Spatio-temporal Adaptive Pricing for Balancing Mobility-on-Demand Networks

SUINING HE and KANG G. SHIN, The University of Michigan, Ann Arbor, USA

Pricing in mobility-on-demand (MOD) networks, such as Uber, Lyft, and connected taxicabs, is done adaptively by leveraging the price responsiveness of drivers (supplies) and passengers (demands) to achieve such goals as maximizing drivers' incomes, improving riders' experience, and sustaining platform operation. Existing pricing policies only respond to short-term demand fluctuations without accurate trip forecast and spatial demand-supply balancing, thus mismatching drivers to riders and resulting in loss of profit.

We propose CAPrice, a novel adaptive pricing scheme for urban MOD networks. It uses a new spatio-temporal deep capsule network (STCapsNet) that accurately predicts ride demands and driver supplies with vectorized neuron capsules while accounting for comprehensive spatio-temporal and external factors. Given accurate perception of zone-to-zone traffic flows in a city, CAPrice formulates a joint optimization problem by considering spatial equilibrium to balance the platform, providing drivers and riders/passengers with proactive pricing "signals." We have conducted an extensive experimental evaluation upon over 4.0×10^8 MOD trips (Uber, Didi Chuxing, and connected taxicabs) in New York City, Beijing, and Chengdu, validating the accuracy, effectiveness, and profitability (often 20% ride prediction accuracy and 30% profit improvements over the state-of-the-arts) of CAPrice in managing urban MOD networks.

CCS Concepts: • **Information systems** → *Spatial-temporal systems*; *Geographic information systems*; *Data mining*;

Additional Key Words and Phrases: Mobility-on-demand, ride sharing, adaptive pricing, deep learning, traffic prediction, flow balancing, sharing economy, smart transportation

ACM Reference format:
Suining He and Kang G. Shin. 2019. Spatio-temporal Adaptive Pricing for Balancing Mobility-on-Demand Networks. *ACM Trans. Intell. Syst. Technol.* 10, 4, Article 39 (July 2019), 28 pages.
https://doi.org/10.1145/3331450

1 INTRODUCTION

With the rapid urbanization and emergence of smart cities [19, 57, 60, 67], the Mobility-on-Demand (MOD) economy [27, 52], pioneered by Uber, Lyft, Didi, and connected taxicabs, has witnessed significant growth in recent years, meeting passengers' demands with immediate access to, and convenient provisioning of urban transportation services. According to Reference [8], the global MOD market is expected to exceed $228 billion by 2022. Given its socio-economic significance, how to design business and management strategies for MOD platforms is an important and challenging problem.

Authors' addresses: S. He and K. G. Shin, The University of Michigan, Ann Arbor, Department of Electrical Engineering and Computer Science, MI, 48109-2121; emails: {suiningh, kgshin}@umich.edu.

2157-6904/2019/07-ART39 $15.00
https://doi.org/10.1145/3331450

Demands (riders/passengers/customers/requesters) and supplies (drivers/cabs/vehicles) in MOD markets dynamically change over time. Unless properly managed, mismatches between demands and supplies (such as a sudden increase of ride-requests in a city zone with fewer drivers/cabs available nearby) can cause significant loss of revenue to MOD businesses and discourage both customers and drivers. Since the drivers of many MOD (e.g., ride-sharing) platforms are often freelancers rather than full-time workers, it is difficult in practice to enforce their spatio-temporal availability directly.

For MOD service to be adaptive to dynamically changing supplies and demands, a platform must monitor the market status and publish appropriate "signals"—*pricing policies*. Specifically, platforms such as Uber and Lyft post the real-time "heatmap" (discretized city zones) of ride-requests to the drivers based on historical and recent data. In case of a sharp demand burst (say, Super Bowl night) exceeding supplies somewhere, the platform increases the fare rate in those "hot" zones by applying a certain multiplier to the standard rate (accompanied by dynamic subsidies to drivers). Such "surge pricing" [6, 38] (or "prime time," as Lyft calls its version of Uber's "surge pricing") has been adopted by many popular ride-sharing platforms.

The surge pricing is to narrow the gap between supply and demand [6, 14]. First, more drivers are enticed with potentially higher earnings, and hence move to hot zones and serve requests there. Second, customers who under-value the service will likely give up and seek alternative transportation means, thus increasing the chance of matching drivers with the remaining customers. This profit-driven market's self-adaptation [9] appears promising, but has not yet been fully realized for the following reasons: Most existing demand/ride forecasts focus on a scalar-based time-series analysis, often modeling trips that connect zones locally/independently of each other. Due to their simplicity and lack of realism, the demands displayed in drivers' UIs have been reported neither credible nor accurate [14]. Many of the drivers chasing surges have found little to low demand when they arrive at the surge zones, earning less and thus getting discouraged. Some MOD communities suggest inexperienced drivers ignore delayed surges,[1] as the actual earnings of blind/greedy surge chasing (so-called "bunch-up" or "wild-goose chase") do not meet expectations in practice.[2]

The absence of proactive "signals" also exacerbates the supply-demand mismatch and incurs long waits for the passengers. Rather than affording surged or even uncapped prices (say, by Uber) again and again, passengers in need of timely rides prefer *proactive* distribution of vehicles *w.r.t.* location and time [7], particularly during rush hours. However, without *joint balancing* of vehicles among zones, reliance on only price surge cannot steer earning/profit-driven supplies towards demands in a timely and effective way. Due to the supply-demand imbalance, the surveys conducted in China [25] in 2017 show that 81.7% of respondents complained about more hailing difficulties and up to 129.2% longer waiting time than in 2016. Starving demand, as intended by surge pricing, is not a sustainable option.

As an inaccurate and delayed market signal, the conventional price surging cannot proactively match the incoming ride requests, while driver supplies are not incentivized *w.r.t.* location and time. After the initial MOD hype [8, 9], long-term service sustainability is thus often compromised by such temporary profit-greedy control of conventional surges. Pricing signals are merely the levers, not the sole and ultimate market goals.

To improve service sustainability and platform utilization, we propose `CAPrice`, a novel adaptive pricing scheme for urban MOD networks. This adaptive pricing for mobility-on-demand depends on accurate perception of the dynamic market status and consequent fine-grained management.

[1] https://www.sfgate.com/business/article/Report-says-Uber-surge-pricing-has-a-twist-some-6597012.php.
[2] https://maximumridesharingprofits.com/advice-new-uber-drivers-dont-chase-surge/.

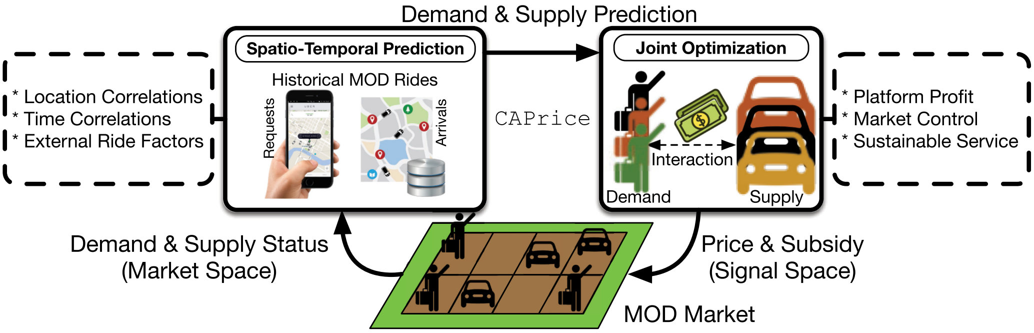

Fig. 1. Illustration of adaptive pricing and balancing for MOD networks.

As shown in Figure 1, it uses the historical MOD service requests and rides to forecast the demand-supply patterns in the near future (say, a few minutes later). Then, it *jointly* optimizes the pricing policies based on profit, market control, and sustainable service (zone balancing), bridging the signal (price, subsidy) and market (demand, supply) spaces via more responsive connections. By adaptively matching the spatio-temporal pricing signals with the market divisions, the supplies meet demands *spatially* and *proactively*.

Specifically, we first design a spatio-temporal ride prediction scheme based on deep capsule neural network, called STCapsNet, which accurately forecasts future demands/supplies via structural and vectorized *capsules*—structured groups of neurons [51]. The neurons in each capsule produce a vector, taking into account essential spatial hierarchies between simple and complex objects in an input image. Considering input pick-up/drop-off distributions as images, STCapsNet captures the inherent correlations between pixels (city zones) by a novel vectorization structure, acquiring far more knowledge than conventional scalar-based approaches, including convolutional neural network (CNN) [51].

Based on more accurate demand/supply predictions, CAPrice formulates a joint optimization framework, anticipating prices and subsidies towards incoming ride-requests and thus incentivizing drivers more responsively to customers than previous greedy surge-chasing. In particular, with the formulation of *spatial equilibrium* in the vehicle (re)distribution and long-term ride-request patterns, CAPrice follows the spirit of dynamic pricing [9] but *jointly* optimizes the distributions of incentive-compatible prices and subsidies for the coming rides. Therefore, CAPrice proactively handles demand-supply imbalances via more responsive driver flows between zones.

The main contributions of this article are:

- *Accurate Prediction of Ride Requests via a Spatio-Temporal Capsule Network* (Section 4): CAPrice leverages STCapsNet to comprehensively capture the inherent pick-up/drop-off relationship among city zones by the novel vectorized neuron structures. Integrated with spatio-temporal ride distributions and external ride factors, STCapsNet achieves highly accurate prediction of rides in complex urban networks.
- *Adaptive Pricing for Mobility-on-Demand (MOD) Platforms* (Section 5): CAPrice augments the MOD platforms with enhanced responsiveness to spatio-temporal demand-supply fluctuations and adaptive pricing for profit and utilization maximization. Specifically, it takes the forecasts of dynamic demand-supply patterns as input and incentivizes the proactive distribution of drivers adaptively towards customer demands, thus achieving profitable and sustainable platform operation.
- *Large-scale Data-driven MOD Network Studies* (Section 6): We have conducted extensive data-driven studies to validate the accuracy, effectiveness, and profitability of CAPrice. Our

experimental evaluation based on large-scale MOD data (more than 4.0×10^8 trip records in total) from New York City, NY, as well as Beijing and Chengdu, China, has shown that CAPrice can accurately predict ride (demand/supply) patterns (often reducing error by more than 25% over state-of-the-art [39, 66]) and provide a more profitable pricing guideline for MOD networks (often 30% more revenues, faster market clearance, and shorter wait time than conventional pricing policies of exiting platforms [14, 35, 37]).

Besides the latest comprehensive validation of the application of capsule-networks [51], we establish critical profit-maximizing insights as deployment references for the MOD platforms, such as pricing the over-demanded zones more spatio-temporally, while subsidizing their unattractive but demanded peers more, especially those unbalanced ones due to their highly attractive neighbors. The importance of ride prediction accuracy to market pricing is also demonstrated. With CAPrice's *joint* formulation of market forecasting and pricing, we expect that the MOD platforms would benefit both by accounting for predictable demand-supply patterns (say, morning rush hours) while smoothing out short-term fluctuations (say, due to weather conditions).

Despite the current prototype studies on urban ride-sharing [5, 21] and connected taxi [25], the general framework of CAPrice can be easily extended to other emerging autonomous or connected MOD services, such as self-driving rental cars [43, 47].

The rest of this article is organized as follows: After discussing the related work (Section 2), we first overview the concepts, problems, and framework (Section 3). Then, we present capsule network design for ride prediction (Section 4) and joint optimization formulation of adaptive pricing (Section 5). Our prototype is evaluated with MOD datasets (Section 6), followed by deployment discussions (Section 7) and concluding remarks (Section 8).

2 RELATED WORK

We briefly review the related work and discuss its differences from CAPrice.

Traffic Prediction: With the growing need of handling more complex city systems [67], there have been numerous efforts in urban traffic analytics, enabled by the advent of mobile/big data science [58, 64, 66] and IoT [20, 41, 60]. Beyond the previous studies applying "shallow" model structures [34, 53, 65], deep learning, powered by growing datasets and advancing parallelism, has emerged as more versatile and promising traffic analytics [54]. Recurrent neural networks (RNNs), enhanced by long-short-term memory (LSTM) techniques [24], learn the scalar-based traffic sequence. Based on the success in understanding photos [30], CNN is explored for traffic monitoring [39], modeling neighborhood regions into scalars and locally capturing (pooling) spatial dependencies of zones.

Though advances have been achieved, existence of close/distant points-of-interests (PoIs) and complex zone-to-zone ride correlations have been overlooked by the scalar and region-based learning of RNNs and CNNs. The integration of both still could not fully capture the macro picture of metropolitan traffic [62] due to their scalar nature. Unlike these previous studies, CAPrice adopts the capsule networks [51] in a novel spatio-temporal manner. A capsule neural network is a further developed and structured version of *parse tree* for joint image segmentation and recognition [22], deriving semantic and interpretable representations from input images. Beyond conventional CNNs, the neurons inside a capsule are activated for various physical properties of the input images for better instantiation of the objects of interest [23]. We take advantage of this strength in CAPrice and model the shared rides into heatmap frames, instantiating the correlated/co-occurring ride demands and supplies across zones. A dynamic routing mechanism [51] is applied further to reduce errors and the size of training sets for capsule network. Compared to scalar-based conventional CNNs, STCapsNet captures far more spatial knowledge between locations due to its comprehensive vector-based neuron structures, hence achieving better accuracy.

MOD Pricing & Balancing: To support MOD development, many analytical designs related to drivers-riders matching, dispatching, and pricing have been proposed in recent years. Orthogonal to our work, Oda et al. [46] and Xu et al. [59] explored the online learning for vehicle fleet management and matching without accounting for incentivization with pricing. Fang et al. [15] analyzed the price, profit, and social welfare maximization for the macro MOD ride-sharing economy. Banerjee et al. [10] studied a queueing-theoretic price setting. However, their work did not consider the spatio-temporal effects of pricing upon dynamic flows, which is essential for practical system management. Bimpikis et al. [11] also explored the impact of price discrimination on ride-sharing but simplified demand patterns and MOD network structures in their balance analysis.

Despite their pioneering studies, few of these analytical designs have considered the interplays of ride flows, equilibrium, and pricing for profitable MOD management. They did not take into account proactive data-driven ride monitoring either, which is critical for platform profitability and sustainability. Via comprehensive data analytics and experimental evaluation, CAPrice fills these gaps and provides important insights in developing a more profitable and sustainable MOD platform.

3 PRELIMINARIES & SYSTEM ARCHITECTURE

We first overview the preliminary concepts and definitions used in CAPrice (Section 3.1) and then discuss its problem framework and system architecture (Section 3.2).

3.1 Preliminary Concepts

Introduced below are the essential concepts and definitions used in CAPrice.

Definition 1. City Zones & Time Intervals: To facilitate the search and visualization across a spacious city area, we discretize the entire map (space) into a total of G non-overlapping zones (cells), each of which is denoted as Z_i ($1 \leq i \leq G$) and $\mathbf{Z} = \{Z_1, \ldots, Z_G\}$. For our prototype, we divide the entire map longitudinally and latitudinally into $W \times H$ rectangular grids of the same size ($G = W \times H$), and their shape and size (subject to map) can be easily customized according to each specific task, balancing between estimation granularity and computational efficiency. Similarly, we discretize time domain with a certain interval (say, every 30m). For simplicity, "the period k" or index k refers to the kth time interval. Rides within each k are aggregated for analysis.

Definition 2. Fulfilled Rides: Each MOD ride (trip) comes with a departure and an arrival. Then, the number of fulfilled departures (demands or requests) aggregated from Z_i in the kth time interval, denoted as D_i^k, refers to the number of cars that pick up the passengers from there. It captures the ride demands in a certain zone.[3]

Similarly, let A_i^k be the number of fulfilled arrivals (actual drop-offs) in Z_i at period k. Given a discretized city map and ride records, at period k the fulfilled demand distributions of all zones form $\mathbf{D}^k = [D_1^k, \ldots, D_i^k, \ldots, D_G^k]$. Likewise, the arrivals are modeled as $\mathbf{A}^k = [A_1^k, \ldots, A_i^k, \ldots, A_G^k]$. The set of fulfilled rides is defined as $\boldsymbol{T}^k = \{\mathbf{D}^k, \mathbf{A}^k\}$, $i \in \{1, \ldots, G\}$.

Definition 3. Mobility-on-Demand (MOD) Network: Given the discretized zones $\mathbf{Z}$ and rides $\boldsymbol{T}^k$ connecting them, we express the city map as networks of zones connected by spatio-temporal

[3] A "weaker" consideration of "ride demand" may refer to the initiated requests via mobile apps, which are often difficult to monitor. Limited by resource and the unavailability of ride-request data from service providers (due to privacy or confidential concerns [14]), as in many state-of-the-art approaches [14, 66], we focus on fulfilled demands (actual departures) when referring to "demands."

MOD transportation. During a certain period, two zones i and j are connected by flows of (m_{ij}, m_{ji}) MOD cars, where m_{ij} represents the volume of car flows from i to j, and vice versa $(i, j \in \{1, \ldots, G\})$.

For MOD network's profitability and sustainability, we need to consider the following two critical questions: (1) *when and where rides will happen* and (2) *how to determine their prices and subsidies proactively*. To answer the first question, we structure the spatio-temporal ride distributions across zones into *frames* of *heatmaps*.

Definition 4. Ride Heatmap Frames: Each frame of fulfilled rides $\boldsymbol{T}^k$ at the kth time interval can be represented by a 3D matrix $\boldsymbol{F}^k$, i.e., in the form of $W \times H \times 2$, where the first two dimensions correspond to the city map discretization (Definition 1) and the third represents two flows of $\mathbf{D}^k$'s and $\mathbf{A}^k$'s (Definition 2). In a heatmap, the warmer colors represent more departures/arrivals.

This way, with each Z_i as a "pixel," we can design a ride prediction algorithm inspired by image processing, leveraging cross spatial and temporal relations within a sequence of historical frames:

Definition 5. Spatio-Temporal Ride Prediction Algorithm: An estimation function $\mathcal{P}(\cdot)$ takes, as input, the geographic zones' information, external factors $\boldsymbol{\Psi}^k$ at target period k, and the w latest consecutive frames $\{\boldsymbol{F}^{k-w}, \ldots, \boldsymbol{F}^{k-1}\}$ to predict or recover an image frame $\widehat{\boldsymbol{F}}^k$ via their cross spatial/temporal relations.

For the second question, given anticipation of rides in the market space (Figure 1), price and subsidy responses in signal space are formed:

Definition 6. Price & Subsidy of a MOD platform: Price p refers to the payments from passengers as their trip fares, while s refers to the subsidies for drivers in (re)locating to a zone for a match (passenger pick-up). The platform strategically determines the tuples of $\{p, s\}$ to steer demand and supply subject to the related regulations and laws. In a spatio-temporal setting as in Section 1, the market is divided and thus has $\{p_i, s_i\}_{i=1}^{G}$.

As a driver's earnings from each trip are also proportional to the service time and distance (standard rate) of that trip, we retrieve and focus on only the dynamically/lawfully manipulated part by the MOD platform [6]. A common practice (say, adopted by Uber/Lyft) is a dynamic *multiplier* applied upon the standard rate. To illustrate this, we also crawled via Uber API and showed in Figure 2 the 12-hour real-world price multipliers (from *Wall Street Journal* headquarters to Goldman Sachs) with surge effects [6] in UberX, NYC, reflecting the hidden dynamics of demand and supply. As the headquarters is located in the central business district of NYC, far more evening departures (hence, higher surges) happen than in the morning.

We also observe that subsidizing is a common incentivization practice for many MOD platforms [15], especially when they are expanding business in new service area, enticing new drivers' entries. Furthermore, compensations may also apply if they relocate to less popular zones due to resultant gasoline/labor expense. Both largely constitute the cost of the MOD platform. Different from subsidies or compensations, price adaptation is considered to cater for demand-supply interaction in a more mature market and their popular zones. The price multipliers, along with subsidies and steered vehicle flows, are to be determined (detailed in Section 5).

Following the common goals of commercial MOD platforms [15], the vanilla pricing problem summarizing above can be considered as a *joint* multi-objective optimization framework, i.e.,

Definition 7. Vanilla Problem for Spatio-Temporal Adaptive Pricing: Given states of drivers and passengers—i.e., interaction of supply and demand—as well as historical records until the $(k-1)$th period, the platform estimates rides in all zones $\{\widehat{\boldsymbol{F}}_i^k\}_{i=1}^{G}$ and proactively determines $\{p_i, s_i\}_{i=1}^{G}$ at

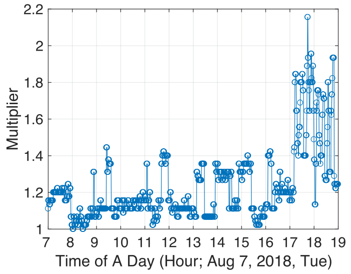

Fig. 2. Illustration of pricing multipliers.

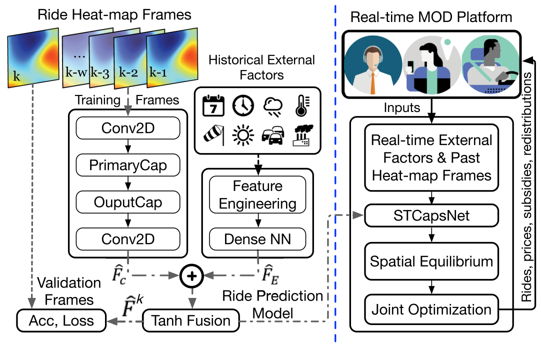

Fig. 3. *CAPrice*'s architecture.

k, such that the aggregate profit (net income) is maximized. The vanilla problem thus takes the following preliminary form:

$$\arg\max_{\{p_i, s_i\}} \quad \sum_{i=1} F_i^k \cdot (p_i - s_i), \tag{1}$$

$$\text{s.t.} \quad \widehat{F}^k = \mathcal{P}(\Psi^k, \{F^{k-w}, \ldots, F^{k-1}\}), \quad p_i \in [p_{\min}, p_{\max}], \quad s_i \in [s_{\min}, s_{\max}], \quad \forall i \in \{1, \ldots, G\},$$

$$\text{flows } F^k \text{ adapted from } \widehat{F}^k \text{ due to } p_i\text{'s and } s_i\text{'s.}$$

Our framework automatically determines the popularity of zones and their consequent $\{p_i, s_i\}$'s. Since demand and pricing have mutual effect in practice, we can adjust the resultant F^k from $\widehat{F}^k$ to account for the pricing effect. We will further extend the settings of this vanilla form in the following sections (including the final core formulation in Section 5). CAPrice is general enough for other more sophisticated objectives customized by the platforms.

3.2 System Architecture

Figure 3 shows the architecture of CAPrice, which consists of two major modules. At the first module on the left, *ride prediction* by STCapsNet (Section 4), historical MOD trips at each time interval, in the form of heatmap frames, are fed to our proposed neural networks. Specifically, the heatmaps in a sliding window are fed to a capsule network where spatio-temporal ride distribution is examined and captured. Meanwhile, relations between rides and external factors like weather are learned by a fully connected neural network. Outputs from both networks are fused via a tanh function [16], and the ride estimation is returned for training validation or testing.

At the second module on the right, given the predicted rides, CAPrice formulates a *joint optimization* problem (Section 5) that maximizes the platform profits and steers drivers towards the corresponding demands with adaptive pricing signals (Section 5.2) for spatial equilibrium or balance (Section 5.3). Optimized pricing signal *w.r.t.* each zone is returned to the MOD platform.

4 SPATIO-TEMPORAL RIDE PREDICTION VIA STCAPSNET

We present a model for predicting spatio-temporal MOD ride requests by (i) analyzing the real-world datasets (Section 4.1) for ride analysis (Section 4.2), and (ii) presenting the capsule network (Section 4.3) and the framework with the prediction model (Section 4.4).

4.1 Data Overview

We use the following MOD datasets to evaluate CAPrice:

Table 1. External Factors for the Four Datasets in Our Experimental Evaluation

Datasets	External Factors		
	Weather	**Holiday**	**Traffic Condition**
NYC Uber	7 types of conditions, temperature, sky visibility, wind speed, wind direction, humidity, pressure, sunrise/sunset time (by NOAA)	Federal and bank holidays	Average vehicle speed, average travel time
NYC Taxi			
Chengdu Didi		\	\
Beijing Taxi	16 types of conditions, temperature, wind speed	National holidays	\

- *Ubers in New York City* [5]: The ride-sharing data (GPS coordinates of pick-up locations and timestamps) of Uber in New York City (NYC) between 2014-Q2Q3 and 2015-Q1Q2 contain 18,804,806 trips, covering a bounding box of $[40.65°N, 40.85°N, -74.05°W, -73.77°W]$.
- *Yellow Taxis in NYC* [4]: The ride data (with pick-up/drop-off locations and timestamps) of Yellow Taxis in 2014, 2015, and 2016-Q1Q2 contain 362,101,984 trips. As both datasets of Uber and Yellow Taxi share the same pick-up/drop-off region identifiers [4], we discretize the NYC map likewise. Taxis are shown to share similar mobility behaviors and patterns with ride-sharing platforms [49, 56], leading to transferable knowledge in our experimental studies.
- *Taxis in Beijing, China*: The connected taxicab data (pick-up/drop-off zones and timestamps) are processed from their real-time GPS trajectories in Beijing between 2013 and 2016, containing 24,193,552 rides [63, 66].
- *Didi Chuxing in Chengdu, China*: Didi Chuxing, Inc., is a major Chinese ride-sharing service provider, handling 20M rides on a daily basis in 2016. The ride-sharing data (with pick-up/drop-off locations and timestamps) provided by Didi [1] contain a total of 6,744,508 trips from the city of Chengdu, Sichuan Province, China, in November 2016. The rides cover a bounding box of $[30.653°N, 30.728°N, 104.043°E, 104.130°E]$.

For each of these cities, `CAPrice` also considers weather, holidays, and traffic conditions as external factors related to the ride distribution, subject to the data availability (detailed in Table 1). Next, we briefly go through some MOD trip patterns pertinent to our model.

4.2 MOD Ride Pattern Analysis

We have conducted an extensive pattern analysis with the above datasets to design $\mathcal{P}(\cdot)$ in Equation (1). Due to space limit and its representative patterns, we will focus on NYC Taxis as an illustration.

A. Spatial Trip Patterns ($\{F_i\}_{i=1}^{G}$): The metropolitan MOD feature of a potential passenger depends highly on her/his working and living places, personal preferences, and many other social/demographic factors [9, 13] that lead to spatial diversity in ride distributions. Figure 4 shows the departure heatmap of NYC on May 31 (00:00–12:00), 2016. We can observe heavy trip demands from midtown Manhattan. We can also see that notable demands from the JFK International Airport and nearby PoIs or recreational areas (say, casinos and resorts), reflecting the spatial complexity of ride distributions.

B. Temporal Demand Patterns ($\{F^{k-w}, \ldots, F^{k-1}\}$): The demand will likely vary with day of week and/or time of day. Figure 5 shows the temporal dynamics of MOD trips within two weeks (with Monday and Saturday flows highlighted), showing different repetitive patterns of weekdays and weekends.

C. External Factors (Ψ^k): Fine-grained external factors may also be subject to various meteorological conditions, demographics, cultures, and social trends, such as:

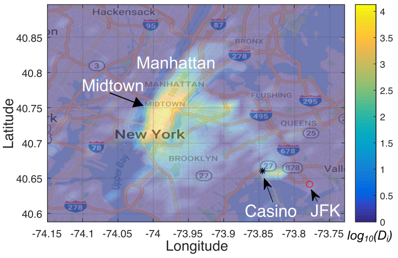

Fig. 4. Spatial distribution ($\log_{10}(\cdot)$) of MOD departures (May 31, 2016, NYC).

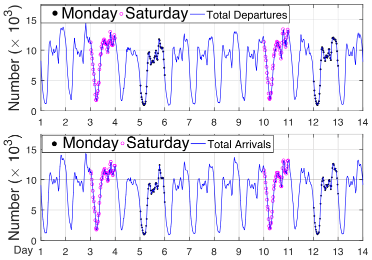

Fig. 5. Temporal dynamics of MOD traffic (two weeks in May 2016, NYC).

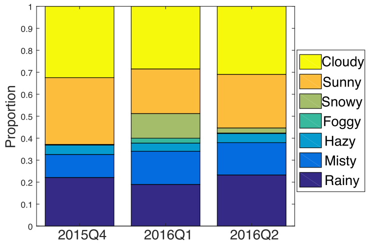

Fig. 6. Proportions of NYC meteorological conditions in three quarters.

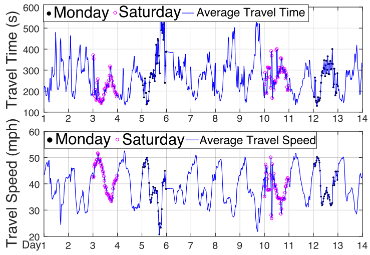

Fig. 7. Illustration of mean travel time (sec) & speed (mph) in NYC.

(1) *Meteorology*[4]: A passenger's preference on MOD service is affected by the weather conditions. To model the correlation, our comprehensive meteorological data for NYC Uber/Taxis and Chengdu Didi include temperature, humidity, air pressure, wind speed/direction, sky visibility, and sunrise/sunset time of that day. Figure 6 illustrates such seasonal diversity in meteorological conditions. Besides, due to the difference of sources, for rides in NYC and Chengdu, we utilize 7 types of binary weather conditions (e.g., sunny or not), while for Beijing we use 16.

(2) *Metropolitan Traffic Condition* (TC): TC characterizes the overall congestion levels of city roads [62], which also influence the fulfilled departures/arrivals for a MOD network. Our model considers the average vehicle speed (unit: mph) and average travel time (unit: sec) of the road segments.[5] Note that TC data from the city monitor are measured independently from the MOD platform. In practice, real-time metropolitan TC can be easily monitored via road-side cameras, magnet-coil sensors at the intersections, and toll-office records, serving as external hints for MOD ride prediction. Figure 7 further illustrates the weekly travel time and speed variations in NYC. We can see the travel time peaks and travel speed valleys in Figure 7, which are correlated with the weekday morning/evening rush hours in Figure 5 (the same weeks).

(3) *Event*: As in Figures 5 and 7, the peaks in weekday/weekend/holiday traffic may differ due to different rush hours, neighboring events, and travel purposes. Our event dataset also includes weekday/weekend parsing and public holidays. In particular, for NYC, `CAPrice`

[4] National Centers for Environmental Information, National Oceanic and Atmospheric Association (NOAA). https://www.ncdc.noaa.gov/cdo-web/datatools/lcd.

[5] New York City (NYC) Real-Time Traffic Speed Data Feed (Archived) Five Minute Intervals, 2015–2018. http://data.beta.nyc/dataset/nyc-real-time-traffic-speed-data-feed-archived.

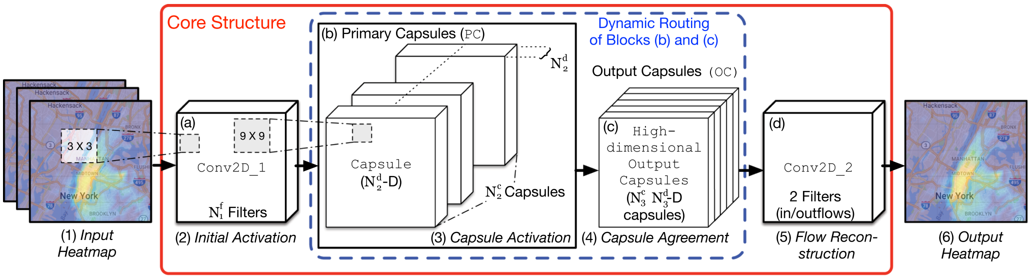

Fig. 8. Structure and phases of capsule network in STCapsNet: (a) input convolutional block, (b) primary capsules, (c) output capsules, and (d) output convolutional block.

uses federal and stock-market/bank holidays, while for Beijing and Chengdu it takes into account national holidays, if any (no public holidays during November in China).

Note that CAPrice does not directly predict the weather and traffic conditions at the target period k. Based on the general spatio-temporal continuity in weather and traffic, we can adopt those at the latest period (say, from $(k-1)$). Estimating weather/traffic conditions from other state-of-the-arts [66] can be also applied for refinement here.

In what follows, we detail the design of STCapsNet, which comprehensively accommodates the above factors for accurate ride prediction.

4.3 Capsule Network in STCapsNet

A capsule, consisting of a structured group of neurons, instantiates fully or partially the input heatmap matrix representing the zone-to-zone MOD ride distributions (including ride co-occurrence of close and distant zones). The major technical insights for a capsule network lie in (i) its layer-to-layer *vector information flow* and (ii) replacement of max-pooling [16, 51] with *capsule routing*. We first overview the structures of the capsule network in our STCapsNet and then discuss its learning mechanism, followed by detailed formulation of capsules.

A. Structure Overview: Figure 8 overviews the phases (1)–(6) with the core blocks (a)–(d) for our capsule-based ride prediction:

(1) Input Heatmap: We first pre-process the aggregated ride patterns into high-dimensional heatmap frames as in Definition 4.

(2) Initial Activation: Each sliding window of heatmap frames is first fed to a traditional 2D convolutional block (Conv2D_1 in (a)) [16] of N_1^f filters (with 3×3 kernels) for initial feature detection. This block helps transform the raw pixel intensities into activities of the local feature detector, leading to detection of the local trip patterns.

(3) Capsule Activation: After Conv2D_1 perceives scalar-based local features of close zones, a group of *Primary Capsules* (PC in (b)) with N_2^c N_2^d-dimensional capsules takes in its predecessor's convolutionalized outputs and constructs *vectorized* feature expressions [51] among all zones. Specifically, a capsule layer is reshaped from a Conv2D module consisting of $N_2^c \times N_2^d$ filters (with 9×9 kernels) [51]. Thus, each capsule (a cuboid in (b)) is an individual group of N_2^d convolutional units (neurons). The output from PC consists of N_2^c vectors, each of which is from a capsule activated by certain correlated zones. This way, correlations, including ride likelihood and direction, between close and distant zones are further extracted by this vectorization structure.

(4) *Capsule Agreement*: A set of N_3^c N_3^d-dimensional capsules, *Output Capsules* (OC in (c)), then concatenates, compresses, and refines the outputs from primary capsules. A linkage (route) between capsules i in PC and j in OC is assigned with a conventional weight Ω_{ij} as well as a coupling probability f_{ij} for agreement computation (detailed later). N_3^d tends to be set high in practice [51]. OC is designed to bridge the output side and PC, updating capsule weights according to training loss via "routing by agreement" [51] of the vectors.

(5)–(6) *Flow Reconstruction & Output Heatmap*: The final 2D convolutional block (Conv2D_2 in (d)) reconstructs and matches OC outputs (vectors) back towards the *Output Heatmap Frames* (scalar-based).

Note that two convolutional blocks, (2) and (5), simply serve as input/output heatmap transformation, while the rest of the structures render capsule network vastly different from CNN. As a general framework for ride prediction, the number/size/dimension of the blocks can be customized (detailed in Section 6.1). Next, by comparing it with CNNs, we discuss how the capsule network learns the rides.

B. Learning by Vectorization & Routing: As in Figure 9, conventional CNNs [23] subdivide the image into many local representations (say, local regions of size 2 × 2, as illustrated in Figure 9), which are then passed through max/min-pooling (downsample the local maximum or minimum) to retrieve features. Two linked neurons in consecutive CNN layers are communicating with scalars representing the local knowledge. Despite the success, one major drawback of such a process within CNNs lies in their deficiency to perceive important spatial hierarchical structures between different parts of a complex image [23], especially for our ride heatmap with many distant/close zones correlated with each other. Max/Min-pooling that requires fine-grained tuning might also lead to information loss and render the model less tractable [51].

Beyond scalar-based CNNs, capsules capture far more comprehensive spatial knowledge via the vector-based representation (due to multiple neuron groups or capsules). Consider a capsule in preceding PC (closer to raw inputs) is activated by a large ride volume in certain zones (say, Manhattan in NYC). Its output vector is then propagated to other capsules in succeeding OC (closer to evaluation criteria or physical meaning), forming some activated routes (linkages). Similar cases may happen to other capsules in PC due to other ride patterns discovered.

Training a STCapsNet can then be regarded as *extracting and refining the active routes* from a preceding capsule layer to a succeeding one. Specifically, an active route means a MOD pattern is "memorized," while a deactivated one represents that the unimportant connections can be "forgotten." As in Figure 9, instead of a single scalar in CNN, a high-dimensional output vector (with values and orientations) of the capsule represents not only the occurrence probability but also the graphical feature (say, geographic direction and co-occurrence of rides) of an instance, like a strong north-east zone-to-zone MOD flow in Manhattan in Figure 4.

To learn these high-dimensional vectors, a highly efficient scheme called *routing-by-agreement* is applied between blocks of PC and OC in Figure 8. Its basic idea is to keep extracting correlations between strongly connected traffic zones via *vector squashing*, while deemphasizing weak or noisy ones. By the dynamic "routing-by-agreement" and knowledge extraction, we can remove max/min-pooling, which is considered less capable of preserving useful information for accurate identification [51].

In what follows, we detail the capsule design and the formulation of the dynamic routing.

C. Capsule Design & Formulation: In terms of *layer-wise propagation* for STCapsNet, let β_{ij} be the logarithm prior probability of a zone-to-zone ride trend captured by a capsule i in PC ($i \in \{1, \ldots, N_2^c\}$) and a succeeding peer j in OC ($j \in \{1, \ldots, N_3^c\}$). Given the activated links between capsule i and all others in the succeeding OC, a softmax function first normalizes each coupling

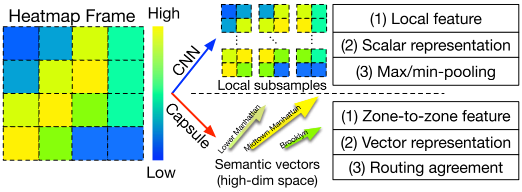

Fig. 9. Comparison of capsules and conventional CNNs.

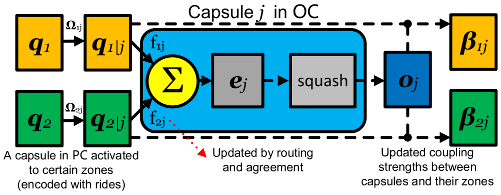
Capsule j in OC
q_1
Ω_{1j}
$q_{1|j}$
f_{1j}
q_2
Ω_{2j}
$q_{2|j}$
f_{2j}
Σ
e_j
squash
o_j
β_{1j}
β_{2j}
A capsule in PC activated to certain zones (encoded with rides)
Updated by routing and agreement
Updated coupling strengths between capsules and their zones

Fig. 10. Illustration of capsule learning.

coefficient f_{ij} between capsules i and j:

$$f_{ij} = \frac{\exp(\beta_{ij})}{\sum_l \exp(\beta_{il})}. \tag{2}$$

The probabilities β_{ij}'s (initialized uniformly before training) are used to preserve the precise location of useful heatmap patterns within a city, capturing the hierarchical structures of the demand correlation between the close and distant zones. The resultant f_{ij} represents the strength that capsule i should be linked with j, which is to be determined during capsule training.

Like the traditional neural network structure [16], we have the weight matrix across capsules i and j, denoted as Ω_{ij}, which is learned through conventional back-propagation algorithm [16]. For each succeeding capsule j, let $\mathbf{q}_i$ be the ride prediction vector returned from a preceding peer i in PC. We then find the product of weights and the prediction vectors as

$$\mathbf{q}_{j|i} = \Omega_{ij}\mathbf{q}_i, \tag{3}$$

which is further coupled with the coefficients derived in Equation (2), and adjusted into a weighted average form, i.e.,

$$\mathbf{e}_j = \sum_i f_{ij}\mathbf{q}_{j|i}. \tag{4}$$

Then, $\mathbf{e}_j$ serves as the vector input for the succeeding capsule j in OC.

To determine the f_{ij} during model training, in *routing-by-agreement* (dynamic routing) between capsules i and j, a squashing function $\mathtt{squash}(\cdot)$ is applied to discriminately adjust and shrink values of vectors $\mathbf{e}_j$ of different lengths. Specifically, short vectors that show less agreement get penalized and shrunk towards 0, while the long ones are converted to values slightly smaller than 1. Formally, the squash function is given as

$$\mathbf{o}_j = \mathtt{squash}(\mathbf{e}_j) = \frac{||\mathbf{e}_j||^2}{1 + ||\mathbf{e}_j||^2} \cdot \frac{\mathbf{e}_j}{||\mathbf{e}_j||}, \tag{5}$$

such that a slight increase in vector length of demand/supply patterns $\mathbf{e}_j$ quickly saturates the output towards 1 and easily gets captured by the entire capsule network. The vector product of $\mathbf{q}_{j|i}$ and $\mathbf{o}_j$ (resulted from $\mathbf{q}_{j|i}$'s) represents their mutual *agreement*, i.e., it is maximized only when their lengths/orientations match. Finally, the logarithm prior probabilities β_{ij}'s are updated in a coupled manner, with this product of prediction vector $\mathbf{q}_{j|i}$ and its adjustment $\mathbf{o}_j$:

$$\beta'_{ij} = \beta_{ij} + \mathbf{q}_{j|i} \cdot \mathbf{o}_j, \tag{6}$$

which is then returned to Equation (2) for f_{ij} if another iteration of dynamic routing is needed. So, a vector agreement, instead of pooling's down-sample, gets more features for better learning. Figure 10 further illustrates and summarizes the above procedures (illustrated with two preceding capsules; solid lines: propagation; dashed lines: routing).

Via iterative routings between PC and OC, routes of important high-dimensional vectors representing zone connections get refined and emphasized, and accurate heatmap predictions are reconstructed.

4.4 STCapsNet's Ride Prediction & Model Training

Finally, we present the entire ride prediction model of STCapsNet and describe how to train it.

Spatio-Temporal Frame Structures: Each high-dimensional frame (with its each grid min-max normalized [16]) describes the ride distributions in a certain period. To further capture sequential trip dynamics across time, we combine several consecutive frames (Figure 3) in a sliding window of size w (subject to the design of prediction accuracy and computational efficiency). The batched set of all w latest frames ($\{\boldsymbol{F}_{k-w}, \ldots, \boldsymbol{F}_{k-1}\}$) is then fed to the capsule network in Section 4.3 for model training, and the predict frame $\widehat{\boldsymbol{F}}_C$ is returned.

External Factors: Taking in lower-dimensional external data $\boldsymbol{\Psi}^k$ in Section 4.2, a fully connected (dense) neural network [16], as shown in Figure 3, learns the patterns of external factors correlated with training frames. Specifically, input values from different extracted features [16] are concatenated as a vector and fed to the neural network stacking two fully connected layers [16], returning ride prediction $\boldsymbol{F}_E$. The patterns with neuron activation are detected by the first layer, whose outputs are then reshaped back to frames by the second one.

Final Estimation Fusion: The outputs of both components, i.e., $\widehat{\boldsymbol{F}}_C$ and $\widehat{\boldsymbol{F}}_E$, are fused (merged) by a hyperbolic tangent function $\tanh(\cdot)$ [16, 66] that matches inputs towards values between −1 and 1:

$$\widehat{\boldsymbol{F}}^k = \tanh(\widehat{\boldsymbol{F}}_C + \widehat{\boldsymbol{F}}_E). \tag{7}$$

Then, predicted values in $\widehat{\boldsymbol{F}}^k$ undergo transformation back to physical space, i.e.,

$$\widehat{\mathbf{D}}^k = \frac{1}{2}(\widehat{\mathbf{D}}^k - (-1)) \cdot (D_{\max} - D_{\min}) + D_{\min}, \tag{8}$$

where $D_{\max}$ ($D_{\min}$) is the maximum (minimum) demand value of the data. $\widehat{\mathbf{A}}^k$ is scaled back similarly.

Model Training: STCapsNet's training process is to determine the model hyperparameters $\boldsymbol{\Phi}$ (including f_{ij}'s and neuron weights $\boldsymbol{\Omega}_{ij}$'s) that minimize the mean squared error between $\widehat{\boldsymbol{F}}^k$ and $\boldsymbol{F}^k$, i.e.,

$$\mathcal{L}(\boldsymbol{\Phi}) = \|\widehat{\boldsymbol{F}}^k - \boldsymbol{F}^k\|_2^2. \tag{9}$$

The iterative training process for STCapsNet involves the stochastic optimizer such as *Adam* (adaptive moment estimation) [29] and back-propagation mechanism [50], which are all available within existing deep-learning libraries [16]. Like its many deep-learning peers for big data analytics [66], STCapsNet may incur considerable training overhead (say, ~5 hours for NYC with our GPU), and its deployment, including model updates (depending on platforms and daily/weekly/monthly data variability), can be facilitated via GPU clusters [66] and online model learning [16], which are beyond our current scope.

5 SPATIO-TEMPORAL ADAPTIVE PRICING FOR MOD NETWORK BALANCING

5.1 Overview of Spatio-Temporal Adaptive Pricing

Given accurate spatio-temporal prediction of demands, we first overview the pricing optimization of CAPrice. Figure 11 overviews the adaptive pricing in balancing the market. A MOD platform consists of two spaces: *signal* (price and subsidy) and *market* (demand and supply). To cater for the two sides, before each target interval (say, of a few minutes), CAPrice takes the following steps for pricing optimization:

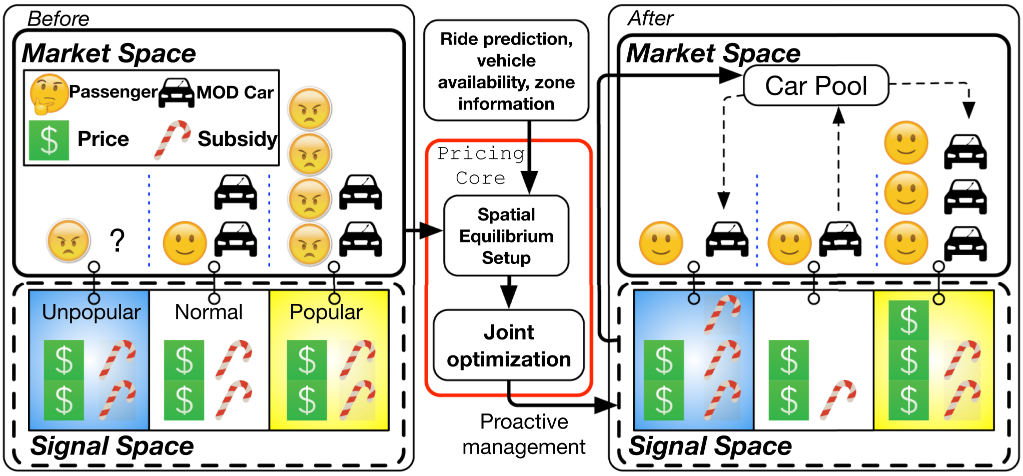

Fig. 11. Market and signal spaces: Basic flow of joint optimization and adaptive pricing.

(1) *Formalizing signal & market spaces*: Customers and drivers are the participants and bases of a MOD platform. Based on accurate STCapsNet and networked MOD participants, CAPrice interprets, characterizes, and forecasts their responsiveness to pricing signals and their market behaviors, including ride tendency, traffic flows, and distributions in each zone in Figure 11. These serve as inputs for the following steps (2) and (3).

(2) *Equilibrium setup*: Beyond conventional profit-hungry surge chasing, forming an equilibrium in a spatio-temporal manner helps smoothen and balance the supply-demand dynamics (such as unpopular, normal, and popular zones in Figure 11) of the MOD market.

(3) *Joint optimization & balancing*: Given inputs in step (1) and setup in step (2) (as the constraints in Equation (1)), CAPrice formulates a joint optimization framework and returns optimized and incentive-compatible pricing signals, which steer or incentivize zone flows to prevent over- or under-supply. With this further step beyond conventional practices [14, 35, 37, 38], the platform can make profit without compromising its sustainability. Outputs can be stored for computation in the next interval/period.

In what follows, we detail both of these spaces (Sections 5.2 and 5.3) and form an equilibrium state for the joint optimization (Section 5.4) and finally discuss its complexity analysis (Section 5.5).

5.2 Signal Space: Price & Subsidy

We first formalize the manageable price multiplier (Definition 6) upon the unchangeable standard rates. Specifically, we let δ_{ij} be the distance between zones Z_i and Z_j ($\delta_{ij} = \delta_{ji}$) [6], π be the fare per distance unit (i.e., fare rate), and r_i ($1 \leq r_i \leq r_{\max}$) be the multiplier related to the pick-up zone Z_i [6]. Meanwhile, the MOD platform may incur a minimum fare p_i^o (usually no less than the local subsidy s_i). Then, the price p_{ij} of each trip from Z_i to Z_j is given by

$$p_{ij} = (p_i^o + \pi\delta_{ij})r_i, \quad 1 \leq r_i \leq r_{\max}, \tag{10}$$

which serves as one steering basis for a profit-driven driver on picking up a passenger or incentive of (re)locating to somewhere.

However, each driver is considered to be rational [13] and have an outside option λ [11, 15], i.e., an amount of expected earning away from the market (say, a basic salary for other lower-skilled jobs or an enforced minimum wage). Intuitively, a driver enters the MOD market if her/his worst earning is at least λ and exits if the expected earning is even lower than λ, matching her/his incentive-compatible expectations. To maintain the drivers' incentives (especially when unmatched) in serving less popular zones, the platform further provides spatially varied subsidies s_i. This way, both exits and entries of the drivers are steered by the difference of platform subsidies (representing the "worst-case earning"), expected earning (detailed in Section 5.3), and outside option λ. Both price multipliers and subsidies are the *optimization variables* to be determined by `CAPrice`.

5.3 Market Space & Spatial Equilibrium

We formalize the market space in the following two perspectives:

(a) **Ride Matching & Earnings**: Given uncertainty of being matched with a passenger, each driver cares how much s/he expects to earn. Let P_i $(0 \le P_i \le 1)$ be the probability that one of a total of m_i cars in Z_i will pick up a passenger. The matched rides D_i^* from Z_i and the probability P_i are, respectively:

$$D_i^* = \min\{m_i, D_i\}, \quad P_i = \min\{D_i/m_i, 1\}. \tag{11}$$

Let a_{ij} be the probability that a matched passenger in Z_i is expected to move to Z_j, and $\sum_{j=1}^{G} a_{ij} = 1$ for each i. This probability can be easily estimated from predicted arrivals $\widehat{\mathbf{A}}$ (Section 4.4).

Consequent earning considers two cases. If matched, then a driver earns part (denoted as $(1-\rho) \in [0,1]$)[6] of the fare paid by the rider [6]. Otherwise, the earning of being unmatched in Z_i can be considered as one from the next "best" zone (based on last period's V_i''s) to migrate to. Based on the above and Section 5.2, the expected earning of a driver on serving Z_i, i.e., her/his incentive measure V_i for (re)locating to i, is:

$$V_i = P_i(1-\rho)\left(p_i^o + \sum_{j=1}^{G} a_{ij}\pi\delta_{ij}\right)r_i + (1-P_i)U_i', \quad U_i' = \max\{V_1', \ldots, V_G'\}, \quad \forall i \in \{1, \ldots, G\}, \tag{12}$$

where U_i' is her/his valuation for the next best relocation.

(b) **Flows & Distributions**: Given signals in Section 5.2 and valuations on matchings/earnings, incentivized drivers form the traffic flows and zone distributions to earn more.

In practice, some drivers become inactive for various non-monetary reasons [32], especially after a long trip.[7] So, our formulation considers that after traveling from Z_i to Z_j, a driver may likely exit the platform with a probability $1 - \kappa_{ij}$ $(0 \le \kappa_{ij} \le 1)$, which is considered to enlarge *w.r.t.* distance δ_{ij} between zones. Inclusion of this also fulfills drivers' general tendency for neighborhood (close zones) migration in practice [13].

Besides matched rides, based on the global supply demand trend, the platform may steer unmatched drivers to (re)locate to other zones that are short of supply via subsidy signals. Basically, the redistribution of drivers to Z_i consists of three parts: *(i)* relocated drivers from other zones Z_j's, each flow of which is denoted as z_{ji}, and *(ii)* newly enticed drivers, denoted as ϵ_i, due to the likely earning. Then, including *(iii)* the matched flows

[6] For example, it has been reported in 2017 that Uber and Lyft (Didi) usually take a commission rate of $\rho = 25\%$ (20%) of the fares in the US (China) [2].

[7] https://www.uber.com/drive/new-york/resources/driving-hour-limits/.

D_j^* (vacant after dispatch), the mass m_i of available cars in Z_i becomes:

$$m_i = \sum_{j=1}^{G} \kappa_{ji}(a_{ji}D_i^* + z_{ji}) + \epsilon_i. \tag{13}$$

After formalizing (a) and (b) above for the market space, we also consider characterizing and setting up the target-balanced states of the market for price optimization.

Equilibrium: Recall that drivers may over-react (bunched-up) or under-react (discouraged) to the greedily surged signals, as the former pricing did not foresee and optimize the consequent driver flows/distributions. Therefore, we take into account the *equilibrium* [31, 33] of vehicle distribution across the MOD networks, i.e., supply-demand matching of each zone, and find the incentive-compatible prices and subsidies such that drivers are better motivated to (re)locate themselves. In other words, the driver flows resulted from the pricing signals are steered to a zone-to-zone equilibrium, i.e.,

Definition 8. Spatial Equilibrium (SE): An *SE* for a MOD platform given $\{p_i, s_i\}_{i=1}^G$ is defined as a time-invariant distribution of drivers/vehicles and passengers at each of the G zones in that interval. Setting up an *SE* fulfills the following three criteria:

(1) *Maximizing expected earnings*: The platform provides each driver with *incentive-compatible* offers [20, 60]. If the highest expected earning, as s/he chases a best chance, and the subsidy in a zone matches at least a driver's outside option:

$$U_i = \max\{V_i\}_{i=1}^G = \lambda \le s_i, \quad \forall i \in \{1, \ldots, G\}, \tag{14}$$

then all the drivers are expected to enter or (re)locate to maximize their expected earnings [18, 33]. Otherwise, drivers cannot get enticed or may even exit the market (Section 5.2).

(2) *Balanced driver distribution and passenger flow*: The car distribution remains overall balanced across zones, or the outgoing number matches the incoming one, i.e., $\sum_{j\neq i} m_{ij} = \sum_{i\neq j} m_{ji} \ge 0$. Considering the *three* different driver flows and the potential exits in Equation (13), we formulate the balanced outgoing/incoming flows at each Z_i as

$$\sum_{j\neq i}^{G} m_{ij} = \sum_{i\neq j}^{G} \kappa_{ji} m_{ji} + \epsilon_i. \tag{15}$$

Equations (11), (13), and (15) are fused into the general *flow balance*:

$$D_i^* + \sum_{j=1}^{G} z_{ij} = \sum_{j=1}^{G} \kappa_{ji}(a_{ji}D_j^* + z_{ji}) + \epsilon_i. \tag{16}$$

(3) *Total number constraints*: In practice, the number of drivers and passengers involved should not exceed their respective total numbers in the city (or subject to the city regulation[8]), i.e.,

$$\sum_{i=1}^{G} m_i \le M, \quad \sum_{i=1}^{G} D_i \le D. \tag{17}$$

Existence of *SE* results from finiteness of both search scopes and potential responses/actions [40]. Given the MOD market settings (finite numbers of zones, drivers, and passengers, plus their incentive-compatible responses), the *SE* exists and is unique [31] (formal proofs are omitted due to space limit).

[8] https://money.cnn.com/2018/08/08/technology/uber-lyft-new-york-legislation/index.html.

In summary, the three criteria in SE, along with the market space formulation in (a) and (b), serve as *constraints* of the optimization framework. (Re)location flows of ϵ_i's and z_{ij}'s, formulated as the optimization variables, are also returned for the platform's management reference.

Projected Departures Due to New Price Multipliers: Recall that in Equation (1) the updated pricing may make the actually fulfilled demands different from the predicted [11]. Thus, following the industrial practices (of Uber and Lyft) [35, 45], we need to adjust/project the resultant $\widehat{D}_i$'s in our optimization based on the difference of the historical and newly estimated multipliers, which are denoted as r_i''s and $\widehat{r}_i$'s, respectively.

Specifically, let θ_i be the latent demands (often inaccessible) independent of the updated/estimated price multiplier $\widehat{r}_i$. Let $F(r_i)$ be the empirical cumulative distribution of passengers' unwillingness to afford a multiplier r_i [11, 17, 18]. Then, $1 - F(r_i)$ represents the likelihood of accepting r_i. $F(r_i)$ increases as r_i grows, meaning that fewer ride requesters would accept the multiplier [11, 17]. The projected demand from θ_i at Z_i due to $\widehat{r}_i$ is then formulated as a generic form:

$$D_i = (1 - F(\widehat{r}_i))\theta_i, \quad 0 \leq F(\widehat{r}_i) \leq 1, \tag{18}$$

where $F(r_i)$ can be parameterized [17] and obtained via surveys [13] (detailed in Section 6.1).

Based on the temporal continuity in the high-frequency MOD flows as considered in traffic monitoring, the predicted fulfilled demand $\widehat{D}_i$ at period k is considered correlated with and thus is estimated from the historical record's pricing multiplier. Specifically, `CAPrice` inherits θ_i, r_i' and $1 - F(r_i')$ from the last period $k - 1$, approximates $\theta_i = \widehat{D}_i/(1 - F(r_i'))$, and returns the adjusted demand D_i at k. Based on Equation (18), the projection function $\Gamma(\cdot)$ from $\widehat{D}_i$ due to multiplier $\widehat{r}_i$ at the target period k is then given by

$$D_i = \Gamma(\widehat{r}_i, r_i', \widehat{D}_i) = (1 - F(\widehat{r}_i))\theta_i = \frac{1 - F(\widehat{r}_i)}{1 - F(r_i')}\widehat{D}_i. \tag{19}$$

In other words, the projection D_i and prediction $\widehat{D}_i$ match if the price multiplier remains unchanged ($\widehat{r}_i = r_i'$), and do not match otherwise ($D_i < \widehat{D}_i$ if $\widehat{r}_i > r_i'$). Note that more sophisticated projection functions and pricing [10, 11, 28] other than Equations (18) and (19) can also be applied in our generic framework. For example, Jorge et al. [28] considered the linear increase of demand with the decrease of price [36].

5.4 Joint Optimization Formulation

We design the following objective function and optimization formulation for `CAPrice`:

Revenue & Subsidy: The total revenues of a platform are proportional to drivers' earnings due to predominantly used fixed commission rate (ρ in Equation (12)) in industry [2]. Then, the revenue objective $\texttt{Obj}_1$ of a platform (with a positive commission rate ρ) is given by the aggregated fees upon the fulfilled demands, i.e.,

$$\texttt{Obj}_1 : \rho \sum_{i=1}^{G} \sum_{j=1}^{G} a_{ij} p_{ij} D_i^*. \tag{20}$$

However, s_i mainly caters for incentive-compatibility in Criterion (1) of Definition 8, especially the drivers' relocation z_{ji}'s and entry ϵ_i's (Section 5.3). Let $T_{ji} \geq 0$ be the estimated travel time (normalized for ease of formulation) when relocating from zone j to i, meaning compensation should be adjusted proportionally. Since (re)location consumes time and gasoline while incurring extra waiting time to the ride-requesters, we introduce T_{ji} as a weight upon the redistribution flow z_{ji} such that long-term but unprofitable driving can be jointly mitigated. Then, the total platform cost (subsidies) $\texttt{Obj}_2$ is given by

$$\texttt{Obj}_2 : \sum_{i=1}^{G} s_i \left(\epsilon_i + \sum_{j=1}^{G} T_{ji} z_{ji} \right). \tag{21}$$

In this work, we estimate the travel time between zones based on a random forest tree trained upon the historical data of rides (other models may also apply [55, 61]). Despite its prototyping with Equations (20) and (21), `CAPrice` is general enough to be extended to other customized objectives [11, 42] by the MOD platforms.

Final Formulation: Given predicted MOD rides, i.e., departures $\widehat{\mathbf{D}}$'s and arrivals $\widehat{\mathbf{A}}$'s, `CAPrice` *jointly* determines the price multipliers and subsidies ($\{\widehat{r}_i, \widehat{s}_i\}$'s) spatio-temporally and (re)distributions of drivers ($\{\widehat{\epsilon}_i, \{\widehat{z}_{ij}\}_{j=1}^{G}\}_{i=1}^{G}$) with both *maximum platform profit* and *spatial equilibrium*. Specifically, we augment and formalize Equation (1) into

$$\begin{aligned}
\underset{\{\widehat{r}_i, \widehat{s}_i, \widehat{\epsilon}_i, \{\widehat{z}_{ij}\}\}_{i=1}^{G}}{\arg\max} \quad & \rho \sum_{i=1}^{G} \sum_{j=1}^{G} a_{ij} p_{ij} D_i^* - \sum_{i=1}^{G} \widehat{s}_i \left(\widehat{\epsilon}_i + \sum_{j=1}^{G} T_{ji} \widehat{z}_{ji} \right), && \text{(Platform profitability)} \quad (22) \\
\text{subject to} \quad & p_{ij} = (p_i^o + \pi \delta_{ij}) \widehat{r}_i, \quad 1 \le \widehat{r}_i \le r_{\max}, \quad D_i = \Gamma(\widehat{r}_i, r_i', \widehat{D}_i), && \text{(Prices \& departures)} \\
& U_i = \lambda \le \widehat{s}_i, \quad \widehat{\epsilon}_i, \widehat{z}_{ij}, \widehat{s}_i \ge 0, \quad \widehat{\epsilon}_i \le \epsilon_{\max}, \quad \widehat{z}_{ij} \le z_{\max}, \quad \widehat{s}_i \le s_{\max}, && \text{(Incentive-compatibility)} \\
& D_i^* = \min\{m_i, D_i\}, \quad D_i^* + \sum_{j=1}^{G} \widehat{z}_{ij} = \sum_{j=1}^{G} \kappa_{ji} (a_{ji} D_j^* + \widehat{z}_{ji}) + \widehat{\epsilon}_i, \ \forall i, && \text{(Balanced flows)} \\
& \sum_{i=1}^{G} m_i \le M, \quad \sum_{i=1}^{G} D_i \le D. && \text{(Total constraint)}
\end{aligned}$$

Note that the incentive-compatible setting in *SE* extrinsically maintains the willingness of drivers' (re)distribution in terms of price/subsidy signals [60]. Considering their exit and continuation (Section 5.3) as well as their fuel/labor cost, drivers tend to (re)locate to nearby zones, hence completing the transition in a reasonably short time [11].

When deploying the above framework, one may further discretize the fare multipliers into Q predefined values (say, ranging from 1.0 to $r_{\max} = 3.75$ evenly spaced with 0.25) for ease of UI display and calculation. Thus, in Equation (22), the continuous variable space is discretized (Section 6.1). The multiplier cap $r_{\max}$ can be customized by the service providers.[9] Similarly, the MOD platform can also customize other continuous variables (including car flows) based on practical settings and market survey [13, 14]. In our prototype, after the optimizing period k, the estimated signals ($\{\widehat{r}_i, \widehat{s}_i\}$'s) and the predicted market states, plus the fulfilled rides, are stored for use in the next period $(k + 1)$, which is similar to a Markov structure [40, 59] and common industrial practices [35]. The resultant demand flows at period k can be stored for `STCapsNet`'s training in the next step. One may periodically initiate or reset spatial prices (say, uniform or proportional to regional price analysis [13]).

5.5 Complexity

Given G zones, their trips, and Q discretized multipliers, there are a total of $O(GQ + G + G^2)$ variables for the optimization formulation. The problem in Equation (22) can be solved using efficient optimization solvers (such as GLPK and JOptimizer) [12], taking asymptotically $O((GQ + G^2)^2)$ [12]. We also observe that at each heatmap frame, only prices/subsidies belonging to the highly demanded subset of the grids [14, 42] need to be adjusted (often ≤15%), making $O(G)$

[9] http://fortune.com/2015/01/26/uber-caps-surge-pricing-during-blizzard-but-people-still-complain/.

generally small. Thus, its deployment price updates can be made efficiently and in real time (say, average 5.13s per frame in the NYC test with our PC configuration in Section 6.1).

STCapsNet's online ride-estimation time is almost negligible (say, 0.12s on average for each frame in NYC). The price-update frequency can be in the order of minutes (every 5 to 10mins for platforms like Uber [14]) or on an hourly basis, up to the platform's customization and the local market's nature. Further computation reduction in pricing can be done by region partitioning [42] and parallel programming, which is omitted due to space limit.

6 EXPERIMENTAL EVALUATION

We first describe the experimental evaluation setup in Section 6.1 and then present the evaluation results in Section 6.2.

6.1 Evaluation Setup

Data Pre-processing: We evaluate CAPrice and related schemes with the datasets presented in Section 4.1. All experimental evaluations are done using a PC with Intel i7-8700K, 32GB RAM, an Nvidia/EVGA GeForce GTX 1080Ti and Windows 10. Each frame (empirically $W = H = 32$, and $G = 1{,}024$) aggregates the rides within 0.5h due to data availability. The form of zone and time discretization follows the conventional practice and state-of-the-arts for ease of prototyping and comparison [66], while further evaluation on their variation is omitted due to space limit and left as future work. Our test of ride prediction algorithms is done upon the frames of the last four weeks (total 1,344 test frames from 28 days) for each dataset (or last two-week frames of our one-month Didi data). The rest of the datasets are used for model training. All frames are min-max normalized [16] to a range of $[-1, 1]$ as in Section 4.4 to fit in the tanh function. External factors in Section 4.2 are parsed and aggregated in the same frequency to rides. We similarly scale the non-categorical meteorological data (say, temperature/wind speed) and traffic conditions (travel speed and time) into $[0, 1]$ for each dimension. For other categorical data (say, date/event, holiday, weather conditions, and sky visibility), we apply the one-hot coding upon them to form binary vectors. For example, given candidate weather conditions of [snow, rain, sunny], a time period with only rain recorded is tagged with vector $[0, 1, 0]$.

Pricing evaluation is done upon the same test frames given their ride predictions of each dataset. To characterize the MOD flows, we find the set of probabilities $\{a_{ij}\}$ in Equation (16) based on the overall predicted incoming (arrival) flows A_j's and generalize it for all passengers within all D_i's from all zones. As the Uber ride data [5] in our hands only provide pick-up locations, from the local taxi and traffic flow data [4] ($\{a_{ij}\}$'s) at the same time, we extract the approximate arrivals for the evaluation of demands and supplies [49, 56]. To estimate the time of relocation T_{ji} in Equation (22) (related to passengers' waiting time), we train a random forest regression model (20 trees as the ensemble estimators; 5×10^{-5} as the minimum fraction of samples required to split an internal node) upon the one-month rides before the test data. The mean time estimation error is 2.7mins upon 50,000 validation rides.

Comparison Schemes: In ride prediction, we compare CAPrice with the following traditional/advanced algorithms:

- *HA & S-HA*: predicts the rides based on the average of historical records of the same periods. For example, we average all data during 8:00–8:30 of all recorded Mondays to predict that of 8:00–8:30 of a specific Monday. (Seasonal) S-HA takes the same periods but of the same seasons [26].
- *ARIMA*: predicts rides based on the traditional Auto Regressive Integrated Moving Average (ARIMA) model [26]. In our setting, ARIMA empirically predicts the rides based on a sliding window of recent five days (240 ride frames).

- *LSTM*: predicts rides with long-short-term-memory-based (LSTM-based) [24] neural network regression learned from k most recent ride frames [26]. We empirically set $k = 72$.
- *ST-CNN*: predicts rides based on CNN [30, 39] and residual networks (ResNet) [66] in a spatio-temporal LSTM structure [24]. Note that we have also provided the external factors presented in Section 4.2 with ST-CNN to compare its core with that of `STCapsNet`. An optimal window of the latest 21 frames is applied for ST-CNN.

For pricing performance, we compare `CAPrice` with the following schemes (all given `STCapsNet`'s predictions; implementation details can be referred to their works):

- *Spatio-temporal Surge Pricing* (ST-SP): is a traditional dynamic pricing scheme adopted by existing ride-sharing service providers (detailed in References [14, 35, 37, 38]), greedily adjusting the zone prices spatio-temporally to entice drivers. Despite the price variation due to spatial and temporal demand changes, it does not take into account price optimization based on zone-to-zone equilibrium and flow balance.
- *Temporal Surge Pricing* (T-SP): only surges temporally [4, 10, 48] based on overall demand-supply patterns without considering the spatial diversities and equilibrium effects.
- *Static/Fixed Pricing* (FP): is a basic and uniform pricing/subsidizing scheme [3] without considering the spatio-temporal effect of the MOD market.

Parameter Setups: Unless otherwise stated, the following default parameters are used for `CAPrice`: In Figure 8, ($N_1^f, N_2^d, N_2^c, N_3^d, N_3^c$)=(128, 8, 8, 64, 16). Each filter in `Conv2D_1` and `Conv2D_2` has 3×3 kernels [51]; while for `PC`, we use the 9×9 kernel. The size of sliding window (w) is set to 12. The batch size is 64, learning rate is 2×10^{-3}, and the max number of epochs is 200 (with early-stop rounds [16] at 20). The number of routing iterations in `STCapsNet` is 3 (Section 4.3).

For all pricing schemes, based on empirical studies and user surveys [11, 14, 31, 32], we set r_i in the range of [1.0, 3.75] (evenly discretized with 0.25). As in many state-of-the-art incentive and analytical pricing designs [11, 17, 18, 38] in characterizing driver responses, $F(\widehat{r}_i) = \alpha \widehat{r}_i$ ($\alpha > 0$) is considered in our prototype evaluation (Equation (18)), and we empirically set $\alpha = 0.2$. We consider $\kappa_{ij} = \beta^{\delta_{ij}}$ in Equation (13), where the continuation rate β can be determined via the driver surveys and statistics [31, 32], and we set $\beta = 0.8$. $s_{\max}$ is set to 20% of the average ride fare based on the ride-sharing policy [6] from the Uber platforms. Initial $\{p_i, s_i\}$'s, rate π per distance unit,[10] commission rate ρ (Equation (20) [2]), outside option λ, and car/passenger availability are also set based on the local MOD markets [4, 6].

Comparison Metrics: The following evaluation metrics are used:

- *Root mean square error* (RMSE) & *Mean average percentage error* (MAPE): which are respectively given by

$$\text{RMSE} = \sqrt{\frac{1}{\mathcal{M}} \sum_{i=1}^{M} \left(\boldsymbol{F}_i - \widehat{\boldsymbol{F}}_i \right)^2}, \quad \text{MAPE} = \frac{1}{\mathcal{M}} \sum_{i=1}^{M} \frac{|\boldsymbol{F}_i - \widehat{\boldsymbol{F}}_i|}{\boldsymbol{F}_i}, \tag{23}$$

  where $\mathcal{M}$, $\boldsymbol{F}_i$ and $\widehat{\boldsymbol{F}}_i$ are the total number of zones (consisting of D_i and A_i) in all test frames, ground-truth and estimation, respectively. For the accuracies of ride predictions, RMSE shows the absolute errors while MAPE focuses on relative proportion *w.r.t.* scale of traffic flows.
- *Operation revenue & Profit*: We compare the profitability of different pricing schemes with respect to the average revenues and profits of all temporal frames. For a clearer comparison,

[10] https://www.numbeo.com/taxi-fare/in/Beijing.

Table 2. Ride Prediction Accuracy with Four Different Datasets

Dataset	Uber, NYC		Taxi, NYC		Taxi, Beijing		Didi, Chengdu	
Method	RMSE	MAPE	RMSE	MAPE	RMSE	MAPE	RMSE	MAPE
HA	15.96	0.245	38.63	0.323	59.48	0.504	37.83	0.539
SHA	12.06	0.192	37.46	0.315	53.79	0.446	37.61	0.536
ARIMA	10.11	0.124	28.15	0.209	46.40	0.384	16.14	0.363
LSTM	9.73	0.093	18.42	0.127	28.55	0.230	11.97	0.252
ST-CNN	9.04	0.079	14.57	0.101	17.71	0.158	11.36	0.237
Proposed	**4.57**	**0.036**	**9.60**	**0.065**	**14.69**	**0.118**	**5.93**	**0.177**

the revenue refers to the income from trips in all zones with estimated fare multipliers $r_i > 1$ ("surge") or subsidies $s_i > 0$ by all pricing schemes, and the profit means the net income after subtracting costs due to applied subsidies from the zones. Others without adapted prices or subsidies are not included due to the same earnings there. This way, we can focus on the effect of adaptive pricing.

- *Market clearance ratio* (MCR) & *Estimated wait time* (EWT): We compare the sustainability of the pricing policies based on how the MOD market gets cleared, i.e., passengers' demands get matched (responded) by the driver supplies. We find the mean MCR of all test frames. We also find the riders' average estimated wait time caused by mismatches and the subsequent vehicle redistributions from neighboring regions. Note that if an event of relocation is estimated to take too long (say, longer than an interval in our setting), for practical evaluation, we consider the ride is canceled and this demand cannot get cleared either.

6.2 Evaluation Results

A. Ride Prediction: We first overview the performance upon the four datasets, followed by a sensitivity analysis.

(1) *Prediction Accuracy*: Table 2 shows the accuracies of different algorithms upon the four datasets. HA and SHA only consider the temporal trend without the spatial trip dependency of the zones, thus experiencing much higher errors in the complex MOD networks. ARIMA and LSTM regress/model the spatio-temporal trends more vigorously. However, their lack of comprehensive learning structure degrades accuracy in handling complex city networks. ST-CNN formalizes the ride distributions via convolutional neurons, but its CNN structure focuses on local features, not global zone-to-zone trends (as in Figure 9). So, it cannot comprehensively capture connections between different distant zones.

In terms of RMSE and MAPE, `CAPrice` is shown in Table 2 to outperform other state-of-the-art ride prediction schemes often by 20% or more. Unlike the above schemes, `STCapsNet` perceives the rides via structured vector-based capsules. Different capsules capture ride connections of close/distant zones, thus gaining more accurate information than others. Given the spacious urban map (1,024 grids) and large MOD traffic volumes (Figure 5), its accuracy will likely make a significant improvement of the urban MOD economy. Figure 12 further visualizes a heatmap of demand prediction (Taxi, NYC; 00:00–12:00, 2016-05-31), which, subject to some noise, closely resembles the ground-truth in Figure 4.

We also observe that all schemes in NYC Uber and Taxi evaluations achieve better accuracy than in the Beijing Taxi, because the rides in NYC have overall more regular and deterministic trip directions (say, around Manhattan and JFK airport) than in Beijing. Beijing has more complex ring road structures and hence more complicated taxi drivers' routes (as

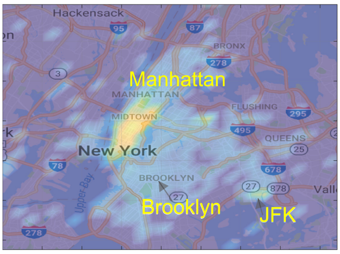

Fig. 12. Visualization of NYC ride demand prediction.

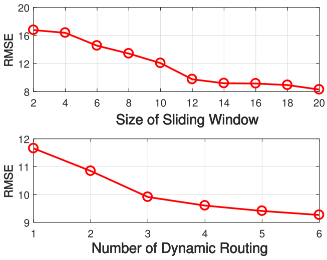

Fig. 13. RMSE vs. frame numbers and routing numbers.

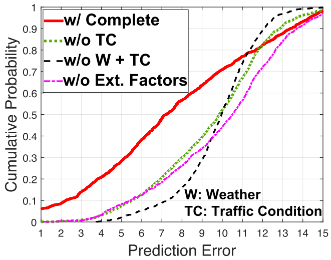

Fig. 14. Error CDFs with or without different external factors.

discussed in References [63, 68]) than NYC. Thus, more randomness and noise is observed in the trips in Beijing. Compared to Taxis in two other metropolitan cities, the test case of Didi in Chengdu is overall easier for all schemes. Nevertheless, due to much smaller training sets, higher prediction errors are still observed than from Uber at NYC. A further (future) enhancement may include integration of detailed city road networks [41, 55].

(2) *Sensitivity Analysis*: We also analyze `CAPrice`'s sensitivity to important parameters/external factors with NYC Taxi data as an illustration, where the validation data (10% left out of the training frames) is different from the above test data.

- *Number of consecutive frames & dynamic routings*: As shown in Figure 13, the more consecutive frames in a window (Section 4.4) and more iterations of routings (Section 4.3), the more accurate `CAPrice` becomes due to more captured ride knowledge. However, as we increase the frames and routings further, the improvement slows down. Inclusion of many frames may come with some noisy measurements over time, thus degrading its learning quality. Further learnable features from too many routing-by-agreements may also vanish.

  Meanwhile, increasing the sliding window and the routing time may also create a more complicated model and cause longer training time. Thus, for the applicability of the model, we choose knee points [12] (where the curves "turn" in Figure 13) of window size $w = 12$ (still much smaller than ST-CNN's 21 frames in Section 6.1) and three routings in our prototype studies.
- *External factors*: Figure 14 shows the error (i.e., the absolute difference between estimation and ground-truth at each Z_i) CDFs of four cases—`STCapsNet` with complete external factors, that without only traffic condition, that without both meteorological data and traffic conditions, and pure capsule network (without any external factors in Section 4.2). We can see with assistance from more factors that `STCapsNet` achieves higher accuracy than others. In what follows, we study `CAPrice` with complete external factors.

B. Adaptive Pricing: Given the ride prediction, we further evaluate the performance of `CAPrice`'s pricing in terms of platform profitability and sustainability.

(1) *Operation Revenue, Profit, Market Clearance Ratio, & Estimated Time of Wait*: Given the ride predictions by `STCapsNet`, Table 3 shows the overall pricing performance in the four different cases. Specifically, we list the means of revenue, profit (both in their common logarithms; from the zones with adapted prices), MCR, and EWT with respect to each ride heatmap frame.

`CAPrice` is shown to outperform the other conventional pricing models often by at least 30% more profits and revenues. FP fails to capture the spatio-temporal ride dynamics, thus resulting in significant profit loss. Despite the consideration of temporal demand-supply

Table 3. Market Analysis with Four Datasets (Means of All Test Frames)

Dataset	Uber, NYC				Taxi, NYC				Taxi, Beijing				Didi, Chengdu			
Method	Revenue ($\log_{10}(\cdot)$)	Profit ($\log_{10}(\cdot)$)	MCR	EWT (min)	Revenue ($\log_{10}(\cdot)$)	Profit ($\log_{10}(\cdot)$)	MCR	EWT (min)	Revenue ($\log_{10}(\cdot)$)	Profit ($\log_{10}(\cdot)$)	MCR	EWT (min)	Revenue ($\log_{10}(\cdot)$)	Profit ($\log_{10}(\cdot)$)	MCR	EWT (min)
FP	3.376	2.955	0.744	14.2	4.555	4.032	0.753	16.1	3.873	3.639	0.858	17.8	3.649	3.647	0.708	10.1
T-SP	3.326	3.007	0.757	8.5	4.638	4.413	0.770	9.7	3.875	3.715	0.841	10.9	3.739	3.730	0.783	7.3
ST-SP	3.481	3.217	0.799	5.2	4.737	4.566	0.820	5.9	3.941	3.782	0.899	6.7	3.869	3.721	0.822	5.4
CAPrice	**3.803**	**3.707**	**0.935**	**3.2**	**4.943**	**4.811**	**0.940**	**3.7**	**4.161**	**4.060**	**0.951**	**4.2**	**3.958**	**3.899**	**0.882**	**4.1**

fluctuations, the absence of spatial differentiation still degrades T-SP. Beyond FP and T-SP, ST-SP perceives more about spatio-temporal flow dynamics. However, it does not jointly consider the prices, subsidies, and driver redistributions based on forecasts and equilibrium. Thus, misdistribution of drivers often happens and the platform's profitability is held down. Meanwhile, the market cannot get cleared in a timely manner, degrading customer experience and the sustainability of platform service.

In contrast, `CAPrice` formulates the spatial equilibrium into the joint optimization, balancing dynamic flows via a spatio-temporal adaptive pricing policy. Vehicles can follow incentive-compatible distribution guidelines while bunch-ups are mitigated, leading to more adaptive and smoother ride-flow distributions. Thus, it enhances the profitability and sustainability of the MOD platforms as well as the predictability of rides. Due mainly to the size difference in reported vehicle volumes, the revenues/profits by Yellow Taxi are found to be much larger than NYC Uber peers and Beijing Taxi/Chengdu Didi markets. The difference between each pair of revenue and profit reflects the total subsidies paid to the drivers. Thanks to the joint flow-balancing and optimization, `CAPrice` achieves higher MCRs at lower EWTs and subsidies.

(2) *Further Market Analysis*: Figure 15 further shows a snapshot of spatial price multipliers and subsides at 09:00, May 4, 2016 (NYC Taxi); that most of the surging prices happen in the popular midtown Manhattan due to large volume of traffic demands there. Meanwhile, we can see higher subsidies from the Upper East Side, Jackson Heights and peripheral areas around Manhattan, to compensate driver redistribution.

Figure 16 further illustrates the dynamic flows of vacant cars. We show the directions of in- (blue sectors) and out-flows (yellow sectors) of vacant cars due to incentivization of $\{r_i, s_i\}$'s. The larger radius the sector, the more incoming or outgoing vehicles there. Corresponding to Figures 4 and 15, unmatched drivers are steered towards locations such as midtown Manhattan (due to prices) and Jackson Heights (due to subsidies) to serve the supply-demand imbalance there. Figures 15 and 16 also show insights for setting higher price multipliers (subsidies) attractive (unattractive) zones to steer drivers. In particular, zones of unbalanced supply-demand due to their far more attractive neighbors (such as Manhattan peripherals around the popular Midtown in Figure 15) should be subsidized more to entice new and redistributed driver flows.

Figure 17 shows (a) temporal profits and (b) temporal MCRs of `CAPrice` of a day, and (c) MCR CDFs with all schemes (Uber, NYC). From (a) and (b), we can see that `CAPrice` maintains overall high MCRs over time while adapting to high demand during morning/afternoon (08:00/18:00) rush hours. Thus, high profits and MCRs can be observed then. Figure 17(c) summarizes the MCR CDFs, showing more than 30% improvement in clearing the market.

(3) *Effect of Ride Prediction over Pricing*: Testing `CAPrice` on Uber data, Figure 18 shows the importance of ride-prediction accuracy over market profitability/sustainability of adaptive pricing. Inaccurate ride prediction in an urban network markedly exacerbates misdistribution of drivers towards customers, resulting in revenue decreases, more steering subsidy costs, and MCR degradation. Note that the CDFs of profits are from their common logarithms. With more accurate forecasts of `STCapsNet`, `CAPrice` gains at least 35% more profits and revenues and 30% higher MCR than with ST-CNN.

7 DISCUSSION ON DEPLOYMENT

Task assignment for humans & machines: Task assignment and pick-up acceptance are also critical for MOD platform operation in redistributing drivers. While incentive-compatible

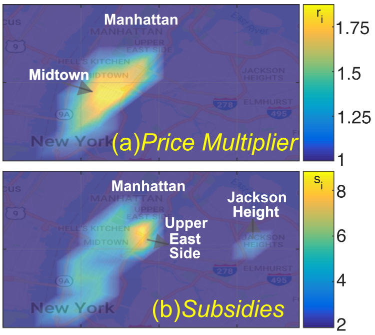

Fig. 15. Heatmaps of (a) r_i and (b) s_i (NYC, 09:00).

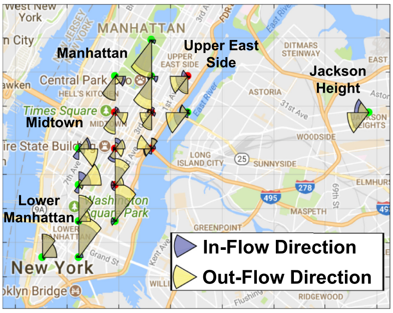

Fig. 16. Relocation flow of vacant cars (NYC, 9:00).

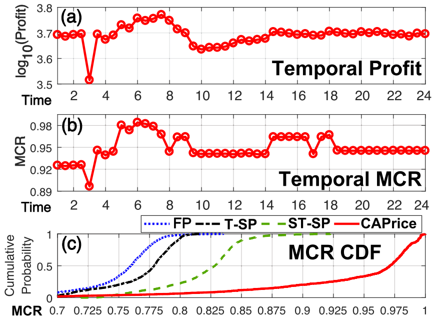

Fig. 17. Market dynamics and summary (Tue. 2015-06-30).

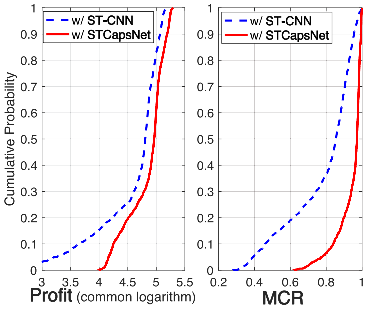

Fig. 18. Prediction effects: Uber profits and MCRs.

monetary subsidies/compensations are often used for this purpose, many social [17, 25], psychological, and human-computer interaction designs, orthogonal to pricing, may increase pick-up/response rates further in unpopular times and/or city zones. Explicit or implicit rules by the platform (bonus or penalty) may also regulate the (re)distribution of drivers [13]. Although it is very challenging to find a complete design, our accurate ride prediction and joint pricing optimization framework is generic enough to be extended to accommodate these factors.

Market of monopoly or duopoly: We focused on a market of monopoly (oligopoly) for ease of prototyping, but CAPrice can be adapted and extended to a more complex one with multiple competing MOD platforms (say, Uber, Lyft, and Didi). A MOD driver survey recently conducted in 2017 [13] still shows that 75% of the drivers indicate they are primarily working for Uber, although a majority of them could be active drivers for other services. Unlike North America, more than 80% of China's ride-sharing market (2018) is held by Didi. In the future, we would like to extend CAPrice to highly competitive market settings [9, 44]. How to properly regulate, instead of letting an unplanned market up to the "invisible hand" [9], is also worth further exploration by city planners.

8 CONCLUSION & ACKNOWLEDGMENT

We proposed CAPrice, a novel adaptive pricing system for MOD networks. CAPrice first predicts MOD trips (departures and arrivals) via a spatio-temporal deep capsule learning network, which comprehensively constructs vector-based zone-to-zone ride relations for accurate ride prediction. It then formulates a joint optimization problem that takes into account the spatial equilibrium to balance the platform, drivers, and passengers. Our extensive experimental evaluations over the real data (Uber, Didi, and Taxi) from NYC, Beijing, and Chengdu have validated the accuracy, effectiveness, and profitability of `CAPrice` on urban MOD networks. In the future, we would like to investigate further the effect of space and time discretization upon the pricing performance.

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

REFERENCES

[1] 2019. Didi Chuxing Inc. Retrieved from: https://www.didichuxing.com.

[2] 2019. Driver payout & take-home. Retrieved from: https://ride.guru/content/resources/driver-payout-take-home.

[3] 2019. Flywheel's rate. Retrieved from: https://bestcompany.com/car-sharing/company/flywheel.

[4] 2019. NYC TLC trip record data. Retrieved from: http://www.nyc.gov/html/tlc/html/about/trip_record_data.shtml.

[5] 2016. Uber Pick-ups in NYC. Retrieved from: https://www.kaggle.com/fivethirtyeight/uber-pickups-in-new-york-city/data.

[6] 2018. Uber Pricing: What is Surge? Retrieved from: https://help.uber.com/h/e9375d5e-917b-4bc5-8142-23b89a440eec; Lyft Prime Time: Retrieved from: https://help.lyft.com/hc/en-us/articles/115012926467-Prime-Time-for-drivers.

[7] 2015. UberPeople. Net: Surging, but no pings? Retrieved from: https://uberpeople.net/threads/surging-but-no-pings.36735/.

[8] 2017. Global Mobility on Demand Market Forecast & Opportunities by 2022. Retrieved from: https://www.researchandmarkets.com/research/8qc94g/global_mobility.

[9] Eduardo M. Azevedo and E. Glen Weyl. 2016. Matching markets in the digital age. *Science* 352, 6289 (2016), 1056–1057.

[10] Siddhartha Banerjee, Ramesh Johari, and Carlos Riquelme. 2015. Pricing in ride-sharing platforms: A queueing-theoretic approach. In *Proceedings of the ACM EC*.

[11] Kostas Bimpikis, Ozan Candogan, and Saban Daniela. 2016. Spatial pricing in ride-sharing networks. *Soc. Sci. Res. Netw.* (2016). Retrieved from SSRN: https://ssrn.com/abstract=2868080.

[12] Stephen Boyd and Lieven Vandenberghe. 2004. *Convex Optimization*. Cambridge University Press.

[13] Harry Campbell. 2017. RSG 2017 Survey Results: Driver Earnings, Satisfaction and Demographics. Retrieved from: https://therideshareguy.com/rsg-2017-survey-results-driver-earnings-satisfaction-and-demographics/.

[14] Le Chen, Alan Mislove, and Christo Wilson. 2015. Peeking beneath the hood of Uber. In *Proceedings of the IMC*. 495–508.

[15] Zhixuan Fang, Longbo Huang, and Adam Wierman. 2017. Prices and subsidies in the sharing economy. In *Proceedings of the WWW*. 53–62.

[16] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. 2016. *Deep Learning*. The MIT Press.

[17] Suiming Guo, Chao Chen, Yaxiao Liu, Ke Xu, and Dah Ming Chiu. 2018. Modelling passengers' reaction to dynamic prices in ride-on-demand services: A search for the best fare. *Proc. ACM IMWUT* 1, 4, Article 136 (Jan. 2018), 23 pages.

[18] S. He, D. H. Shin, J. Zhang, J. Chen, and P. Lin. 2017. An exchange market approach to mobile crowdsensing: Pricing, task allocation, and Walrasian equilibrium. *IEEE JSAC* 35, 4 (Apr. 2017), 921–934.

[19] Suining He and Kang G. Shin. 2018. (Re)Configuring bike station network via crowdsourced information fusion and joint optimization. In *Proceedings of the ACM MobiHoc*. 1–10.

[20] S. He and K. G. Shin. 2018. Steering crowdsourced signal map construction via Bayesian compressive sensing. In *Proceedings of the IEEE INFOCOM*. 1016–1024.

[21] Suining He and Kang G. Shin. 2019. Spatio-temporal capsule-based reinforcement learning for mobility-on-demand network coordination. In *Proceedings of the WWW*. 2806–2813.

[22] Geoffrey E. Hinton, Zoubin Ghahramani, and Yee Whye Teh. 2000. Learning to parse images. In *Proceedings of the NIPS*. 463–469.

[23] Geoffrey E. Hinton, Alex Krizhevsky, and Sida D. Wang. 2011. Transforming auto-encoders. In *Proceedings of the ICANN*. 44–51.

[24] Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural Comput.* 9, 8 (1997), 1735–1780.

[25] Josh Horwitz. 2017. One year after the Uber-Didi merger, it's only getting harder to hail a ride in China. Retrieved from: 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.

[26] Rob J. Hyndman and George Athanasopoulos. 2014. *Forecasting: Principles and Practice*. OTexts.

[27] Takuro Ikeda, Takushi Fujita, and Moshe E. Ben-Akiva. 2015. Mobility on demand for improving business profits and user satisfaction. *FUJITSU Sci. Tech. J* 51, 4 (2015), 21–26.

[28] Diana Jorge, Goran Molnar, and Gonçalo Homem de Almeida Correia. 2015. Trip pricing of one-way station-based carsharing networks with zone and time of day price variations. *Trans. Res. Part B: Meth.* 81 (2015), 461–482.

[29] Diederik P. Kingma and Jimmy Ba. 2015. Adam: A method for stochastic optimization. In *Proceedings of the ICLR*.

[30] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. 2012. ImageNet classification with deep convolutional neural networks. In *Proceedings of the NIPS*. 1097–1105.

[31] Ricardo Lagos. 2000. An alternative approach to search frictions. *J. Polit. Econ.* 108, 5 (2000), 851–873.

[32] Ryan Lawler. 2015. Uber study shows its drivers make more per hour and work fewer hours than taxi drivers. Retrieved from: https://techcrunch.com/2015/01/22/uber-study/.

[33] Jonathan Levin. 2006. General equilibrium. *Microecon. Notes* (2006). Retrieved from: https://web.stanford.edu/ jdlevin/Econ%20202/General%20Equilibrium.pdf.

[34] Yuxuan Liang, Zhongyuan Jiang, and Yu Zheng. 2017. Inferring traffic cascading patterns. In *Proceedings of the ACM SIGSPATIAL*. Article 2, 10 pages.

[35] H. H. Lin, T. C. Kalanick, and E. Wang. 2014. System and method for providing dynamic supply positioning for on-demand services. Patent No. US20140011522.

[36] K. Lippoldt, T. Niels, and K. Bogenberger. 2018. Effectiveness of different incentive models in free-floating carsharing systems: A case study in Milan. In *Proceedings of the ITSC*. 1179–1185.

[37] Paul Lo, Andrew Tsukahira, and Henry Vogel. 2018. Mitigating surge pricing in ridesharing services. Patent No. US20180005145A1. Retrieved on January 4, 2018 from https://patents.justia.com/patent/20180005145.

[38] Alice Lu, Peter I. Frazier, and Oren Kislev. 2018. Surge pricing moves Uber's driver-partners. In *Proceedings of the ACM EC*. 3–3.

[39] Xiaolei Ma, Zhuang Dai, Zhengbing He, Jihui Ma, Yong Wang, and Yunpeng Wang. 2017. Learning traffic as images: A deep convolutional neural network for large-scale transportation network speed prediction. *Sensors* 17, 4 (2017), 818.

[40] Eric Maskin and Jean Tirole. 2001. Markov perfect equilibrium: I. Observable actions. *J. Econ. Theor.* 100, 2 (2001), 191–219.

[41] Chuishi Meng, Xiuwen Yi, Lu Su, Jing Gao, and Yu Zheng. 2017. City-wide traffic volume inference with loop detector data and taxi trajectories. In *Proceedings of the ACM SIGSPATIAL*. Article 1, 10 pages.

[42] F. Miao, S. Han, A. M. Hendawi, M. E. Khalefa, J. A. Stankovic, and G. J. Pappas. 2017. Data-driven distributionally robust vehicle balancing using dynamic region partitions. In *Proceedings of the ACM/IEEE ICCPS*. 261–272.

[43] J. Miller and J. P. How. 2017. Predictive positioning and quality of service ridesharing for campus mobility on demand systems. In *Proceedings of the IEEE ICRA*. 1402–1408.

[44] Afshin Nikzad. 2017. Thickness and competition in ride-sharing markets. *Soc. Sci. Res. Netw.* (2017). Retrieved from SSRN: https://ssrn.com/abstract=3065672.

[45] Kevin Mark Novak and Travis Cordell Kalanick. 2013. System and method for dynamically adjusting prices for services. Patent No. US20130246207A1. Retrieved on September 19, 2013 from https://patents.google.com/patent/US20130246207A1/en.

[46] Takuma Oda and Carlee Joe-Wong. 2018. MOVI: A model-free approach to dynamic fleet management. In *Proceedings of the IEEE INFOCOM*. 2708–2716.

[47] Marco Pavone. 2016. Autonomous mobility-on-demand systems for future urban mobility. In *Autonomous Driving*. Springer, 387–404.

[48] X. Qian and S. V. Ukkusuri. 2017. Time-of-day pricing in taxi markets. *IEEE T-ITS* 18, 6 (June 2017), 1610–1622.

[49] 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. *Trans. Policy* 45 (2016), 168–178. DOI: https://doi.org/10.1016/j.tranpol.2015.10.004.

[50] David E. Rumelhart, Geoffrey E. Hinton, and Ronald J. Williams. 1986. Learning representations by back-propagating errors. *Nature* 323, 6088 (1986), 533.

[51] Sara Sabour, Nicholas Frosst, and Geoffrey E. Hinton. 2017. Dynamic routing between capsules. In *Proceedings of the NIPS*. 3859–3869.

[52] Kevin Spieser, Kyle Treleaven, Rick Zhang, Emilio Frazzoli, Daniel Morton, and Marco Pavone. 2014. Toward a systematic approach to the design and evaluation of automated mobility-on-demand systems: A case study in Singapore. In *Road Vehicle Automation*. 229–245.

[53] Yongxin Tong, Yuqiang Chen, Zimu Zhou, Lei Chen, Jie Wang, Qiang Yang, Jieping Ye, and Weifeng Lv. 2017. The simpler the better: A unified approach to predicting original taxi demands based on large-scale online platforms. In *Proceedings of the ACM KDD*. 1653–1662.

[54] Dong Wang, Wei Cao, Jian Li, and Jieping Ye. 2017. DeepSD: Supply-demand prediction for online car-hailing services using deep neural networks. In *Proceedings of the IEEE ICDE*. 243–254.

[55] Dong Wang, Junbo Zhang, Wei Cao, Jian Li, and Yu Zheng. 2018. When will you arrive? Estimating travel time based on deep neural networks. In *Proceedings of the AAAI*. 2500–2507.

[56] L. Wang, B. Guo, and Q. Yang. 2018. Smart city development with urban transfer learning. *Computer* 51, 12 (Dec. 2018), 32–41.

[57] Shuai Wang, Tian He, Desheng Zhang, Yuanchao Shu, Yunhuai Liu, Yu Gu, Cong Liu, Haengju Lee, and Sang H. Son. 2018. BRAVO: Improving the rebalancing operation in bike sharing with rebalancing range prediction. *Proc. ACM IMWUT* 2, 1, Article 44 (Mar. 2018), 22 pages.

[58] Xiaoyang Xie, Fan Zhang, and Desheng Zhang. 2018. PrivateHunt: Multi-source data-driven dispatching in for-hire vehicle systems. *Proc. ACM IMWUT* 2, 1, Article 45 (Mar. 2018), 26 pages. DOI: http://doi.acm.org/10.1145/3191777.

[59] 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 *Proceedings of the ACM KDD*. 905–913.

[60] Dejun Yang, Guoliang Xue, Xi Fang, and Jian Tang. 2012. Crowdsourcing to smartphones: Incentive mechanism design for mobile phone sensing. In *Proceedings of the ACM MobiCom*. 173–184.

[61] Yu Yang, Fan Zhang, and Desheng Zhang. 2018. SharedEdge: GPS-free fine-grained travel time estimation in state-level highway systems. *Proc. ACM IMWUT* 2, 1, Article 48 (Mar. 2018), 26 pages. DOI: https://doi.org/10.1145/3191780.

[62] Huaxiu Yao, Fei Wu, Jintao Ke, Xianfeng Tang, Yitian Jia, Siyu Lu, Pinghua Gong, Jieping Ye, and Zhenhui Li. 2018. Deep multi-view spatial-temporal network for taxi demand prediction. In *Proceedings of the AAAI*. 2588–2595.

[63] J. Yuan, Y. Zheng, X. Xie, and G. Sun. 2013. T-Drive: Enhancing driving directions with taxi drivers' intelligence. *IEEE TKDE* 25, 1 (Jan. 2013), 220–232.

[64] Yukun Yuan, Desheng Zhang, Fei Miao, John A. Stankovic, Tian He, George Pappas, and Shan Lin. 2018. Dynamic integration of heterogeneous transportation modes under disruptive events. In *Proceedings of the ACM/IEEE ICCPS*. 65–76.

[65] Desheng Zhang, Tian He, and Fan Zhang. 2017. Real-time human mobility modeling with multi-view learning. *ACM TIST* 9, 3, Article 22 (Dec. 2017), 25 pages.

[66] Junbo Zhang, Yu Zheng, and Dekang Qi. 2017. Deep spatio-temporal residual networks for citywide crowd flows prediction. In *Proceedings of the AAAI*. 1655–1661.

[67] Yu Zheng, Licia Capra, Ouri Wolfson, and Hai Yang. 2014. Urban computing: Concepts, methodologies, and applications. *ACM TIST* 5, 3, Article 38 (Sept. 2014), 55 pages.

[68] Yu Zheng, Yanchi Liu, Jing Yuan, and Xing Xie. 2011. Urban computing with taxicabs. In *Proceedings of the ACM UbiComp*. 89–98.

Received April 2019; accepted May 2019