Spatio-Temporal Capsule-Based Reinforcement Learning for Mobility-on-Demand Coordination

Suining He, *Member, IEEE* and Kang G. Shin, *Life Fellow, IEEE*

Abstract—As an alternative means of convenient and smart transportation, mobility-on-demand (MOD), typified by online ride-sharing and connected taxicabs, has been rapidly growing and spreading worldwide. The large volume of complex traffic and the uncertainty of market supplies/demands have made it essential for many MOD service providers to *proactively* dispatch vehicles towards ride-seekers. To meet this need effectively, we propose STRide, an MOD coordination learning mechanism reinforced spatio-temporally with capsules. We formalize the adaptive coordination of vehicles into a reinforcement learning framework. STRide incorporates spatial and temporal distributions of supplies (vehicles) and demands (ride requests), customers' preferences and other external factors. A novel spatio-temporal capsule neural network is designed to predict the provider's rewards based on MOD network states, vehicles and their dispatch actions. This way, the MOD platform adapts itself to the supply-demand dynamics with the best potential rewards. We have conducted extensive data analytics and experimental evaluation with five large-scale datasets (~27 million rides from Uber, NYC/Chicago Taxis, Didi and Car2Go). STRide is shown to outperform state-of-the-arts, substantially reducing request-rejection rate and passenger waiting time, and also increasing the service provider's profits.

Index Terms—Mobility-on-demand, ride-sharing platform, human and vehicle mobility, coordination, smart transportation, reinforcement learning, spatio-temporal capsule network, smart city

1 Introduction

Accelerating urbanization and rising population have led to a rapid growth of vehicle fleet size, making it essential to tackle traffic congestion and air pollution problems. By integrating online information of traffic demands and supplies, transit network operation and communication, cooperative mobility-on-demand (MOD) systems, such as Uber, Lyft, Didi and connected taxicabs, have created unprecedented transportation alternatives. With an estimated compound annual growth rate of 19.81 percent since 2017, the value of global MOD market is expected to grow to USD$276 billion by 2025 [1].

Given its significant economic and social values, we must coordinate the MOD operations, i.e., dispatching (matching) supplies—available vehicles/drivers—towards (with) demands—passengers/requesters/riders. A coordination policy/strategy usually uses current observations to determine where and when to relocate vehicles for maximization of MOD service providers' profit and satisfaction of riders' desire/requirement. During each phase of coordination, the idle/vacant MOD vehicles are dispatched towards different service zones (discretization of a city map), and matched with their nearest requesters. The resultant rides "connect" the zones and thus form an MOD *network*.

- *S. He is with the Department of Computer Science and Engineering, The University of Connecticut, Storrs, CT 06269 USA. E-mail: suining.he@uconn.edu.*
- *K.G. Shin is with the Department of Electrical Engineering and Computer Science, The University of Michigan, Ann Arbor, MI 48109 USA. E-mail: kgshin@umich.edu.*

Manuscript received 25 Mar. 2019; revised 25 Apr. 2020; accepted 30 Apr. 2020. Date of publication 4 May 2020; date of current version 3 Feb. 2022. (Corresponding author: Suining He.) Recommended for acceptance by L. Xiong. Digital Object Identifier no. 10.1109/TKDE.2020.2992565

However, increasingly complex urban traffic networks and mis-coordination of demand-supply dynamics often under/over-serve many service zones, thus degrading the providers' profitability, service quality, passenger satisfaction and drivers' enthusiasm. As illustrated in Fig. 1, the problem is particularly severe during rush hours when people travel in similar directions between their home and work or recreational/commercial zone. A survey of ride-hailing customers in China [2] has shown that 81.7 percent of the respondents experience more difficulties in hailing vehicles in 2017 than in 2016, including up to 129.2 percent longer waiting times. In 2019, Uber drivers in Seattle, Washington report there were days they could spend up to 16 hours on call in order to get a few passengers [3]. Despite numerous efforts and enormous historical ride-data available, designing an MOD coordination mechanism remains to be difficult for the following reasons.

First, due to urbanization and MOD market expansion, the static coordination based on old data in some zones cannot be applied and scaled throughout the dynamically-changing MOD network, thus calling for a new adaptive mechanism. Second, coordinating complex MOD supplies and demands exhibits sequential and long-term effects. A single vehicle dispatching action may introduce profound consequences to the environment and other vehicles, that cannot be easily foreseen by a heuristic coordination mechanism. Third, there usually exists *multi-level periodicity* within traffic routines, commute patterns, annual festival events, and ride preferences. The weather can also affect the ride requests, which, for example, will surge during rainy hours. Without comprehensive modeling of these factors, the conventional coordination methods cannot easily and accurately capture the repeating patterns.

1041-4347 © 2020 IEEE. Personal use is permitted, but republication/redistribution requires IEEE permission.
See https://www.ieee.org/publications/rights/index.html for more information.

Authorized licensed use limited to: University of Michigan Library. Downloaded on February 07,2022 at 02:22:07 UTC from IEEE Xplore. Restrictions apply.

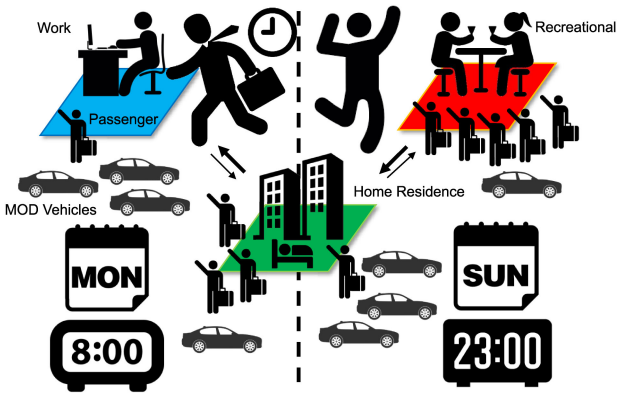


Fig. 1. Illustration of MOD demand and supply. For example, on a weekday morning there are more residence-to-work commutes while a weekend night meets travels back to residence, leading to demand-supply imbalance at origins/destinations.

We meet these challenges by proposing STRide, deep reinforcement learning (RL) based on spatio-temporal capsules for coordinating the MOD network. Specifically, we design an RL framework with a data-driven emulator, adjusting the coordination policy with an online *self-adapting* mechanism. A *state* in this RL represents the spatial MOD demand, supply and external temporal factors, and each *action* there represents the zones to which vehicles can be relocated. The multi-objective reward function characterizes the platform profitability, cost and service coverage, whose long-term *values* are maximized by STRide. The framework also takes into account the contextual scopes of vehicles and travel time estimation to emulate a fine-grained learning environment.

Since it is difficult to specify the long-term coordination effects on future demands and supplies, we design a novel capsule-based neural network to comprehensively learn the relationship between observed states, coordination actions and potential rewards. Conventional scalar-based machine learning models like CNNs (convolutional neural networks) [4] communicate simply between two linked neurons of consecutive layers with scalars. Different from them, *capsules*, as structured groups of neurons [5] correlating the reward distributions, outperforms the conventional scalar-based neural networks by capturing the complex co-occurrence patterns of rewards at different zones of the city. A link between two consecutive capsules is represented by a vector, and such vectorized representation is propagated between layers. Ride patterns, including spatial co-existence of multiple demand surges during rush hours or rainy days, are hence captured. Using capsules, STRide foresees an upcoming demand–supply imbalance and proactively provides learned decisions, achieving fine-grained coordination. It incorporates the spatial demand-supply dynamics, temporal external influence factors and ride preferences to capture the complex periodicity in MOD ride demands. The optimized coordination policy is stored in the capsule network for efficient online use by service providers.

To summarize, this paper makes the following three major contributions:

- *Comprehensive Learning Framework for Proactive & Efficient MOD Coordination*: We have formalized the dynamic vehicle dispatching and rides matching into a spatio-temporal RL framework. STRide accounts for spatial and temporal distributions of supplies and demands, ride preferences and other external factors. Using contextual scope processing and the learned Q-network, STRide dispatches each participating vehicle effectively and efficiently.
- *Spatio-Temporal Capsule-based Policy Learning*: Within the RL framework, we integrate a novel spatio-temporal capsule neural network, accurately and efficiently mapping the observed MOD network states, vehicles and dispatch actions to the provider's rewards. This way, STRide finds the spatio-temporally adaptive coordination policy with the best platform profitability, service quality, passenger satisfaction and driver incentivization.
- *Extensive Data-driven Model Analytics & Evaluation*: Based on the above platform and framework, we have conducted large-scale data analytics (a total of 27,612,831 rides) and comprehensive experimental evaluation of STRide. Our results based on the data sets of Uber/Yellow Taxis in New York City, Didi in Chengdu, China, Car2Go in Turin, Italy, Taxis in Chicago show STRide to outperform other state-of-the-arts, lowering request rejection rate, and passenger waiting time, and also enhancing the platform's profits.

STRide focuses on spatio-temporal configuration design (prototyped with double deep Q-network [6] or double DQN), but it can be integrated with other emerging RL models [7], [8], [9] for more advanced applications. Despite our prototyped scope upon classical services like ride-sharing or connected taxicabs, the spatio-temporal model can be easily extended to vehicle dispatching/allocation in many emerging connected transportation and smart city problems [10], including autonomous ride-sharing [11], rental car service (like Enterprise Rent-A-Car and Zipcar), bus dispatching, bike station reconfiguration [12], and charging dock deployment [13] for electric vehicles.

An earlier and shorter version of this work can be found in [14]. Besides more elaboration on the core and implementation of STRide (Section 4) and discussions on practical deployment (Section 6), this version expands and improves it significantly in the following three perspectives:

1) More comprehensive data-driven studies on traffic flows and rider preferences based on the MOD ride data in Sections 2 and 3 (including Section 3.2);
2) More design analysis studies (including model/setting variations) and performance comparison with many other state-of-the-arts in addition to the preliminary results reported in [14] (Section 5).
3) More extensive and detailed experimental studies (additional MOD datasets from Car2Go in Turin and taxis in Chicago) (Sections 3.1 & 5).

The rest of our paper is organized as follows. We first state the problem and describe the system in Section 2. Section 3 then presents STRide with a learning emulator, followed by the core algorithm in Section 4. We present the experimental evaluation in Section 5, and discuss its deployment in Section 6. We later review the related work in Section 7, and finally conclude the paper in Section 8.

Authorized licensed use limited to: University of Michigan Library. Downloaded on February 07,2022 at 02:22:07 UTC from IEEE Xplore. Restrictions apply.

TABLE 1
Major Symbols in STRide Formulation

Notations	Definitions
R, K	Numbers of zones and time intervals
k, m, $M^{(k)}$	Index of learning step, index and number of vehicles
$\mathcal{S}^{(k)}$	Observation at time interval k
$\mathcal{U}_m$	Dispatch action space for vehicle v_m
$\mathcal{Q}_m^{(k)}$	Long-term values of remaining steps for vehicle m
$\tau_m^{(k)}$, π	v_m's reward value at step k and the policy
γ	Weight parameter on temporal proximity
$\mathbf{H}^{(k)}$	A transition sample of consecutive states at k
$P_m^{(k)}$	Earning score of v_m at interval k
$\omega(i,j)$, $\Gamma(i,j)$	Ride preference metric & relocation travel time of z_i & z_j
$F_m^{(k)}$	Relocation cost related to dispatching & idle driving time
f_{ij}	Coupling coefficient between capsules i and j
$\mathbf{o}_j(\cdot)$	Squashing function at the capsule j
$\mathbf{e}_j$	Input to capsule j as the weighted average by f_{ij}
$\mathbf{q}_{j\|i}$	Propagated prediction vector from capsules i to j
θ_{ij}	Logarithm prior probabilities in the routing iteration

2 Problem Formulation & System Overview

We first present the preliminary concepts of STRide in Section 2.1, and then the problem formulation in Section 2.2, followed by the proposed framework in Section 2.3. We summarize the important symbols used in STRide in Table 1.

2.1 Preliminaries

Zones, Rides & MOD Network. A large-area city map is discretized into R zones, $\mathbf{Z} = \{z_1, \ldots, z_R\}$, while balancing between coordination granularity and computation efficiency. The shape and size of a zone can be subject to the performance goal and platform customization. We discretize the entire map into $L_{\text{lon}} \times L_{\text{lat}}$ rectangular "zones", the shape of each of which may be altered to reflect the existence of buildings, rivers, roads, etc.

Let $\mathbf{T}(i,j)$ be the set of directed MOD rides from z_i to z_j, and $\mathbf{T} = \{\mathbf{T}(i,j);\ i,j = 1, \ldots, z_R\}$. The MOD network is then a directed graph $\mathbb{G}(\mathbf{Z}, \mathbf{T})$ and formed by the end-to-end (e2e) MOD rides $\mathbf{T}$ across R different zones. The thus-formed $\mathbb{G}(\mathbf{Z}, \mathbf{T})$ serves as the *environment* of our coordination learning.

Discretization of Time Domain. The data structure for STRide's training is prepared by following the practice in RL framework [15] and slicing ride records $\mathbf{T}$ (sorted by their pick-up timestamps) into N_{epi} identical non-overlapping chunks. Each chunk, or *episode*, is a time period within a day, during which the MOD platform maximizes the financial returns. Simiarly to the map discretization, the time domain of each trip chunk is discretized into N_{step} equal intervals (30 min each in our prototype). Each interval $k \in \{1, \ldots, K\}$ corresponds to a *learning step*. For each z_i in a step k, let $D_i^{(k)}$ be the number of aggregated pick-ups (requests).

Fig. 2. Spatial distribution & heatmap frame of pick-ups.

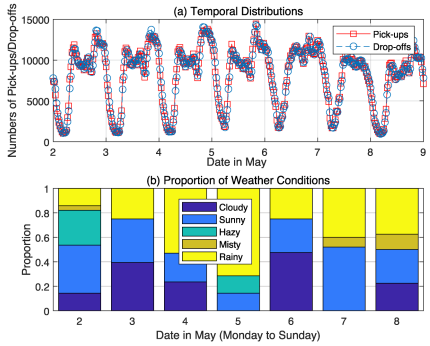


Fig. 3. Temporal distributions & weather conditions of a week.

Vehicles. Considering the connectivity of the MOD network, the service provider monitors the status of $M^{(k)}$ participating vehicles in step k. Each of these vehicles, denoted as v_m ($m \in \{1, \ldots, M^{(k)}\}$), contains its unique identifier, the current location (longitude and latitude *loc*; zone z_i), availability (vacant or not), and destination of ride/dispatch, if any. Each vehicle is in one of the following 4 states: *dispatching* (relocating to another zone for potential pick-ups), *matching* (heading to the pick-up location after accept), *occupied* (between pick-up and drop-off), and *vacant* (staying in the same zone after "dispatching" or "occupied" is over). Besides, each v_m is associated with estimated time of arrival, ETA_m.

2.2 Problem Formulation

Spatio-Temporal Features. Fig. 2 show the spatial (aggregated rides from 00:00 to 12:00 on May 31, 2016; the warmer colors indicate more pick-ups) of MOD rides (pick-ups/drop-offs) in the NYC Taxi dataset. Fig. 3 shows (a) the temporal distributions per 30 min of a day and (b) the daily proportion of different weather conditions, both of which correspond to the same week in May 2016. Most rides are shown to take place during the rush hours around Manhattan and locations of recreation/interests like casinos and airports. We also observe the complex multi-level periodicity (hourly, daily and weekly) in the rides requested and served. Different weather conditions may further alter and add complexity to the ride periodicity. Fig. 4 further visualizes the variations (in terms of totals and standard deviations) of trips per day. Clearly, weekdays generally experience more daily variations compared with weekends. Such spatial and temporal dynamics call for the need of more comprehensive learning of coordination.

Coordination Problem. Given the above spatial and temporal discretization and observations, we would like to proactively decide on *where* and *when* each available vehicle should be coordinated to serve ride requests so as to maximize the service provider's profitability and passenger-perceived service quality.

This problem is characterized with five major components $\{\mathcal{S}, \mathcal{U}, \tau, \pi, \mathcal{Q}\}$:

a) *State* $\mathcal{S}$: We assume that an MOD network or environment is coordinated by a web-based/online coordination

Authorized licensed use limited to: University of Michigan Library. Downloaded on February 07,2022 at 02:22:07 UTC from IEEE Xplore. Restrictions apply.

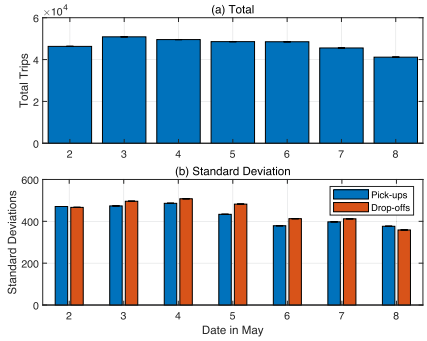


Fig. 4. Standard deviations of pick-ups/drop-offs per day.

center, or an *agent*, connecting a large group of spatially-distributed vehicles and passengers with mobile apps. The state space $\mathcal{S}^{(k)}$ that the agent observed at the learning step k consists of:

- *MOD vehicles* $\mathbf{V}$: the 2-D (an $L_{\text{lon}} \times L_{\text{lat}}$ matrix) distribution of all vehicles v_m's. From $\mathbf{V}$, we derive the 2-D distribution of vacant/available vehicles, denoted as $\mathbf{A}$. Both $\mathbf{V}$ and $\mathbf{A}$ capture the participating vehicles.
- *Departures/demands/pick-ups* $\mathbf{D}$: the 2-D location distribution ($L_{\text{lon}} \times L_{\text{lat}}$) of requests or departures of passengers.
- *External factors* $\mathbf{E}$: Since the different events (e.g., holidays or not), meteorological metrics (e.g., wind speed) and weather conditions (e.g., rainy or snowy) affect the demand/supply [16] as well as the resultant coordination performance, we form them into an L_{ext}-D vector as the additional hints for model training.

We model distributions of $\mathbf{D}$, $\mathbf{A}$ and $\mathbf{V}$ into frames of *heatmaps* (each is an $L_{\text{lon}} \times L_{\text{lat}}$ matrix; warmer colors indicate more passenger requests or vehicles) such that they can be processed by our spatio-temporal learning algorithm as *input features*. At each step k, we find the comprehensive state or observation $\mathcal{S}^{(k)}$ as the important spatial features characterizing the MOD environment, i.e.,

$$\mathcal{S}^{(k)} = \left\{\mathbf{D}^{(k)}, \mathbf{A}^{(k)}, \mathbf{V}^{(k)}, \mathbf{E}^{(k)}\right\}. \tag{1}$$

b) *Action* $\mathcal{U}$: An action is a solution to a coordination problem. The action space $\mathcal{U}_m$ for each vehicle v_m is defined as a set of discrete transits to any of its neighboring rectangle zones, plus staying where it is. Let L be the number of neighboring zones that a vehicle can relocate to ($L \leq R$). For each v_m, we consider $\mathcal{U}_m$ of size $(L+1)$ is centered at her/his current zone, and d_{ml} represents a destination zone l relative to the current center. Then, the dispatch *action space* for v_m is

$$\mathcal{U}_m = \{d_{m0}, d_{m1}, \ldots, d_{ml}, \ldots, d_{mL}\}, \tag{2}$$

where d_{m0} represents the action of staying where it resides.

c) *Reward* τ: Given the settings of states $\mathcal{S}$ and actions $\mathcal{U}$, each of $M^{(k)}$ available vehicles dispatched by the agent, arrives at the next state, and is returned with a value of immediate *reward*, i.e., $\mathcal{S} \times \mathcal{U} \times \mathcal{S} \rightarrow \mathbb{R}$ ($\tau \in \mathbb{R}$).

Specifically, each vehicle (driver) m at learning step k is associated with an instant reward function value $\tau_m^{(k)}$. $\tau_m^{(k)}$ takes into account the platform revenues and coordination cost (often subsidized by MOD service providers [17]), such that maximizing individual rewards can also maximize the platform profitability (Section 3.2). Driver incentivization [18], [19] and passenger satisfaction are also figured in the fine-grained multi-objective formulation.

d) *Policy* π & *Long-term Value* $\mathcal{Q}$: Intuitively, the MOD coordination actions have long-term effects upon the vehicle distributions. STRide aims at maximizing the expected reward in each episode, and mitigating the demand-supply imbalance. In other words, a zone with long-term oversupply would be considered less worthy (lower value) for a vehicle to relocate to, while an under-served one counts (higher value) due to more pick-up/earning opportunities. From the platform's perspectives, this coordination policy, as a joint mapping function, not only predicts future demand-supply gaps based on current market status, but also yields the highest reward by relocating vacant vehicles.

Specifically, at each step k, we find the subsequent cumulative returns based on a certain policy of coordination $\mathcal{S} \times \mathcal{U} \rightarrow \pi$, with a weight parameter $\gamma \in (0, 1)$ that differentiates rewards in terms of temporal proximity. For each v_m, at step k STRide expects the long-term values in the remaining steps of the episode as

$$\mathcal{Q}_m^{(k)} = \tau_m^{(k)} + \gamma\tau_m^{(k+1)} + \cdots + \gamma^{K-k}\tau_m^{(K)} = \sum_{\widetilde{k=k}}^{K} \widetilde{\gamma^{k-k}}\widetilde{\tau_m^{(k)}}. \tag{3}$$

Then, the optimal value function $\mathcal{Q}_m^*(\cdot)$ is the maximum expected long-term reward of all candidate dispatch decisions, i.e.,

$$\mathcal{Q}^* m\left(\mathcal{S}^{(k)}, \mathcal{U}^{(k)}\right) \triangleq \max_{\pi} \mathbb{E}\left[\mathcal{Q}^{(k)} m \middle| \mathcal{S}^{(k)}, \mathcal{U}^{(k)}, \pi\right], \tag{4}$$

which fulfills the Bellman equation [15] for iterative learning as: $\mathcal{Q}_m^*(\mathcal{S}^{(k)}, \mathcal{U}^{(k)}) =$

$$\mathbb{E}_{\mathcal{S}'}\left[\tau_m^{(k)} + \gamma \max_{\mathcal{U}'} \mathcal{Q}_m(\mathcal{S}', \mathcal{U}') \middle| \mathcal{S}^{(k)}, \mathcal{U}^{(k)}\right], \tag{5}$$

where $\mathcal{S}'$ and $\mathcal{U}'$ represent the given state and action of subsequent step $(k+1)$. Note that vehicles with the same spatio-temporal states are considered homogeneous. In other words, vehicles in the same zone and step (time interval) share the same coordination policy and value function. Due to the difficulty of specifying the sophisticated long-term value $\mathcal{Q}$, we design a spatio-temporal deep capsule network as the *Q-network* (Section 4) to store policy π.

2.3 System Framework & Flow

Fig. 5 overviews the STRide's system design with two phases: *offline learning* and *online coordination*. The entire system consists of following 3 major components:

1) *Feature Extraction & Processing*: Given the MOD data of rides from mobile apps and other external factors from the MOD (online) environment (Section 3.1), STRide first extracts features, structuring the batched data into states $\mathcal{S}$ as Eq. (1) for ease of the following offline emulation and model learning. The historical (offline learning) and real-time (online coordination) external knowledge can be obtained via weather station records or Internet [16].

Authorized licensed use limited to: University of Michigan Library. Downloaded on February 07,2022 at 02:22:07 UTC from IEEE Xplore. Restrictions apply.

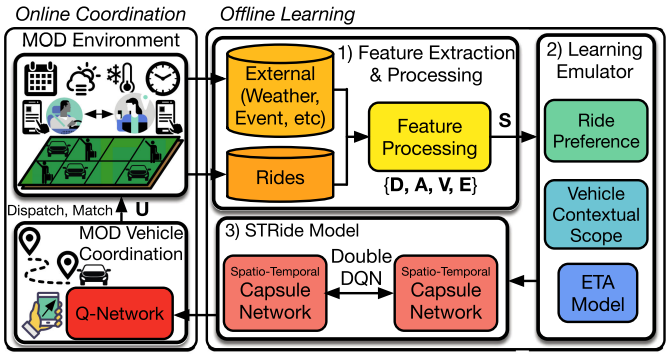


Fig. 5. The system framework of STRide.

2) *Learning Emulator*: In our RL design, the emulator [6], [15] provides the offline and emulated environment derived from real-world rides data and the city map for model training. Its design (Section 3.2) accounts for the ride preferences and the contextual scope for each individual vehicle. Meanwhile, STRide finds the estimated time of arrival (ETA) for each matching or dispatching transit. Note that the emulator can be cold-started by several episodes of rides without STRide's coordination. Historical rides can also be fed for offline learning to train an initial model of STRide.

3) *STRide Model*: The environment states, the agent's coordination actions and resultant rewards at each step are used to train our deep capsule network model (Section 4.2) for policy learning, minimizing the $\mathcal{Q}$ estimation loss. During model training, the emulator is reset given each episode of rides data, while the model is updated *w.r.t.* each learning step k.

After taking multiple steps, the reinforcement learning module (double DQN in our prototype) in STRide learns the coordination policy, and the STRide model is returned for the next episode. The offline learning ends when all episodes of data are examined. The learned capsule network returns the enhanced policy and actions for online coordination, e.g., routing vacant vehicles to destinations via mobile apps.

3 Data-Driven Learning Emulator

Given the above concepts and system, we first present the data sets in Section 3.1 and then describe the design of the learning emulator in Section 3.2, followed by the emulation process in Section 3.3.

3.1 Data Sets for Analytics & Evaluation

Our emulator is built for data analytics and performance evaluation based on the following five large-scale MOD datasets:

- *Uber, NYC* [20]: The ride sharing data of Uber in New York City (NYC), June 2015, contains a total of 2,816,895 rides covering the area of [40.60° N, 40.90° N, -74.04° W, -73.75° W].
- *Yellow Taxis, NYC* [21]: The ride data of Yellow Taxis in May 2016 contains 11,588,760 rides, their pick-up/drop-off locations and timestamps. According to recent data-driven studies [22], taxis share many similar characteristics with ride-sharing/ride-hailing vehicles (e.g., demand market, service coverage and trip length), making this feasible for mobility-on-demand study.

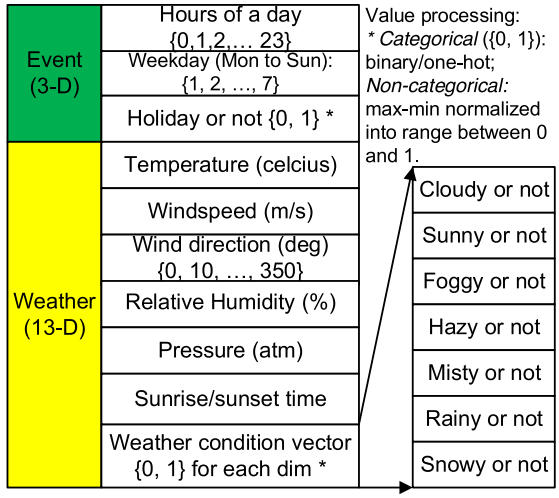


Fig. 6. Illustration of external factors & processing for $\mathbf{E}$ in our prototype (total 16 dimensions).

- *Didi, Chengdu, China* [23]: The ride sharing data provided by Didi Chuxing [23], contains a total of 6,744,508 rides covering the area of [30.65° N, 30.73° N, 104.04° W, 104.13° W] (with pick-up/drop-off locations and timestamps) from the city of Chengdu, Sichuan Province, China in November 2016.
- *Car2go, Turin, Italy*: The ride data of the ride-sharing service provider called Car2Go [24] contain 122,017 rides from September 1st to November 1st, 2017, their pick-up/drop-off locations in the City of Turin, Italy, covering the area of [45.1895° N, 45.0109° N, 7.7318° W, 7.6052° W].
- *Taxis, Chicago, IL*: The ride data of the taxis [25] contain 6,340,651 rides from March 1st to June 30th in 2016, their pick-up/drop-off locations and timestamps in the city of Chicago, IL, covering the area of [41.6446° N, 42.0229° N, -87.9395° W, -87.5245° W].

For each of these datasets, we also include local weather [26], weekday/weekend and festivals/events as the external factors ($\mathbf{E}$) in the model, as summarized in Fig. 6. We also obtain the road network (street center lines) from OpenStreetMap (OSM) [27].

3.2 Design of Comprehensive Learning Emulator

Based on the above data sets, in order for STRide to comprehensively train and evaluate the learned coordination policy model, we design and implement a *data-driven* learning platform that emulates the real-world MOD platform. By feeding real-world historical MOD rides to the learning emulator, STRide can model the spatio-temporal interactions between supplies and demands, and find a better policy reflecting a practical setting. The operation of the MOD agent (coordinator) is driven by its actual ride requests and the demand-supply forecast, and updates the state when a pick-up/request or drop-off/arrival event takes place.

Our emulator is designed with the following considerations.

a) *Contextual setting*: A driver usually focuses on her/his neighborhood observations or *contexts* for ease of (re)location. The relocation policy for each vehicle is made fine-grained by cropping and padding (zeros) [28] in the heatmap frames in $\mathcal{S}$

Authorized licensed use limited to: University of Michigan Library. Downloaded on February 07,2022 at 02:22:07 UTC from IEEE Xplore. Restrictions apply.

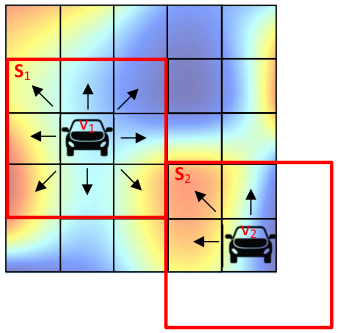


Fig. 7. Illustration of contextual scopes describing $\mathcal{S}$, and action spaces $\mathcal{U}$ of vehicles v_1 and v_2.

such that the *local* scope $\mathcal{S}_m$ of each v_m (i.e., the observed contexts in its neighborhood) is centered around its current zone. This way, STRide can learn the coordination policy based on each vehicle's neighborhood observations.

This also reflects the drivers' tendency in (re)locating to nearby zones (the search scope can be determined by surveying drivers [29] and customizing the platform) for less time/fuel consumption. STRide will also pay less computation overhead. At step k, we find the contextual scopes of $M^{(k)}$ available vehicles from the global state $\mathcal{S}^{(k)}$ in Eq. (2). The state of each vehicle m to be dispatched is

$$\mathcal{S}_m^{(k)} = \{\mathbf{D}_m^{(k)}, \mathbf{A}_m^{(k)}, \mathbf{V}_m^{(k)}, \mathbf{E}_m^{(k)}\}. \tag{6}$$

Each of $\mathbf{D}_m^{(k)}$, $\mathbf{A}_m^{(k)}$ and $\mathbf{V}_m^{(k)}$ is cropped from $\mathcal{S}^{(k)}$ into a subframe of smaller size $L'_{\text{lon}} \times L'_{\text{lat}}$ ($L'_{\text{lon}} < L_{\text{lon}}$ and $L'_{\text{lat}} < L_{\text{lat}}$), and centered at m's current zone like $\mathcal{U}_m$. $\mathbf{E}_m^{(k)}$ is an $(L_{\text{ext}}+2)$-D vector consisting of L_{ext} external factors and v_m's current 2-D location.

Using the above learning, STRide captures more fine-grained market features in vehicles' specific scopes instead of looking at a single heatmap frame, and reduces the computation overhead. Fig. 7 illustrates two examples of the contextual scopes (each is 3×3). The background heatmap (5×5) represents the spatial distribution of $\mathcal{Q}$ in a certain step. We also show the action spaces (with dispatching arrows; size may differ from $\mathcal{S}_m$) of vehicles 1 and 2. One may also impose a map constraint layer upon $\mathcal{U}_m$, by forcing the rewards associated with some d_{ml}'s outside service zones to be inaccessible (masked) by the vehicles (say, v_2 at the peripheral of Fig. 7). This way, the area outside our target service zones will not be considered for coordination, thus avoiding unnecessary computation.

In a sequential learning setting [15], we consider all idle vehicles sequentially determine where to relocate to. Each vehicle considers all other peers' present status, while its dispatch decision is independent of the peers' next moves/actions [30]. In state $\mathcal{S}_m^{(k)}$, v_m's relocation to a neighboring zone $d_{ml}^{(k)}$ at step k leads to a subsequent $\mathcal{S}_m^{(k+1)}$ observation and an instant reward $\tau_m^{(k)}$. Then, a *transition sample* $\mathbf{H}^{(k)}$ of consecutive states at k and $(k+1)$ is given by

$$\mathbf{H}^{(k)} = \{\mathcal{S}_m^{(k)}, d_{ml}^{(k)}, \mathcal{S}_m^{(k+1)}, \tau_m^{(k)}\}, \tag{7}$$

which is stored in the *experience replay* $\mathbf{O}$ and resampled for further training of STRide model in Section 4.

b) *Multi-objective ride reward function*: For each vehicle, STRide accounts for the platform profitability (in proportion to the vehicle's earning), service coverage and configuration cost in characterizing the agent's reward function. Intuitively, more rewards are expected if more ride fares are earned and less time of dispatching/idle driving is consumed.

To accommodate this, we consider for each v_m in the kth step two critical perspectives: earning score $P_m^{(k)}$ in terms of fulfilled rides and revenues, and the relocation cost $F_m^{(k)}$ related to dispatching and idle driving time (and fuel consumption). We adopt the vehicle-wise reward function since in practice each driver of the MOD or ride sharing platform have a smartphone app (like the component shown in the bottom left of Fig. 5) which updates with service and demand status. Each driver can access the knowledge of the platform and the imbalance between the passenger demand and their peer supply, and then make their decisions to fulfill their pursuits of profits. This way, each driver's incentive can be further reflected, which has been discussed in [18], [19].

Then, v_m's reward at step k is defined as the sum of earning scores minus the relocation costs in a window of w steps, i.e.,

$$\tau_m^{(k)} \triangleq \sum_{\widetilde{k}=k-w}^{k} \left(\alpha \widetilde{P_m^{(k)}} - \widetilde{F_m^{(k)}}\right), \tag{8}$$

where $\alpha > 0$ is an adjustable parameter. In our prototype, we empirically set $w = 15$. In other words, the more pick-up revenues and the less relocation time and fuel consumption, the higher reward a vehicle could achieve. The *earning score* $P_m^{(k)}$ is defined as the weighted sum of serviced ride fares among the zones based on the historical ride preferences $\omega(i,j)$'s, i.e.,

$$P_m^{(k)} \triangleq \sum_{i=1}^{R}\sum_{j=1}^{R} \left|\mathbf{T}_m^{(k)}(i,j)\right| \cdot e_{ij}, \tag{9}$$

where $\left|\mathbf{T}_m^{(k)}(i,j)\right|$ represents the number of actual rides from zones i to j provided by v_m, and e_{ij} is the resultant earning. In our emulator prototype, the driver revenue or ride fare is set as

$$e_{ij} = a \cdot \delta_{ij} + b, \quad a > 0, \quad b > 0, \tag{10}$$

where a is the unit price *w.r.t.* distance δ_{ij} and b is the base price (our prototype uses the local ride rates [20], [23]).

c) *Ride preference & zone-to-zone connectivity*: The drivers and passengers usually have frequent travel patterns among zones due to their commute routes and ride preferences. Fig. 8 shows the ride distributions $\mathbf{T}(i,j)$'s from start to destination zones during two different time periods of a day according to Didi Chengdu. Intensive morning flows (warmer colors) among central and business city zones (indexed by 1 to 6 and 10) can be observed in Fig. 8a, while Fig. 8b shows more flows among suburban city zones at night, indicating the spatio-temporal change of MOD commute routes and ride preferences. In coordinating the MOD vehicles, ignoring such ride tendency in coordination policy learning may discourage the drivers and passengers, resulting in profit loss of the MOD service. Therefore, we design a weighting scheme within

Authorized licensed use limited to: University of Michigan Library. Downloaded on February 07,2022 at 02:22:07 UTC from IEEE Xplore. Restrictions apply.

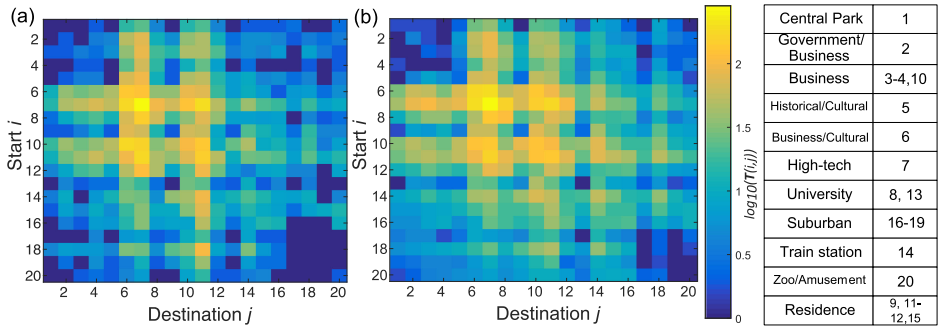


Fig. 8. Ride distributions across zones (Nov 1, 2016; Didi): (a) 06:00 – 12:00; (b) 18:00 – 24:00. For ease of visualization, the Chengdu map is discretized into 4×5 rectangular zones based on local information.

STRide to account for the effect of historical spatio-temporal ride preference [12], [31].

Considering the MOD network $\mathbb{G}$ connecting the zones $\mathbf{Z}$ with rides $\mathbf{T}$, inspired by the *first-order proximity* in network embedding [31], we design a *ride preference metric* $\omega(i,j)$ for rides $\mathbf{T}(i,j)$ between z_i and z_j as

$$\omega(i,j) \triangleq \frac{1}{1+\exp\left(-\vec{c}_{ij}\cdot\vec{c}_{ji}\right)}, \tag{11}$$

where the relative proportion of rides $\vec{c}_{ij}$ is defined as a vector of

$$\vec{c}_{ij} \triangleq \left[\frac{|\mathbf{T}(i,j)|}{\sum_{k=1,k\neq i}^{R}|\mathbf{T}(i,k)|}, \quad 1-\frac{|\mathbf{T}(i,j)|}{\sum_{k=1,k\neq i}^{R}|\mathbf{T}(i,k)|}\right]. \tag{12}$$

That is, the more rides recorded between z_i and z_j, the higher $\omega(i,j)$, indicating a stronger connectivity between the two zones.

We incorporate the above ride tendency of zones into STRide's formulation, not relying only upon individually-aggregated demand values. Our prototype discretizes each day into four 6-hour periods, and aggregates rides of the same period for different sets of $\omega(i,j)$'s. Then, to calculate $F_m^{(k)}$, we consider in Eq. (8) the relocation travel time $\Gamma(i,j)$, and the recent weight $\omega(i,j)$ between z_i and z_j belonging to the same historical periods (say, 06:00 – 12:00 of the same weekday in its preceding week), and find the weighted sum of

$$F_m^{(k)} \triangleq \sum_{i=1}^{R}\sum_{j=1}^{R}\frac{\beta\Gamma(i,j)}{\omega(i,j)}, \tag{13}$$

where $\beta > 0$ is the unit cost related to the relocation time or petrol prices [32] (say, a subsidy rate by the MOD platform for dispatching [33]). Rides starting and ending within the same zone, i.e., when $i = j$, are also considered within Eq. (11).

$\omega(i,j)$'s, as the soft margins, help differentiate and cluster different pairs of zones. In other words, STRide de-emphasizes the relocation cost between zones with high ride preferences and connectivities. The inner-rebalancing actions are encouraged within these closely-connected zones, better matching the MOD zone preferences as in Fig. 8. Relocating between such highly-connected zones is more likely to generate more pick-up opportunities and much inner demand-supply balance, thus reducing idle driving times in future. Given the above considerations, the travel preference is incorporated within the reward function modeling, achieving more fine-grained MOD coordination.

d) *Estimated time of arrivals*: For fine-grained evaluation, the learning emulator also calculates the estimated travel time between two locations for characterizing: (1) the time of travel from current location to destination when passengers are served (shown in Fig. 12); (2) the time of passenger wait if vehicles and certain passengers are matched; and (3) the time of idle driving if vehicles are dispatched to another location for potential requests.

Specifically, we implement a random forest regression [34] to estimate the travel time ETA_m between one location to another given the input vector of start/end location coordinates, length of the shortest path between them, day of a week and hour/minute of a day belonging to the start time. The length of interim road segments during the vehicle's travel are derived from OSM [27]. Our ETA model achieves a mean error of 2.7 min over 30,000 validation rides from all datasets (10,000 each). Despite our focus on designing a coordination framework, other more advanced models for ETAs [35], [36], [37] can also be integrated for better performance.

3.3 Emulation Process

Fig. 9 illustrates the time line inside the MOD emulation process. Each episode consists of multiple learning steps. In each step, STRide takes in the *new MOD requests* derived from historical records. Given current vehicle status, STRide finds the nearest driver–passenger *matching*, and updates the vacant vehicle status. Then, STRide proactively conducts *dispatching* of the remaining idle ones towards zones of potential demands based on the core $\mathcal{Q}$-network. If a dispatching/occupied vehicle arrives at its destination before another request comes, it automatically becomes vacant and stays there for the next coordination. After a pick-up, the vehicle arrives at the destination in the actual time of the ride record. After each step, STRide summarizes the rewards and vehicle distribution for next-step model training. In the vehicle emulation, the online/offline operations and the self-movements of each vehicle (driver) are also fed to determine the available

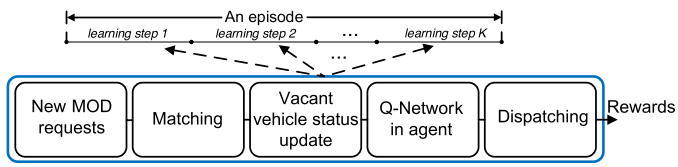


Fig. 9. MOD emulation process in one learning step of an episode.

Authorized licensed use limited to: University of Michigan Library. Downloaded on February 07,2022 at 02:22:07 UTC from IEEE Xplore. Restrictions apply.

vehicles for redistribution and pick-up in different city zones, which reflects the practical settings in the MOD platform [19]. Algorithm 1 summarizes this MOD emulation process.

To balance between the learned policy (maximization in Eq. (5)) and space exploration, STRide dispatches the vehicles based on the ϵ-greedy mechanism [15]. Specifically, if the selected random variable is greater than a preset threshold $\epsilon \in (0, 1)$, a vehicle is sent to the zone with maximum $\mathcal{Q}$, or will otherwise randomly explore one of the reachable zones in $\mathcal{U}$ as its destination.

Algorithm 1. Emulation Process in STRide

Input: Rides $\mathbf{T}$, external factors $\mathbf{E}$, ETA model eta_mod.
Output: Updated Agent and environment Env.

Agent.reset($\mathbf{T}$, $\mathbf{E}$); /* Initialization of the agent */
for k *in* range(K) **do**
  /* (1) Matching */
  $\mathbf{D}^{(k)} \leftarrow$ load_rides($\mathbf{T}$, k);
  $M' \leftarrow \min(\text{len}(\mathbf{D}^{(k)}), M)$; /* Feasible matched number */
  for v_m, $\mathbf{D}^{(k)}[l]$ *in* nearest_neighbors($\mathbf{A}^{(k)}$, $\mathbf{D}^{(k)}$, M') **do**
    if dist(v_m, $\mathbf{D}^{(k)}[l]$) < REJ_DIST **then**
      $\text{ETA}_m \leftarrow$ eta_mod(v_m, $\mathbf{D}^{(k)}[l]$) /* Waiting */
      Env.Match($v_m, \mathbf{D}^{(k)}[l], \text{ETA}_m$);
      $\mathbf{A}^{(k)} \leftarrow$ Env.Update($\mathbf{A}^{(k)}$);
    end
  end
  /* (2) Dispatching */
  $\{\widehat{\mathbf{d}}\} \leftarrow$ Agent.get_action($\mathbf{D}^{(k)}, \mathbf{A}^{(k)}, \mathbf{V}^{(k)}, \mathbf{E}$);
  for $\widehat{v}_m$, $\widehat{d}_{ml}$ *in* $\{\widehat{\mathbf{d}}\}$ **do**
    $\text{ETA}_m \leftarrow$ eta_mod($\widehat{v}_m$, $\widehat{d}_{ml}$); /* Reloc. time */
    Env.Assign($\widehat{v}_m$, $\widehat{d}_{ml}$, ETA_m); /* Dispatch assign */
    $\mathbf{A}^{(k)} \leftarrow$ Env.Update($\mathbf{A}^{(k)}$); /* Update of vehicles */
  end
end

4 Capsule-Based Coordination Learning

Given the above emulator and settings, we first present the motivations of our capsule network design in Section 4.1, followed by the structures of the spatio-temporal coordination learning in Section 4.2.

4.1 Motivations

Characterizing $\mathcal{Q}$, the relationship between states, actions and long-term values, is essential for STRide to decide on the best coordination policy π while adapting to the environment. Due to the complexity of large-scale MOD networks, it is often difficult to specify the mapping representation. Furthermore, the inherent relationship dynamically changes over the time. Therefore, we design a novel spatio-temporal capsule neural network as the deep $\mathcal{Q}$-network [6] within STRide, dynamically capturing and learning $\mathcal{Q}_m(\mathcal{S}_m, \mathcal{U}_m)$. That is, given state observations of $\mathcal{S}_m$, we want to (i) accurately predict the future distribution of $\mathcal{Q}_m$ values which has the same dimensions as $\mathcal{U}_m$, and then (ii) determine the better neighborhood for coordination using ϵ-greedy scheme.

There are usually long-term coordination effect and multi-level periodicity that cannot be easily learned by conventional scalar-based models like convolutional neural networks (CNNs) [4], where two linked neurons of consecutive layers communicate simply with a scalar. To address this, we propose the use of capsule-based $\mathcal{Q}$-network for more comprehensive MOD coordination learning. A *capsule* is a structured group of neurons [5], and many capsules can be grouped together in one layer. A link between two capsules in consecutive layers becomes a vector. Such vectorized (instead of scalar) representation of data is propagated between layers. Hence, more comprehensive ride patterns, including spatial co-existence of multiple demand surges during rush hours or rainy days, are captured with the vector representation as *instantiated entities* [38], thus yielding better coordination performance than the learning mechanism based on conventional CNNs [18].

Specifically, the vector-based capsule network characterizes observations of the input environment (i.e., distributions of demands, vehicle availability, and all vehicles) through the instantiation of the zones. For the image processing, capsules have been proposed to leverage the high-dimensional coincidence filtering [38], where a target object can be identified via agreement between the votes for the relative spatial relationship. In our formulation, we model the series of measurements on demand (pick-ups $\mathbf{D}$), supply (available vehicles $\mathbf{A}$) and the vehicle distributions $\mathbf{V}$ into time sequences of spatial heatmap images for Q-values. Therefore, like image processing, zones with have concurrent surges in the above distributions are structured together as instances and captured by the vectors of neurons. Conventional scalar-based networks cannot characterize these and hence degrade performance in the spatio-temporal scenarios.

4.2 Structures of Spatio-Temporal Coordination Learning

Fig. 10 illustrates the processing structures of the main and external state features, as detailed below.

a) *Main spatial features* $\{\mathbf{D}, \mathbf{A}, \mathbf{V}\}$: Given the input heatmap frames, $\{\mathbf{D}_m^{(k)}, \mathbf{A}_m^{(k)}, \mathbf{V}_m^{(k)}\}$, the core deep neural network in STRide captures the *spatial* relationship between states, actions and rewards. Specifically, a fine-grained capsule network takes in the vehicle's 2-D state representation, and returns the estimated $\mathcal{Q}$-values, denoted as $\mathcal{Q}^{\text{main}}$, *w.r.t.* the 2-D action space $\mathcal{U}_m$.

We further illustrate the basic structures and learning process of capsule network in STRide. Specifically, given the input heatmap $\mathbf{X}^{\text{input}} = \{\mathbf{D}_m^{(k)}, \mathbf{A}_m^{(k)}, \mathbf{V}_m^{(k)}\}$ processed from state $\mathcal{S}_m^{(k)}$, the capsule network consists of four sequential major components, i.e.,

$$\begin{aligned} \mathbf{X}_1 &= \texttt{Conv}\left(\mathbf{X}^{\text{input}}\right), \quad \mathbf{X}_2 = \texttt{PC}(\mathbf{X}_1), \\ \mathbf{X}_3 &= \texttt{OC}(\mathbf{X}_2), \quad \mathcal{Q}^{\text{main}} = \texttt{Conv}(\mathbf{X}_3), \end{aligned} \tag{14}$$

where the first and fourth two-dimensional convolutional layers (Conv) are used for transformation between the physical heatmap frames (scalar-based) and hidden capsule layers (vector-based), *Primary Capsule* (PC) and *Output Capsule* (OC).

In our implementation, each capsule contains a structured group of neurons reshaped and regrouped from convolutional layers [5]. Each cuboid in PC/OC of Fig. 10, as a capsule,

Authorized licensed use limited to: University of Michigan Library. Downloaded on February 07,2022 at 02:22:07 UTC from IEEE Xplore. Restrictions apply.

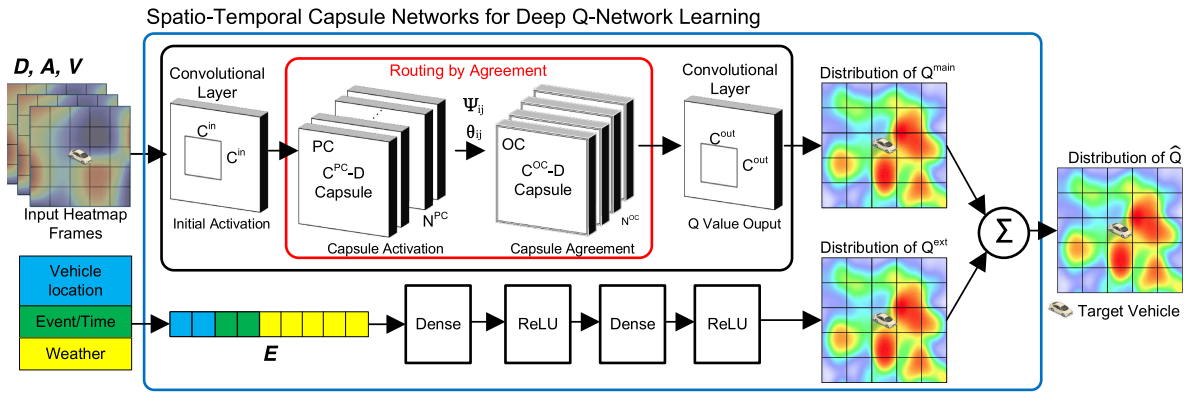


Fig. 10. Core module of spatio-temporal capsule-based $\mathcal{Q}$-network characterizing and predicting the distributions of rewards.

corresponds to a group of convolutional units or neurons (each neuron as a dimension is with a 9×9 kernel and a stride of 2 in our prototype). Specifically, PC is comprised of N^{PC} C^{PC}-D capsules, while OC is made of N^{OC} C^{OC}-D capsules. The input Conv layer has N^{in} $C^{\text{in}} \times C^{\text{in}}$ convolutional kernel filters, while the output one has N^{out} $C^{\text{out}} \times C^{\text{out}}$ filters.

Fig. 11 shows the training/learning process of the capsule network in STRide. In a nutshell, the parameters of STRide's capsules, consisting of traditional *neuron weights* and additional *capsule probabilities*, are propagated and refined between the layers of PC and OC. Specifically, let θ_{ij} be the logarithm prior probability captured by a preceding capsule i in PC and its succeeding peer j in OC. A softmax function [28] is then applied upon the θ_{ij}'s, returning the coupling coefficient f_{ij} (normalized) as

$$f_{ij} = \frac{\exp(\theta_{ij})}{\sum_l \exp(\theta_{il})}. \tag{15}$$

θ_{ij}'s capture the strengths of vehicle distributions in the map.

Similarly to the traditional neuron structure, the capsule network also has the weight coefficient across capsules i and j, denoted as Ψ_{ij}, learned through the conventional back-propagation algorithm [28]. All coefficients thus form an $N^{\text{PC}} \times N^{\text{OC}}$ weight matrix $\mathbf{\Psi}$. For each succeeding capsule j, let $\mathbf{q}_i$ be the ride prediction vector returned from a preceding peer i in PC. The propagated vector from capsules i to j, denoted as $\mathbf{q}_{j|i}$, is given by the product of neural network weights Ψ_{ij} and prediction vectors $\mathbf{q}_i$, i.e.,

$$\mathbf{q}_{j|i} = \Psi_{ij}\mathbf{q}_i, \tag{16}$$

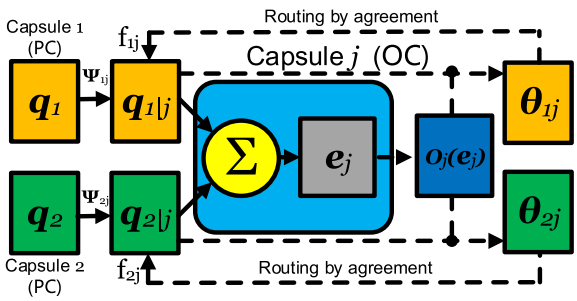


Fig. 11. Illustration of routing-by-agreement in STRide's capsules.

is then fed to capsule j as a weighted average by f_{ij}'s, i.e.,

$$\mathbf{e}_j = \sum_i f_{ij}\mathbf{q}_{j|i}. \tag{17}$$

A *routing-by-agreement* between capsules [5] is used to differentiate the vectors by their strengths of mutual agreement. Capsule training can then be regarded as extracting and refining the active *routes* from a preceding capsule layer (PC) to a succeeding one (OC). An active route across layers means a specific coordination strategy with certain zones, as an entity, is "memorized", while a deactivated one represents that the unimportant connections can be "forgotten".

Specifically, a *squash function* $\mathbf{o}_j(\cdot)$ is applied first upon $\mathbf{e}_j$ to characterize its length [5], which is given by

$$\mathbf{o}_j(\mathbf{e}_j) \triangleq \frac{\|\mathbf{e}_j\|^2}{1 + \|\mathbf{e}_j\|^2} \cdot \frac{\mathbf{e}_j}{\|\mathbf{e}_j\|}. \tag{18}$$

In other words, an increase in the vector length of the spatial ride distribution saturates the output towards one, which is identified and captured by the capsule network.

The logarithm prior probabilities θ_{ij} are updated with the product of prediction $\mathbf{q}_{j|i}$ and adjustment $\mathbf{o}_j(\mathbf{e}_j)$, i.e.,

$$\theta_{ij} \leftarrow \theta_{ij} + \mathbf{q}_{j|i} \cdot \mathbf{o}_j(\mathbf{e}_j). \tag{19}$$

The resultant θ_{ij} is returned to Eq. (15) for another routing iteration. Via routing-by-agreement, STRide finds the vectors with higher agreement, preserving the ride correlations across zones.

b) *External temporal features* $\mathbf{E}$: Inclusion of $\mathbf{E}$ further augments the capsule network in handling *temporal* ride dynamics, leading to a spatio-temporal formulation. This way, STRide can capture better the multi-level periodicity in the market dynamics.

Due to $\mathbf{E}$'s lower dimension than vehicle and passenger distributions, we form it into a vector of the external features, and design a fully-connected neural network (Dense) to learn the coordination. Specifically, the sequential model with two layers of dense/fully-connected networks (with respective output dimensions C_1^{Den} and C_2^{Den}) and subsequent ReLU activation functions [28] is given by

$$\begin{aligned} &\mathbf{E}_1 = \texttt{Dense}\big(\mathbf{E}^{\text{input}}\big), \quad \mathbf{E}_2 = \texttt{ReLU}(\mathbf{E}_1), \\ &\mathbf{E}_3 = \texttt{Dense}(\mathbf{E}_2), \quad \mathcal{Q}^{\text{ext}} = \texttt{Reshape}(\texttt{ReLU}(\mathbf{E}_3)). \end{aligned} \tag{20}$$

Authorized licensed use limited to: University of Michigan Library. Downloaded on February 07,2022 at 02:22:07 UTC from IEEE Xplore. Restrictions apply.

where the final output $\mathcal{Q}^{\text{ext}}$ is reshaped from a vector back to 2-D matrix *w.r.t.* the vehicle's action space $\mathcal{U}_m$.

E is processed as in Fig. 6. Note that considering the temporal continuity of weather records in **E** (temperature, wind-speed, *etc.*), STRide leverages data from the last learning step (k) as the external inputs of the next ($k+1$). STRide is also generic enough to be integrated with other more advanced weather forecast models [16].

Finally, the estimated $\mathcal{Q}$-value function *w.r.t.* each state and action is given by merging $\mathcal{Q}^{\text{main}}$ and $\mathcal{Q}^{\text{ext}}$, i.e.,

$$\widehat{\mathcal{Q}} = \mathcal{Q}^{\text{main}} + \mathcal{Q}^{\text{ext}}. \tag{21}$$

Then, in coordination STRide finds the zone with the maximum $\mathcal{Q}$-value in $\widehat{\mathcal{Q}}$ for dispatching (subject to ϵ-greedy in Section 2.2). The set of parameters to be trained, denoted as $\mathbf{\Omega}$, is hence made of parameters contributing to $\mathcal{Q}^{\text{main}}$ (including Ψ_{ij}'s and θ_{ij}'s) and those neuron weights for $\mathcal{Q}^{\text{ext}}$.

Further learning approximates the inherent and sophisticated mapping function as closely as possible without specifying the representation. While in this prototype we adopt the double deep Q-network (Double DQN) [6], which is detailed in the supplemental material, which can be found on the Computer Society Digital Library at http://doi.ieeecomputersociety.org/10.1109/TKDE.2020.2992565, STRide can be easily extended to other reinforcement learning methods or mechanisms (evaluated in Section 5).

In summary, STRide model enables: (1) *joint prediction and decision*: the model simultaneously forecasts the zone demand-supply gap, and determines those with higher values for enhanced coordination; (2) *self-adaptation*: the model can be dynamically updated with new input data, adapting to urbanization and road network changes; (3) *computational efficiency*: once the model is learned, an action decision can be made efficiently given the fast propagation of state inputs through neural network layers.

5 Experimental Evaluation

Given the datasets in Section 3.1, we first describe the experimental settings in Section 5.1, and then present the results in Section 5.2.

5.1 Experimental Settings

Comparison Schemes. We compare STRide with the following typical algorithms:

- *CRL*: an online learning framework [39] leveraging the conventional **c**onvolutional neural network (CNN) for **r**einforced coordination policy **l**earning [18].
- *NRL*: an online learning leveraging the three-layer dense **n**eural network for **r**einforced coordination policy **l**earning [40].
- *L&P*, a sequential decision-making and optimization algorithm, realizing **l**earning and **p**lanning [30], [41] for order dispatch.
- *CONT*: a conventional model-based approach, formulating a traffic **cont**rol problem [42] to find the dispatching and matching strategy based on the demand-supply balancing model. The complex city network is simplified for ease of model specification.

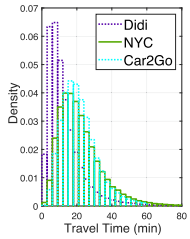


Fig. 12. Travel time.

- *MRL*: a **m**ulti-agent **r**einforcement **l**earning method [9] for ride sharing dispatching and matching.
- *GD*: a heuristic vehicle dispatching without optimizing or learning the ride dynamics. MOD vehicles are **g**reedily **d**ispatched towards zones with the highest demand-supply imbalances [22], [42].

We also compare in the sensitivity studies the performance variation by replacing the capsule module in STRide with neural network (NN) and convolutional neural network (settings of the NN/CNN layers follow [18], [40]), which are denoted as w/ NN and w/ CNN, respectively. We have also studied the performance with Asynchronous Advantage Actor-Critic (A3C) [43], deep Q-network (DQN), dueling double deep Q-network (Dueling DQN), which are denoted as w/ A3C, w/ DQN, and w/ Duel DQN, respectively. For A3C we adopt 10 actor-learners or workers. All evaluated schemes (detailed parameter settings can be referred to their work) use the same rides data, the module for estimated time of arrival (travel time data of NYC Taxi, Didi and Car2Go are shown in Fig. 12) and learning emulator settings. We then comparatively evaluate the performance based on the following metrics:

- *Profits* (platform): characterizes the overall profitability of the platform. In our experiment, we show the mean platform profit (fulfilled ride revenues subtracted by relocation subsidies on fuel costs [9]) *w.r.t.* each step, and normalize the values by global maximum of each dataset.
- *Idle driving time* (driver): is the time when the vehicle is not occupied but still incurs the driving costs. Note that in practice, the resultant costs including gasoline and wear of vehicles also incur higher subsidies paid by the platform inside a step. We show its mean (in minute) *w.r.t.* each driver before a pick-up.
- *Rejection rate* (platform): the number of rejects due to unavailability of vehicles nearby versus that of the total requests. We present the mean rejection rate *w.r.t.* the step.
- *Waiting time* (passenger): time between the event that a request is accepted and that the matched vehicle picks up the passenger. We show the mean (in minute) *w.r.t.* each matched request.
- *Coordination time* (platform): the computation time of each online decision for MOD request matching and vehicle dispatching, characterizing computation overhead and deployment efficiency. Specifically, we show the mean coordination time (in second) *w.r.t.* the step.

We have implemented the framework of STRide as well as its learning environment in python/tensorflow, and

Authorized licensed use limited to: University of Michigan Library. Downloaded on February 07,2022 at 02:22:07 UTC from IEEE Xplore. Restrictions apply.

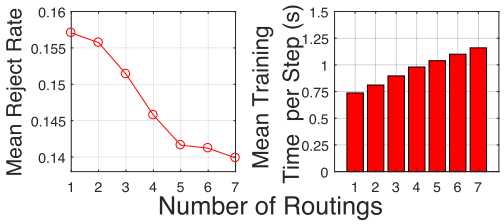


Fig. 13. Number of routings (Uber).

the models of STRide and other state-of-the-arts are trained and evaluated upon a desktop server with Intel i7-8700K 3.70 GHz, 16 GB RAM and Nvidia GTX 1080Ti with 11 GB DDR5.

Parameters: Unless otherwise stated, we use the following parameters by default. In the learning emulator, we empirically set total number of vehicles M = 8,000, α = 1.0, β = 0.3 and ϵ = 0.1. The total number of episodes is set to 12, while each of them lasts for 30 hours. Each step lasts for 30 min. We consider that a rejection happens if the distance between a request and the nearest vacant vehicle is greater than 5.0 km. For Uber/Taxis, $a = 5$ and $b = 3$; for Didi/Car2Go, $a = 5$ and $b = 1$. The platform revenues are 20 percent of the ride fares, and the subsidies are 20 percent of the relocation costs based on the surveys in [2], [22], [33]. The city map is partitioned into $L_{\text{lon}} \times L_{\text{lat}} = 219 \times 219$ zones (rectangular shapes are adjusted according to OSM map accessibility), while each vehicle is considered to move in its neighborhood of 15×15 zones ($\mathcal{U}_m$). Each vehicle's scope is set $L'_{\text{lon}} \times L'_{\text{lat}} = 23 \times 23$.

In the core $\mathcal{Q}$-network, we set each layer of the spatio-temporal capsule network as follows: $\left(N^{\texttt{PC}}, C^{\texttt{PC}}\right) = (8, 8)$, $\left(N^{\texttt{OC}}, C^{\texttt{OC}}\right) = (8, 50)$, $\left(N^{\texttt{in}}, C^{\texttt{in}}\right) = (64, 9)$, $\left(N^{\texttt{out}}, C^{\texttt{out}}\right) = (64, 6)$ (each Conv has a stride of 1 and ReLU activation) for main feature components in Fig. 10; $\left(C_1^{\texttt{Den}}, C_2^{\texttt{Den}}\right) = (10, 15^2)$ for the rest components handling the 18 external features ($L_{\text{ext}} = 16$ and 2-D location in $\mathbf{E}$ as in Fig. 6).

5.2 Experimental Results

With the above settings, we present the experimental results.

Module & Sensitivity Analysis. Taking NYC Uber as an example, we first show the performance variation of STRide *w.r.t.* the settings of several important design parameters (other datasets are left out here due to space limit). Note that we take the first 48 hours out of all one-month rides for the parameter and system validation, while the rest of them are used for evaluation/testing.

Number of Iterative Routings. Fig. 13 shows the mean rejection rate and mean training time per step (in s) *w.r.t.* the number of iterative routings in STRide (Section 4.2). Involvement

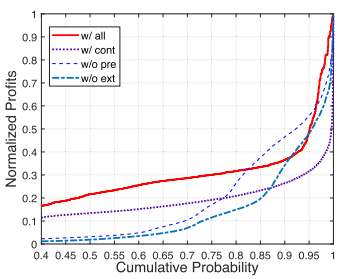


Fig. 14. Components (Uber).

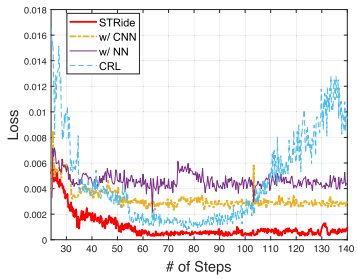


Fig. 15. Training loss (Uber).

of more routings by agreement may lead to better extraction of important ride features, at the cost of longer training time. Meanwhile, the extractable features tend to decrease, leading to the diminishing returns in the rejection rate improvement. Therefore, to balance these, we choose 5 routings in our prototype study.

Ride Preference/External Factors. Fig. 14 shows STRide's normalized profits without one of ride preference (pre), external factors (ext) and contextual scope (cont), and the complete model with all components. Introducing ride preferences helps STRide relocate vehicles across popular zones with strong connectivities, thus enhancing the probability of picking up more passengers and making resultant profits. Inclusion of auxiliary factors like wind speed, temperature and weather conditions related to MOD rides help STRide capture more intrinsic routines in the spatio-temporal ride distribution, hence achieving more profits than that without external knowledge. The contextual scope helps STRide to better capture the observations around each vehicle's neighborhood.

Training Loss. We also show in Fig. 15 the model learning loss of STRide and CRL *w.r.t.* the number of steps based on the validation data. Without a comprehensive learning structure, CRL's training curve easily diverges due to the complexity in metropolitan MOD network dynamics. Thanks to more comprehensive capsule network structures that combine different influential factors, STRide outperforms CRL with better learning capability. We also show the loss of STRide w/ NN and w/ CNN as comparison, from which we can observe that our capsule designs help STRide have a better convergence than NN and CNN. We have conducted convergence experiments upon the datasets of other MOD platforms (other similar results are omitted due to space limit). Further theoretical studies will be considered is part of our future work.

Change of Q-Network Modules. We further compare in Fig. 16 the performance—in terms of profits and waiting time—of STRide with the spatio-temporal capsule network (w/ Cap), the one with deep neural network (w/ DNN), and the one with convolutional neural network (w/ CNN). Using the proposed capsule network, STRide better characterizes the relationship between the states, actions and the long-term values. Without capsule's vectorization, NN-based and CNN-based Q-networks cannot capture the spatial dependencies of the close and distant zones. Therefore, one can observe that STRide w/ Cap outperforms other settings, yielding more profits and shorter waiting time (by often more than 28 and 13 percent improvements, respectively, in our test).

Change of Reward Functions. Taking NYC Taxi as an example, we compare the coordination performance of STRide

Authorized licensed use limited to: University of Michigan Library. Downloaded on February 07,2022 at 02:22:07 UTC from IEEE Xplore. Restrictions apply.

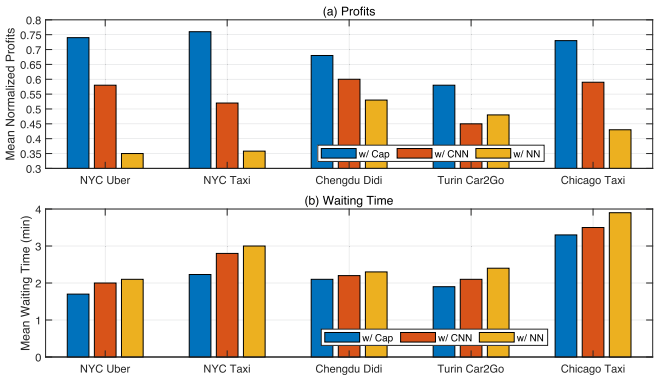


Fig. 16. Performance comparison of w/ Cap, w/ CNN & w/ NN.

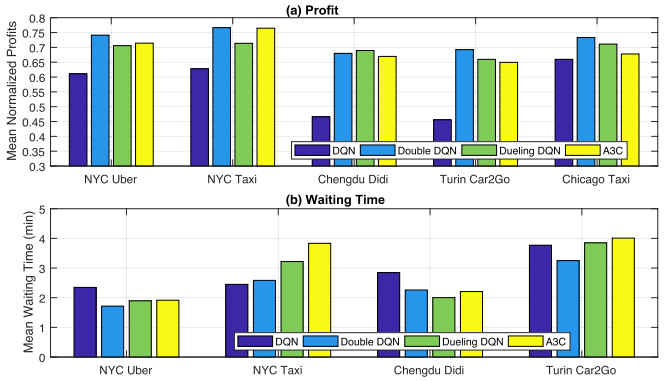


Fig. 19. Performance comparison of STride with double DQN, DQN, dueling DQN and A3C.

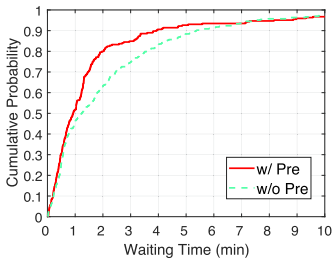


Fig. 17. CDFs: waiting time.

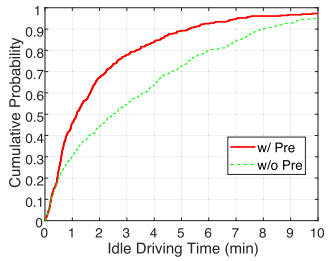


Fig. 18. CDFs: idle driving time.

TABLE 2
Mean Profits (Normalized) for the Five Datasets

Schemes	NYC Uber	NYC Taxi	Chengdu Didi	Turin Car2Go	Chicago Taxi
CRL	0.4235	0.5847	0.4199	0.4871	0.4926
NRL	0.4060	0.3415	0.3909	0.4519	0.4301
L&P	0.3780	0.3554	0.3055	0.4116	0.4071
CONT	0.1461	0.2917	0.2718	0.3520	0.3567
MRL	0.5420	0.6462	0.4993	0.5087	0.5318
GD	0.1354	0.2267	0.2547	0.3119	0.2466
STRide	**0.7413**	**0.7665**	**0.6799**	**0.5925**	**0.7333**

with and without the reward function designs in Section 3.2. In particular, we compare in Figs. 17 and 18, respectively, the waiting time and idle driving time distributions of STride with and without the weighted design (denoted as *w/ Pre* and *w/o Pre*) of Eq. (13). These two metrics specify the quality of the MOD services. We can observe that inclusion of ride preference better steers the MOD vehicles towards the connected zones, thus benefiting both passengers and drivers by reducing the waiting and idle driving time (respectively by 17.21 and 45.52 percent in our test).

Change of Reinforcement Learning Methods. By changing the reinforcement learning mechanisms within STride, we show in Fig. 19 the performance comparison of double DQN with DQN, dueling DQN and A3C, in terms of mean profits and waiting time. We can observe that double DQN outperforms DQN in terms of the mean profits. The performances of double DQN and dueling DQN are close to each other since the learned MOD features suffice to maintain good coordination performance based on the dataset. One can also see that the coordination performance of A3C compared with double DQN may vary with data sets mainly because A3C focuses on lightweight learning, but does not necessarily improve the learning performance. Also, note that our focus is on the spatial and temporal learning framework for coordination.

Platform Profitability, Efficiency & Service Quality. Given aforementioned sensitivity studies, we further present the performance of our trained model with evaluation data.

Profits & Efficiency. Table 2 shows the mean profit (normalized *w.r.t.* each dataset) of STride in comparison with other state-of-the-arts. The schemes like L&P and CONT may not fully capture the spatial-temporal complexity in MOD ride flows. STride is shown to achieve much higher rewards (often by more than 30 percent) than other schemes. The advanced spatio-temporal capsule network helps STride accurately capture the dynamic imbalance between demands and supplies. With step-by-step loss minimization, the RL framework in STride proactively finds a coordination policy with more pick-ups and less relocation effort, yielding more profits for the MOD platforms.

Taking the large-scale NYC taxi as an example, we show the CDFs of computation time in Fig. 20. Due to the fast weight propagation across the layers, STride achieves much better computational efficiency than other more sophisticated control and heuristic mechanisms, which is essential for real-time coordination.

Taking Chengdu Didi as an example, we further show in Fig. 21 the temporal profits of STride during a day (of 24 hours) compared to CRL. The dynamics of temporal rewards also reflect the relation between supplies and demands during the rush hours.

We also show in Fig. 22 the MOD demand-supply dynamics over the time in a business zone of Chengdu. We plot the relative gap when demand surpasses supply there, and show that the state values (Q) grow with the gap, meaning

Authorized licensed use limited to: University of Michigan Library. Downloaded on February 07,2022 at 02:22:07 UTC from IEEE Xplore. Restrictions apply.

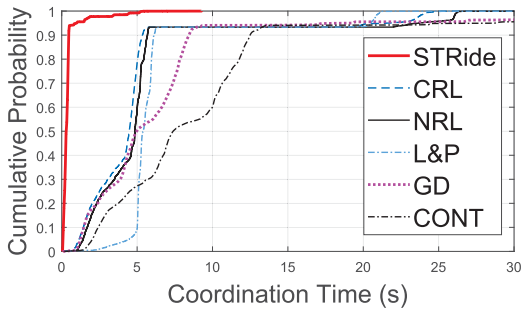


Fig. 20. Online coordination overhead, NYC Taxi.

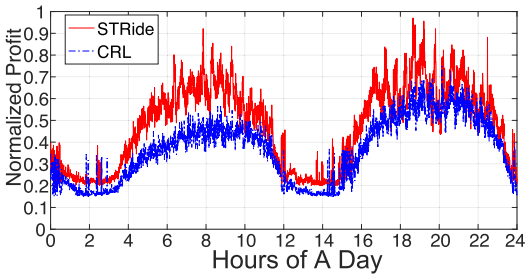


Fig. 21. Profit dynamics of STRide within a day (Didi).

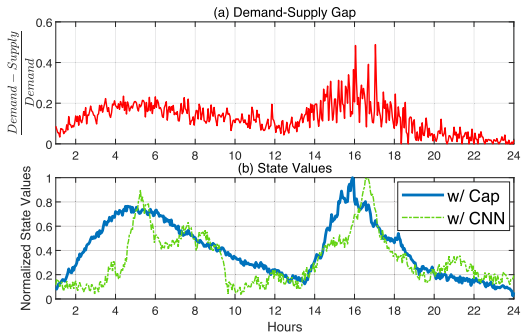


Fig. 22. Supply-demand dynamics & values within a day (Didi).

the locations are more valuable for relocation. We can also see that the adapted state values of STRide w/ Cap captures the dynamic gaps more accurately and proactively than that w/ CNN. The result validates effectiveness and proactiveness of STRide in learning and coordinating the MOD networks.

Idle Driving Time. Table 3 shows that a shorter average idle driving time (min) *w.r.t.* the five datasets. It is due mainly to the highly accurate Q-value approximation within STRide, capturing the spatio-temporal relations among states, actions and consequent rewards. Less idle driving may also lead to less subsidies and costs from the MOD platform owner, enhancing its overall profitability. For NYC, taxis may show shorter idle travels due to more demands recorded than the Uber cars (Section 3.1). On the other hand, due to smaller city size and population, the MOD services in Turin may not be as active as New York City and Chicago, leading to a slightly higher idle driving time.

Rejection Rate. Table 4 shows the lower mean rejection rates of the schemes *w.r.t.* the five different data sets. Thanks to STRide's proactive relocation of MOD vehicles, fewer ride requests are rejected. On the other hand, we can infer

TABLE 3
Mean Idle Driving Time (Min) for the Five Datasets

Schemes	NYC Uber	NYC Taxi	Chengdu Didi	Turin Car2Go	Chicago Taxi
CRL	4.45	3.65	4.22	3.83	4.35
NRL	7.09	4.85	6.74	3.93	5.43
L&P	9.44	6.74	9.36	4.29	6.48
CONT	11.46	9.2	9.61	5.10	9.93
MRL	4.41	3.63	3.85	3.63	4.35
GD	12.09	11.07	11.72	9.88	11.49
STRide	**2.87**	**2.38**	**2.69**	**2.94**	**3.34**

TABLE 4
Mean Rejection Rates for the Five Datasets

Schemes	NYC Uber	NYC Taxi	Chengdu Didi	Turin Car2Go	Chicago Taxi
CRL	0.2172	0.1941	0.1617	0.2036	0.2389
NRL	0.2201	0.2019	0.2172	0.2081	0.2510
L&P	0.2900	0.2201	0.2262	0.2157	0.2798
CONT	0.2863	0.2413	0.2616	0.2655	0.3078
MRL	0.2063	0.2022	0.1880	0.1912	0.2030
GD	0.2925	0.2705	0.2689	0.3036	0.3802
STRide	**0.1376**	**0.1162**	**0.0863**	**0.1637**	**0.2020**

TABLE 5
Mean Waiting Time (Min) fo the Five Datasets

Schemes	NYC Uber	NYC Taxi	Chengdu Didi	Turin Car2Go	Chicago Taxi
CRL	2.80	3.36	3.01	2.12	4.52
NRL	4.12	4.06	4.55	2.37	4.85
L&P	3.83	4.60	4.66	3.54	6.55
CONT	5.23	5.68	6.37	3.96	6.65
MRL	3.84	3.68	3.42	3.42	4.40
GD	5.98	6.68	7.45	4.31	8.44
STRide	**1.70**	**2.23**	**2.13**	**1.97**	**3.27**

that STRide accurately learns the locations of potentially high demands, and determines proactive dispatch regions within the rejection distance threshold. Due to a larger service coverage, we observe higher rejection rates from the trials in NYC than in Chengdu.

Waiting Time. Table 5 shows the shorter passenger average waiting time (min) of STRide than other related algorithms. Via more proactive dispatching, the supplies match demands in time, hence resulting in shorter waits. This way, the MOD platforms can improve their service quality, and hence will likely achieve higher passenger satisfaction. Due to the greater freedom of pick-ups/drop-offs in practice, Uber cars experience a shorter waiting time (15.11 percent shorter, on average, in our test) than taxis in NYC.

Coordination Visualization. To better illustrate the performance, we also qualitatively visualize the state values (normalized Q) and coordination results by STRide in Figs. 23 and 24, respectively. Taking NYC Yellow Taxi as an example, we plot the estimated Q distribution of Manhattan island in Fig. 23, characterizing the expected demand-supply imbalances. The warmer colors indicate higher state values for relocation. We also show the resultant out-flows of vacant vehicles dispatched by STRide in several selected zones there in Fig. 24. The radius of a sector represents the size of the out-flow. We can see that more MOD vehicles are relocated from the Manhattan peripherals to midtown due to the significant supply-and-demand imbalance there.

Authorized licensed use limited to: University of Michigan Library. Downloaded on February 07,2022 at 02:22:07 UTC from IEEE Xplore. Restrictions apply.

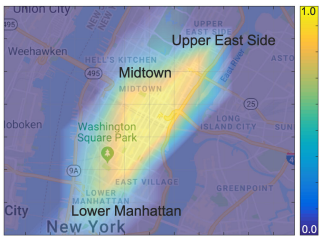


Fig. 23. Heatmap of visualized state values, NYC Taxi (09:00).

Fig. 24. Visualized relocation flows, NYC Taxi (09:00).

6 Discussion

We briefly discuss the deployment of STRide from the following two perspectives:

- *Single/Multi-Agent Reinforcement Learning*: We focus on the spatial and temporal learning of the MOD systems like Uber and Didi, and hence leverage the single-agent settings for ease of implementation and prototyping, especially when we deal with large-scale MOD ride flows. We take into account the neighborhood observations of each vehicle as the contextual scope, thus realizing more fine-grained decision learning. While our work leveraged single-agent reinforcement learning in the prototype, our core designs, including the spatio-temporal capsule network, can be easily extended to a multi-agent framework. Specifically, we can consider and model each individual driver as an agent for the observation and decision. Possible approaches include probabilistic modeling of the interactions among the driver agents [44], as well as hierarchical regional coordination [8], to address scalability problem, which is part of our future work.

  In our experimental studies, we have also investigated STRide upon different reinforced learning frameworks, showing its adaptability. Despite the dynamic environments and vehicle-wise rewards, STRide has been shown to achieve better coordination performance than other schemes, demonstrating its adaptability and learning capability.
- *Demands, Requests & Pick-ups*: While self-movements of drivers have been considered in our framework, there is still room to reflect the practical MOD settings. A passenger's ride demand may not always lead to an actual request and subsequent pick-up. When a potential rider launches the MOD app, her/his request of a ride may be discouraged by the surged price [19] or longer waiting time, leading to a difference of "demand" and pick-up. Due to the unavailability of the aforementioned data, we did not formulate the above into the current framework, but will consider the above scenarios in our future work.

7 Related Work

Mobility-on-Demand. Driven by increasing vehicular connectivity [45], [46], cloud computing and big data analytics [47], [48], the emerging online platforms for MOD services, typified by ride-sharing and ride-hailing [49], have recently attracted significant attention. Various problems, including privacy-preserving [50], traffic flow [51], and responses to dynamic pricing [17], [52], [53], [54] have been investigated for better MOD services. In contrast, we focused on the MOD coordination problem which is essential for an MOD platform, and presented our novel and comprehensive learning framework.

Transportation Management. Numerous *optimization*-based schemes have been proposed [42], [55], [56], including control-based methodology [51], combinatorial optimization [30], and queueing theory [57]. Despite the reported promising results, the optimization-based models are usually difficult to solve efficiently, particularly for large-scale MOD networks. Furthermore, it is still very challenging to specify beforehand a generic optimization formulation to accommodate highly dynamic and uncertain traffic environments.

Powered by exploding big data [58] and facilitating parallelism [7], [59], we have witnessed unprecedented advances in *learning*-based transportation management [4], [8], [60], [61]. Wen *et al.* [40] conducted preliminary studies upon the learning-based MOD rebalancing. Lin *et al.* [9] studied an RL-based mechanism managing the fleet with scalar-based neural network. Similarly, Xu *et al.* [30] explored the order dispatching, focusing on the sequential dispatch optimization problem. Oda *et al.* [18] proposed a fleet management system based on convolutional neural networks, finding the policy for relocating the connected taxicabs.

Multi-agent reinforcement learning has also been studied for ride sharing recently. Li *et al.* [44] studied mean field multi-agent reinforcement learning for efficient order dispatching. Lin *et al.* [9] investigated the contextual multi-agent reinforcement learning for large-scale management.

Unlike the above-mentioned studies, STRide incorporates several practical aspects, such as the distributed and contextual vehicle scopes, heatmap frames representing ride distributions, and external weather/event factors. We have also designed in STRide a novel capsule-based Q-network for comprehensive learning. Furthermore, in contrast with the usual focus on the snapshot of ride distribution [39], [40], STRide accounts for multi-level complexity, periodicity and preference in spatio-temporal ride patterns, thus achieving better performance with the real-world data sets. Albeit orthogonal to the multi-agent settings, our work can be adapted further to muli-agent designs to enhance the system performance and deployability.

Coordination Factors. There are many other internal and external factors affecting the success of transportation coordination, including social [62], demographic [12], [63], personalization [41], [64], governmental and psychological [53] perspectives. Despite the challenges to design a complete coordination model, our spatio-temporal RL mechanism is a good and comprehensive direction to accommodate the

Authorized licensed use limited to: University of Michigan Library. Downloaded on February 07,2022 at 02:22:07 UTC from IEEE Xplore. Restrictions apply.

above factors. Further mechanism design for incentive-compatible monetary rewards in driver redistribution can also be referred to other orthogonal works [57], [65], [66].

8 Conclusion

We have proposed STRide, a spatio-temporal reinforcement learning framework for MOD coordination. Given the MOD ride data, STRide forms a learning emulator for enhanced coordination policy training, which takes into account the ride preference, contextual scopes of vehicles, and travel time. We have designed a spatio-temporal capsule network in STRide to map the states and dispatch actions towards the expected future rewards. Spatial distributions of demands and supplies, and the temporal external factors like events and weather conditions, are jointly considered. With a structured group of neurons capturing the complex ride correlations, STRide comprehensively learns the improved policies for the vehicle coordination. We have conducted extensive data analytics and experimental evaluation on five large-scale datasets (Uber, Taxis, Car2Go and Didi). STRide is shown to outperform many other state-of-the-arts, with lower request rejection rates, shorter waiting time, and higher platform profitability compared to state-of-the-arts.

Acknowledgments

We would like to thank DiDi Chuxing GAIA Open Dataset Initiative for the shared ride data.

References

[1] Ride hailing market by service type (e-hailing, car sharing, station-based, car rental), data service (navigation, information, payment, others), connectivity (3G, 4G, 5G, Wi-Fi and V2V, V2I, V2P, V2N), vehicle type & region - global forecast to 2025, 2017. [Online]. Available: https://www.marketsandmarkets.com/Market-Reports/mobility-on-demand-market-1 98699113.html

[2] J. Horwitz, "One year after the Uber-Didi merger, it's only getting harder to hail a ride in China," 2017. [Online]. Available: https://qz.com/1045268/ and http://news.sina.com.cn/c/2017–07-26/doc-ifyinryq6222913.shtml

[3] Uber drivers report 80-plus hour workweeks and a lot of waiting, 2019. [Online]. Available: http://theconversation.com/uber-drivers-report-80-plus-hour-workweeks-a nd-a-lot-of-waiting-115782

[4] 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.

[5] S. Sabour, N. Frosst, and G. E. Hinton, "Dynamic routing between capsules," in *Proc. 31st Int. Conf. Neural Inf. Process. Syst.*, 2017, pp. 3856–3866.

[6] H. Van Hasselt, A. Guez, and D. Silver, "Deep reinforcement learning with double Q-learning," in *Proc. 30th AAAI Conf. Artif. Intell.*, 2016, vol. 2, pp. 2094–2100.

[7] Q. Cai, A. Filos-Ratsikas, P. Tang, and Y. Zhang, "Reinforcement mechanism design for e-commerce," in *Proc. World Wide Web Conf.*, 2018, pp. 1339–1348.

[8] L. Pan, Q. Cai, Z. Fang, P. Tang, and L. Huang, "A deep reinforcement learning framework for rebalancing dockless bike sharing systems," in *Proc. 33rd AAAI Conf. Artif. Intell.*, 2018, pp. 1393–1400.

[9] 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. Discov. Data Mining*, 2018, pp. 1774–1783.

[10] Y. Zheng, L. Capra, O. Wolfson, and H. Yang, "Urban computing: Concepts, methodologies, and applications," *ACM Trans. Intell. Syst. Technol.*, vol. 5, no. 3, pp. 38:1–38:55, Sep. 2014.

[11] S. Aoki and R. R. Rajkumar, "Dynamic intersections and self-driving vehicles," in *Proc. ACM/IEEE 9th Int. Conf. Cyber-Physical Syst.*, 2018, pp. 320–330.

[12] S. He and K. G. Shin, "(Re)Configuring bike station network via crowdsourced information fusion and joint optimization," in *Proc. 18th ACM Int. Symp. Mobile Ad Hoc Netw. Comput.*, 2018, pp. 1–10.

[13] B. Du, Y. Tong, Z. Zhou, Q. Tao, and W. Zhou, "Demand-aware charger planning for electric vehicle sharing," in *Proc. 24th ACM SIGKDD Int. Conf. Knowl. Discov. Data Mining*, 2018, pp. 1330–1338.

[14] S. He and K. G. Shin, "Spatio-temporal capsule-based reinforcement learning for mobility-on-demand network coordination," in *Proc. World Wide Web Conf.*, 2019, pp. 2806–2813.

[15] R. S. Sutton and A. G. Barto, *Introduction to Reinforcement Learning*, vol. 135. Cambridge, MA, USA: MIT Press, 1998.

[16] J. Zhang, Y. Zheng, and D. Qi, "Deep spatio-temporal residual networks for citywide crowd flows prediction," in *Proc. 31st AAAI Conf. Artif. Intell.*, 2017, pp. 1655–1661.

[17] Z. Fang, L. Huang, and A. Wierman, "Prices and subsidies in the sharing economy," in *Proc. 26th Int. Conf. World Wide Web*, 2017, pp. 53–62.

[18] T. Oda and C. Joe-Wong, "MOVI: A model-free approach to dynamic fleet management," in *Proc. IEEE Conf. Comput. Commun.*, 2018, pp. 2708–2716.

[19] K. Bimpikis, O. Candogan, and D. Saban, "Spatial pricing in ride-sharing networks," *Operations Res.*, vol. 67, no. 3, pp. 744–769, 2019.

[20] Uber pickups in New York City, 2018. [Online]. Available: https://www.kaggle.com/fivethirtyeight/uber-pickups-in-new-york-city/da ta

[21] TLC Trip Record Data, 2018. [Online]. Available: http://www.nyc.gov/html/tlc/html/about/trip_record_ data.shtml

[22] L. Rayle, D. Dai, N. Chan, R. Cervero, and S. Shaheen, "Just a better Taxi? A survey-based comparison of taxis, transit, and ride-sourcing services in San Francisco," *Transport Policy*, vol. 45, pp. 168–178, 2016.

[23] Didi Chuxing Technology Co, 2019. [Online]. Available: www.didiglobal.com

[24] M. Cocca, D. Giordano, M. Mellia, and L. Vassio, "Free floating electric car sharing design: Data driven optimisation," *Pervasive Mobile Comput.*, vol. 55, pp. 59–75, 2019.

[25] Chicago taxi rides 2016: Details of taxi rides in chicago, 2017. [Online]. Available: https://www.kaggle.com/chicago/chicago-taxi-rides-2016

[26] National Centers for Environmental Information, National Oceanic and Atmospheric Association (NOAA) – Data Tools: Local Climatological Data (LCD), 2018. [Online]. Available: https://www.ncdc.noaa.gov/cdo-web/datatools/lcd

[27] Open Street Map, 2018. [Online]. Available: https://www.openstreetmap.org/

[28] I. Goodfellow, Y. Bengio, A. Courville, and Y. Bengio, *Deep Learning*, vol. 1. Cambridge, MA, USA: MIT Press, 2016.

[29] H. Campbell, "RSG 2017 survey results: Driver earnings, satisfaction & demographics," 2017. [Online]. Available: https://the rideshareguy.com/rsg-2017-survey-results-driver-earnings-sat isfaction-and-demographics/

[30] 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. Discov. Data Mining*, 2018, pp. 905–913.

[31] J. Tang, M. Qu, M. Wang, M. Zhang, J. Yan, and Q. Mei, "LINE: Large-scale information network embedding," in *Proc. 24th Int. Conf. World Wide Web*, 2015, pp. 1067–1077.

[32] Global petrol prices, 2018. [Online]. Available: https://www.globalpetrolprices.com/gasoline_prices/

[33] R. Dovey, "5 Florida cities team up to subsidize Uber rides," 2017. [Online]. Available: https://nextcity.org/daily/entry/five-florida-cities-subsidize-uber-rides

[34] J. Han, J. Pei, and M. Kamber, *Data Mining: Concepts and Techniques*. Amsterdam, Netherlands: Elsevier, 2011.

[35] Y. Li, K. Fu, Z. Wang, C. Shahabi, J. Ye, and Y. Liu, "Multi-task representation learning for travel time estimation," in *Proc. 24th ACM SIGKDD Int. Conf. Knowl. Discov. Data Mining*, 2018, pp. 1695–1704.

[36] D. Wang, W. Cao, J. Li, and J. Ye, "DeepSD: Supply-demand prediction for online car-hailing services using deep neural networks," in *Proc. IEEE 33rd Int. Conf. Data Eng.*, 2017, pp. 243–254.

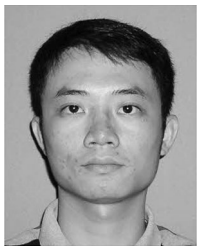
[37] Z. Wang, K. Fu, and J. Ye, "Learning to estimate the travel time," in *Proc. 24th ACM SIGKDD Int. Conf. Knowl. Discov. Data Mining*, 2018, pp. 858–866.

[38] G. E. Hinton, S. Sabour, and N. Frosst, "Matrix capsules with EM routing," in *Proc. Int. Conf. Learn. Representations*, 2018.

[39] Y. Gao, D. Jiang, and Y. Xu, "Optimize taxi driving strategies based on reinforcement learning," *Int. J. Geographical Inf. Sci.*, vol. 32, no. 8, pp. 1677–1696, 2018.

Authorized licensed use limited to: University of Michigan Library. Downloaded on February 07,2022 at 02:22:07 UTC from IEEE Xplore. Restrictions apply.

[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.*, 2017, pp. 220–225.
[41] L. Zhang *et al.*, "A taxi order dispatch model based on combinatorial optimization," in *Proc. 23rd ACM SIGKDD Int. Conf. Knowl. Discov. Data Mining*, 2017, pp. 2151–2159.
[42] N. Agatz, A. Erera, M. Savelsbergh, and X. Wang, "Optimization for dynamic ride-sharing: A review," *Eur. J. Oper. Res.*, vol. 223, no. 2, pp. 295–303, 2012.
[43] V. Mnih *et al.*, "Asynchronous methods for deep reinforcement learning," in *Proc. 33rd Int. Conf. Mach. Learn.*, 2016, pp. 1928–1937.
[44] M. Li *et al.*, "Efficient ridesharing order dispatching with mean field multi-agent reinforcement learning," in *Proc. World Wide Web Conf.*, 2019, pp. 983–994.
[45] V. W. Zheng, Y. Zheng, X. Xie, and Q. Yang, "Collaborative location and activity recommendations with GPS history data," in *Proc. 19th Int. Conf. World Wide Web*, 2010, pp. 1029–1038.
[46] K. D'Silva, K. Jayarajah, A. Noulas, C. Mascolo, and A. Misra, "The role of urban mobility in retail business survival," *Proc. ACM Interactive Mobile Wearable Ubiquitous Technol.*, vol. 2, no. 3, 2018, Art. no. 100.
[47] C. Zhang *et al.*, "Regions, periods, activities: Uncovering urban dynamics via cross-modal representation learning," in *Proc. 26th Int. Conf. World Wide Web*, 2017, pp. 361–370.
[48] S. Yao, S. Hu, Y. Zhao, A. Zhang, and T. Abdelzaher, "DeepSense: A unified deep learning framework for time-series mobile sensing data processing," in *Proc. 26th Int. Conf. World Wide Web*, 2017, pp. 351–360.
[49] S. Jiang, L. Chen, A. Mislove, and C. Wilson, "On ridesharing competition and accessibility: Evidence from Uber, Lyft, and Taxi," in *Proc. World Wide Web Conf.*, 2018, pp. 863–872.
[50] X. Xie, F. Zhang, and D. Zhang, "PrivateHunt: Multi-source data-driven dispatching in for-hire vehicle systems," *Proc. ACM Interactive Mobile Wearable Ubiquitous Technol.*, vol. 2, no. 1, pp. 45:1–45:26, Mar. 2018.
[51] 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.
[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] S. Guo, C. Chen, Y. Liu, K. Xu, and D. M. Chiu, "Modelling passengers' reaction to dynamic prices in ride-on-demand services: A search for the best fare," *Proc. ACM Interactive Mobile Wearable Ubiquitous Technol.*, vol. 1, no. 4, pp. 136:1–136:23, Jan. 2018.
[54] S. Guo *et al.*, "A simple but quantifiable approach to dynamic price prediction in ride-on-demand services leveraging multi-source urban data," *Proc. ACM Interactive Mobile Wearable Ubiquitous Technol.*, vol. 2, no. 3, pp. 112:1–112:24, Sep. 2018.
[55] E. Walraven, M. T. Spaan, and B. Bakker, "Traffic flow optimization: A reinforcement learning approach," *Eng. Appl. Artif. Intell.*, vol. 52, pp. 203–212, 2016.
[56] H. Zheng and J. Wu, "Online to offline business: Urban taxi dispatching with passenger-driver matching stability," in *Proc. IEEE 37th Int. Conf. Distrib. Comput. Syst.*, 2017, pp. 816–825.
[57] S. Banerjee, R. Johari, and C. Riquelme, "Pricing in ride-sharing platforms: A queueing-theoretic approach," in *Proc. 16th ACM Conf. Econ. Comput.*, 2015, pp. 639–639.
[58] C. Miao, Q. Li, L. Su, M. Huai, W. Jiang, and J. Gao, "Attack under disguise: An intelligent data poisoning attack mechanism in crowdsourcing," in *Proc. World Wide Web Conf.*, 2018, pp. 13–22.
[59] J. Zhang, Y. Zheng, D. Qi, R. Li, X. Yi, and T. Li, "Predicting city-wide crowd flows using deep spatio-temporal residual networks," *Artif. Intell.*, vol. 259, pp. 147–166, 2018.
[60] Y. Li, Y. Zheng, and Q. Yang, "Dynamic bike reposition: A spatio-temporal reinforcement learning approach," in *Proc. 24th ACM SIGKDD Int. Conf. Knowl. Discov. Data Mining*, 2018, pp. 1724–1733.
[61] H. Wei, G. Zheng, H. Yao, and Z. Li, "IntelliLight: A reinforcement learning approach for intelligent traffic light control," in *Proc. 24th ACM SIGKDD Int. Conf. Knowl. Discov. Data Mining*, 2018, pp. 2496–2505.
[62] Y. Zheng, L. Zhang, X. Xie, and W.-Y. Ma, "Mining interesting locations and travel sequences from GPS trajectories," in *Proc. 18th Int. Conf. World Wide Web*, 2009, pp. 791–800.
[63] F. Kooti, M. Grbovic, L. M. Aiello, N. Djuric, V. Radosavljevic, and K. Lerman, "Analyzing Uber's ride-sharing economy," in *Proc. 26th Int. Conf. World Wide Web Companion*, 2017, pp. 574–582.
[64] Y. Li, H. Su, U. Demiryurek, B. Zheng, T. He, and C. Shahabi, "PaRE: A system for personalized route guidance," in *Proc. 26th Int. Conf. World Wide Web*, 2017, pp. 637–646.
[65] M. K. Lee, D. Kusbit, E. Metsky, and L. Dabbish, "Working with machines: The impact of algorithmic and data-driven management on human workers," in *Proc. 33rd Annu. ACM Conf. Hum. Factors Comput. Syst.*, 2015, pp. 1603–1612.
[66] D. Tomaras, I. Boutsis, and V. Kalogeraki, "Modeling and predicting bike demand in large city situations," in *Proc. IEEE Int. Conf. Pervasive Comput. Commun.*, 2018, pp. 1–10.



Suining He (Member, IEEE) received the PhD degree from the Department of Computer Science and Engineering, The Hong Kong University of Science and Technology (HKUST), Hong Kong. He is currently working as an assistant professor with the Department of Computer Science and Engineering, The University of Connecticut (UConn), Storrs, Connecticut. Before joining UConn, he worked as a postdoctoral research fellow at the Real-Time Computing Lab (RTCL), Department of Electrical Engineering and Computer Science, the University of Michigan, Ann Arbor, Michigan from 2016 to 2019. He is a Google PhD fellow, 2015. His research interest includes smart transportation, urban data science, and mobile computing. He is an ACM member.

Kang G. Shin (Life Fellow, IEEE) is the Kevin & Nancy O'Connor professor of computer science in the Department of Electrical Engineering and Computer Science, The University of Michigan, Ann Arbor. His current research focuses on QoS-sensitive computing and networking as well as on embedded real-time and cyber-physical systems, such as autonomous vehicles. He has supervised the completion of 85 PhDs, and authored/coauthored close to 1,000 technical articles, a textbook and about 60 patents or invention disclosures, and received numerous awards, including 2019 Caspar Bowden Award for Outstanding Research in Privacy Enhancing Technologies, and the best paper awards from the 2011 ACM International Conference on Mobile Computing and Networking (MobiCom 2011), the 2011 IEEE International Conference on Autonomic Computing, the 2010 and 2000 USENIX Annual Technical Conferences, as well as the 2003 IEEE Communications Society William R. Bennett Prize Paper Award, and the 1987 Outstanding IEEE Transactions of Automatic Control Paper Award. He has also received several institutional awards, including the Research Excellence Award, in 1989, Outstanding Achievement Award, in 1999, Distinguished faculty Achievement Award, in 2001, and Stephen Attwood Award, in 2004 from The University of Michigan (the highest honor bestowed to Michigan Engineering faculty); a Distinguished Alumni Award of the College of Engineering, Seoul National University in 2002; 2003 IEEE RTC Technical Achievement Award; and 2006 Ho-Am Prize in Engineering (the highest honor bestowed to Korean-origin engineers). He has chaired Michigan Computer Science and Engineering Division for three years starting 1991, and also several major conferences, including 2009 ACM MobiCom, 2008 IEEE SECON, 2005 ACM/USENIX MobiSys, 2000 IEEE RTAS, and 1987 IEEE RTSS. He is the fellow of the ACM. He has also served or is serving on numerous government committees, such as the US NSF Cyber-Physical Systems Executive Committee and the Korean Government R&D Strategy Advisory Committee. He has also helped founding a couple of startups and is currently serving as an executive advisor for Samsung Research.

▷ **For more information on this or any other computing topic, please visit our Digital Library at** www.computer.org/csdl.

Authorized licensed use limited to: University of Michigan Library. Downloaded on February 07,2022 at 02:22:07 UTC from IEEE Xplore. Restrictions apply.