Spatio-Temporal Capsule-based Reinforcement Learning for Mobility-on-Demand Network Coordination

Suining He
The University of Michigan–Ann Arbor
suiningh@umich.edu

Kang G. Shin
The University of Michigan–Ann Arbor
kgshin@umich.edu

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 three large-scale datasets (~21 million rides from Uber, Yellow Taxis and Didi). 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, often making 30% improvement over state-of-the-arts.

CCS Concepts

• **Information systems → Spatial-temporal systems**; **Data mining**;

Keywords

Mobility-on-demand, ride-sharing platform, smart transportation coordination, reinforcement learning, capsule network, smart city.

ACM Reference Format:
Suining He and Kang G. Shin. 2019. Spatio-Temporal Capsule-based Reinforcement Learning for Mobility-on-Demand Network Coordination . In *Proceedings of the 2019 World Wide Web Conference (WWW '19), May 13–17, 2019, San Francisco, CA, USA.* ACM, New York, NY, USA, 7 pages. https://doi.org/10.1145/3308558.3313401

1 Introduction

By integrating online information of rides demands and supplies, transit network operation and communication, cooperative mobility-on-demand (MOD) systems, such as Uber, Lyft, Didi and connected taxicabs, have provided unprecedented transportation alternatives.

Given its significant economic and social values, we should coordinate the MOD operations, *i.e.,* dispatching (matching) supplies (i.e., available vehicles/drivers) towards demands (*i.e.,* 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 (a discretized city map), and matched with their nearest requesters. The resultant rides also "connect" the zones, forming an MOD network.

However, increasingly complex urban traffic networks and miscoordination of demand-supply dynamics often under/over-serve many service zones, thus degrading the providers' profitability, service quality, passenger satisfaction and drivers' enthusiasm. The problem is particularly severe during rush hours when people travel in similar directions between home and work. 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 upon old data in some zones cannot be applied and scaled throughout the dynamically-changing MOD network, thus calling for an adaptive mechanism. Second, coordination of complex MOD supplies and demands makes 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. Weather can also affect the ride requests, which will, for example, surge during rainy hours. Without comprehensive modeling of these factors, the conventional coordination methods cannot easily and accurately capture the repeating patterns.

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 that vehicle can be relocated to. 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

WWW '19, May 13–17, 2019, San Francisco, CA, USA

ACM ISBN 978-1-4503-6674-8/19/05.
https://doi.org/10.1145/3308558.3313401

neural network to comprehensively learn the relationship between observed states, coordination actions and potential rewards. This way, STRide foresees an upcoming demand–supply imbalance and proactively provides optimal decisions, achieving fine-grained coordination. STRide 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.

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 double deep Q-network (DQN) learning and contextual scope processing, STRide learns how to dispatch 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 Analytics & Evaluation*: Based on above, we have conducted large-scale (a total of 21,150,163 rides) data analytics and comprehensive experimental evaluation of STRide. Our results based on data of Uber/Yellow Taxi, New York City and Didi, Chengdu, China show STRide to outperform other state-of-the-arts, lowering request reject rate, and passenger waiting time, and also enhancing the platform's profits (often making more than 30% improvement over the state-of-the-arts).

2 Problem Formulation & System Overview

2.1 Preliminary

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. In our prototype, the entire map is discretized 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 [29] 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. Similarly 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 ($k \in \{1,\ldots,K\}$) corresponds to a *learning step*. Then, for each z_i in a step k, we let $D_i^{(k)}$ be the number of aggregated pick-ups (requests).

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

Coordination Problem: Given the spatio-temporal distributions of MOD ride pick-ups and drop-offs, 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 the following five major components: $\{\boldsymbol{\mathcal{S}}, \boldsymbol{\mathcal{U}}, \tau, \pi, Q\}$.

a) **State $\boldsymbol{\mathcal{S}}$**: We consider that an MOD network or environment is coordinated by a web-based/online coordination center, or an *agent*, connecting a large group of spatially-distributed vehicles and passengers with mobile apps. The state space $\boldsymbol{\mathcal{S}}^{(k)}$ 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 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 [38] 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 $\boldsymbol{\mathcal{S}}^{(k)}$ as the important spatial features characterizing the MOD environment, *i.e.,*

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

b) **Action $\boldsymbol{\mathcal{U}}$**: An action is a coordination solution. The action space $\boldsymbol{\mathcal{U}}_m$ 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 ($\leq R$) be the number of neighboring zones that a vehicle can relocate to. For each v_m, we consider $\boldsymbol{\mathcal{U}}_m$ 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 $\boldsymbol{\mathcal{U}}_m = \{d_{m0}, d_{m1}, \ldots, d_{ml}, \ldots, d_{mL}\}$, where d_{m0} represents the action of staying where it resides.

c) **Reward τ**: Given the settings of states $\boldsymbol{\mathcal{S}}$ and actions $\boldsymbol{\mathcal{U}}$, each of $M^{(k)}$ available vehicles dispatched by the agent, arrives

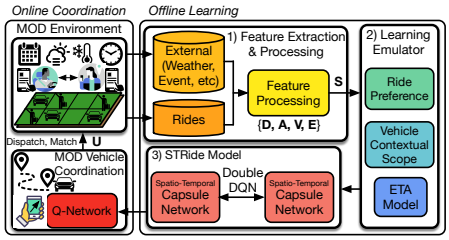

Fig. 1: The system framework of STRide.

at the next state, and is returned with an immediate *reward*, *i.e.,* $\mathcal{S} \times \mathcal{U} \times \mathcal{S} \rightarrow \tau$. Specifically, each vehicle (driver) m at learning step k is associated with an instant reward function $\tau_m^{(k)}$. $\tau_m^{(k)}$ takes into account the platform revenues and coordination cost (often subsidized by MOD service providers [11]), such that maximizing individual rewards can also maximize the platform profitability (Sec. 3.2). Driver incentivization and passenger satisfaction are also figured in the fine-grained multi-objective formulation.

d) **Policy** π & **Long-term Value** 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. From the platform's perspectives, this coordination policy, as a joint mapping function, not only predicts gaps between future demands and supplies based on current market status, but also yields the highest reward by relocating idle 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

$$Q_m^{(k)} = \tau_m^{(k)} + \gamma \tau_m^{(k+1)} + \cdots + \gamma^{K-k} \tau_m^{(K)}. \quad (2)$$

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

$$Q_m^*\left(\mathcal{S}^{(k)}, \mathcal{U}^{(k)}\right) \triangleq \max_{\pi} \mathbb{E}\left[Q_m^{(k)} \middle| \mathcal{S}^{(k)}, \mathcal{U}^{(k)}, \pi\right], \quad (3)$$

which fulfills the Bellman equation [29] for iterative learning as

$$Q_m^*\left(\mathcal{S}^{(k)}, \mathcal{U}^{(k)}\right) = \mathbb{E}_{\mathcal{S}'}\left[\tau_m^{(k)} + \gamma \max_{\mathcal{U}'} Q_m\left(\mathcal{S}', \mathcal{U}'\right) \middle| \mathcal{S}^{(k)}, \mathcal{U}^{(k)}\right], \quad (4)$$

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 Q, we design a spatio-temporal deep capsule network as the Q-*network* (Sec. 4) to store policy π.

2.3 System Framework & Flow

Fig. 1 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 (Sec. 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 [38].

2) **Learning Emulator**: In our RL design, the emulator [29, 31] provides the offline and emulated environment derived from real-world ride data and the city map for model training. Its design (Sec. 3.2) accounts for the ride preferences, 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 ride data can also be used 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 for policy learning, minimizing the Q estimation loss. During model training, the emulator is reset given each episode of ride data, while the model is updated *w.r.t.* each learning step k. After taking multiple steps, the double DQN mechanism learns the coordination policy, and the STRide model is returned for next episode. The offline learning ends when all episodes of data are examined. The learned capsule network returns the optimal policy (subject to ϵ-greedy mechanism) and actions for online coordination, e.g., routing vacant vehicles to destinations via mobile apps [19].

3 Data-Driven Learning Emulator

3.1 Data Sets for Analytics & Evaluation

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

- *Uber, NYC* [5]: The ride sharing data of Uber in New York City (NYC), June 2015, contains a total of 2,816,895 rides.
- *Yellow Taxis, NYC* [4]: The ride data of Yellow Taxis in May 2016 contains 11,588,760 rides, their pick-up/drop-off locations and timestamps. Taxis share many similar characteristics with ride-sharing/hailing vehicles (e.g., driving distance and time), making this experimental study feasible [27].
- *Didi, Chengdu* [6]: The ride-sharing data provided by Didi Chuxing [6], contains a total of 6,744,508 rides (with pick-up/drop-off locations and timestamps) from the city of Chengdu, Sichuan Province, China in November 2016.

For each of these datasets, we also include local weather [2], weekday/weekend and festivals/events as the external factors (**E**) in the model, as summarized in Fig. 3. We also obtain the road network from OpenStreetMap (OSM) [3].

3.2 Design of Comprehensive Learning Emulator

We design and implement a *data-driven* learning platform that emulates a real-world MOD platform with the following considerations.

a) **Contextual setting**: A driver usually focuses on her/his neighborhood observations or *contexts* for (re)location. The relocation policy for each vehicle is made fine-grained by cropping and padding (zeros) [13] in the heatmap frames in $\mathcal{S}$ 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 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 $\mathcal{S}^{(k)}$. The state of each vehicle m to be dispatched is $\mathcal{S}_m^{(k)} = \{\mathbf{D}_m^{(k)}, \mathbf{A}_m^{(k)}, \mathbf{V}_m^{(k)}, \mathbf{E}_m^{(k)}\}$.

Each of $\mathbf{D}_m^{(k)}, \mathbf{A}_m^{(k)}$ and $\mathbf{V}_m^{(k)}$ is cropped from $\boldsymbol{\mathcal{S}}^{(k)}$ into a sub-frame 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 $\boldsymbol{\mathcal{U}}_m$. $\mathbf{E}_m^{(k)}$ is an $(L_{\text{ext}}+2)$-D vector consisting of L_{ext} external factors and v_m's current 2-D location.

In a sequential-learning setting [29], 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 [37]. In state $\boldsymbol{\mathcal{S}}_m^{(k)}$, v_m's relocation to a neighboring zone $d_{ml}^{(k)}$ at step k leads to a subsequent $\boldsymbol{\mathcal{S}}_m^{(k+1)}$ 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

$$\mathbf{H}^{(k)} = \left\{\boldsymbol{\mathcal{S}}_m^{(k)}, d_{ml}^{(k)}, \boldsymbol{\mathcal{S}}_m^{(k+1)}, \tau_m^{(k)}\right\}, \tag{5}$$

which is stored in the *experience replay* $\mathbf{O}$ and resampled for further training of STRide model in Sec. 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 k-th 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). 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.,*

$$\tau_m^{(k)} \triangleq \sum\nolimits_{\widetilde{k}=k-w}^{k} \left(\alpha P_m^{(\widetilde{k})} - F_m^{(\widetilde{k})}\right), \tag{6}$$

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.,*

$$P_m^{(k)} \triangleq \sum\nolimits_{i=1}^{R} \sum\nolimits_{j=1}^{R} \left|\mathbf{T}_m^{(k)}(i,j)\right| \cdot e_{ij}, \tag{7}$$

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

$$e_{ij} = a \cdot \delta_{ij} + b, \quad a > 0, \quad b > 0, \tag{8}$$

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 [5, 6]).

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. Considering the MOD network $\mathbb{G}$ connecting the zones $\mathbf{Z}$ with rides $\mathbf{T}$, inspired by the *first-order proximity* in network embedding [30], we design a ride preference metric $\omega(i,j)$ for rides $\mathbf{T}(i,j)$ between z_i and z_j as

$$\omega(i,j) \triangleq \left(1 + \exp\left(-\vec{c}_{ij} \cdot \vec{c}_{ji}\right)\right)^{-1} \tag{9}$$

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

$$\vec{c}_{ij} \triangleq \left[\frac{|\mathbf{T}(i,j)|}{\sum_{k=1,k\neq i}^{R} |\mathbf{T}(i,k)|}, \quad 1 - \frac{|\mathbf{T}(i,j)|}{\sum_{k=1,k\neq i}^{R} |\mathbf{T}(i,k)|}\right]. \tag{10}$$

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. (6) 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

$$F_m^{(k)} \triangleq \sum\nolimits_{i=1}^{R} \sum\nolimits_{j=1}^{R} \frac{\beta\Gamma(i,j)}{\omega(i,j)}, \tag{11}$$

where $\beta > 0$ is the unit cost related to the relocation time or petrol prices [1] (say, a subsidy rate by the MOD platform for dispatching [10]). Rides starting and ending within the same zone, *i.e.,* when $i = j$, are also considered within Eq. (9).

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; (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 [14] (other more advanced models [18, 33, 34] may apply) to estimate the travel time ETA_m between one location to another given the input vector of start/end 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 [3].

4 Capsule-based Coordination Learning

Characterizing $\boldsymbol{Q}$, 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. We propose the use of capsule-based $\boldsymbol{Q}$-network for more comprehensive MOD coordination learning, dynamically capturing and learning $\boldsymbol{Q}_m(\mathcal{S}_m, \boldsymbol{\mathcal{U}}_m)$. A *capsule* is a structured group of neurons [28], 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* [16].

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

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

$$\begin{aligned} &\mathbf{X}_1 = \texttt{Conv}\left(\mathbf{X}^{\text{input}}\right), \quad \mathbf{X}_2 = \texttt{PC}\left(\mathbf{X}_1\right), \\ &\mathbf{X}_3 = \texttt{OC}\left(\mathbf{X}_2\right), \quad \boldsymbol{Q}^{\text{main}} = \texttt{Conv}\left(\mathbf{X}_3\right), \end{aligned} \tag{12}$$

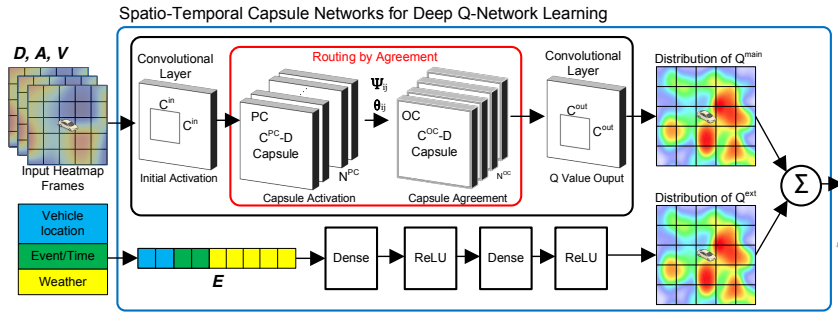

Fig. 2: Core module of spatio-temporal capsule-based Q-network.

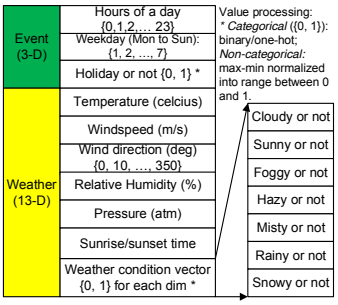

Fig. 3: External factors E.

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 [28]. Each cuboid in PC/OC of Fig. 2, as a capsule, 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} $C^{\mathsf{in}} \times C^{\mathsf{in}}$ convolutional kernel filters, while the output one has N^{out} $C^{\mathsf{out}} \times C^{\mathsf{out}}$ filters.

The learning process is presented as follows. 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 [13] is then applied upon the θ_{ij}'s, returning the coupling coefficient f_{ij} as

$$f_{ij} = \exp(\theta_{ij}) \cdot \left(\sum\nolimits_l \exp(\theta_{il}) \right)^{-1}. \tag{13}$$

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 [13]. All coefficients thus form an $N^{\mathsf{PC}} \times N^{\mathsf{OC}}$ weight matrix $\boldsymbol{\Psi}$. 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.*, $\mathbf{q}_{j|i} = \Psi_{ij}\mathbf{q}_i$, is then fed to capsule j as a weighted average by f_{ij}'s, *i.e.*,

$$\mathbf{e}_j = \sum\nolimits_i f_{ij}\mathbf{q}_{j|i}. \tag{14}$$

A *routing-by-agreement* between capsules [28] 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 [28], which is given by

$$\mathbf{o}_j(\mathbf{e}_j) \triangleq \frac{\|\mathbf{e}_j\|^2}{1 + \|\mathbf{e}_j\|^2} \cdot \frac{\mathbf{e}_j}{\|\mathbf{e}_j\|}. \tag{15}$$

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.*,

$$\theta_{ij} \leftarrow \theta_{ij} + \mathbf{q}_{j|i} \cdot \mathbf{o}_j(\mathbf{e}_j). \tag{16}$$

The resultant θ_{ij} is returned to Eq. (13) 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 E**: E is processed as in Fig. 3. Due to 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 [13] is given by

$$\begin{aligned} &\mathbf{E}_1 = \mathtt{Dense}(\mathbf{E}^{\mathsf{input}}), \quad \mathbf{E}_2 = \mathtt{ReLU}(\mathbf{E}_1), \\ &\mathbf{E}_3 = \mathtt{Dense}(\mathbf{E}_2), \quad \boldsymbol{Q}^{\mathsf{ext}} = \mathtt{Reshape}(\mathtt{ReLU}(\mathbf{E}_3)). \end{aligned} \tag{17}$$

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

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

$$\widehat{\boldsymbol{Q}} = \boldsymbol{Q}^{\mathsf{main}} + \boldsymbol{Q}^{\mathsf{ext}}. \tag{18}$$

Then, in coordination STRide finds the zone with the maximum Q-value in $\widehat{\boldsymbol{Q}}$ for dispatching (subject to ϵ-greedy). The set of parameters to be trained, denoted as $\boldsymbol{\Omega}$, is hence made of those contributing to $\boldsymbol{Q}^{\mathsf{main}}$ (including Ψ_{ij}'s and θ_{ij}'s) and those neuron weights for $\boldsymbol{Q}^{\mathsf{ext}}$. We implement a double deep Q-network (DQN) learning mechanism [31] within STRide to train the core deep Q-network.

5 Experimental Evaluation

Experimental settings: We compare STRide with the following typical algorithms: (1) *CRL*: an online learning framework [12] leveraging the conventional convolutional neural network (CNN) for coordination policy learning [25, 36]. (2) *CONT*: a conventional model-based approach, formulating a traffic control problem [7, 25] to find the dispatching and matching strategy based on the demand-supply balancing model. (3) *GD*: a heuristic vehicle dispatching without optimizing or learning the ride dynamics. MOD vehicles are greedily dispatched towards zones with the highest demand-supply imbalances [7, 27]. All evaluated schemes (the detailed parameter settings can be found in the corresponding references) use the same ride data, ETA module and learning emulator settings.

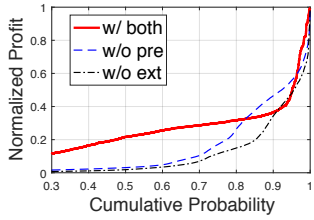

Fig. 4: Components (Uber).

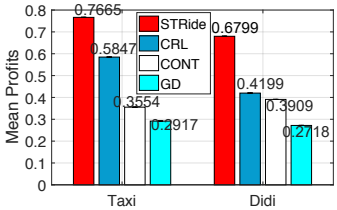

Fig. 5: Mean profit (Taxi&Didi).

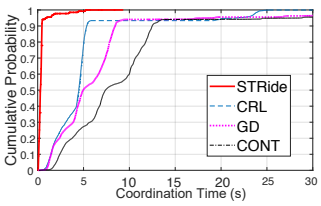

Fig. 6: Online overhead (Taxi).

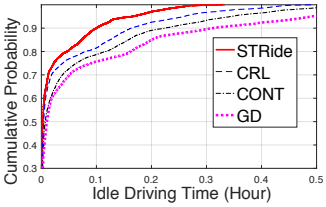

Fig. 7: Idle time (Taxi).

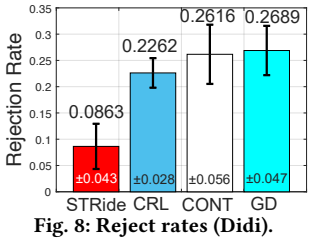

Fig. 8: Reject rates (Didi).

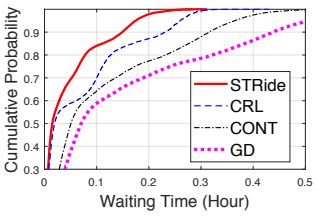

Fig. 9: Waiting time (Didi).

We comparatively evaluate the performance based on the following metrics: profits (platform; rides revenues minus relocation subsidies on fuel costs), idle driving time (the time when a vehicle is not occupied but still incurs the driving cost), reject rate (due to unavailability of vehicles nearby), (passenger's) waiting time, and coordination time (overhead of each online decision for MOD request matching and vehicle dispatching).

Unless stated otherwise, we use the following default parameters. In the learning emulator, we empirically set the total number of vehicles $M = 8,000$, $\alpha = 1.0$, $\beta = 0.3$, and $\epsilon = 0.1$. The total number of episodes is set to 12, while each of them 30 hours long. Each step lasts for 30 min. A rejection happens if the distance between a request and the nearest vacant vehicle is greater than 5.0 km. For Uber/Taxi, $a = 5$ and $b = 3$; for Didi, $a = 5$ and $b = 1$. The platform revenues are 20% of the ride fares, and the subsidies are 20% of the relocation costs based on studies in [10, 17, 27]. The city map is partitioned into $L_{\text{lon}} \times L_{\text{lat}} = 219 \times 219$ zones, 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 Q-network, we set each layer of the spatio-temporal capsule network as follows: $(N^{\text{PC}}, C^{\text{PC}}) = (8, 8)$, $(N^{\text{OC}}, C^{\text{OC}}) = (8, 50)$, $(N^{\text{in}}, C^{\text{in}}) = (64, 9)$, $(N^{\text{out}}, C^{\text{out}}) = (64, 6)$ (each Conv has a stride of 1 and ReLU activation) for main feature components in Fig. 2; $(C_1^{\text{Den}}, C_2^{\text{Den}}) = (10, 15^2)$ for the rest components handling the 18 external features ($L_{\text{ext}} = 16$ and 2-D location in $\mathbf{E}$ as in Fig. 3). With the above settings, we obtained the following experimental results.

Experimental Results: *Profits & Efficiency*: Fig. 4 shows STRide's normalized profits without either ride preference (pre) or external factors (ext), and the complete model with both components. Introducing ride preferences helps STRide relocate vehicles across popular zones with strong connectivities, leading to more pick-ups and hence more profits. Inclusion of auxiliary factors related to MOD rides help STRide capture more intrinsic routines in the spatio-temporal ride distribution, hence achieving more profits. Fig. 5 shows the mean profit (normalized *w.r.t.* each dataset) of all schemes. STRide is shown to achieve much higher profits than other schemes. Taking the large-scale NYC taxi dataset as an example, we show the CDFs of computation time in Fig. 6. Due to fast weight propagation across the layers, STRide achieves much better computational efficiency, which is essential for real-time coordination.

Idle driving time: Figs. 7 shows that shorter idle driving time for NYC Taxis. 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 lower subsidies and costs from the MOD platform, thus enhancing its overall profitability.

Rejection rate: Fig. 8 shows the lower mean rejection rates of the schemes (with standard deviation) in the Didi dataset. Thanks to STRide's proactive relocation of MOD vehicles, ride requests are rejected less. On the other hand, STRide accurately learns the locations of potentially high demands, and determines proactive dispatch regions within the rejection distance threshold.

Waiting time: Fig. 9 shows the shorter passenger waiting time of STRide than other related algorithms. With more proactive dispatching, the supplies match demands in time and hence overall shorter waits are needed. This way, the MOD platforms can enhance service quality and passengers' satisfaction.

6 Related Work

Numerous *optimization*-based schemes have been proposed for transportation management [7, 15, 32, 40], including control-based methodology [24], combinatorial optimization [37], and queueing theory [8]. Powered by exploding big data [23] and facilitating parallelism [9, 39], we have witnessed unprecedented advances in *learning*-based transportation management [20, 22, 26, 35]. Wen *et al.* [36] conducted preliminary studies upon learning-based MOD rebalancing. Lin *et al.* [21] studied an RL-based mechanism managing the fleet with a scalar-based neural network. Similarly, Xu *et al.* [37] explored the order dispatching, focusing on the sequential dispatch optimization problem. Oda *et al.* [25] proposed a fleet management system based on convolutional neural networks (CNNs), finding the optimal policy for relocating connected taxicabs.

7 Conclusion & Acknowledgment

We have proposed STRide, a spatio-temporal reinforcement learning framework for MOD coordination. Given MOD ride data, STRide builds a learning emulator for coordination policy training. 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 temporal external factors like events and weather conditions, are considered together to learn the coordination policy. We have conducted extensive data analytics and experimental evaluation on three large-scale datasets. STRide is shown to outperform state-of-the-arts, with lower request rejection rates, shorter waiting times, and higher platform profitability, often by more than 30% improvement.

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

References

[1] 2018. Global Petrol Prices. https://www.globalpetrolprices.com/gasoline_prices/.
[2] 2018. National Centers for Environmental Information, National Oceanic and Atmospheric Association (NOAA) – Data Tools: Local Climatological Data (LCD). https://www.ncdc.noaa.gov/cdo-web/datatools/lcd.
[3] 2018. Open Street Map. https://www.openstreetmap.org/.
[4] 2018. TLC Trip Record Data. http://www.nyc.gov/html/tlc/html/about/trip_record_data.shtml.
[5] 2018. Uber pickups in New York City. https://www.kaggle.com/fivethirtyeight/uber-pickups-in-new-york-city/data.
[6] 2019. Didi Chuxing Technology Co. www.didiglobal.com.
[7] Niels Agatz, Alan Erera, Martin Savelsbergh, and Xing Wang. 2012. Optimization for dynamic ride-sharing: A review. *European Journal of Operational Research* 223, 2 (2012), 295 – 303.
[8] Siddhartha Banerjee, Ramesh Johari, and Carlos Riquelme. 2015. Pricing in Ride-Sharing Platforms: A Queueing-Theoretic Approach. In *Proc. ACM EC*. 639–639.
[9] Qingpeng Cai, Aris Filos-Ratsikas, Pingzhong Tang, and Yiwei Zhang. 2018. Reinforcement Mechanism Design for e-Commerce. In *Proc. WWW*. 1339–1348.
[10] Rachel Dovey. 2017. 5 Florida Cities Team Up to Subsidize Uber Rides. https://nextcity.org/daily/entry/five-florida-cities-subsidize-uber-rides.
[11] Zhixuan Fang, Longbo Huang, and Adam Wierman. 2017. Prices and Subsidies in the Sharing Economy. In *Proc. WWW*. 53–62.
[12] Yong Gao, Dan Jiang, and Yan Xu. 2018. Optimize taxi driving strategies based on reinforcement learning. *IJGIS* 32, 8 (2018), 1677–1696.
[13] Ian Goodfellow, Yoshua Bengio, Aaron Courville, and Yoshua Bengio. 2016. *Deep Learning*. Vol. 1. MIT Press Cambridge.
[14] Jiawei Han, Jian Pei, and Micheline Kamber. 2011. *Data mining: Concepts and techniques*. Elsevier.
[15] Suining He and Kang G. Shin. 2018. (Re)Configuring Bike Station Network via Crowdsourced Information Fusion and Joint Optimization. In *Proc. ACM MobiHoc*. 1–10.
[16] Geoffrey E Hinton, Sara Sabour, and Nicholas Frosst. 2018. Matrix capsules with EM routing. In *Proc. ICLR*.
[17] Josh Horwitz. 2017. One year after the Uber-Didi merger, it's only getting harder to hail a ride in China. Respondent: https://qz.com/1045268/one-year-after-the-uber-didi-merger-its-only-getting-harder-to-hail-a-ride-in-china/ and waiting time: http://news.sina.com.cn/c/2017-07-26/doc-ifyinryq6222913.shtml.
[18] Yaguang Li, Kun Fu, Zheng Wang, Cyrus Shahabi, Jieping Ye, and Yan Liu. 2018. Multi-task Representation Learning for Travel Time Estimation. In *Proc. ACM KDD*. 1695–1704.
[19] Yaguang Li, Han Su, Ugur Demiryurek, Bolong Zheng, Tieke He, and Cyrus Shahabi. 2017. PaRE: A System for Personalized Route Guidance. In *Proc. WWW*. 637–646.
[20] Yexin Li, Yu Zheng, and Qiang Yang. 2018. Dynamic Bike Reposition: A Spatio-Temporal Reinforcement Learning Approach. In *Proc. ACM KDD*. 1724–1733.
[21] Kaixiang Lin, Renyu Zhao, Zhe Xu, and Jiayu Zhou. 2018. Efficient Large-Scale Fleet Management via Multi-Agent Deep Reinforcement Learning. In *Proc. ACM KDD*. 1774–1783.
[22] Zhidan Liu, Zhenjiang Li, Kaishun Wu, and Mo Li. 2018. Urban Traffic Prediction from Mobility Data Using Deep Learning. *IEEE Network* 32, 4 (July 2018), 40–46.
[23] Chenglin Miao, Qi Li, Lu Su, Mengdi Huai, Wenjun Jiang, and Jing Gao. 2018. Attack Under Disguise: An Intelligent Data Poisoning Attack Mechanism in Crowdsourcing. In *Proc. WWW*. 13–22.
[24] Fei Miao, Shuo Han, Shan Lin, John A Stankovic, Desheng Zhang, Sirajum Munir, Hua Huang, Tian He, and George J Pappas. 2016. Taxi Dispatch With Real-Time Sensing Data in Metropolitan Areas: A Receding Horizon Control Approach. *IEEE Trans. Automation Science and Engineering* 13, 2 (April 2016), 463–478.
[25] Takuma Oda and Carlee Joe-Wong. 2018. MOVI: A Model-Free Approach to Dynamic Fleet Management. In *Proc. IEEE INFOCOM*. 2708–2716.
[26] Ling Pan, Qingpeng Cai, Zhixuan Fang, Pingzhong Tang, and Longbo Huang. 2018. A Deep Reinforcement Learning Framework for Rebalancing Dockless Bike Sharing Systems. In *Proc. AAAI*.
[27] Lisa Rayle, Danielle Dai, Nelson Chan, Robert Cervero, and Susan Shaheen. 2016. Just a better Taxi? A survey-based comparison of taxis, transit, and ridesourcing services in San Francisco. *Transport Policy* 45 (2016), 168–178.
[28] Sara Sabour, Nicholas Frosst, and Geoffrey E Hinton. 2017. Dynamic routing between capsules. In *Proc. NIPS*. 3856–3866.
[29] Richard S Sutton and Andrew G Barto. 1998. *Introduction to Reinforcement Learning*. Vol. 135. MIT Press Cambridge.
[30] Jian Tang, Meng Qu, Mingzhe Wang, Ming Zhang, Jun Yan, and Qiaozhu Mei. 2015. LINE: Large-scale Information Network Embedding. In *Proc. WWW*. 1067–1077.
[31] Hado Van Hasselt, Arthur Guez, and David Silver. 2016. Deep Reinforcement Learning with Double Q-Learning.. In *Proc. AAAI*, Vol. 2. 5.
[32] Erwin Walraven, Matthijs TJ Spaan, and Bram Bakker. 2016. Traffic flow optimization: A reinforcement learning approach. *Engineering Applications of Artificial Intelligence* 52 (2016), 203–212.
[33] Dong Wang, Wei Cao, Jian Li, and Jieping Ye. 2017. DeepSD: Supply-demand prediction for online car-hailing services using deep neural networks. In *Proc. IEEE ICDE*. 243–254.
[34] Zheng Wang, Kun Fu, and Jieping Ye. 2018. Learning to Estimate the Travel Time. In *Proc. ACM KDD*. 858–866.
[35] Hua Wei, Guanjie Zheng, Huaxiu Yao, and Zhenhui Li. 2018. Intellilight: A reinforcement learning approach for intelligent traffic light control. In *Proc. ACM SIGKDD*. 2496–2505.
[36] Jian Wen, Jinhua Zhao, and Patrick Jaillet. 2017. Rebalancing shared mobility-on-demand systems: A reinforcement learning approach. In *Proc. IEEE ITSC*. 220–225.
[37] Zhe Xu, Zhixin Li, Qingwen Guan, Dingshui Zhang, Qiang Li, Junxiao Nan, Chunyang Liu, Wei Bian, and Jieping Ye. 2018. Large-Scale Order Dispatch in On-Demand Ride-Hailing Platforms: A Learning and Planning Approach. In *Proc. ACM KDD*. 905–913.
[38] Junbo Zhang, Yu Zheng, and Dekang Qi. 2017. Deep Spatio-Temporal Residual Networks for Citywide Crowd Flows Prediction.. In *Proc. AAAI*. 1655–1661.
[39] Junbo Zhang, Yu Zheng, Dekang Qi, Ruiyuan Li, Xiuwen Yi, and Tianrui Li. 2018. Predicting citywide crowd flows using deep spatio-temporal residual networks. *Artificial Intelligence* 259 (2018), 147 – 166.
[40] Huanyang Zheng and Jie Wu. 2017. Online to Offline Business: Urban Taxi Dispatching with Passenger-Driver Matching Stability. In *Proc. IEEE ICDCS*. 816–825.