}} \mathcal{U}_{(t,h,h')}. \quad (3)$$

Using Equations (2) and (3), the Q_I matrix is updated in line 16 of Algorithm 1 using the evidence obtained via simulations in form of utilities $\mathcal{U}_{(t,h,h')}$ of both the wait and relocate actions.

3) Choice of coordinated action: The choice of coordinated action is more intricate and non-standard, and we next explain it in detail. To guide the coordinated behavior of drivers in line 12 of Algorithm 1, we solve a reward-maximizing rebalancing operation between city zones experiencing supply-demand imbalances. There are two principal components driving the coordinated behavior: *degree of coordination* (ξ) which controls the need of coordination in a particular zone at a time, and *coordinated learning matrix* (Q_C) which determines the choice of action as a response to the need of coordination. Thus, each coordinated action is associated with a probability for it to participate in the rebalancing operation that is stored in the matrix Q_C. Note that Q_C contains learned probabilities, as against the usual action-value nature of Q_I.

Let the policy learned at the end of k-th episode during training be denoted by π_k. Following this policy induces a driver supply S^{π_k} during the $(k+1)$-th episode of training. For each zone h, at time t, the coordinated action (a^c) in line 12 of Algorithm 1 is obtained by uniformly sampling from the probability vector $Q_C(t,h)$.

Imbalance matrix (Δ): A matrix $\Delta \in \mathcal{R}^{|\mathcal{T}| \times m}$ such that each entry $\delta(t,h)$ denotes the supply-demand imbalance experienced at zone h at time t during the $(k+1)$-th episode. Specifically, each entry of the imbalance matrix can be given by, $\delta(t,h) = S^{\pi_k}(t,h) - \sum_{h'} d_t(h,h')$. We mask the imbalance matrix using an imbalance threshold parameter Λ such that,

$$\delta(t,h) = \begin{cases} \delta(t,h) & \text{if } \big|\delta(t,h)\big| \geq \Lambda \\ 0 & \text{otherwise.} \end{cases}$$

Using this parameter allows us to control the level of imbalances that the framework should attempt to mitigate.

Rebalancing graph ($\mathcal{G}$): Based upon the supply-demand imbalance matrix induced at the end of an episode, we create the *rebalancing graph* $\mathcal{G} = (V,E)$ consisting of imbalanced zones as nodes and edges as corresponding relocation actions between them. This is a bipartite graph with nodes $V = \big\{V_+ \cup V_-\big\}$ where V_+ is the set of nodes with excess supply, i.e., $\delta(t,h) > 0$ and V_- is the set of nodes with supply deficit, i.e., $\delta(t,h) < 0$. Thus each node $v_i \in V$ in the rebalancing graph is associated with three attribues: *imbalanced zone* (v_i^h), *time of imbalance* (v_i^t) and *magnitude of imbalance* $\big(\delta(v_i^t, v_i^h)\big)$. The edge set E consists of directed edges from the nodes in V_+ to nodes in V_- and they model feasible relocations. Thus: $E = \big\{e_{ij} : v_i \in V_+, v_j \in V_-, v_i^t + \tau(v_i^h, v_j^h) \leq v_j^t\big\}$. The travel time constraint filters out edges where a relocating driver from an excess supply node cannot reach the deficit node in time. Each edge e_{ij} is associated with utility:

$$\mathcal{U}_{(i,j)} = \underbrace{Q_I(v_j^t, v_j^h, v_j^h)}_{\text{wait action at } v_j^h} - \underbrace{\text{cost}(v_i^h, v_j^h)}_{\text{relocation cost}} - \underbrace{Q_I(v_i^t, v_i^h, v_i^h)}_{\text{wait action at } v_i^h}.$$

Thus, the utility of an edge measures the net value for a driver relocating along it during coordinated behavior.

Rebalancing operation: Given a rebalancing graph $\mathcal{G}$, we wish to relocate drivers from supply excess zones to supply deficit zones. We aim to find a matching that maximizes the net reward of all relocations, in order to maximize the driver earnings. Such a rebalancing operation can be achieved by solving a *Minimum Cost Flow* problem expressed in the form of the linear program below.

$$\begin{aligned} \text{maximize} \quad & \textstyle\sum_{e_{ij} \in E} f_{ij} \times \mathcal{U}_{(i,j)} \\ \text{s.t.,} \quad & \\ \forall e_{ij} \in E, \quad & f_{ij} \geq 0 \\ \forall v_i \in V_+, \quad & \textstyle\sum_{v_j \in V_-} f_{ij} \leq \delta(v_i^t, v_i^h) \\ \forall v_j \in V_-, \quad & \textstyle\sum_{v_i \in V_+} f_{ij} \leq |\delta(v_j^t, v_j^h)| \end{aligned}$$

Here, we calculate the number of excess drivers who should relocate from an excess node to a deficit node and store it in

the form of a flow vector $f \in \mathcal{R}^{|E|}$ indexed along the edges set such that f_{ij} denote the flow from v_i to v_j.

If the platform aims to maximize demand fulfillment, we can formulate it as a *Maximum Flow* problem by setting the utility associated with each edge $\mathcal{U}_{(i,j)} = 1$.

As the constraint matrices – in both problems – are unimodal, the solutions of the linear programs are integral flow vectors and are thus optimal. Note that the size of the constraint matrix increases with a decrease in the Λ parameter. However, we can greatly reduce the sizes of corresponding linear programs and hence the computation time by solving a set of smaller linear programs; one for each connected component of the rebalancing graph.

4) Coordinated learning: Based upon the computed imbalance matrix (Δ) and the solution to the rebalancing operation above, we are now in a position to update the coordinated learning matrix (Q_C) and the degrees of coordination (ξ) as described below. It should be noted that while the choice of coordinated action from the matrix Q_C is influenced by the reward-maximizing rebalancing described above, the degree of coordination ξ is merely influenced by the supply-demand imbalances induced as a result of the policy learnt so far.

Updating Q_C for rebalancing operation: We capture the rebalancing operation in form of a rebalance matrix $\mathbf{Z} \in \mathbb{R}^{|\mathcal{T}|\times m\times m}$ where each entry $\zeta(t,h,h')$ denotes a probability of a rebalancing relocation from zone h to zone h' being required at time t. For every edge $e_{ij} \in E$, we update $\mathbf{Z}$ as follows,

$$\begin{aligned}\zeta(v_i^t, v_i^h, v_i^h) &= \frac{\delta(v_i^t, v_i^h) - \sum_{v_j \in V_-} f_{ij}}{\delta v_i^t, v_i^h} \\ \zeta(v_i^t, v_i^h, v_j^h) &= \frac{f_{ij}}{\delta(v_i^t, v_i^h)}.\end{aligned}$$

Using the rebalance matrix, we update Q_C in line 18 of Algorithm 1 as follows,

$$Q_C(t,h,h') \leftarrow (1-\alpha)Q_C(t,h,h') + \alpha\zeta(t,h,h'). \quad (4)$$

Updating degree of coordination (ξ): At the end of each training episode $(k+1)$, we use the realized imbalance matrix (Δ) to determine the degree of coordination required within each zone at every time step. We update the degree of coordination as follows:

$$\xi_{k+1}(t,h) = (1-\alpha)\xi_k(t,h) + \alpha\mu(t,h), \quad (5)$$

where the rebalancing raio μ is computed as:

$$\mu(t,h) = \begin{cases} \frac{\delta(t,h)}{S^{\pi_k}(t,h)} & \text{if } \delta(t,h) > 0. \\ \frac{\left|\delta(t,h)\right|}{\sum_{h'} d_t(h,h')} & \text{if } \delta(t,h) < 0 \text{ and } \xi_k(t,h) > 0. \end{cases}$$

While the former condition encourages driver relocations in zones with supply excess, the latter condition discourages it in zones with supply deficit. Thus, we use Equation (5) to update the degree of coordination for each zone in line 17 of Algorithm 1.

V. Data and Experiments

In this section, we begin by describing the pre-processing we did in order to use the New York City Yellow taxi rides public dataset and then we evaluate our framework.

A. Data pre-processing

To train our model, we need to construct the time-evolving city matrices - $\mathbf{D}$, $\mathbf{R}$, and $\mathbf{T}$ described in Section III.

Hexagonal binning of New York City: We employ the popular methodology of hexagonal binning to discretize the city into a set $\mathcal{H}$ of 250 non-overlapping uniform-sized hexagonal zones. The distance from the center of a zone to its vertices is about 1 mile.

Forming time-evolving matrices: We begin with the NYC Taxi dataset (2015), which contains street-hail records of over 200,000 taxi rides per day with information regarding pickup and dropoff locations and times, fare, trip distances, etc., from before the significant confounding effects of ride-sharing platforms like Uber, Lyft, etc. For each ride in the dataset, we evaluate its pickup and dropoff zones based on location coordinates. Assuming that passengers do not hail a taxi for short distances, we ignore a small percentage of rides which begin and end within the same zone.

We discretize a 24-hour day into 288 time-slices of duration 5 minutes each, indexed by their start time. Thus, to populate the entries of the matrices $\mathbf{D}$, $\mathbf{R}$ and $\mathbf{T}$ at time t, we use the rides from the dataset in the 5 minutes time-slice beginning at time t. Due to variations in the popularity of particular pickup and dropoff zones at specific times of the day, the $\mathbf{R}$ and $\mathbf{T}$ matrices obtained using this method are sparse. However, to compute the best policies, our framework requires the availability of complete information regarding rewards and travel times. Hence, we estimate the missing values in these matrices using linear regression models including fixed-effects for the time of the day, the source and destination zones [2]. The performance of our model is not sensitive to the choice of a specific linear regression modeling technique.

B. Experimental results

1) Settings: For all experiments, we use a multiprocessor implementation of our algorithm on a 24-core 2.9 GHz Intel Xeon E5 processor with 512 GB memory. The model training time for 100 episodes of training takes less than an hour. The model testing time is less than 5 minutes. Our code has been made publicly available for reproducibility purposes [19]. All our experiments use learning rate $\alpha = 0.01$ and discount factor $\gamma = 0.99$. During independent learning, the exploration factor (ϵ) used in ϵ-greedy Q-learning decreases exponentially as the training progresses. Unless mentioned otherwise, we train 5,000 drivers over 200 episodes and set the imbalance threshold (Λ) to 2. Experimental results presented in this paper

[2]To compute the travel time entry $\tau(i,j)$ at time t, we fit a linear regression model $\tau(i,j,t) = \overline{\beta_0}X_{i,j,t} + \beta_1\alpha_i + \beta_2\alpha_j + \beta_3\alpha_t + \epsilon_{i,j,t}$ where $X_{i,j,t}$ are the time-variant predictors, the α_i, α_j, and α_t are time-invariant fixed-effects for source, destination and time of the day respectively, while $\epsilon_{i,j,t}$ is standard normal error.

are obtained by training models over a representative day viz., first Monday of September 2015 with a demand of over 232,000 rides. However, our results generalize to any day.

2) Model performance: First, we address the question: *how well does our reinforcement learning-based algorithm learn the driver dispatch policy?* In Figure 1, we observe the improvement in mean driver earnings and demand fulfillment as the training progresses. We split the 200 training episodes into independent learning episodes ($E_{IL} = 160$) and coordinated learning episodes ($E_{CL} = 60$). This can be achieved by setting the degree of coordination (ξ) to 1 until episode number $E - E_{CL}$ on line 9 of Algorithm 1. Consequently, episodes $[140, 160]$ utilize both independent and coordinated learning. In Figure 1, we observe a significant improvement in the objective in the interval denoted by a shaded region. As expected, coordinated learning appropriately relaxes some of the constraints imposed by single-agent MDP and leads to significantly better performance.

In Figure 2, we plot the total demand at various times in the day, along with its fulfilled and unfulfilled portions by drivers following our policy. About 95% of the total demand during the day is satisfied with our framework. We consider a ride request fulfilled if an idle driver is present in the same zone at the time of the request. We find that 10% of unfulfilled demand can be fulfilled by nearby drivers by adding 10 minutes of passenger wait, and 75% of unfulfilled demand with 15 minutes of wait. At the beginning of a day, for lack of better alternative, we initialize drivers uniformly across the city zones. Hence, our model requires a "warm-up" time for the drivers to reposition themselves in order to fulfill the demand. This warm-up interval contributes significantly to the unfulfilled demand at the beginning of the day from 12AM-1AM. One may left-pad the training interval to alleviate this issue.

The explicit coordination in our model allows us to visualize the market conditions in which it is utilized. In Figure 3, we plot snapshots of coordination in form of a heatmap with probabilities of coordinated wait actions i.e. $Q_C(t, h, h)$ at 6 AM during the early morning commute and at 6 PM during the evening commute[3]. Without coordination, we would expect all the drivers in the city to relocate to Manhattan in order to satisfy the extremely high volume of demand during the morning commute. However, as observed in Figure 3, our model recommends a certain proportion of drivers to wait in the outer boroughs of New York City for the early morning commute to Manhattan. Notably, the model is able to learn demand trends in time-dependent hotspots such as the J.F.K. airport to the south-east of the city. In contrast, during the evening commute to outer boroughs, the model strongly recommends that the drivers wait inside Manhattan.

3) Impact of independent and coordinated learning: The overlap between the independent and the coordinated learning during training is a crucial aspect of our framework. In this

[3]More detailed visualizations depicting evolution of coordinated actions and degree of coordination across the city and through the time of the day are available at [19].

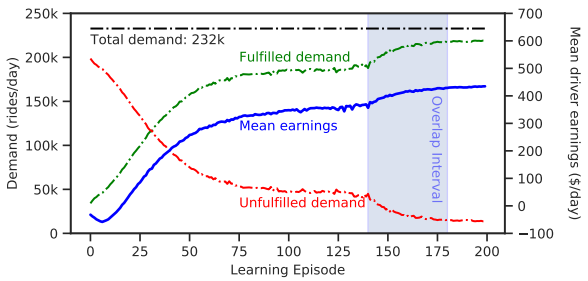

Fig. 1. A representative illustration of improvement in mean driver earnings during training.

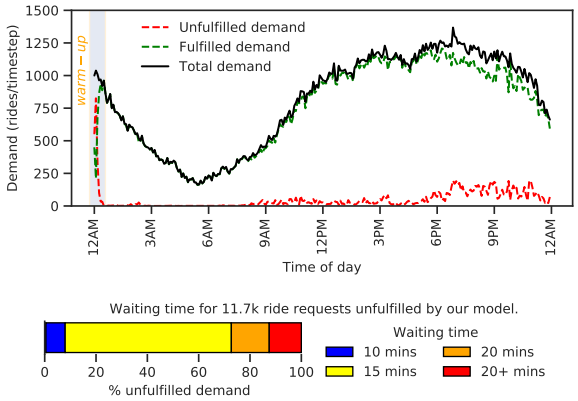

Fig. 2. Top: Demand fulfillment by a trained policy at different times on a representative day. Bottom: Waiting times for demand not immediately fulfilled by the model.

section, we address the question: *how do we determine the appropriate number of independent learning and coordinated learning episodes during training?* Given a fixed number of training episodes E, we assume that our model trains the initial E_{IL} episodes with independent learning and the final E_{CL} episodes with coordinated learning. When $E_{IL} + E_{CL} \geq E$, we have $E_{IL} + E_{CL} - E$ episodes of overlap between independent and coordinated learning. In Figure 4, we use 200 episodes of training, and we vary the values of E_{IL} and E_{CL} in the range $[20, 200]$ to achieve various overlaps[4]. We then plot the mean driver earnings for each learned policy. We show that for a large interval of values of E_{IL} and E_{CL}, our framework provides stable and high performance with up to \$535 mean earnings per day when $E_{IL} = 60$ and $E_{CL} = 160$, denoted by a green marker in the figure. This observation supports our claim that our framework is robust to imperfections in hyperparameter tuning. Note that we have used different values of E_{IL} and E_{CL} in Figure 1 in order to clearly portray the incremental impact of coordinated learning on mean driver earnings per day.

4) Impact of Driver Supply: We next answer the question: *what is an appropriate number of drivers to fulfill the ride demand?* To study this question, we vary the driver supply in

[4]Note that there is no overlap between the independent and the coordinated learning phases in the lower triangle of Figure 4 when $E_{IL} + E_{CL} < E$.

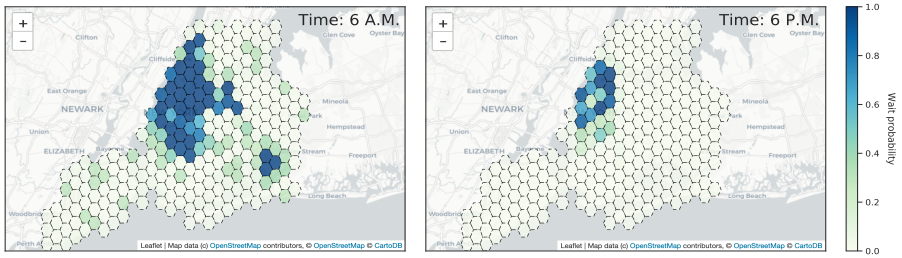

Fig. 3. Heatmaps of probability of coordinated wait i.e., $Q_C(t, h, h)$ during morning commute at 6 A.M. (left) and evening commute at 6 P.M. (right).

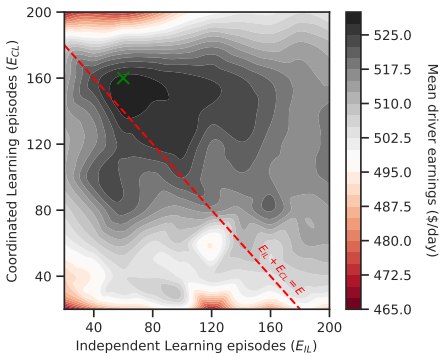

Fig. 4. Performance stability over a wide range of overlaps between the independent and the coordinated learning.

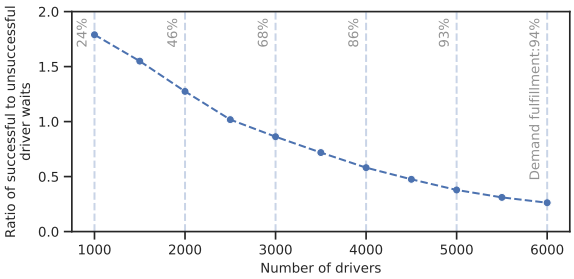

Fig. 5. Impact of supply size on the ease of finding a passenger on a representative day.

the range $[1000, 6000]$, where the units are individual drivers. Given a fixed supply size, we plot the ratio of the number of successful driver waits resulting into passenger rides to the number of unsuccessful driver waits while taking into account the overall demand fulfillment. When the number of drivers is small compared to the demand, the drivers should have an easier time finding a passenger. On the other hand, a city saturated with drivers should result in a higher number of unsuccessful driver waits. In Figure 5, we observe that the framework validates our expectations. The "warm-up" period described in Figure 2 causes underestimation of demand fulfillment while it simultaneously causes overestimation of the number of unsuccessful driver waits. Excluding the warm-up interval, this experiment provides evidence that over 96% demand of New York City can be fulfilled by about 5,000

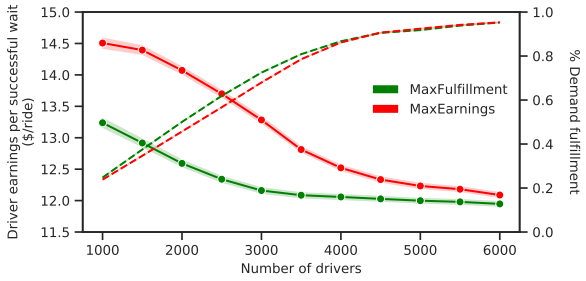

Fig. 6. Differential impact of platform objectives. Left Y-axis: Driver earnings per passenger ride with 90% confidence interval. Right Y-axis: % Demand fulfillment for both objectives on a representative day.

drivers. Note that in September 2015, New York City had over 13,500 operational taxicab medallions [30]. It also justifies our decision to use 5,000 drivers in most of our experiments.

5) Impact of platform objectives: So far, our experiments focused on the MAXEARNINGS problem. A natural question is: *should a platform optimize driver dispatches to maximize their earnings or to maximize demand fulfillment?* Note that while maximizing the demand fulfillment might help retain customers over a longer-term, it can be detrimental to drivers' earnings. To solve MAXFULFILLMENT (see Section III), the framework rewards (resp. penalizes) a successful passenger pickup (resp. unsuccessful wait) by +1 (resp. -1) net reward. Figure 6 depicts that mean driver earnings per passenger ride can be over a $1 lower in a policy optimized for maximizing demand fulfillment relative to one optimized for earnings. The additional rides covered by the solution to MAXFULFILLMENT may direct drivers to sub-optimal locations and compromise their future earnings for the day. As the supply increases over the minimum number of required drivers, the two objectives converge while a statistically significant difference in the driver earnings per ride persists. Note that higher rewards/penalties while solving MAXFULFILLMENT result in larger divergence between the two objectives.

6) Advantage of strategic behavior: Next, we address the question: *does our model provide consistently higher earnings for all the drivers?* To explore this, we model the taxi driver population of the city as comprised of strategic drivers who follow the model recommendations and naive drivers who act upon heuristics learned via experience. We expect the mean earnings of drivers to decrease as the number of strategic

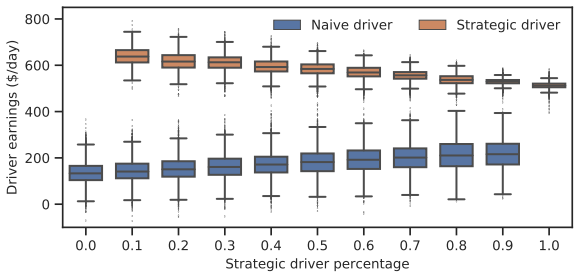

Fig. 7. Earnings advantage of the strategic drivers over the naive drivers on a representative day.

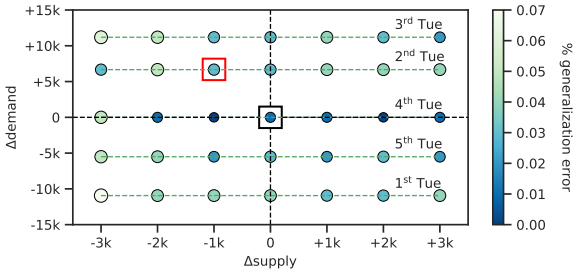

Fig. 8. Model robustness: Baseline model (enclosed within a black box at origin) is deployed on other Tuesdays.

drivers on the platform increases. While modeling a naive driver, we assume that taxi drivers, over time, learn the popular spots in the city. If they are unable to locate passengers reasonably quickly in other parts of the city, they head back to the popular spots. We designate 15 zones as popular zones based on the historical demand data. Furthermore, we assume that an idle naive driver looking for a passenger decides to head back to one of the popular zones with a fixed probability of 0.25. Upon choosing to relocate, the naive driver picks the target popular zone with a probability inversely proportional to its distance from the current location.

In Figure 7, we plot the earnings of the two categories of drivers while varying the percentage of strategic drivers. As expected, an increase in the number of strategic drivers causes their individual earnings to decline. Overall, the strategic drivers not only earn more than the naive drivers, but also the variance in their earnings is significantly lower. Thus, our framework is *envy-free*, i.e., drivers at the same location and time do not envy each other's future earnings.

7) Model generalizability: In this section, we explore the question of model generalizability: *does our model perform well when deployed on days with considerably different supply-demand conditions compared to the day it was trained on?* We cross-validate our model by evaluating the policy of a trained model on different days.

For illustrative purposes, we choose as baseline – m_0 – a model trained to satisfy the demand of 288,000 rides observed on the fourth Tuesday of September using 7,000 drivers. We test the policy $\pi(m_0)$ recommended by our baseline model by deploying it on other Tuesdays of the month. Note that

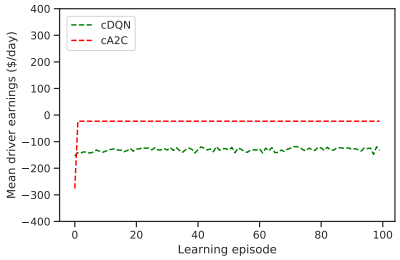

Fig. 9. Performance of cDQN and cA2C deep-learning approaches from [14].

the observed demand, as well as the number of active drivers might vary on other Tuesdays compared to our baseline model. To capture this potential for change in supply, we vary the number of simulated drivers during testing in the range $[4000, 10000]$. In Figure 8, enclosed within a red square box is an illustration of the generalization error associated with deploying our baseline model's recommended policy on the second Tuesday of the month with just 6,000 drivers. Importantly, the policy $\pi(m_0)$ now attempts to fulfill an increased demand of about 7,000 extra rides ($\Delta demand$) using 1,000 fewer drivers ($\Delta supply$) than it was trained for. To evaluate its performance in this task, we compare it with a model m_* which was explicitly trained to fulfill the demand of the second Tuesday with exactly 6000 drivers. Thus, we compute the baseline policy's generalization error as

$$\%\text{generalization error} = \frac{\mathcal{F}^{\pi}(m_*) - \mathcal{F}^{\pi}(m_0)}{\mathcal{F}^{\pi}(m_*)},$$

where $\mathcal{F}^{\pi}(m)$ denotes the demand fulfillment of model m.

Figure 8 shows that our framework generalizes well to perturbations in both supply and demand. We also observe that decreasing the number of drivers excessively impacts harms its generalization performance. As a result, we recommend deploying models trained with a reasonably higher number of drivers than minimally required so that they generalize better in cases of varying demand. For brevity, we have presented a single illustrative example here; the generalizability result holds true across all the models.

8) Comparison with baselines: A major challenge in comparative studies in this domain is the lack of reproducibility due to proprietory datasets and simulators. To the best of our knowledge, although [14] uses coordinated deep reinforcement learning approach, it is most similar to ours with respect to modeling assumptions. In the absence of the Didi Chuxing's proprietary driver simulator and datasets, direct comparison of our works is impossible. We make an effort to compare our approaches by re-implementing their deep reinforcement learning based algorithms (cDQN and cA2C) with minimal modifications to fit our setting which computes future driver distributions based on simulating passenger pickups and dropoffs, instead of predicting them using proprietary models.

In [14], the authors do not train the neural network from its randomly initialized state. Instead, they bootstrap the network based on a *pre-trained value networks based on historical*

means from the aforementioned simulator. As a direct and fair comparison with our model which does not rely on external pre-trained inputs, our implementations of their algorithms also attempt to learn *from scratch*.

Figure 9 shows mean driver earnings per day over the course of model training. Even after extensive hyperparameter tuning, the baselines failed to learn meaningful strategies, with driver earning net negative rewards of -$20 over a day. In the absence of a pre-trained value network, the algorithms proposed in [14] are unable to explore the policy space effectively. Moreover, the reward sharing assumption used in [14] results in a superficial coordination behavior which fails to learn in a more realistic scenario such as ours, which simulates actual passenger pickups and dropoffs. Our implementations of contextual DQN (cDQN) and contextual actor-critic (cA2C) algorithms are publicly available at [19].

VI. Conclusions

In this paper, we studied the problem of maximizing earnings of drivers employed by ride-sharing platforms like Uber, Lyft, etc. Our work confirms the idea that even in a high-dimensional and big data domain such as ride-sharing, the inherent structure of the data can be leveraged to develop a simple, interpretable, fair and highly efficient framework that aims to achieve this goal. Extensive simulations based on New York City taxi datasets showed that our framework is easy to calibrate due to its robustness to imperfections in hyperparameter tuning. Our experiments provided evidence for the differential impact of the platform's objectives on driver earnings. Finally, we demonstrated that our model generalizes well to fluctuations in supply and demand. We make available an OpenAI gym environment for comparative studies.

Acknowledgements

This research was partially funded by NSF CAREER 1253393 and NSF 1813406 awards. The authors also thank the anonymous reviewers for their helpful comments.

References

[1] Business Traveller, "Global ride sharing industry valued at more than $61 Billion," https://www.businesstraveller.com/business-travel/2019/01/04/value-of-global-ride-sharing-industry-estimated-at-more-than-61-billion/.

[2] C. Yan, H. Zhu, N. Korolko, and D. Woodard, "Dynamic Pricing and Matching in Ride-Hailing Platforms," Oct. 2018.

[3] O. Besbes, F. Castro, and I. Lobel, "Surge Pricing and Its Spatial Supply Response," May 2019.

[4] S. Banerjee, C. Riquelme, and R. Johari, "Pricing in Ride-Share Platforms: A Queueing-Theoretic Approach," Feb. 2015.

[5] J. C. Castillo, D. T. Knoepfle, and E. G. Weyl, "Surge Pricing Solves the Wild Goose Chase," Mar. 2018.

[6] N. Garg and H. Nazerzadeh, "Driver Surge Pricing," May 2019.

[7] D.-H. Lee, H. Wang, R. L. Cheu, and S. H. Teo, "Taxi Dispatch System Based on Current Demands and Real-Time Traffic Conditions," *Transp. Res. Rec.*, vol. 1882, no. 1, pp. 193–200, Jan. 2004.

[8] R. Zhang and M. Pavone, "Control of robotic mobility-on-demand systems: A queueing-theoretical perspective," *Int. J. Rob. Res.*, vol. 35, no. 1-3, pp. 186–203, Jan. 2016.

[9] K. T. Seow, N. H. Dang, and D. Lee, "A Collaborative Multiagent Taxi-Dispatch System," *IEEE Trans. Autom. Sci. Eng.*, vol. 7, no. 3, pp. 607–616, Jul. 2010.

[10] 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.

[11] L. Zhang, T. Hu, Y. Min, G. Wu, J. Zhang, P. Feng, P. Gong, and J. Ye, "A Taxi Order Dispatch Model Based On Combinatorial Optimization," in *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, ser. KDD '17. New York, NY, USA: ACM, 2017, pp. 2151–2159.

[12] V. Mnih, K. Kavukcuoglu, D. Silver, A. Graves, I. Antonoglou, D. Wierstra, and M. Riedmiller, "Playing Atari with Deep Reinforcement Learning," Dec. 2013.

[13] 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. New York, NY, USA: ACM, 2019, pp. 1780–1790.

[14] 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.

[15] J. Wen, J. Zhao, and P. Jaillet, "Rebalancing shared mobility-on-demand systems: A reinforcement learning approach," 2017.

[16] Z. Wang, Z. Qin, X. Tang, J. Ye, and H. Zhu, "Deep Reinforcement Learning with Knowledge Transfer for Online Rides Order Dispatching," in *2018 IEEE International Conference on Data Mining (ICDM)*, Nov. 2018, pp. 617–626.

[17] J. Vincent, "AI systems should be accountable, explainable, and unbiased, says EU," *TheVerge*, Apr. 2019.

[18] G. Brockman, V. Cheung, L. Pettersson, J. Schneider, J. Schulman, J. Tang, and W. Zaremba, "OpenAI Gym," 2016.

[19] GitHub Repository, "Learn to earn," https://chdhr-harshal.github.io/learn-to-earn/, 2020.

[20] 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 PERCOM Workshops*, Mar. 2011, pp. 63–68.

[21] 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.

[22] N. J. Yuan, Y. Zheng, L. Zhang, and X. Xie, "T-Finder: A Recommender System for Finding Passengers and Vacant Taxis," *IEEE Trans. Knowl. Data Eng.*, vol. 25, no. 10, pp. 2390–2403, Oct. 2013.

[23] H. A. Chaudhari, J. W. Byers, and E. Terzi, "Putting Data in the Driver's Seat: Optimizing Earnings for On-Demand Ride-Hailing," in *Proceedings of the Eleventh ACM International Conference on Web Search and Data Mining*, ser. WSDM '18. New York, NY, USA: ACM, 2018, pp. 90–98.

[24] K. Bimpikis, O. Candogan, and D. Saban, "Spatial Pricing in Ride-Sharing Networks," Nov. 2016.

[25] H. Ma, F. Fang, and D. C. Parkes, "Spatio-Temporal Pricing for Ridesharing Platforms," Jan. 2018.

[26] T. Sühr, A. J. B. Meike, Zehlike, K. P. Gummadi, and A. Chakraborty, "Two-Sided Fairness for Repeated Matchings in Two-Sided Markets: A Case Study of a Ride-Hailing Platform," in *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. New York, NY, USA: ACM, Jul. 2019, pp. 3082–3092.

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

[28] H. Chen, W. Wang, K. Kjelstrøm, and E. Reinhold, "Gaining Insights in a Simulated Marketplace with Machine Learning at Uber," https://eng.uber.com/simulated-marketplace/, Jun. 2019.

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

[30] Wikipedia contributors, "Taxicabs of new york city," https://en.wikipedia.org/w/index.php?title=Taxicabs_of_New_York_City, 2019.