Context-Aware Taxi Dispatching at City-Scale Using Deep Reinforcement Learning

Zhidan Liu, *Member, IEEE*, Jiangzhou Li, and Kaishun Wu, *Member, IEEE*

***Abstract*— Proactive taxi dispatching is of great importance to balance taxi demand-supply gaps among different locations in a city. Recent advances primarily rely on deep reinforcement learning (DRL) to directly learn the optimal dispatching policy. These works, however, are still not sufficiently efficient because they overlook several pieces of valuable context information. As a result, they may generate quite a few improper actions and introduce unnecessary coordination costs. To improve existing works, we present *COX* – a context-aware taxi dispatching approach that incorporates rich contexts into DRL modeling for more efficient taxi reallocations. Specifically, rather than simply dividing the service area into grids, *COX* proposes a road connectivity aware clustering algorithm to divide the road network graph into zones for practical taxi dispatching. In addition, *COX* comprehensively analyzes zone-level taxi demands and supplies through accurate taxi demand prediction and timely updates of taxi statuses. *COX* improves the DRL modeling by integrating these derived contexts, *e.g.*, state representation with complete demand/supply data and sequential action generation with full coordination among idle taxis. In particular, we implement an environment simulator to train and evaluate *COX* using a large real-world taxi dataset. Extensive experiments show that *COX* outperforms state-of-the-art approaches on various performance metrics, *e.g.*, on average improving the total order values by 6.74%, while reducing the number of unserved taxi orders and passengers' waiting time by 4.92% and 44.84%, respectively.**

***Index Terms*— Taxi dispatching, deep reinforcement learning, road network clustering, taxi demand prediction.**

I. Introduction

THE emerging large-scale modern ride-hailing platforms, *e.g.*, Uber [3] and Didi Chuxing [1], have greatly benefited our daily travel by allowing passengers to book a trip in advance and matching available taxis with ride requests in real time. Although such systems could serve millions of taxi ride requests everyday in a city, a large number of requests remain unserved due to the lack of available taxis nearby [36]. On the other hand, however, there are a plenty of idle taxis seeking for passengers in other places [49]. Such taxi demand-supply imbalances among different geographic locations widely exist in urban cities [47], which will severely deteriorate the system efficiency and result in terrible user experience.

As a key enabler for the intelligent ride-hailing systems, *taxi dispatching* is expected to balance the difference between taxi demands and supplies by proactively reallocating available taxis to some locations for better serving future ride requests [18]. An efficient taxi dispatching strategy can significantly increase the number of requests being served, reduce the idle cursing distances of taxis and passengers' waiting time [25]. Proactive taxi dispatching over a large city, however, encounters two major challenges [30]. First, a ride-hailing platform usually manages and regulates tens of thousands of taxis to pick-up and deliver passengers across the whole city. It thus requires extremely complex coordination among taxis. The demand-supply balancing may further lead to sequential and long-term impacts that are difficult or even impossible to be well modelled. Second, taxi demands and supplies are dynamically changing over time, which thus poses many sources of future uncertainties for the effective taxi dispatching.

In the literature, many efforts have been made to achieve efficient taxi dispatching. Based on historical taxi data, some studies attempt to explicitly build taxi demand/supply models [47], and then dispatch idle taxis given these models through techniques like receding horizon control [25]. The model-based approaches, however, are inherently limited by the specified model and cannot evolve with the dynamic taxi service network. Therefore, some model-free approaches [9], [18], [30], [33] have been recently proposed. They primarily rely on the deep reinforcement learning (DRL) theory [5] to directly learn the best dispatching policies by interacting with the taxi transportation environment. Although these approaches have achieved better performances when compared with traditional ones, they are still not sufficiently efficient due to the neglects of some context information. As a result, existing DRL based approaches usually use numerous states to represent the environment and generate improper or even inconsistent actions, leading to huge computational and coordination costs.

To improve existing works, we present a context-aware taxi dispatching approach, named as *COX*. As a data-driven model-free approach, *COX* also makes use of the DRL

Manuscript received January 18, 2020; revised July 8, 2020; accepted September 30, 2020. This work was supported in part by the China NSFC under Grant 61802261, Grant 61872248, and Grant U1736207; in part by the Guangdong Basic and Applied Basic Research Foundation under Grant 2020A1515011502; in part by the Guangdong NSF under Grant 2017A030312008; in part by the Guangdong Science and Technology Foundation under Grant 2019B111103001 and Grant 2019B020209001; in part by the Shenzhen Science and Technology Foundation under Grant ZDSYS20190902092853047; and in part by the GDUPS (2015). The Associate Editor for this article was Y. Lv. *(Corresponding author: Kaishun Wu.)*

Zhidan Liu and Jiangzhou Li are with the College of Computer Science and Software Engineering, Shenzhen University, Shenzhen 518060, China (e-mail: liuzhidan@szu.edu.cn; lijiangzhou2018@email.szu.edu.cn).

Kaishun Wu is with the College of Computer Science and Software Engineering, Shenzhen University, Shenzhen 518060, China, and also with the Guangzhou HKUST Fok Ying Tung Research Institute, Guangzhou 511458, China (e-mail: wu@szu.edu.cn).

Digital Object Identifier 10.1109/TITS.2020.3030252

framework to search for the optimal actions rather than explicitly modeling the system. Specifically, *COX* views the dispatching center as the agent and exploits double deep Q-network (DQN) learning technique [26] to directly learn the dispatching policies, since DQN can comprehensively capture the relationship among observed states, actions, and long-term rewards. More important, *COX* derives and incorporates several pieces of important yet neglected context information into the system design.

(1) *road connectivity aware zone formation*. Existing works [9], [18], [30] usually divide the city into grids, and taxis are dispatched from its currently locating grid to one of the adjacent grids, so as to reduce the action space. Such a grid based taxi dispatching, however, omits the underlying road network, and thus possibly leads to improper dispatching between two grids with no direct road connectivity or no accessibility. Instead, *COX* divides a city into zones by clustering on the road network graph, and dispatches an idle taxi by taking both road connectivity and practical travel cost into consideration.

(2) *Comprehensive demand/supply aware states*. The appropriate state representation will potentially help DRL models to better understand the environment and thus determines the optimal dispatching actions. Previous works merely consider currently idle taxis as the supply [9], [18], [30] and implicitly encode future taxi demand information into DRL states [18]. As a result, their DRL models poorly describe the environment and bring unacceptable computation overheads. Instead, *COX* carefully counts both current and future available taxis as the overall supply, and accurately predicts zone-level demands by exploiting the graph convolutional network (GCN) model [12], which well fits with our irregular zones.

(3) *Action aware coordination*. Existing works [9], [18], [30] separately generate an independent action for each available taxi with no coordination at all, resulting in inefficient and redundant dispatching [18]. *COX* improves the coordination efficiency by allowing each available taxi to encode its own decision on the state representation, so that subsequent taxis can perceive such supply information and make wiser decisions accordingly.

The contributions of our work are summarized as follows.

- We identify the limitations of existing DRL based taxi dispatching approaches, and thus propose *COX* to achieve more efficient taxi dispatching at city scale by incorporating rich context information.
- We implement a realistic environment simulator to train, test, and evaluate *COX* design and other taxi dispatching approaches based on a large scale real-world taxi dataset. In order to inspire more future studies, we have publicly opened the source code of our simulator.[1]
- We conduct extensive experiments to evaluate *COX* using the environment simulator and large datasets. The experimental results demonstrate that *COX* significantly outperforms state-of-the-art approaches on various metrics, *e.g.*, averagely reducing the number of unserved requests (due to the unavailability of idle taxis nearby) and passengers' waiting time by 4.92% and 44.84%, respectively, while improving the total order values by 6.74%.

[1]Code is available at https://github.com/szlhl1040/Simulator.

The rest of this paper is organized as follows. We review the related works in Section II. We present the problem statement in Section III. The *COX* design is elaborated and evaluated in Section IV and Section V, respectively. Finally, Section VI concludes this paper.

II. Related Work

A. Taxi Dispatching

Proactive taxi dispatching is an imperative part of fleet management systems to balance taxi demands and supplies among different locations [29]. Traditionally, people have studied the demand-supply equilibrium of taxi services with regulations on fare structure and fleet size [47]. With the wide availability of taxi data, many data-driven approaches have been proposed [24], [31], [49]. For example, previous works recommend drivers to find potential passengers along a profitable driving route [31] or stay at some hot-spots [49] by analyzing massive historical taxi data. These methods have no coordination among taxis at all. In addition, some works explicitly model taxi demand/supply based on taxi data, and then dispatch taxis according to the model and real-time GPS locations of taxis through various techniques, *e.g.*, receding horizon control [25], mixed-integer program [44] and combinatorial optimization algorithm [42]. However, model-based approaches are inherently limited by the pre-specified model and cannot be adapted to the dynamic environment [30].

Recently, some model-free approaches have been proposed to address the taxi dispatching problem [9], [18], [30], [33], [40]. These approaches mainly make use of deep reinforcement learning [5] to directly learn appropriate action policies, rather than accurately modeling taxi demand/supply, by instructing all taxis to interact with the external environment. As the action space could be extremely large for taxi dispatching in a city, deep Q-network learning [26] has been adopted by the state-of-the-art approaches [9], [18], [30] to accelerate the policy learning process. Although these works have indeed improved the system performances when compared with the traditional ones, they are still not sufficiently efficient since they overlook some important context information, *e.g.*, road network connectivity and future taxi demands. Furthermore, these works do not well coordinate the available taxis, and as a result introduce large dispatching costs. In this paper, *COX* carefully derives and incorporates such context information into the design to further optimize the performance.

B. Order Dispatching

Different from taxi dispatching, order dispatching corresponds to the process of searching a proper vehicle to serve a received ride request [50]. Previously, greedy methods are widely used by assigning the nearest available taxi to a ride request [17]. Although simple, these methods omit the global demands and supplies, and thus cannot achieve the optimal performance in the long run.

Recent works utilize complete demand-supply information to achieve automatically order dispatching with the optimized long-term performances [45], *e.g.*, maximizing the success rate of taxi-order matches [50]. To this end, Xu *et al*. model order dispatching as a sequential decision-making problem and address it with the reinforcement learning theory [45]. Wang *et al*. further propose a transfer learning method to increase the learning adaptability and efficiency, where the learned order dispatching model can be transferred to other cities [39]. Li *et al*. propose a multi-agent reinforcement learning solution to address order dispatching in large-scale ridesharing scenarios [15]. Zhou *et al*. simultaneously maximize both accumulated driver income and served orders by exploiting double *Q*-learning and KL-divergence optimization [53]. Other factors, *e.g.*, pricing [52] and preferences of passengers [51], have also been considered. Our work differs from these works by proactively dispatching taxis to serve future unknown requests.

C. Ride-Hailing Demand Prediction

It is necessary and essential for intelligent ride-hailing platforms to be aware of the future mobility demands, which can help them to efficiently allocate resources in advance [27]. Thanks to the deep learning theory [13], [38] and the availability of tremendous amount of mobility data [22] in recent years, many research efforts have been made on predicting ride-hailing demands. To derive more accurate forecasting results, these works capture the complex spatial-temporal relations in the transportation network using various deep learning models, including recurrent neural network [43], multi-graph convolutional network [8], and deep multi-view spatial-temporal network [48]. In particular, Tong *et al*. propose a unified approach to predict the original taxi demands, which refer to the number of taxi-calling requests [36]. Wang *et al*. study a new perspective of demand modeling by predicting origin-destination matrix, which contains the number of taxi demands from one region to another region [19]. These works could benefit taxi dispatching, since they provide hints on determining proper dispatching actions.

D. Deep Reinforcement Learning

Deep reinforcement learning combines the principles of deep learning [13] and reinforcement learning [11] to intelligently learn the best actions from the observed states and received rewards based on sequential trail and error [5]. In recent years, deep reinforcement learning has been successfully applied to various challenging problems, *e.g.*, ridesharing [4], express system [16], network congestion control [41], and App usage prediction [35]. By comparing with existing works [9], [18], [30], we propose a context-aware approach to improve the deep reinforcement learning based taxi dispatching.

III. Problem Statement

In this section, we will define the taxi dispatching problem, and briefly discuss existing deep reinforcement learning based taxi dispatching approaches to motivate our design. The key notations and abbreviations used in this paper are summarized in Table I and Table II, respectively.

TABLE I
SUMMARY OF KEY NOTATIONS

Notation	Description
$z_i \in \mathbb{Z}$	The i-th zone of $\mathbb{Z} = \{z_1, z_2, \cdots, z_m\}$
$t_j \in \mathbb{T}$	The j-th time slot of $\mathbb{T} = \{t_0, t_1, \cdots, t_n\}$
Δt	Size of time slots
$s_t \in \mathcal{S}$	State of external environment at time t
$a_t \in \mathcal{A}$	Action taken at time t
$r_t \in \mathcal{R}$	Reward gained at time t
γ	The discount factor
$\mathtt{G}(\mathtt{V}, \mathtt{E})$	Road network graph
$cost(\cdot)$	Function to calculate travel costs on the road network
$\mathtt{C}_i \in \mathbb{C}$	Road connectivity aware clusters $\mathbb{C} = \{\mathtt{C}_i, i = 1, \cdots, k\}$
$\mathtt{G}^z(\mathtt{Z}, \mathbf{A})$	Zone graph with zone vertices $\mathtt{Z}$ and adjacency matrix $\mathbf{A}$

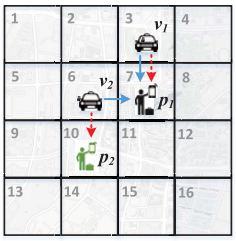
TABLE II
LIST OF KEY ABBREVIATIONS

Abbreviation	Term
COX	Context-aware taxi dispatching approach
DRL	Deep reinforcement learning
DQN	Deep *Q*-network
CARnet	Connectivity-aware road network clustering algorithm
GCN	Graph convolutional network
ReLU	Rectified linear unit
CNN	Convolutional neural network

A. Preliminary

We consider a modern ride-hailing platform, where a dispatching center manages a large number of geographically distributed taxis to serve passengers who can issue their ride requests online through the smartphones. The dispatching center continuously tracks the real-time location and availability status of each taxi, receives passengers' online ride requests, and assigns a proper taxi to serve each request given intelligent taxi-order matching algorithms [20]. In a city, the amounts of ride requests across different time of a day and among different locations can be distinctly different, resulting in taxi demand-supply imbalances that will harm the quality and efficiency of taxi service [47]. Therefore, the ride-hailing platforms usually proactively dispatch some available taxis to the location with larger demand-supply gap than their current locations, in the hope of serving more passengers with better experience [18].

To facilitate taxi allocations, the dispatching center usually divides a large city into a set of disjoint zones, denoted by $\mathbb{Z} = \{z_1, z_2, \cdots, z_m\}$, and splits the time into a sequence of time slots, denoted by $\mathbb{T} = \{t_0, t_1, \cdots, t_n\}$, where the size of all time slots is set as Δt. The sizes of both zone and time slot can be adjusted to balance the dispatching granularity and computation overhead [9]. Therefore, rather than dispatching a taxi to a specific location, existing works [9], [18], [29], [30], [33], [40] pre-allocate each idle taxi to a nearby zone within each time slot, so as to reduce the overall dispatching complexity. For simplicity, these works usually divide a large



(a)

Time / Zone	t_0	t_1	t_1
z_3	$[v_1]$	$\emptyset$	$\emptyset$
z_6	$[v_2]$	$\emptyset$	$\emptyset$
z_7	$\emptyset$	$[v_1, v_2, p_1]$	$[v_1, p_1]$
z_{10}	$\emptyset$	$[p_2]$	$[v_2, p_2]$

(b)

Fig. 1. A simple motivation example. (a) The whole area is divided into zones (*i.e.*, grids) to track statuses of taxis and ride requests, where at time t_0 taxi v_1 and v_2 are in zone z_3 and z_6, respectively, and at time t_1 passenger p_1 and p_2 will appear in zone z_7 and z_{10}, respectively. The blue and red arrows correspond to two different taxi dispatching policies. (b) The spatial-temporal illustration of taxi dispatching policy for time t_1, where the red policy is better than the blue one as it considers both future demands (*i.e.*, p_2) and the coordination among idle taxis (*i.e.*, v_1 and v_2).

city into grids [9], [30], [40] (or hexagonal grids [18]). Figure 1(a) shows the grid-world system adopted by previous works to track taxis and ride requests.

Problem statement: given the real-time locations and availability statuses of all taxis and the information of ride requests (including pick-up location, drop-off location, and the release time), the taxi dispatching problem aims to decide *which zone and when* each available taxi should be reallocated, so as to maximize the total number of ride requests being served and the passenger-perceived service quality (*e.g.*, the waiting time).

B. Deep Reinforcement Learning Based Solutions

Due to the uncertainties and dynamics of taxi demands and supplies in the urban city, however, proactive taxi dispatching is quite challenging [47]. To tackle this tough problem, many research efforts have been already made [29], [49], while most of the recent advances [9], [18], [30], [40] primarily rely on deep reinforcement learning (DRL) [5] to directly learn the best dispatching policies rather than accurately modeling taxi demand/supply. Specifically, DRL instructs the agent to accomplish a challenging task by trail and error in the process of interacting with the external environment [11]. In general, these recent works [9], [18], [30], [33], [40] characterize taxi dispatching problem with following five major components:

- *Agent:* Some works [9], [30], [40] view the dispatching center as an agent to interact with the external environment through a sequence of *observed states*, *actions*, and *rewards*. Besides, some other works [18], [33] adopt a multi-agent setting, where each available taxi is considered as an agent to partially observe the environment.
- *State* $s_t \in \mathcal{S}$*:* The perceived information at time t are represented as the state s_t, which is the input of an agent to determine the corresponding action. Specifically, state s_t in prior works [18], [30], [40] includes the zone-level spatial distribution of currently available taxis and ride requests, and other external factors like time of the day, day of the week, weather conditions, and *etc.*., [9].
- *Action* $a_t \in \mathcal{A}$*:* An action is a coordination solution made by the agent. Action a_t will instruct an available taxi to travel to a target zone (which may be in short of taxi supplies) or stay at its current zone. In order to reduce the coordination cost (*e.g.*, travel cost to the target zone), the action space $\mathcal{A}^v$ for each available taxi v could be defined as a set of discrete transits to any of its neighboring zones and staying where it is. Taking taxi v_2 locating in zone z_6 in Figure 1(a) as an example, its action space is $\mathcal{A}^{v_2} = \{z_1, z_2, z_3, z_5, z_6, z_7, z_9, z_{10}, z_{11}\}$.
- *Reward* $r_t \in \mathcal{R}$*:* Each applied action will affect the environment and thus get a feedback from the environment. Such a feedback can be quantified using a reward function $f_{\mathcal{R}}$ to calculate an immediate reward r_t. In general, $f_{\mathcal{R}}$ will take the revenues (*e.g.*, serving passengers at the target zone) and coordination costs into consideration. In DRL, the agent usually aims to maximize the long-term discount reward $\mathbf{Q}(\mathcal{S}, \mathcal{A})$ that is defined as

$$\mathbf{Q}(\mathcal{S}, \mathcal{A}) = \mathbb{E}[\sum_{t=0}^{\infty} \gamma \cdot r_t | s_0], \tag{1}$$

  where r_t is the reward of time t, s_0 is the initial state, and γ is the predefined discount factor.
- *Policy* π*:* As the core of an agent, policy π takes the state as the input to generate an action. By extensively interacting with the environment, the agent will learn a mapping between the states and proper actions. Considering the complexity of real-world problems, deep learning models, *e.g.*, deep neural networks [30], [35], are frequently used to learn the dispatching policy [5], [26].

C. Motivation

Although recent DRL based solutions [9], [18], [30], [40] have shown great advantages than traditional approaches [24], [25], [31], [47], [49], they are still not sufficiently efficient yet. Specifically, we observe at least three limitations of existing works, which will affect their efficiency and practicality.

1) Zone Formation With No Consideration of Road Connectivity: Despite the simplicity, most of existing works [9], [18], [30] divide a city area into grids with no consideration of the underlying road network. These grids, however, will lead to some improper action spaces due to the neglects of road connectivity. For example, an action may be infeasible if the target grid is occupied by a lake with no accessible roads. In addition, taxis may not timely arrive at the target grids when dispatching decisions are made with no consideration of real travel costs on the road network. Improper action spaces will not only harm the taxi dispatching performances, but also introduce unnecessary computation overheads.

2) Inadequate Coordination Among Taxis: Although multi-agent DRL setting can reduce the complexity by decomposing the dispatching task to each idle taxi, it makes the coordination among taxis more difficult. As a result, multi-agent DRL based solutions [18], [40] cannot adequately coordinate all agents to achieve the global taxi demand-supply balance. Meanwhile, other works [9], [30], which view the dispatching center as the agent, separately select an independent action for each available taxi, with no coordination as well. In fact, the action taken for one taxi would affect the decision-making of other taxis that are waiting for dispatching.

3) Incomplete Taxi Supply/Demand Information: Previous works [18], [29], [40] primarily rely on already known information of supplies (*i.e.*, currently idle taxis) and demands (*i.e.*, received yet unserved ride requests) for state representations, and exploit the long-term reward effect to implicitly perceive demands and supplies in the near future for making dispatching decisions. However, they need enormous states to describe the environment and thus introduce tremendous training and computation overheads. Although some works [9], [30] have explicitly considered future taxi demands that are predicted by some models, they still cannot derive the comprehensive and accurate taxi supply/demand information for taxi dispatching.

Figure 1 further illustrates above arguments. Based on the distribution of available taxis, the agent of existing approaches may generate two independent actions for taxi v_1 and v_2 by dispatching them to the same zone z_7 where passenger p_1 locates (as shown in the third column of Figure 1(b)). There is only one request in zone z_7, while the agent sends two idle taxi there. Such a dispatching, however, will result in a waste of resources (*e.g.*, energy and time) with no benefit for the taxi that finally gets no passenger. In fact, if the agent can predict the arrival of ride request p_2 in zone z_{10} at time t_1, a better dispatching plan would be that the agent dispatches taxi v_1 to zone z_7 and taxi v_2 to zone z_{10} (as shown in the forth column of Figure 1(b)). These dispatching decisions are well coordinated among taxis based on more comprehensive taxi demand/supply information, and thus would be more beneficial for taxi drivers, passengers, and the ride-hailing platform.

Challenges. To improve the recent advances, rich context information, including road connectivity, explicit coordination, and comprehensive supply/demand information, are desired to be incorporated into the DRL modeling for more effective and efficient taxi dispatching. However, it is non-trivial to realize such a system mainly due to following two challenges. First, it is challenging to accurately derive and represent these context information, *e.g.*, predicting future taxi demands and counting possible taxi supplies are difficult, since taxi demands/supplies actually are extremely dynamic. Second, considering the large number of taxis to operate in a city, it is necessary yet difficult to well refine both state space and action space. Previous works [9], [30] include many features (*e.g.*, taxi supply/demand and some external factors like weather conditions) into the state representation to minutely describe the environment, however, it leads to enormous state spaces and thus interminable training process of the DRL model. In addition, an appropriate action space should be defined for each idle taxi to produce effective dispatching while retaining the coordination among taxis.

IV. Design of *COX*

In this section, we first present the system overview of *COX*, and then elaborate the design of each component.

A. Design Overview

Figure 2 illustrates the system architecture of *COX*, which consists of three major modules, *i.e.*, *Context Acquisition*, *DQN Model*, and *Environment Simulator*. At high level, *COX* aims to derive a deep Q-network (DQN) model by extensively interacting with the environment simulator, which emulates a practical ride-hailing scenario based on real-world taxi data.

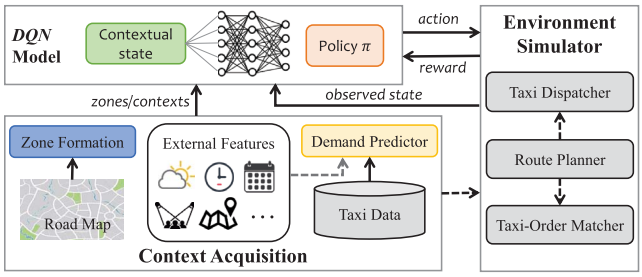


Fig. 2. The system architecture of *COX*.

Specifically, the *Context Acquisition* module acquires useful context features to represent the external environment. On one hand, it divides the road network rather than the city area into connectivity-aware zones for fine-grained taxi dispatching. On the other hand, it makes use of external features (*e.g.*, weather conditions, time of the day, day of the week, festival/event, points of interest, and so on) and historical taxi data to build a demand predictor, which can provide accurate zone-level future taxi demands. These contexts together with observed state delivered by the simulator form the contextual DRL states. The *DQN Model* module will train the taxi dispatching model via deep Q-network learning with a plenty of episodes. At each episode, contextual states, agent's coordination actions, and resultant rewards are used to train the DQN model for policy learning. In particular, both state representation and action space are refined by *COX* to optimize the training process. Lastly, the *Environment Simulator* module will execute *taxi dispatcher* and *taxi-order matcher*, both of which are supported by the *route planner* to find a travel route for each taxi on the road network. A ride request could be either served by an idle taxi nearby or be rejected by the ride-hailing platform if there are no idle taxis within a given time deadline.

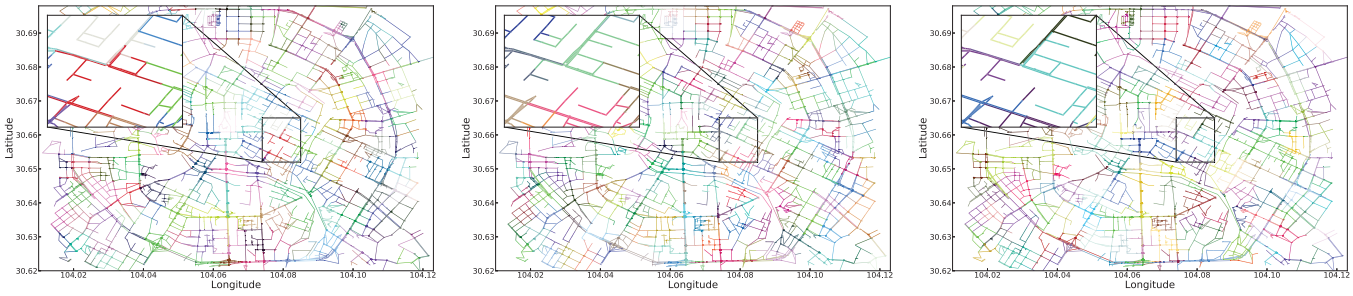
B. Context Acquisition

In order to derive rich context information, *COX* proposes a connectivity-aware road network clustering (*CARnet*) algorithm to form the zones, and builds a demand predictor to predict zone-level future taxi demands for better capturing the demand-supply gaps. We introduce them as follows.

1) Connectivity-Aware Zone Formation: In order to preserve road connectivity among zones, *COX* proposes to cluster on the road network rather than the city area to form zones $\mathbf{Z}$.

To this end, we formulate the road network as a directed graph $\mathtt{G}(\mathtt{V}, \mathtt{E})$, where each vertex in $\mathtt{V}$ represents a geo-location (*e.g.*, road intersection), and each edge $e \in \mathtt{E}$ represents a road segment, which is associated with a travel cost $cost(e)$[2] as the weight. Then, some clustering algorithms, *e.g.*, k-means [10] and spectral clustering [46], can be applied

[2]Function $cost(\cdot)$ can calculate the travel time on road network graph $\mathtt{G}$ for a given route or any two locations based on the distance and travel speeds.



(a) Grid (b) k-means (c) *CARnet*

Fig. 3. Demonstrations of applying (a) grid clustering; (b) k-means clustering; and (c) *CARnet* algorithm on the road network graph of Chengdu city, China. The vertices and edges belonging to different clusters are differentiated by colors. Since spectral clustering has similar results as k-means, we thus omit its results due to space limit.

on $\mathtt{G}$ to classify vertices into different clusters. Edges are assigned along with their source vertices as well. At last, each cluster will form a zone. These algorithms, however, primarily group vertices (and edges) based on their geographical locations, and thus even vertices and edges belonging to the same zone may still be disconnected. In addition, the resultant cluster sizes vary greatly, resulting in biased dispatching costs among zones. To preserve both inter-zones and intra-zone road connectivity, we instead present *CARnet* algorithm that works as follows.

① *Initializing clusters.* To obtain the uniformly distributed clusters, we firstly divide road network graph $\mathtt{G}$ using k grids. For each grid, we select the vertex u that is the most closest to the grid center as the *centroid* to initialize a cluster $\mathtt{C}$. Edges are classified into clusters along with their source vertices respectively. Each cluster $\mathtt{C}$ maintains following information: the centroid $\mathtt{C}.c$, vertex set $\mathtt{C}.V$, edge set $\mathtt{C}.E$, and total weight $\mathtt{C}.w$ that is the weight sum of edges belonging to this cluster. Next, we will classify all unassigned vertices, denoted by set $\mathbb{U}$, to clusters $\mathbb{C} = \{\mathtt{C}_i, i = 1, \cdots, k\}$.

② *Selecting target cluster.* We select the cluster $\mathtt{C}_i$ with the minimum total weight $\mathtt{C}_i.w$ in $\mathbb{C}$ to add new vertex/edge. The intuition behind is that we would like to balance the sizes of all clusters, so that dispatching actions executed on these clusters would be more operable and efficient.

③ *Adding unassigned vertex/edge.* We scan all unassigned vertices, and select the vertex $u \in \mathbb{U}$ with the minimum vertex-cluster cost. The vertex-cluster cost d_u^i, with respect to vertex u and cluster $\mathtt{C}_i$, is defined as the sum of travel cost from u to cluster centroid $\mathtt{C}_i.c$ and the minimum travel cost from u to any vertex in $\mathtt{C}_i.V$. If d_u^i is smaller than a threshold d_{vc}, we add u (and its associated edges) to cluster $\mathtt{C}_i$, and remove u out from $\mathbb{U}$; Otherwise, we move to the vertex with the second minimum vertex-cluster cost.

We repeat step ② and ③ until the set $\mathbb{U}$ becomes empty. In practice, due to the irregular road network graph structures, there may exist a few vertices that cannot be included into any cluster even after multiple iterations. We can gradually increase the threshold d_{vc} to relax the constraint so that these vertices (and edges) can be finally accepted by some clusters.

Figure 3 compares the zones formed by different clustering algorithms for the road network graph of Chengdu city, China. As shown in the zoom-in figures in Figure 3(a)(b), we see that vertices/edges of the same cluster generated by either grid clustering or k-means clustering are disconnected. In addition, we find that the zones formed by *CARnet* not only preserve the road connectivity, but also have similar cluster sizes, as shown in Figure 3(c). Such properties will benefit taxi dispatching.

2) Taxi Demand Predictor: Previous works [18] implicitly encode future taxi demand information into state representations, resulting in major issues like numerous state spaces and inefficient DRL model training. Thus *COX* separately builds a taxi demand predictor to decouple the two correlative tasks of taxi dispatching and demand prediction. There are three reasons for this design choice. First, demand prediction can be well handled by supervised machine learning models based on historical taxi orders. Second, we can migrate the complex external factors to the demand predictor, so as to keep the DRL states simple yet clear. Third, accurate future taxi demands will boost DRL modeling, since these information could greatly reduce the complexity of state spaces.

Recent advances on predicting taxi demands mainly resort to deep learning models, *e.g.*, convolutional neural networks (CNN) [48] and recurrent neural networks (RNN) [43]. In particular, recent taxi dispatching studies [9], [30] treat the whole city divided by regular grids as an image and utilize CNN models to predict taxi demands. CNN models have been successfully used to process Euclidean domain data that are with regular grid structures (*e.g.*, images and text) [6], while our connectivity-aware zones are quite different from grids but with irregular structures in the non-Euclidean domain. Thus, previous CNN model based predictors cannot work well here.

For taxi demand predictions over irregular zones, we model the zones as a graph, and exploit emerging graph convolutional network (GCN) [12] to derive accurate zone-level demands. GCN model has recently been proposed to process the non-Euclidean data, *e.g.*, graphs, and gained remarkable performances. Specifically, we define each zone as a vertex, and an edge is formed if two zones are immediately neighboring. Given the distribution of zones, we build a zone graph $\mathtt{G}^{\mathtt{Z}} = (\mathtt{Z}, \mathbf{A})$, where $\mathtt{Z}$ is the set of zone vertices and $\mathbf{A} \in \mathbb{R}^{|\mathtt{Z}| \times |\mathtt{Z}|}$ is the adjacency matrix, indicating the connections between vertices. In addition, we define the graph

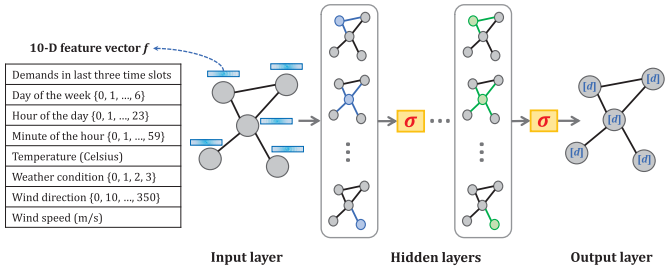


Fig. 4. The framework of GCN based taxi demand predictor and the structure of feature vector f. We adopt *ReLU* as the activation function σ.

Laplacian matrix as

$$\mathbf{L} = \mathbf{I} - \mathbf{D}^{-\frac{1}{2}}\mathbf{A}\mathbf{D}^{-\frac{1}{2}}, \tag{2}$$

where $\mathbf{I} \in \mathbb{R}^{|\mathbb{Z}|\times|\mathbb{Z}|}$ is an identity matrix, $\mathbf{D} \in \mathbb{R}^{|\mathbb{Z}|\times|\mathbb{Z}|}$ is the degree matrix, in which all of the diagonal elements are the degrees of vertices [12]. With matrix $\mathbf{L}$, GCN is able to capture non-Euclidean pair-wise correlations among distant zones on the taxi demands. This is achieved by the message passing function, which is defined as:

$$\mathbf{H}^{\ell+1} = \sigma(\mathbf{L}\,\mathbf{H}^{\ell}\,\mathbf{W}^{\ell}), \tag{3}$$

where $\mathbf{H}^{\ell}$ denotes the features in the ℓ-th layer, $\mathbf{W}^{\ell}$ is a trainable weight matrix for the ℓ-th layer, and σ is a non-linear activation function. In recent years, the rectified linear unit, *i.e.*, *ReLU* [28], has become the most popular activation function for many types of neural networks because a model that uses *ReLU* is easier to train and often achieves better performance. We thus adopt *ReLU* as the activation function. In order to incorporate most factors that will influence taxi orders in a zone, we construct a feature vector $\bar{f}_i \in \mathbb{R}^{10}$ for each zone z_i, which includes the numbers of taxi demands in previous three time slots, day of the week, hour of the day, minute of the hour, weather condition, temperature, wind direction, and wind speed. Therefore, the input of GCN, *i.e.*, $\mathbf{H}^0$, is summarized in a $|\mathbb{Z}| \times 10$ feature matrix, and Equation (3) captures local and global structural patterns for the final demand prediction. The predictor is trained independently from the DQN model, and thus the whole training overhead of *COX* will be alleviated.

Figure 4 illustrates the framework of our GCN based taxi demand predictor and the detailed structure of feature vector $\bar{f}_i$ for zone z_i. Specifically, we discretize the features of day of the week, hour of the day, minute of the hour, weather condition (*i.e.*, 0-*sunny*, 1-*rainy*, 2-*cloudy*, and 3-*others*), and wind direction, as illustrated in Figure 4. In addition, we normalize remaining features using the *Min-Max* normalization method. In fact, extra features on some special events, *e.g.*, accidents, can be included into the feature vector to further enhance the predictor's capability to handle unusual situations. The final output is the zone-level future taxi demands.

C. DQN Model

We consider the dispatching center as the agent, which can continuously track the real-time information (*e.g.*, location and status) of all taxis and ride requests, and thus could achieve the optimized taxi demand-supply balance. At the beginning of each time slot, the agent exploits the DQN model to generate an action for each available taxi based on the contextual states.

In practice, it is inefficient to dispatch an available taxi to a zone far away. Similarly, the states of distant zones also have ignoble impacts on the dispatching action of a taxi. We thus refine the state space and action space for all available taxis in the same zone, so as to reduce the computation complexity and enable *COX* work for city-scale ride-hailing services. Taxi dispatching among adjacent zones can be effective and fast to alleviate the demand-supply imbalances. Therefore, for a given zone z_i, we define the top-κ nearest zones as its neighbors $\mathbb{N}_{z_i} = \{z_j, j = 1, \cdots, \kappa\}$, where the distance between two zones is calculated as the travel cost on road network between their corresponding centroids. To avoid dispatching taxis to distant zones, we search neighbors for each zone only within the travel cost threshold d_{nb}. Furthermore, instead of making the same decision for all available taxis in the same zone [18] or generating actions for taxis independently [9], [30], *COX* takes actions for all available taxis sequentially, so as to guarantee the coordination among taxis. The intuition behind is that once an idle taxi has been sent to a specific zone, it has essentially changed the demand-supply environment that will affect the actions of other subsequent taxis. Based on these considerations, we design the DQN model of *COX* as follows.

1) Contextual State: Since we migrate all external factors, *e.g.*, weather condition, to the taxi demand predictor model, thus we can adopt a simple state representation that mainly contains zone-level demand-supply information. Specifically, the state of an available taxi's locating zone z_i includes the zone ID i, z_i's demand and supply data, and the demand/supply data of z_i's all neighbor zones. If z_i has insufficient ($< \kappa$) neighbors, the remaining fields are padded with zeros.

For each zone z_i, its taxi demand $\hat{D}_{z_i}^{t_j}$ of time slot t_j is provided by the demand predictor, while its taxi supply $\hat{P}_{z_i}^{t_j}$ can be comprehensively estimated as:

$$\hat{P}_{z_i}^{t_j} = N_{drop}^{t_j} + N_{stay}^{t_j} + N_{disp}^{t_{j-1}}, \tag{4}$$

where $N_{drop}^{t_j}$, $N_{stay}^{t_j}$, and $N_{disp}^{t_{j-1}}$ represent the number of taxis that drop off passengers in zone z_i at time t_j, the number of available taxis that prefer to stay in zone z_i at time t_j, and the number of taxis that are dispatched to zone z_i at time t_{j-1} and will arrive in zone z_i at time t_j, respectively.

As a concrete example, we illustrate the state s_t^z for zone z at time t in Equation (5), where we set $\kappa = 5$. Thus s_t^z includes the demand and supply data of z (in blue), the demand/supply information of z's 4 neighbors (in orange), and the remaining fields are padded with zeros (in gray).

$$s_t^z = [\,\mathbf{i},\ \mathbf{5},\ \mathbf{10},\ \mathbf{4},\ \mathbf{1},\ \mathbf{2},\ \mathbf{2},\ \mathbf{15},\ \mathbf{20},\ \mathbf{7},\ \mathbf{4},\mathbf{0},\ \mathbf{0}\,]. \tag{5}$$

2) Dispatching Action: Each available taxi has $(\kappa + 1)$ possible actions, each of which dispatches the taxi to a specific zone. Specifically, $a_t = i\ (0 < i \leq \kappa)$ indicates dispatching the taxi to the i-th neighbor zone of its locating zone at time t, while $a_t = 0$ suggests this taxi to stay at current zone at time t.

3) Immediate Reward: Since the actions are sequentially taken for all idle taxis, we thus calculate an immediate reward for each taxi separately according to its dispatching order. For the action that dispatches taxi x from zone z_i to target zone z_g at time t_{j-1}, we calculate a reward for taxi x at time t_j according to this action's impact on the supply-demand situations of both current zone and target zone. Hence, we define supply-demand ratio ω_{z_i} for zone z_i with respect to taxi x as:

$$\omega_{z_i} = \frac{P_{z_i}^{t_{j-1}}}{D_{z_i}^{t_{j-1}}}, \tag{6}$$

where $P_{z_i}^{t_{j-1}}$ and $D_{z_i}^{t_{j-1}}$ represent actual supplies and actual demands for zone z_i at time t_{j-1}. Specifically, $D_{z_i}^{t_{j-1}}$ can be observed by the agent at time t_j, and the agent will dynamically calculate $P_{z_i}^{t_{j-1}}$ for each dispatched taxi according to its dispatching order. Specifically, $P_{z_i}^{t_{j-1}}$ consists of the number of taxis that actually drop off passengers in zone z_i at time t_j, the number of idle taxis that have been dispatched to zone z_i before the action taken for taxi x at time t_{j-1}, and the number of idle taxis in zone z_i, which will be processed after dispatching taxi x. In particular, we set $\omega_{z_i} = +\infty$ if $D_{z_i}^{t_{j-1}} = 0$ for Equation (6).

For the action that dispatches an idle taxi from its locating zone z_i to target zone z_g, *COX* calculates an immediate reward r_t using the reward function $f_{\mathcal{R}}$ defined as Equation (7) based on ω_{z_i} and ω_{z_g}. In principle, if z_i is in short of taxi supplies, staying action will get a positive reward and other actions are penalized. If there are more supplies than demands in zone z_i, the action will get more rewards if target zone z_g has more demands than supplies. For the case when both current zone and nearby zones have sufficient supplies (*i.e.*, $\omega_{z_i} > 1$ and $\omega_{z_g} > 1$), dispatching an idle taxi out of its current zone will get a penalizing reward while staying action gets zero reward. *COX* implicitly encourages idle taxis to stay at their current zones to avoid unnecessary taxi dispatching in this case.

$$r_t = \begin{cases} 5 & 0 \le \omega_{z_i} \le 1 \ \&\ i == g, \\ -5 & 0 \le \omega_{z_i} \le 1 \ \&\ i \ne g, \\ \dfrac{1}{\omega_{z_g}} & \omega_{z_i} > 1 \ \&\ 0 \le \omega_{z_g} \le 1, \\ 0 & \omega_{z_i} > 1 \ \&\ \omega_{z_g} > 1 \ \&\ i == g, \\ -\omega_{z_g} & \omega_{z_i} > 1 \ \&\ \omega_{z_g} > 1 \ \&\ i \ne g. \end{cases} \tag{7}$$

Based on above modeling, we utilize powerful DQN model [26] to dynamically learn the best policy for taxi dispatching. As the core of DQN models, Q-learning is an off-policy temporal-difference learning approach and aims to obtain the maximum long-term discount reward $\mathbf{Q}(\mathcal{S}, \mathcal{A})$, as expressed in Equation (1). In particular, *COX* utilizes a deep neural network to approximate the Q function (see Figure 2). During the training phase, the total Q-value (*i.e.*, rewards) is updated as:

$$\mathbf{Q}^*(s,a) \leftarrow \mathbf{Q}(s,a) + \alpha[\, r + \gamma \max_{a^*} \mathbf{Q}(s,a^*) - \mathbf{Q}(s,a)\,], \tag{8}$$

where α is the learning rate and γ is the discount factor.

In order to address the instability problem of DQN training, we adopt two techniques: target network [37] and prioritized experience replay [32]. Specifically, the target network is a copy of the estimated value function that is kept frozen to serve as a stable target for a number of steps. During training, parameters of target network are updated to match policy network. In addition, experience replay memory stores experiences in the form of transition tuples, denoted as $(s_{t-1}, a_{t-1}, r_{t-1}, s_t)$ with states, actions, rewards, and successor states at some time index t, in a cyclic buffer, and thus enables the agent to sample from and train on the previously observed data. Instead of uniformly sampling on the memorized tuples, prioritizing samples based on temporal-difference (TD) error δ would be more effective for learning [32]. For each transition tuple, we compute its δ as

$$\delta = r_{t-1} + \gamma \max_{a^*} \mathbf{Q}(s_t, a^*) - \mathbf{Q}(s_{t-1}, a_{t-1}). \tag{9}$$

When updating Q-network weights, the mean-squared loss function $\mathcal{L}(\theta)$ is used to calculate the difference between the predicted Q-values and the target Q-values, *i.e.*,

$$\mathcal{L}(\theta) = \mathbb{E}\,[\,((r + \gamma \max_{a^*} \mathbf{Q}(s,a^*;\theta')) - \mathbf{Q}(s,a;\theta))^2\,], \tag{10}$$

where θ and θ' are the weights of behavior network and target network, respectively. In this equation, the optimal values are approximated with a target value $r + \gamma \max_{a^*} \mathbf{Q}(s,a^*;\theta')$, with the weights θ' that are kept from some previous iterations.

Algorithm 1 presents the pseudocode of double Q-learning with prioritized experience replay. At first, we initialize replay memory $\mathcal{M}$, and set both behavior Q-network's θ and target Q-network's θ' with random weights. Then we train the DQN model with a specified maximum episodes by exploiting our environment simulator (detailed in next subsection). In each episode, we use the taxi data of $|\mathbb{T}| = n$ time slots to train the model with total n steps. At each step, we conduct dispatching actions and store the transition tuples into $\mathcal{M}$ (line 9-11). The transition tuples are sampled with different priorities to update behavior Q-network weights θ (line 12-19), and we update target Q-network's weights θ' as θ for every 144 steps (line 20). Finally, we take both predicted demands and supply statistics into account to generate coordination actions for all available taxis (line 21-26). Note that actions are sequentially generated, so that *COX* can take an action for each taxi by referring to other taxis' actions to achieve better coordination.

In the algorithm, exponent ψ determines how much prioritization is used, with $\psi = 0$ corresponding to the uniform sampling case. The exponent β is used to adjust the importance of sampling weights, and the exponent η is a coefficient for updating behavior Q-network's weights. In this paper, we set these exponents as the default values in [32].

D. Environment Simulator

We design and implement a simulator that can emulate the external environment to train DRL algorithms based on real-world datasets. The simulator models the whole procedure of how a ride-hailing platform manages taxis and processes ride requests. Specifically, the simulator includes a *route planner* that will find a travel path on the road network for a

Algorithm 1 DQN With Prioritized Experience Replay

Input: mini-batch b, replay period B, exponents ψ, β, and η.
Initialize: memory $\mathcal{M} = \emptyset$ and size N, $\Delta = 0$, $p_* = 1$;
Initialize: behavior Q-network θ with random weights;
Initialize: target Q-network θ' with random weights;
for *epi* $= 1$ *to max-episodes* **do**
  Reset the simulator with initial state s_0;
  for *step* $t = 1$ *to* n **do**
    Execute taxi dispatching actions;
    for *each dispatched taxi* i **do**
      Observe state s_t^i and calculate reward r_t^i;
      Store tuple $(s_{t-1}^i, a_{t-1}^i, r_{t-1}^i, s_t^i)$ into $\mathcal{M}$;
    if $t \equiv 0$ *mod* B **then**
      for $i = 1$ *to* b **do**
        Sample transition tuple $i \sim P(i) = \frac{p_i^\psi}{\sum_j p_j^\psi}$;
        Compute importance-sampling weight $w_i = \frac{(N \times P(i))^{-\beta}}{\max_j w_j}$;
        Compute TD-error δ_i using Equation (9);
        Update transition priority $p_i \leftarrow |\delta_i|$;
        Accumulate weight-change $\Delta \leftarrow \Delta + w_i \delta_i \mathbf{Q}(s^i, a^i; \theta)$;
      Update $\theta \leftarrow \theta + \eta\Delta$, and reset $\Delta = 0$;
      Set $\theta' = \theta$ after replay period of 144 steps;
    Predict zone-level taxi demands for step $t + 1$;
    Create a random sequence $\mathcal{X}$ of all available taxis;
    for *each available taxi* $x \in \mathcal{X}$ **do**
      Observe state s^x;
      Generate an action a^x for taxi x given s^x;
      Update demand/supply statistics of taxi x's current zone and target zone;

taxi given its dispatching action or assigned order, a *taxi-order matcher* that assigns each ride request to an appropriate taxi, and a *taxi dispatcher* that executes a dispatching action for an available taxi according to some taxi dispatching policy. Therefore, this simulator serves as the training environment for different DRL algorithms, and also can be used for the realistic evaluations of various taxi dispatching approaches. We detail each component of the environment simulator as follows.

1) Real-World Dataset: We build the simulator using a public taxi dataset that is provided by Didi's GAIA initiative.[3] This dataset totally includes 7065907 taxi orders in the downtown area of Chengdu city, China, in November 2016. Each taxi order consists of a transaction ID, a taxi ID, pick-up/drop-off locations, and a release timestamp. We obtain the road network of Chengdu city from OpenStreetMap [2] and build the road network graph G (as shown in Figure 3), which consists of 214440 vertices and 466330 edges, covering an area of more than $70\,km^2$. Besides, we have downloaded the corresponding weather data (including temperature, weather condition, and wind direction/speed) via Internet as the external factors for building the taxi demand predictor.

2) Route Planner: For realistic evaluations, the route planner computes the shortest path between two locations using the *Dijkstra*'s algorithm [7] on road network graph G. It serves both taxi-order matcher and taxi dispatcher to estimate: (1) the arrival time of an occupied taxi that is delivering passengers to the destination, so that this taxi supply can be taken into account for future taxi demand-supply balance; (2) the time of passengers to wait for their assigned taxis; (3) the travel cost of dispatching an available taxi from its current location to the center position of target zone. For simplicity, we assume the constant taxi speed $15\,km/h$ as with previous works [52], and calculate the travel cost for a given route through $cost(\cdot)$.

Real-time traffic conditions, which can be derived from the transportation agency or inferred from traffic samplings by exploiting advanced traffic estimation methods [21], [23], could be incorporated to dynamically update the edge weights. Over the dynamic road network graph, some fast shortest path query algorithms [14] can be used to calculate more accurate travel costs. Since our dataset does not contain such traffic condition data, we thus consider a static road network graph as previous works [9], [18], [30], [33]. We leave the traffic-aware route planning as the future work.

3) Taxi-Order Matcher: When a new ride request arrives, the simulator assigns the closest idle taxi to serve it. The assigned taxi will travel to pick-up the passengers and then deliver them to the destination. If a ride request cannot be assigned with an idle taxi within Δt minutes (*i.e.*, we hope that each ride request can be served within a dispatching time slot), the request is instead *rejected*. In practice, taxi ride-hailing platforms usually expect to minimize the number of rejected requests.

4) Taxi Dispatcher: This component will execute the actions generated by the DQN model to dispatch each available taxi to the center of target zone. Meanwhile, it tracks the statuses of all taxis and ride requests to form observed states for the DQN model. Given the actual taxi demands and supplies in each zone, it will calculate the rewards for these actions taken in last time slot using Equation (7), serving as a feedback of the environment to the agent to update the dispatching policy.

V. Performance Evaluation

In this section, we conduct extensive experiments to evaluate the performance of *COX* using the environment simulator.

A. Experimental Setup

We will compare *COX* with six representative taxi dispatching approaches on some typical performance metrics.

1) Baseline Approaches: The baseline approaches for performance comparisons are described as follows.

- *Simulation*. Once a taxi drops off the passengers, it will stay in place to wait for a new ride request. This baseline simulates the scenario with no taxi dispatching strategy.
- *Random*. This baseline randomly dispatches an available taxi to one of the neighbor zones with a 50% probability.

[3]Didi's GAIA: https://outreach.didichuxing.com/research/opendata/.

- *Greedy*. This baseline dispatches an available taxi to the neighbor zone, which currently has the fewest idle taxis, in a greedy manner.
- *cDQN*. This baseline is one of state-of-the-art approaches. This approach dispatches idle taxis among zones using the multi-agent DRL models, where each taxi is regarded as an agent. It implicitly encodes future taxi demands into states, and explicitly includes the geographic context and the collaborative context into DRL modeling [18].
- *MOVI*. This baseline is one of state-of-the-art approaches as well. Its DRL modeling also explicitly incorporates future taxi demands, which are predicted by CNN models, for the demand-supply balances [30].
- *Adaptive cooperative rebalance* (*ACR*). This baseline could be viewed as a variant of our *COX*, which can also perceive comprehensive zone-level taxi demands/supplies but dispatches taxis in a greedy manner. For each zone z_i, it computes a supply-demand ratio ω_{z_i} using Equation (6) with the estimated supplies and predicted demands. Then it dispatches each available taxi in zone z_i to the neighbor zone that has the smallest supply-demand ratio.

All the baseline approaches form the zones using the grids. Since our *CARnet* algorithm initializes the clusters using grids as well, thus all approaches will operate on the same number of zones for fair comparisons.

2) Performance Metrics: We comparatively evaluate the performances of all approaches based on the following metrics.

(1) *Reject rate*, which is calculated as the ratio between the rejected ride requests (due to unavailability of idle taxis) and the total number of received ride requests.

(2) *Average repositions*, which is the average reallocation times of dispatching a taxi out of its current zone.

(3) *Average coordination cost*, which measures the average travel time to the target zone for all dispatched taxis.

(4) *Average waiting time*, which indicates the average time of all served ride requests waiting for their assigned taxis.

(5) *Total order values (TOV)*. The total order values are the revenues gained by all taxis for serving ride requests, where we approximate the revenue of a taxi order as the trip distance. For a clear comparison, the total order values of each approach is normalized by the total order values of *Simulation*.

3) Implementation: We implement *COX* and the six baseline approaches in Python 3.7.3 with Keras 2.3.1 and TensorFlow 1.15.0 for building various deep learning and DRL models. For evaluations, we keep the taxi data from the last week of November, 2016 for testing and all the rest as historical data for training the models, *e.g.*, CNN or GCN based demand predictors and DRL models of different approaches. On average, we have 246871 ride requests per day for the testing. For each ride request, its pick-up/drop-off locations are map matched [34] to the closest vertices of graph $\mathtt{G}$. In addition, we fix the total number of taxis, whose initial locations are randomly chosen from the vertices of graph $\mathtt{G}$. We set time slot size $\Delta t = 10$ minutes. Besides, we divide the entire service area with the grid size of $800\,m \times 800\,m$, and in total we have 192 zones. Since *cDQN* [18] performs better on smaller grids, we thus conduct extra experiments for *cDQN* using the grid size of $400\,m \times 400\,m$, with totally 768 zones. In particular, we denote its results on the smaller grids as *cDQN**.

We set *COX*'s parameters as follows. For *CARnet* algorithm, we set $d_{vc} = 30$ minutes to classify vertices into clusters, and search at most $\kappa = 7$ neighbors for each zone with threshold $d_{nb} = 10$ minutes. For demand predictor, we build the GCN model with 3 convolutional layers, each of which has 512 units, and use *ReLU* as the activation function. For the DQN model, we set both behavior Q-network and target Q-network with the same architecture, which consists of 4 fully connected layers with 400 units per layer. We use *ReLU* as the activation function as well. We follow the settings in [32] to configure the prioritized experience replay, and set $N = 3 \times 10^4$, $b = 256$, and $B = 12$. We set learning rate $\alpha = 10^{-4}$ and discount factor $\gamma = 0.9$. Furthermore, for each baseline approach we adopt the settings that can achieve its best performance.

All experiments are conducted on a powerful server with Intel Core i9-9900K CPU@3.60GHz, NAVIDA GeForce RTX 2080 Ti GPU, and 32GB memory. Each experiment setting is repeated 5 times and the average results are reported.

B. Performance Comparison

We compare *COX* with baseline approaches by varying the total number of taxis, *i.e.*, 6000, 9000, and 12000. The results are summarized in Table III, where for the given number of taxis the best result for each metric is marked in bold.

In general, more taxis could serve more ride requests, and thus both reject rate and passengers' waiting time of all the approaches can be largely reduced. From Table III, we find that *cDQN** outperforms *cDQN* on the metrics of both reject rate and coordination cost. This is because *cDQN* [18] prefers to work with smaller zones. Among all the approaches except *cDQN*, *Simulation* has the largest reject rate and on average introduces passengers' waiting time about 0.95 minutes. On the other hand, we find that *MOVI*, *ACR*, and *COX* generally have smaller reject rates and shorter passengers' waiting time. These results demonstrate that efficient taxi dispatching indeed helps taxis to approach potential taxi demands, and meanwhile can improve the taxi service quality (with reduced passengers' waiting time) and the ride-hailing platform's revenues (with increased *Normalized TOV*), as shown in Table III.

Compared to the two naive taxi dispatching approaches (*i.e.*, *Random* and *Greedy*), other advanced approaches (except *cDQN*) achieve much better performances on all the metrics. Compared to *Simulation*, *Random* and *Greedy* derive higher *Normalized TOV* at the cost of more average repositions. For example, *Random* and *Greedy* reposition each taxi with 11.23 times and 24.54 times, respectively, when we have 9000 taxis.

Among these DRL based solutions, *COX* beats the other two state-of-the-art approaches, *i.e.*, *cDQN*/*cDQN** and *MOVI*, with significant advantages on the four metrics of reject rate, average repositions, passengers' waiting time, and *Normalized TOV*. Note that *cDQN** has the smallest coordination cost

TABLE III

PERFORMANCE COMPARISONS OF DIFFERENT APPROACHES ON THE FIVE METRICS, WHERE THE UNITS FOR BOTH COORDINATION COST AND WAITING TIME ARE *Minutes*, N/A REPRESENTS *Not Available*, AND *cDQN** INDICATES THE RESULTS OF *cDQN* ON THE SMALLER GRIDS

# of taxis	*Approach*	*Reject rate*	*Average repositions*	*Coordination cost*	*Waiting time*	*Normalized TOV*
6000	*Simulation*	40.94%	N/A	N/A	1.10	100.00%
	Random	31.28%	8.65	6.79	0.96	116.16%
	Greedy	31.98%	18.72	6.77	1.83	115.64%
	cDQN	43.24%	10.17	6.29	1.11	96.10%
	*cDQN**	28.41%	14.20	**4.03**	1.23	122.01%
	MOVI	28.77%	8.76	6.30	0.98	120.46%
	ACR	28.06%	8.51	7.03	0.94	121.81%
	COX	**24.32%**	**7.04**	5.92	**0.68**	**127.71%**
9000	*Simulation*	30.62%	N/A	N/A	0.93	100.00%
	Random	19.36%	11.23	6.81	0.68	116.04%
	Greedy	21.91%	24.54	6.72	0.77	112.94%
	cDQN	37.18%	10.62	6.98	0.99	91.03%
	*cDQN**	19.32%	16.77	**4.00**	1.03	116.20%
	MOVI	16.40%	11.91	6.20	0.74	120.40%
	ACR	14.64%	9.64	6.98	0.62	122.98%
	COX	**11.48%**	**9.60**	5.60	**0.46**	**126.95%**
12000	*Simulation*	24.71%	N/A	N/A	0.82	100.00%
	Random	13.87%	13.03	6.85	0.52	114.21%
	Greedy	16.13%	28.08	6.62	0.64	111.67%
	cDQN	34.04%	9.43	6.84	0.91	88.03%
	*cDQN**	12.24%	16.15	**4.04**	0.85	116.31%
	MOVI	10.28%	11.81	6.15	0.60	119.09%
	ACR	8.67%	9.46	6.88	0.44	121.10%
	COX	**7.15%**	**8.31**	5.63	**0.35**	**122.77%**

because it dispatches taxis among the smaller grids.[4] However, the finer granularity of dispatching zones introduces much more computation overheads as *cDQN* enlarges the action space by 4 ($= \frac{768}{192}$) times. Compared to *cDQN** and *MOVI*, on average *COX* reduces reject rate by 5.67% and 4.17%, respectively, reduces passengers' waiting time by 52.96% and 36.71%, respectively, and improves the *Normalized TOV* by 7.64% and 5.83%, respectively. These statistics indicate that *COX* has taken more effective dispatching actions than the other two DRL based approaches. It can be explained as that *COX* is able to accurately localize the zones with insufficient taxi supplies, and then accordingly reallocate nearby idle taxis there, which is proved by our case study later.

It is interesting to find that *ACR* outperforms *cDQN*/*cDQN** and *MOVI* in most cases. This is possibly because the comprehensive taxi demand/supply information help *ACR* to better understand the demand-supply gaps for decision-making. *COX* performs better than *ACR*, mainly because the DQN model can learn a wiser dispatching policy than *ACR*'s greedy manner.

Last but not the least, *COX* has the smallest average coordination cost and passengers' waiting time among all approaches except *cDQN** (as it runs on smaller grids). The reason behind is that *COX* dispatches taxis among the connectivity-preserved zones rather than grids that omit the underlying road network. Our *CARnet* algorithm restrains the travel cost between any two vertices within a cluster, so that passengers' waiting time is potentially bounded. In addition, *COX* refines the action space for each zone by restricting its neighbor zones within a travel cost threshold d_{nb}, where we set it as Δt so that dispatched taxis can serve requests in the next time slot.

[4]We run *COX* on the smaller grids as well, and find *COX* still significantly outperforms *cDQN** on all metrics. For example, when we have 12000 taxis, *COX* achieves reject rate as 9.88%, average repositions as 7.15, coordination cost as 3.31, waiting time as 0.76, and *NTOV* as 119.39%.

In summary, the results in Table III demonstrate that proactive taxi dispatching benefits both the platform and passengers. Furthermore, rich contexts derived by *COX* indeed help DRL modeling better understand the external environment and thus learn much better dispatching policies.

C. Evaluations of COX Design

Next, we conduct experiments to evaluate the design choices of *COX* by comparing with some alternative designs.

1) Computation Efficiency: To investigate *COX*'s efficiency for city-scale dispatching, we run *COX* with 12000 taxis and record the execution time for each component. The experiment results show that on average *COX* can complete the simulation of each time slot within 3.53 seconds. More specifically, GCN based demand predictor takes 14.24 milliseconds to perform demand predictions for the next time slot, and the DQN model takes about 2.52 seconds to make dispatching decisions for all available taxis within a time slot. In addition, *COX* uses 0.99 seconds to simulate taxi-order matching and route planning for serving all ride requests in a time slot. The results demonstrate that *COX* can perform real-time taxi dispatching at city scale.

2) Impact of Zone Formations: We compare our *CARnet* algorithm with the grid based zone formation [9], [18], [30] and two classical clustering algorithms, *i.e.*, k-means [10] and spectral clustering [46]. More specifically, we use each of these algorithms to form the zones and then run *COX* on them.

Since *Normalized TOV* can be inferred by the reject rate and all methods have similar average repositions (because we use the same DQN model to process the same demands/supplies), we thus only report their results on the metrics of reject rate and waiting time in Figure V-C.2. We find that the zones derived by clustering algorithms generally lead to lower reject

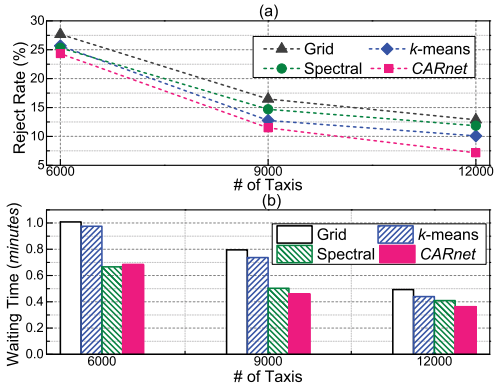


Fig. 5. Comparisons of different zone formation methods on the metric of (a) reject rate; and (b) waiting time.

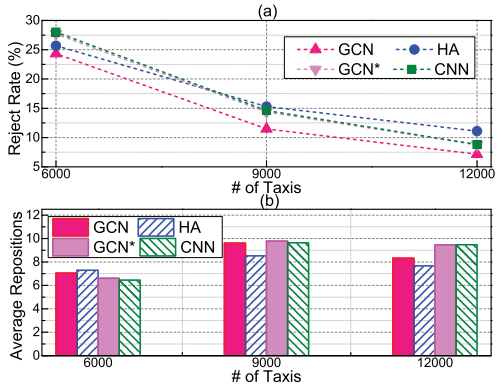


Fig. 6. Comparisons of different demand predictors on the metric of (a) reject rate; and (b) average repositions.

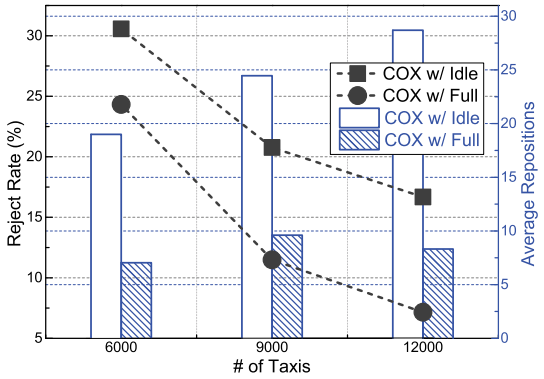


Fig. 7. Comparisons of partial and full taxi supply information on the metrics of reject rate and average repositions.

rates (see Figure V-C.2(a)) and shorter passengers' waiting time (see Figure V-C.2(b)) than the grids. Therefore, it is necessary to consider the underlying road network structure when making taxi dispatching decisions. *CARnet* performs even better than k-means and spectral clustering by further reducing reject rate and waiting time, *e.g.*, averagely reducing by 2.26% and 0.16 minutes when we have 9000 taxis. The reason may be that *CARnet*, different from the two algorithms, can preserve both inter-zones and intra-zone road connectivity, just as shown in Figure 3, and thus is beneficial for taxi reallocation.

3) Impact of Demand Prediction: COX builds a taxi demand predictor that best fits our irregular zones with the GCN model [12]. To evaluate its performance, we compare our predictor with a baseline predictor named *HA* and *MOVI*'s CNN based predictor [30]. Specifically, *HA* adopts the same zones as *COX*, and predicts a zone's demands of next time slot by averaging the numbers of taxi orders at the same time slots of all previous days. The CNN based predictor divides the city into grids, and applies the CNN technique to these grids, which are viewed as an image, for predicting grid-level demands.

At first, we compare their prediction accuracy based on the root mean squared error (*RMSE*) over the predictions of all zones and time slots using the testing data. Experiments show that the *RMSE*s of GCN predictor, *HA*, and CNN predictor are 3.81, 5.81, and 5.66, respectively. CNN slightly outperforms *HA*, while GCN derives much more accurate predictions. In particular, GCN predictor achieves low *RMSE* as 3.90 in the rainy days. It implies that *COX* is able to handle some unusual situations like bad weather conditions.

Then we individually input their demand predictions into *COX*'s DQN model for taxi dispatching, and present their impacts on the metrics of reject rate and average repositions in Figure 6. In this experiment, we also apply GCN on the same grids as CNN for comparison, and its results are denoted by GCN* in Figure 6. As shown in Figure 6(a), GCN* has similar reject rates as CNN, and they both perform a bit better than *HA*. GCN outperforms the three models with the lowest reject rates. It proves that GCN model indeed captures demand correlations among the irregular zones. Figure 6(b) shows that GCN has the largest average repositions, while GCN* and CNN have similar results on this metric. We find that average repositions have an inverse relation with the reject rate, *i.e.*, high average repositions usually correspond to low reject rate. It is possibly because GCN provides more accurate demand predictions, and *COX* accordingly makes more effective dispatching actions to meet these future demands.

4) Impact of Supply Information: In addition to the currently idle taxis in zone z, *COX* further considers the potential taxi supplies, including the taxis that will drop off passengers in zone z right away and the taxis that have been dispatched to zone z, to derive more comprehensive supply information, as expressed in Equation (4). However, previous works [9], [18], [30] only encode the number of currently idle taxis into DRL states for generating actions. In Figure V-C.3, we compare the performances of *COX* with partial supply information (*i.e.*, "*COX w/ Idle*") and with full supply information (*i.e.*, "*COX w/ Full*"). It reports that the full supply information can reduce the reject rate by 6.25%, 9.29%, and 9.52% for the three settings of total taxis, respectively. Moreover, the reject rate reductions are achieved with much fewer repositions, *e.g.*, on average *COX w/ Full* has reduced the average repositions by 71.03% with 12000 taxis. The comparisons imply that comprehensive supply information enable *COX* have a clear understanding of the external environment to take the right actions.

5) Case Study: To understand the rationality of how *COX* takes dispatching actions, we log the intermediate states of all zones in a typical workday. For each zone z_i and a given time slot t_j, the corresponding record includes the demand predictions predicted at time t_{j-1}, currently idle taxis at time

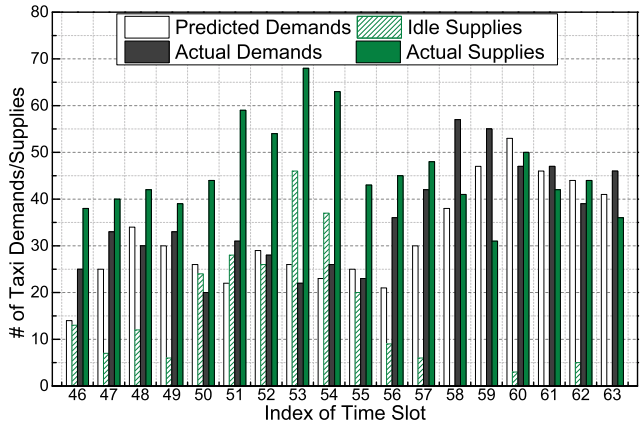


Fig. 8. The taxi demand and supply statuses of a zone in the peak hours of a typical workday.

t_{j-1}, and actual demands and supplies observed at time t_j. As a study case, we present the state information of a randomly selected zone during the peak hours (*i.e.*, 7:40AM-10:30AM) in Figure 8. In the peak hours, there are many ride requests in this zone, and we find the predicted demands are quite close to the actual demands. From Figure 8, we see in most time slots this zone has insufficient taxi supplies to serve the potential taxi demands. Thanks to the contextual DQN model, *COX* can perceive this situation and proactively dispatches available taxis to this zone in advance, where we see the actual supplies is sufficiently enough for the actual demands. By comparing idle supplies and actual supplies, we find *COX* actually reallocates quite a few taxis to this busy zone. After the 58-th time slot (about 9:40AM), taxi demand-supply gap becomes increasingly larger, as there are even no idle taxis in some time slots. By perceiving this situation, *COX* still tries to allocate many available taxis to this zone (referring to the difference between idle supplies and actual supplies). Although these dispatched taxis cannot serve all the demands, *COX* still serves most of them and thus potentially reduces the reject rate through effective taxi dispatching.

VI. Conclusion and Future Works

In this paper, we present *COX* to improve the existing taxi dispatching approaches by incorporating rich context information. Specifically, *COX* divides the road network graph into connectivity-preserved zones and encodes comprehensive taxi demand/supply information into DRL state representation to derive effective and coordinated actions. In addition, we implement a realistic environment simulator to train and evaluate *COX* using a large real-world taxi dataset. Extensive experiments demonstrate that *COX* significantly outperforms the state-of-the-art approaches, *e.g.*, on average reducing reject rate and passengers' waiting time by 4.92% and 44.84%, respectively, while improving the total order values by 6.74%.

As future works, we plan to improve *COX*'s capability on handling unusual situations, *e.g.*, special events and accidents. The nature of unusual situations means that available information of these events are usually sparse, and how to train a deep learning model from such sparse data calls for research efforts. In addition, we would like to design a unified simulator, which can be used to evaluate different taxi dispatching approaches in a wide range of scenarios. Specifically, the scenarios should cover different road networks, different number of vehicles, various amounts of ride requests, and others.

References

[1] *Didi Chuxing*. Accessed: Jul. 7, 2020. [Online]. Available: https://www.didiglobal.com/
[2] *OpenStreetMap*. Accessed: Jul. 7, 2020. [Online]. Available: http://www.openstreetmap.org/
[3] *Uber*. Accessed: Jul. 7, 2020. [Online]. Available: https://www.uber.com
[4] A. O. Al-Abbasi, A. Ghosh, and V. Aggarwal, "DeepPool: Distributed model-free algorithm for ride-sharing using deep reinforcement learning," *IEEE Trans. Intell. Transp. Syst.*, vol. 20, no. 12, pp. 4714–4727, Dec. 2019.
[5] K. Arulkumaran, M. P. Deisenroth, M. Brundage, and A. A. Bharath, "Deep reinforcement learning: A brief survey," *IEEE Signal Process. Mag.*, vol. 34, no. 6, pp. 26–38, Nov. 2017.
[6] M. Defferrard, X. Bresson, and P. Vandergheynst, "Convolutional neural networks on graphs with fast localized spectral filtering," in *Proc. NIPS*, 2016, pp. 3844–3852.
[7] E. W. Dijkstra, "A note on two problems in connexion with graphs," *Numerische Math.*, vol. 1, no. 1, pp. 269–271, Dec. 1959.
[8] X. Geng *et al.*, "Spatiotemporal multi-graph convolution network for ride-hailing demand forecasting," in *Proc. AAAI*, 2019, pp. 3656–3663.
[9] S. He and K. G. Shin, "Spatio-temporal capsule-based reinforcement learning for mobility-on-demand network coordination," in *Proc. World Wide Web Conf. (WWW)*, 2019, pp. 2806–2813.
[10] A. K. Jain, "Data clustering: 50 years beyond K-means," *Pattern Recognit. Lett.*, vol. 31, no. 8, pp. 651–666, Jun. 2010.
[11] L. P. Kaelbling, M. L. Littman, and A. W. Moore, "Reinforcement learning: A survey," *J. Artif. Intell. Res.*, vol. 4, no. 1, pp. 237–285, Jan. 1996.
[12] T. N. Kipf and M. Welling, "Semi-supervised classification with graph convolutional networks," in *Proc. ICLR*, 2017, pp. 1–13.
[13] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," *Nature*, vol. 521, pp. 436–444, May 2015.
[14] L. Li, M. Zhang, W. Hua, and X. Zhou, "Fast query decomposition for batch shortest path processing in road networks," in *Proc. IEEE 36th Int. Conf. Data Eng. (ICDE)*, Apr. 2020, pp. 1189–1200.
[15] M. Li *et al.*, "Efficient ridesharing order dispatching with mean field multi-agent reinforcement learning," in *Proc. World Wide Web Conf.*, May 2019, pp. 983–994.
[16] Y. Li, Y. Zheng, and Q. Yang, "Efficient and effective express via contextual cooperative reinforcement learning," in *Proc. 25th ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining*, Jul. 2019, pp. 510–519.
[17] Z. Liao, "Real-time taxi dispatching using global positioning systems," *Commun. ACM*, vol. 46, no. 5, p. 81, 2003.
[18] K. Lin, R. Zhao, Z. Xu, and J. Zhou, "Efficient large-scale fleet management via multi-agent deep reinforcement learning," in *Proc. 24th ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining*, Jul. 2018, pp. 1774–1783.
[19] L. Liu, Z. Qiu, G. Li, Q. Wang, W. Ouyang, and L. Lin, "Contextualized spatial–temporal network for taxi origin-destination demand prediction," *IEEE Trans. Intell. Transp. Syst.*, vol. 20, no. 10, pp. 3875–3887, Oct. 2019.
[20] Z. Liu, Z. Gong, J. Li, and K. Wu, "Mobility-aware dynamic taxi ridesharing," in *Proc. IEEE 36th Int. Conf. Data Eng. (ICDE)*, Apr. 2020, pp. 961–972.
[21] Z. Liu, Z. Li, M. Li, W. Xing, and D. Lu, "Mining road network correlation for traffic estimation via compressive sensing," *IEEE Trans. Intell. Transp. Syst.*, vol. 17, no. 7, pp. 1880–1893, Jul. 2016.
[22] Z. Liu, Z. Li, K. Wu, and M. Li, "Urban traffic prediction from mobility data using deep learning," *IEEE Netw.*, vol. 32, no. 4, pp. 40–46, Jul. 2018.
[23] Z. Liu, P. Zhou, Z. Li, and M. Li, "Think like a graph: Real-time traffic estimation at city-scale," *IEEE Trans. Mobile Comput.*, vol. 18, no. 10, pp. 2446–2459, Oct. 2019.
[24] F. Miao, S. Han, A. M. Hendawi, M. E. Khalefa, J. A. Stankovic, and G. J. Pappas, "Data-driven distributionally robust vehicle balancing using dynamic region partitions," in *Proc. 8th Int. Conf. Cyber-Phys. Syst.*, Apr. 2017, pp. 261–271.
[25] F. Miao *et al.*, "Taxi dispatch with real-time sensing data in metropolitan areas: A receding horizon control approach," *IEEE Trans. Autom. Sci. Eng.*, vol. 13, no. 2, pp. 463–478, Apr. 2016.
[26] V. Mnih *et al.*, "Human-level control through deep reinforcement learning," *Nature*, vol. 518, no. 7540, p. 529, 2015.

[27] L. Moreira-Matias, J. Gama, M. Ferreira, J. Mendes-Moreira, and L. Damas, "Predicting taxi–passenger demand using streaming data," *IEEE Trans. Intell. Transp. Syst.*, vol. 14, no. 3, pp. 1393–1402, Sep. 2013.
[28] V. Nair and G. E. Hinton, "Rectified linear units improve restricted Boltzmann machines," in *Proc. ICML*, 2010, pp. 807–814.
[29] M. Nourinejad and M. Ramezani, "Developing a large-scale taxi dispatching system for urban networks," in *Proc. IEEE 19th Int. Conf. Intell. Transp. Syst. (ITSC)*, Nov. 2016, pp. 441–446.
[30] T. Oda and C. Joe-Wong, "MOVI: A model-free approach to dynamic fleet management," in *Proc. IEEE Conf. Comput. Commun. (INFOCOM)*, Apr. 2018, pp. 2708–2716.
[31] M. Qu, H. Zhu, J. Liu, G. Liu, and H. Xiong, "A cost-effective recommender system for taxi drivers," in *Proc. 20th ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining*, 2014, pp. 45–54.
[32] T. Schaul, J. Quan, I. Antonoglou, and D. Silver, "Prioritized experience replay," in *Proc. ICLR*, 2016, pp. 1–21.
[33] K. Tian Seow, N. Hai Dang, and D.-H. Lee, "A collaborative multiagent taxi-dispatch system," *IEEE Trans. Autom. Sci. Eng.*, vol. 7, no. 3, pp. 607–616, Jul. 2010.
[34] Z. Shen, W. Du, X. Zhao, and J. Zou, "DMM: Fast map matching for cellular data," in *Proc. 26th Annu. Int. Conf. Mobile Comput. Netw.*, Sep. 2020, pp. 1–14.
[35] Z. Shen, K. Yang, W. Du, X. Zhao, and J. Zou, "DeepAPP: A deep reinforcement learning framework for mobile application usage prediction," in *Proc. 17th Conf. Embedded Networked Sensor Syst.*, Nov. 2019, pp. 153–165.
[36] Y. Tong *et al.*, "The simpler the better: A unified approach to predicting original taxi demands based on large-scale online platforms," in *Proc. 23rd ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining*, Aug. 2017, pp. 1653–1662.
[37] H. Van Hasselt, A. Guez, and D. Silver, "Deep reinforcement learning with double Q-learning," in *Proc. AAAI*, 2016, pp. 2094–2100.
[38] M. Veres and M. Moussa, "Deep learning for intelligent transportation systems: A survey of emerging trends," *IEEE Trans. Intell. Transp. Syst.*, vol. 21, no. 8, pp. 3152–3168, Aug. 2020.
[39] Z. Wang, Z. Qin, X. Tang, J. Ye, and H. Zhu, "Deep reinforcement learning with knowledge transfer for online rides order dispatching," in *Proc. IEEE Int. Conf. Data Mining (ICDM)*, Nov. 2018, pp. 617–626.
[40] J. Wen, J. Zhao, and P. Jaillet, "Rebalancing shared mobility-on-demand systems: A reinforcement learning approach," in *Proc. IEEE 20th Int. Conf. Intell. Transp. Syst. (ITSC)*, Oct. 2017, pp. 220–225.
[41] R. Xie, X. Jia, and K. Wu, "Adaptive online decision method for initial congestion window in 5G mobile edge computing using deep reinforcement learning," *IEEE J. Sel. Areas Commun.*, vol. 38, no. 2, pp. 389–403, Feb. 2020.
[42] X. Xie, F. Zhang, and D. Zhang, "PrivateHunt: Multi-source data-driven dispatching in for-hire vehicle systems," *ACM Interact., Mobile, Wearable Ubiquitous Technol.*, vol. 2, no. 1, pp. 45:1–45:26, 2018.
[43] J. Xu, R. Rahmatizadeh, L. Boloni, and D. Turgut, "Real-time prediction of taxi demand using recurrent neural networks," *IEEE Trans. Intell. Transp. Syst.*, vol. 19, no. 8, pp. 2572–2581, Aug. 2018.
[44] J. Xu, R. Rahmatizadeh, L. Boloni, and D. Turgut, "Taxi dispatch planning via demand and destination modeling," in *Proc. IEEE 43rd Conf. Local Comput. Netw. (LCN)*, Oct. 2018, pp. 377–384.
[45] Z. Xu *et al.*, "Large-scale order dispatch in on-demand ride-hailing platforms: A learning and planning approach," in *Proc. 24th ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining*, Jul. 2018, pp. 905–913.
[46] D. Yan, L. Huang, and M. I. Jordan, "Fast approximate spectral clustering," in *Proc. 15th ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining*, 2009, pp. 907–916.
[47] H. Yang, S. C. Wong, and K. I. Wong, "Demand–supply equilibrium of taxi services in a network under competition and regulation," *Transp. Res. B, Methodol.*, vol. 36, no. 9, pp. 799–819, Nov. 2002.
[48] H. Yao *et al.*, "Deep multi-view spatial-temporal network for taxi demand prediction," in *Proc. AAAI*, 2018, pp. 2588–2595.
[49] 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.
[50] L. Zhang *et al.*, "A taxi order dispatch model based on combinatorial optimization," in *Proc. 23rd ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining*, Aug. 2017, pp. 2151–2159.
[51] B. Zhao, P. Xu, Y. Shi, Y. Tong, Z. Zhou, and Y. Zeng, "Preference-aware task assignment in on-demand taxi dispatching: An online stable matching approach," in *Proc. AAAI*, 2019, pp. 2245–2252.
[52] L. Zheng, L. Chen, and J. Ye, "Order dispatch in price-aware ridesharing," *Proc. VLDB Endowment*, vol. 11, no. 8, pp. 853–865, Apr. 2018.
[53] M. Zhou *et al.*, "Multi-agent reinforcement learning for order-dispatching via order-vehicle distribution matching," in *Proc. 28th ACM Int. Conf. Inf. Knowl. Manage.*, Nov. 2019, pp. 2645–2653.

Zhidan Liu (Member, IEEE) received the Ph.D. degree in computer science and technology from Zhejiang University, Hangzhou, China, in 2014. After that, he worked as a Research Fellow with Nanyang Technological University, Singapore. In 2017, he joined Shenzhen University, Shenzhen, China, as an Assistant Professor. His research interests include distributed sensing and mobile computing, big data analytics, the Internet of Things, and urban computing.

Jiangzhou Li received the B.S. degree in software engineering from Qingdao University, Qingdao, China, in 2018. He is currently a master's degree with the College of Computer Science and Software Engineering, Shenzhen University, under the supervision of Dr. Z. Liu. His research interests include big data, including data analysis, urban computing, and reinforcement learning.

Kaishun Wu (Member, IEEE) received the Ph.D. degree in computer science and engineering from The Hong Kong University of Science and Technology (HKUST), Hong Kong, in 2011.

After that, he worked as a Research Assistant Professor with HKUST. In 2013, he joined Shenzhen University as a Distinguish Professor. He has coauthored two books and published more than 100 high-quality research papers in international leading journals and primer conferences, such as the IEEE TRANSACTIONS ON MOBILE COMPUTING (TMC), the IEEE TRANSACTIONS ON PARALLEL AND DISTRIBUTED SYSTEMS (TPDS), ACM MobiCom, and IEEE INFOCOM. He is the Inventor of six U.S., and more than 100 Chinese pending patent. He is an IET Fellow. He received the 2012 Hong Kong Young Scientist Award, the 2014 Hong Kong ICT awards: Best Innovation, and the 2014 IEEE ComSoc Asia-Pacific Outstanding Young Researcher Award.