An On-line Truthful and Individually Rational Pricing Mechanism for Ride-sharing

Mohammad Asghari
University of Southern California
Los Angeles, CA, USA
masghari@usc.edu

Cyrus Shahabi
University of Southern California
Los Angeles, CA, USA
shahabi@usc.edu

ABSTRACT

Ride-sharing has the potential of addressing many socioeconomic challenges related to transportation. The rising popularity of ride-sharing platforms (e.g., Uber, Lyft, DiDi) in addition to the emergence of new applications like food delivery and grocery shopping which use a similar platform, calls for an in-depth and detailed evaluation of various aspects of this problem.

Auction frameworks and mechanism design, have been widely used for modeling ride-sharing platforms. A key challenge in these approaches is preventing the involving parties from manipulating the platform for their personal gain which in turn, can result in a less satisfactory experience for other parties and/or loss of profit for the platform provider. We introduce a latent space transition model for ride-sharing platforms which drivers can exploit and predict the future supply of the drivers (i.e., available drivers) to their own advantage. Following, we propose a pricing model for ride-sharing platforms which is both truthful and individually rational based on Vickery auctions and show how we can manage the loss of revenue in this approach. We compare our predicting model and pricing model with competing approaches through experiments on New York City's taxi dataset. Our results show that our model can accurately learn the transition patterns of people's ride requests. Furthermore, our pricing mechanism forces drivers to be truthful and takes away any unfair advantage the drivers can achieve by bidding untruthfully. More importantly, our pricing model forces truthfulness without sacrificing much profit unlike what is typical with second-price auction schemes.

CATEGORIES AND SUBJECT DESCRIPTORS

H.3.5 [**Information Storage and Retrieval**]: Online Information Services-*Commercial services*

KEYWORDS

Ride-sharing, Revenue Maximization, Spatial Crowdsourcing

SIGSPATIAL'17, November 7–10, 2017, Los Angeles Area, CA, USA

ACM ISBN 978-1-4503-5490-5/17/11…$15.00
https://doi.org/10.1145/3139958.3139991

1 INTRODUCTION

Traffic congestions, high costs of car ownership and greenhouse emissions are just a few problems that have made road transportation one of the main challenges of our era. Ride-sharing is an alternative transportation model which helps mitigate the negative effects of road transportation. According to [1], in only one month, UberPool reduced individual rides in San Francisco for about 674,000 miles which accounts for more than 13,000 gallons of gasoline and about 120 metric tons of CO2 emissions. In addition to well-known ride-sharing platforms (e.g., Lyft, Uber, DiDi), other applications like Instacart and AmazonFresh, which use the same model as ride-sharing platforms, can also benefit from a real-time ride-sharing framework.

Due to its economical, environmental and social benefits, many researchers have studied ride-sharing platforms [5, 8–11, 15, 19]. Many former studies [8, 9, 15] focus on improving the efficiency of the passenger-to-driver assignments by minimizing the total traveled distance which does not always satisfy the monetary constraints of passengers and drivers [3]. Furthermore, none of the proposed approaches in these studies can scale due to their centralized architecture.

Auction methods have been effectively used for assignment problems in dynamic multi-agent environments [13, 16]. The main advantage of auction methods are their simplicity and the fact that they allow for decentralized implementation. Consequently, in recent years, ride-sharing platforms have been studied in the context of auction methods [5, 10, 19]. One of the key requirements of such platforms is the ability to perform a real-time *on-line* assignment of passengers to drivers as soon as a ride request is received. Most of the previous studies do not support on-line assignments[5, 10, 11].

In [3], we proposed a ride-sharing platform, named APART, with a pricing model and a passenger-to-driver assignment algorithm. The main objective of APART is to maximizes the platform provider's profits, without compromising the passengers' and drivers' monetary incentives (i.e., either charging the passengers more or paying the drivers less). This is achieved by introducing the concept of *profiles* as a mechanism for both parties to report their monetary constraints to the platform. The assignment algorithm guarantees these constraints are met when assigning a passenger to a driver. One drawback of the pricing model of APART is that drivers have an incentive to misreport their profiles, in order to increase their own income. For example, at certain times during the day when the demand (i.e., ride requests) to supply (i.e., available drivers) ratio is high, there will be less competition among drivers for each ride request. In such demand-driven markets, the drivers can increase their reported costs without significantly reducing their chance of getting matched with a passenger.

In this paper, first we present a statistical analysis that proves the drivers' dominant strategy in APART's pricing model is to misreport their profiles. That is, if a driver can estimate the number of available drivers in the vicinity of a new ride request then he can exploit this information to misrepresent his profile and still bid competitively. Subsequently, we present a latent space transition model (LSTM) which the drivers can utilize to predict the number of available drivers in every location at any point in time. We build this model by learning the transitional pattern of ride requests from historical data. We model the historical data as a set of spatial documents and utilize a *Latent Dirichlet Allocation (LDA)*[4] approach to learn the parameters of the transitional pattern.

To address the problem of misreporting profiles, we present a new pricing mechanism for APART based on the second-price auction scheme with a reserved price (SPARP). We show that SPARP is both *individually rational* (i.e., the drivers are incentivized to participate in the platform) and *truthful* (i.e., the dominant strategy of drivers is to report their true profiles). We exploit the concept of *reserved prices* in auction methods to prevent the platform from losing too much profit as a result of the second-price scheme and show how the value of the reserved price can be computed for different requests in APART.

We conducted extensive experiments on a large scale New York City taxi dataset. We compare the prediction accuracy of LSTM with a state-of-the-art approach [23]. We show that our model can learn the transition patterns more accurately and hence, results in a 50% more accurate prediction. Furthermore, we compare the utility of the drivers in a first-price auction scheme when a portion of the drivers bid untruthfully. We show that bidding untruthfully in a first-price auction scheme can result in up to 25% higher income for drivers. Finally, we show that unlike a typical second-price auction scheme, by using reserved price in our mechanism, we can prevent the framework from losing too much profit.

The remainder of this paper is organized as follows. In Section 2, we briefly review the key components of APART that are relevant to this paper. In Section 3, we show that the dominant strategy of drivers in APART is to misreport their profiles. We present our latent space transition model in Section 4 which the drivers can utilize to predict the number of available drivers. We introduce our new pricing mechanism in Section 5. In Section 6, we report the experiment results. Section 7 reviews the related works and we conclude the paper in Section 8.

2 BACKGROUND

In this section, we briefly review the key components of the APART framework[3].

2.1 Definitions

We first define three key concepts of the APART framework.

Definition 2.1 (Ride Request). A ride request r can be represented as $\langle s_r, e_r, w_r, \epsilon_r, \lambda_r \rangle$ consisting of a starting point s_r and an end point e_r. Each request also specifies w_r as the maximum time the rider is willing to wait after making a request and ϵ_r denotes the relative detour the rider can tolerate. In addition, a rider's profile $\lambda_r : \mathbb{R}_+ \rightarrow [0, 1]$, specifies the relative discount in exchange for an incurred detour of $\Delta \in \mathbb{R}_+$.

Definition 2.2 (Driver). A driver d is represented as $\langle \mathcal{R}_d, n_d, \theta_d \rangle$ where $\mathcal{R}_d \subset \mathcal{R}$ is the set of ride requests assigned to d, and n_d is the maximum number of requests d can accept at any point in time. A driver also has a profile $\theta_d : \mathbb{R}_+ \rightarrow \$$[1] which specifies the monetary cost of d providing service to its assigned request for a duration $\omega \in \mathbb{R}_+$.

Definition 2.3 (Schedule). Schedule π_d for driver d on the set of requests $\mathcal{R}_d$ with n requests, is an ordered sequence $\langle x_1, \cdots, x_{2n} \rangle$, of pick-up and drop-off points of requests in $\mathcal{R}_d$, where for each $r \in \mathcal{R}_d$, s_r precedes e_r in π_d.

The driver follows the sequence of picking up and dropping off riders. For every two nodes $x_a, x_b \in \pi_d$ where x_a precedes x_b. We show the cost of traveling from x_a to x_b in schedule π_d with $\phi_{\pi_d}(x_a, x_b)$.

2.2 Dispatch Requests

APART considers drivers as bidders and ride requests as goods. The server plays the role of a central auctioneer in APART. Figure 1 explains how ride requests are dispatched to drivers at a very high level: (1) Everything starts with a passenger submitting a new ride request to the server. (2) Once the server receives a new task, it notifies the available drivers in the vicinity of the pick-up location about the new request. (3) Each driver independently computes his bid by finding the optimal schedule that can fit the new request into his current schedule. The bidding process is performed as a *sealed-bid auction* where drivers simultaneously submit bids and no other driver knows how much the other drivers have bid.(4) Once all the bids are received, the server assigns the passenger to the driver with the optimal bid.

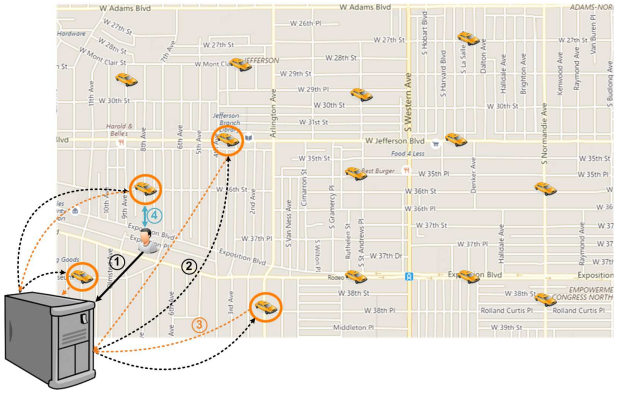


Figure 1: Simple Scenario

Algorithm 1 outlines the process of assigning an incoming request r, where $\mathcal{D}_r$ is the set of eligible drivers for request r (line 4). For each candidate driver d, the *ComputeBid* method (line 5) is executed to perform scheduling and compute d's bid (Section 2.3). Subsequently, the platform chooses the driver with the highest bid. In case of a tie in line 8, the algorithm randomly selects one driver among the ones with the highest bid. Notice that in practice all the iterations of the **for** loop in Algorithm 1 (lines 4-7) run in parallel.

[1] In this paper, we show monetary values with $

Algorithm 1 Dispatch($\mathcal{D}, r, startTime$)

Input: $\mathcal{D}$ is the set of currently available drivers, r is a new request and $startTime$ is the current time
Output: $d \in \mathcal{D}$ as the driver that request r is assigned to

1: $d_{opt} \leftarrow null$
2: $Bids_r \leftarrow \emptyset$
3: $\mathcal{D}_r \leftarrow$ EligibleDrivers(r)
4: **for** $d \in \mathcal{D}_r$ **do**
5: $\quad bid_d^r \leftarrow$ ComputeBid($d, \pi_d, r, startTime$)
6: $\quad Bids_r \leftarrow Bids_r \cup \{bid_d^r\}$
7: **end for**
8: $d_{opt} \leftarrow \arg\max_d \left\{bid_d^r \mid bid_d^r \in Bids\right\}$
9: **return** d_{opt}

2.3 Bid Computation

Once a driver is notified of a new request, it has to compute a bid. The bid each driver generates reflects the profit the system can gain if the request is assigned to that driver. Once the ride-sharing application on the driver's phone receives the request, it generates a bid and submits the bid to the server. When a new request is assigned to a driver, he will be notified with an updated schedule. This means that the human driver's interaction with APART is limited to configuring and reporting his profile on the ride-sharing application.

Algorithm 2 ComputeBid($d, \pi_d, r, startTime$)

Input: d is a driver with schedule π_d, r is a new request and $startTime$ is the current time.
Output: additional $profit$ that d can generate by accepting r

1: $newProfit \leftarrow$FindBestSchedule($d, \pi_d, r, startTime$)
2: $oldProfit \leftarrow$ GetProfit($d, \pi_d, startTime$)
3: $additionProfit \leftarrow newProfit - oldProfit$
4: **return** $additionProfit$

Algorithm 2 outlines the bid computation process. First, the algorithm calls *FindBestSchedule* (line 1) which finds the best valid schedule and its corresponding profit. Because each driver's bid is the *additional* profit that the new request can generate for the platform, the algorithm calculates $oldProfit$ for d's original schedule using the *GetProfit* method (line 2). Finally, the additional profit that d can generate by accepting r is the difference between $newProfit$ and $oldProfit$. The *FindBestSchedule* method uses a branch-and-bound technique to search for the optimal schedule that can fit the new request into a driver's existing schedule. The *GetProfit* method takes a valid schedule (π) and a driver d as input and computes the total fare collected from the passengers serviced with π and the total cost of d performing π and returns the difference between the collected fare and cost as the profit of the schedule. In Section 5 we explain how the fare and cost of rides are computed. Further details of the *FindBestSchedule* and *GetProfit* methods are out of the scope of this paper and can be found in [3].

Once drivers submit their bids, the server selects the driver with the highest bid and assigns the new request to that driver.

3 COMPETITIVE BIDDING

With APART, a driver's income is as much as his reported cost. This is called the first-price auction scheme. Here, we explain how bidders can compute their bids in a first-price scenario to manipulate the system and increase their own income. We assume the bidders know how many other bidders are participating in the auction and also know the distribution of their bids, but not the exact value for everyone else's bids. We denote the *probability density function* and the *cumulative distribution function* of the bids with $f(.)$ and $F(.)$, respectively. Also, for every bidder i with valuation v_i, we assume there exists a *strategy function* $s_i(.)$ that bidder i applies to its valuation to compute what bid to submit. We are interested to find the optimal $s_i(.)$ such that bidder i's *expected utility* $E[u_i]$ is maximized.

Before we continue, we make two assumptions:

(1) The *strategy function* $s_i(.)$ for every user is strictly increasing. In other words, if $v_1 < v_2$ then $s_i(v_1) < s_i(v_2)$.
(2) We will restrict our search to symmetric equilibria (i.e., all bidders use the same equilibria strategy).

We show everything from the point of view of bidder 1. However, since we are considering only symmetric equilibria, the computation will be the same for all other bidders. We start by defining the expected utility of bidder 1 as:

$$E[u_1] = (v_1 - s_1(v_1)) \cdot Prob[win_1] \tag{1}$$

where $Prob[win_1]$ is the probability of bidder 1 winning.

To compute $Prob[win_1]$, first we consider a single bidder i. For bidder 1 to win over bidder i we need to have $s_i(v_i) < s_1(v_1)$.

$$\begin{aligned} Prob[s_i(v_i) < s_1(v_1)] &= Prob\left[v_i < s_i^{-1}(s_1(v_1))\right] \\ &= F\left[s_i'(s_1(v_1))\right] \\ &= F(v_1) \end{aligned}$$

The last equation holds because all bidders use the same strategy function and thus, $s_i(x) = s_j(x)$ for every bidder i and j.

Bidder 1 wins if her bid is higher than all other $n-1$ bidders. Since every bidder i's bid ($i \neq 1$) is independent of other bidders we can say:

$$\begin{aligned} Prob[win_1] &= (Prob[s_i(v_i) < s_1(v_1)])^{n-1} \\ &= F(v_1)^{n-1} \end{aligned}$$

Now we can rewrite Equation (1) as:

$$E[u_1] = (v_1 - s_1(v_1))\ F(v_1)^{n-1} \tag{2}$$

We can maximize $E[u_1]$ by differentiating Equation (2) w.r.t. b_1 and setting it equal to zero, where b_1 is bidder 1's bid. In other words, $b_1 = s_1(v_1)$.

$$\frac{\partial}{\partial b_1} E[u_1] = 0$$

$$\frac{\partial}{\partial b_1}(v_1 - b_1)\ F\left(s_1^{-1}(b_1)\right)^{n-1} = 0$$

Using the Chain Rule and the Product Rule we get:

$$(v_1 - b_1) \cdot \frac{\partial F\left(s_1^{-1}(b_1)\right)^{n-1}}{\partial F\left(s_1^{-1}(b_1)\right)} \cdot \frac{\partial F\left(s_1^{-1}(b_1)\right)}{\partial\left(s_1^{-1}(b_1)\right)} \cdot \frac{\partial\left(s_1^{-1}(b_1)\right)}{\partial b_1} = F\left(s_1^{-1}(b_1)\right)^{n-1}$$

We know that the derivative of the cumulative probability function $F(.)$ is the probability density function $f(.)$. Also:

$$\frac{\partial}{\partial b_1}\left(s_1^{-1}(b_1)\right) = \frac{1}{s_1'\left(s_1^{-1}(b_1)\right)}$$

Then we will have:

$$\frac{(v_1 - b_1)(n-1)\cdot F\left(s_1^{-1}(b_1)\right)^{n-2}\cdot f\left(s_1^{-1}(b_1)\right)}{s_1'\left(s_1^{-1}(b_1)\right)} = F\left(s_1^{-1}(b_1)\right)^{n-1} \tag{3}$$

$$\frac{(v_1 - b_1)(n-1)\cdot f\left(s_1^{-1}(b_1)\right)}{s_1'\left(s_1^{-1}(b_1)\right)} = F\left(s_1^{-1}(b_1)\right) \tag{4}$$

Knowing that $s_1^{-1}(b_1) = v_1$, we can re-write Equation (4) as:

$$(v_1 - b_1)(n-1)\cdot f(v_1)\cdot\frac{1}{s_1'(v_1)} = F(v_1) \tag{5}$$

We can set $b_1 = s_1(v_1)$ and re-arrange Equation (5) and get:

$$s_1'(v_1) = (n-1)\left(\frac{f(v_1)\ (v_1 - s_1(v_1))}{F(v_1)}\right) \tag{6}$$

Solving the differential equation of Equation (6) yields the optimal strategy function $s_1(.)$ for bidder 1 which gives her what to bid for a valuation of v_1. To solve Equation (6), we need to know $f(.)$ and $F(.)$ (i.e., the probability density function and cumulative density functions of the bids). For example, assuming the bids are uniformly distributed in the range $[0, v_{max}]$ we will have:

$$\forall x \in [0, v_{max}] \qquad f(x) = \frac{1}{v_{max}} \qquad \text{and} \qquad F(x) = \frac{x}{v_{max}}$$

Using these values for $f(.)$ and $F(.)$ in Equation (6) we get:

$$s_1(v_1) = \left(\frac{n-1}{n}\right)v_1 \tag{7}$$

Which give the optimal bidding strategy if every driver's valuation is a uniform random variable in $[0, v_{max}]$.

4 THE LATENT SPACE TRANSITION MODEL

In Section 3, we showed how the drivers can take advantage of the platform if they know how many bids are being submitted and the distribution of the bids. In this section, we show how the drivers can estimate the number of bids, based on historical data.

We assume for each location p and time t, there are a potential of β_p^t ride requests submitted to the system. A transition matrix A^t called the *network demand at time t* shows the fraction of riders moving from one location to another location at each point in time. The pq-th entry of A^t, shown as α_{pq}^t, gives the fraction of riders going from location p to location q at time t. This means that the *number* of riders requesting a ride from p to q at time t is given by $\beta_p^t\,\alpha_{pq}^t$.

We can compute the number of available drivers at location p at the start of time t as:

$$v_p^t = \sum_q \alpha_{qp}^{t-1}\min\left\{v_q^{t-1}, \beta_q^{t-1}\right\} + \delta_p^t \tag{8}$$

where δ_p^t is the number of drivers who enter the system at time t in location p.

To predict the number of drivers that enter the platform (i.e., δ_p^t) we use the approach in [6] where a grid-based Gaussian mixture model is introduced to predict the number of passengers for taxi bookings at a specific time and location.

The other key component in estimating the number of available drivers at each location (Equation (8)) is learning the network demand matrix at each point in time. In the remainder of this section, we introduce a *Latent Space Transition Model (LSTM)* where we model the demand network as a *Latent Dirichlet Allocation (LDA)*[4] model. We first explain how we can model the network demand matrix as an LDA and subsequently explain how we can learn the parameters of the model using historic data.

4.1 Network Demand Model

An LDA is a generative probabilistic model for collections of discrete data, in which each item of a collection is modeled as a finite mixture over an underlying set of topics. Each topic is, in turn, modeled as an infinite mixture over an underlying set of topic probabilities[4]. For example LDA is widely used in natural language processing where the observations are words in documents. LDA then assumes each document is a mixture of a small number of topics and tends to find patterns that probabilistically associate the words in the document to topics.

With regard to Ride-sharing environments, various factors (hereafter, topics) such as locality (i.e., source and destination neighborhoods), time and weather, can influence the network demand. Row p of the matrix A^t (i.e., A_p^t), gives the probability distribution over the destination of requests submitted at time t in location p. Therefore, we can think of A_p^t as a collection of destinations where the different topics impact the probability of each location being the destination of those specific requests. For example, a high demand for rides from a business district in a city to residential neighborhoods can be attributed to the locality topic.

In the remainder of this section we assume that the entire geographical space is discretized into smaller regions using a grid index. Similarly, we assume the temporal space is discretized into equal length time slots. For a set of ride request data $\chi_1, \chi_2, \cdots, \chi_d$ in the form of $(source, destination, time)$, we define a spatial document as:

Definition 4.1 (Spatial Document). For every source region p and time t, we define the Spatial Document X_p^t as a vector of length w such that x_{pq}^t for $1 \le q \le w$ records the number of requests originated in location p for destination q at time t.

We assume each spatial document is a k component *Multinomial Mixture Model (MMM)*. Consequently, each spatial document is modeled as an independent draw from the probability mass function:

$$p(x_i) = \sum_{j=1}^{k}\sum_{p=1}^{w} x_{ip}\cdot\psi_{ij}\cdot\mu_{jp} \tag{9}$$

where,

- $\mu_j \in \mathbb{R}^w$, w is the total number of regions and μ_{jp} is the unknown probability of selecting region p from topic j.
- ψ_{ij} is the unknown probability of selecting topic j for document i, such that $\sum_{j=1}^{k}\psi_{ij} = 1$.

Consequently, the generative model for each destination region in spatial document i with latent variables $(\psi_i, \mu_{1:k})$ can be written as:

(1) We draw latent topic indicators $z_{ip} \overset{iid}{\sim} \text{Mult}(\psi_i)$.
(2) For each topic z_{ip}, we can draw x from the topic's multinomial distribution:

$$x|z_{ip} \overset{ind}{\sim} \text{Mult}(\mu_{z_{ip}})$$

4.2 Parameter Inference

In order to learn the parameters of the model introduced in Section 4.1, we utilize the *expectation maximization (EM)* algorithm. The EM algorithm iteratively finds the maximum likelihood estimates of parameters of a statistical model which depends on unobserved latent variables. Each iteration of the EM algorithm consists of two steps. The *expectation (E)* step which estimates each latent variable z's conditional on the observed data $p(z_{1:n,1:w}|x_{1:n,1:w}; \psi_{1:n}, \mu_{1:k})$. Subsequently, the *maximization (M)* step, finds the corresponding parameters that maximize the expected log-likelihood w.r.t. the latent variables estimated in the E step.

Before we explain each step in more details, we need to have the complete log-likelihood function for the model introduced in Section 4.1. The log-likelihood for our model can be computed as:

$$\begin{aligned}\mathcal{L}_{\mu,\psi} &= log\, p(x_{1:n,1:w}, z_{1:n,1:w}; \psi_{1:n}, \mu_{1:k}) \\ &= \sum_{i=1}^{n}\sum_{p=1}^{w} log\, p(x_{ip}, z_{ip}; \psi_i, \mu_{1:k}) \\ &= \sum_{i=1}^{n}\sum_{p=1}^{w} log \sum_{j=1}^{k} \mathbb{I}\left\{z_{ip}=j\right\}\, x_{ip}\, \psi_{iz_{ip}} \mu_{z_{ip}p}\end{aligned} \tag{10}$$

where $\mathbb{I}$ is the indicator function.

4.2.1 E-Step. In this step, we estimate the conditional distribution of each z_{ip} given x_{ip} ($p(z_{ip} = j|x_{ip}; \psi_i, \mu_{1:k})$). By Bayes's rule:

$$p(z_{ip} = j|x_{ip}; \psi_i, \mu_{1:k}) \propto p(z_{ip} = j; \psi_i)\, p(x_{ip}|z_{ip}; \mu_{1:k}) \tag{11}$$

By plugging in the densities based on Equation (9) we get:

$$p(z_{ip} = j|x_{ip}; \psi_i, \mu_{1:k}) = \frac{\psi_{ij}\, \mu_{jp}}{\sum_{l=1}^{k} \psi_{il}\, \mu_{lp}} \tag{12}$$

For simplicity, we show this estimated conditional distribution as τ_{ipj}.

4.2.2 M-Step. This step will maximizes the expected complete log-likelihood. Given the updated latent variables from the E-Step, it can be derived that the following setting updates the log-likelihood of our model:

$$\psi_{ij} = \frac{\sum_{p=1}^{w} x_{ip}\, \tau_{ijp}}{\sum_{p=1}^{w} x_{ip}} \qquad \mu_{jp} = \frac{\sum_{i=1}^{n} x_{ip}\, \tau_{ijp}}{\sum_{q=1}^{w}\sum_{i=1}^{n} x_{iq}\, \tau_{ijq}} \tag{13}$$

Algorithm 3 shows the overall process of inferring the parameters of the statistical model introduced in Section 4.1. In Lines 2 and 5, $Dir()$ represents the Dirichlet distribution and is used to initialize values for ψ and μ. We used the Dirichlet distribution since it is the conjugate prior for the Multinomial distribution. In Line 7 the log-likelihood of the data is computed with the initial values of ψ and μ. The **while** loop (Line 9-17) iteratively performs the *E-Step* and *M-Step* to update ψ, μ and the auxiliary variable τ. After each step, it computes the log-likelihood with the updated parameters and if the improvement of the log-likelihood is less than a predefined ϵ, it terminates the algorithm and returns the final parameters.

Algorithm 3 EM($X_{1:n}, \boldsymbol{\gamma_1}, \boldsymbol{\gamma_2}$)

Input: $X_{1:n}$ are n spatial documents, $\boldsymbol{\gamma_1}$ and $\boldsymbol{\gamma_2}$ are vectors of positive reals with size k and w respectively.
Output: μ, ψ as the parameters of the Multinomial Mixture Model

1: **for** $i = 1 : n$ **do**
2: $\quad\psi_i \leftarrow Dir(\boldsymbol{\gamma_1})$
3: **end for**
4: **for** $k = 1 : K$ **do**
5: $\quad\mu_k = Dir(\boldsymbol{\gamma_2})$
6: **end for**
7: $\mathcal{L}^0_{\mu,\psi}$ = compute likelihood from Equation (10)
8: $converge$ = false, $n \leftarrow 1$
9: **while** (!$converge$) **do**
10: $\quad$update τ from Equation (12)
11: $\quad$update μ and ψ from Equation (13)
12: $\quad\mathcal{L}^n_{\mu,\psi}$ = compute likelihood from Equation (10)
13: $\quad$**if** $(\mathcal{L}^n_{\mu,\psi} - \mathcal{L}^{n-1}_{\mu,\psi} < \epsilon)$ **then**
14: $\quad\quad converge$ = true
15: $\quad$**end if**
16: $\quad n \leftarrow n + 1$
17: **end while**
18: **return** μ, ψ

Each spatial document X, corresponds to one row of the transition matrix, A. Assuming the corresponding spatial document of A^t_p is X_i, each cell of matrix A can be derived as follows:

$$\alpha^t_{pq} = \sum_{j=1}^{k} \psi_{ij} \cdot \mu_{jq} \tag{14}$$

where index i in ψ_{ij} refers to the spacial document X^t_p.

5 THE *SPARP* MECHANISM

5.1 Pricing Model

Every request r has a default fare based on the duration of the shortest trip, $\phi(s_r, e_r)$ (For convenience, we show this as ϕ_r). We define function $F : \mathbb{R}_+ \rightarrow \$$ such that $F(\phi_r)$ is the default fare of a ride. In addition, we show the *actual* route between the two end points of a ride with $\hat{\phi}_r$ and define the detour of a ride as $\Delta_r = \hat{\phi}_r - \phi_r$. As explained in definition Theorem 2.1, each request is associated with a profile as a tool for the rider to specify how much discount he expects to receive in return for a certain amount of detour on his trip.

Subsequently, for a request r with shortest trip ϕ_r, detour Δ_r and a profile λ_r, the final fare is represented as:

$$fare(r) = F(\phi_r) \cdot \lambda_r(\Delta_r) \tag{15}$$

Every driver has a profile which allows them to set their minimum expectations for participating in the platform. A driver's profile represents the cost of the driver participating in the platform. The profile can depend on any number of parameters; e.g., the length of the trip, the number of passengers, etc. In [3] we studied the effect of various profiles for the drivers. Here, we assume a driver's profile only depends on the duration for which the driver provides service.

Drivers can misreport their actual profiles if it is to their best interest. We show driver d's reported profile as $\hat{\theta_d} \in \Theta$ where Θ is the set of all possible profiles. At any point in time, each driver has a schedule. Therefore, for every driver d, the cost of servicing schedule π_d based on the reported profile $\hat{\theta_d}$ is:

$$cost(\pi_d, \hat{\theta_d}) = \int_{start_{\pi_d}}^{end_{\pi_d}} \mathbb{I}\left\{\pi_d^t \neq \langle\rangle\right\} .\hat{\theta_d}(t)dt \tag{16}$$

Where $\mathbb{I}$ is the indicator function and π_d^t is the driver's schedule at time t. In addition, $start_{\pi_d}$ and end_{π_d} are the first pick-up time and last drop-off time of π_d.

For every request r assigned to a driver, depending on the driver's reported profile ($\hat{\theta_d}$) the driver has to pay a *profit* to the platform provider ($\rho(r)$). We can define driver d's payments to the platform and income for servicing schedule π_d as:

$$payment(\pi_d, \hat{\theta_d}) = \sum_{r \in \pi_d} \rho_r \tag{17}$$

$$income(\pi_d, \hat{\theta_d}) = \sum_{r \in \pi_d} fare(r) - payment(\pi_d, \hat{\theta_d}) \tag{18}$$

For every driver d, we define the *utility* as the difference between his income and cost for servicing schedule π_d:

$$u(\pi_d, \hat{\theta_d}, \theta_d) = income(\pi_d, \hat{\theta_d}) - cost(\pi_d, \theta_d) \tag{19}$$

If a driver does not participate in the ride-sharing platform, both the cost and income is zero. We say the platform is *individually rational* if no driver receives a negative utility by participating in the platform. Another crucial aspect of the framework should be preventing the drivers to strategically manipulate the platform by misreporting their profiles, which is known as *truthfulness*.

Definition 5.1 (Truthfulness). A platform is *truthful* if and only if for every driver d, $\forall \hat{\theta_d} \in \Theta \wedge \hat{\theta_d} \neq \theta_d, u(\pi_d, \hat{\theta_d}, \theta_d) \leq u(\pi_d, \theta_d, \theta_d)$.

That is, a platform is truthful if the dominant (most profitable) strategy for drivers is to report their profiles truthfully.

The overall goal of the platform is to maximize the revenue of the platform provider.

Definition 5.2 (Revenue). Given the matching $M(\mathcal{D}, \mathcal{R})$ between the set of drivers $\mathcal{D}$ and requests $\mathcal{R}$, the revenue of the platform provider is

$$revenue(M(\mathcal{D}, \mathcal{R})) = \sum_{d \in \mathcal{D}} payments(\pi_d, \hat{\theta_d}) \tag{20}$$

We call a framework *budget balanced* if $revenue(M(\mathcal{D}, \mathcal{R})) \geq 0$. Otherwise, we say the framework runs a *deficit*.

5.2 Payments

In Section 5.1 we mentioned that for every request r assigned to a driver, the driver has to pay ρ_r to the platform provider. In this section we explain what ρ_r should be so that the platform be:

(1) *individually rational*; i.e., the drivers do not end up with a negative utility.
(2) *truthful*; i.e., the drivers cannot manipulate the framework by misreporting their profiles.

Based on Algorithm 2 the value of the bid driver d submits to the server for request r (bid_d^r) is equal to the *additional profit* driver d generates by accepting a new request r.

Theorem 5.3. *If for every request r assigned to driver d, $\rho_r \leq bid_d^r$ then the ride-sharing platform is individually rational.*

Proof. From Equation (17) we know:

$$\begin{aligned} income(\pi_d, \hat{\theta}_d) &= \sum_{r \in \pi_d} fare(r) - \sum_{r \in \pi_d} \rho_r \\ &\geq \sum_{r \in \pi_d} fare(r) - \sum_{r \in \pi_d} bid_d^r \end{aligned}$$

for every request r and driver d, bid_d^r is the difference between the profit d can make after accepting r (γ_r^d) and the profit d can make before accepting r (γ_{r-}^d). Therefore:

$$income(\pi_d, \hat{\theta}_d) \geq \sum_{r \in \pi_d} fare(r) - \sum_{r \in \pi_d} (\gamma_r^d - \gamma_{r-}^d)$$

Assuming $\langle r_1, r_2, \cdots, r_n \rangle$ shows the order in which driver d accepts the request we can say $\gamma_{r_i-}^d = \gamma_{r_{i-1}}^d$ and hence:

$$income(\pi_d, \hat{\theta}_d) \geq \sum_{r \in \pi_d} fare(r) - (\gamma_{r_n}^d - \gamma_{r_1-}^d)$$

Before accepting the first request, the driver cannot generate any profit (i.e., $\gamma_{r_1-}^d = 0$). Furthermore, the profit each driver generates for the platform is equal to the difference the total collected fares and the cost of that driver:

$$\gamma_{r_n}^d = \sum_{r \in \pi_d} fare(r) - cost(\pi_d, \hat{\theta}_d)$$

Therefore:

$$\begin{aligned} income(\pi_d, \hat{\theta}_d) &\geq \sum_{r \in \pi_d} fare(r) - \left(\sum_{r \in \pi_d} fare(r) - cost(\pi_d, \hat{\theta}_d) \right) \\ &\geq cost(\pi_d, \hat{\theta}_d) \end{aligned}$$

In other words, the driver's income is always at least as much as his costs which implies the utility is always non-negative and the platform is individually rational. □

Intuitively, by adopting a first-price auction scheme (i.e., for every request r, $\rho_r = bid_{d_{opt}}^r$ where d_{opt} is the driver with the highest bid), the platform can maximize its revenue while remaining individually rational. However, computing the profit of a schedule depends on the driver's reported profile. Consequently, a driver's reported profile can eventually affect the driver's bid. The disadvantage of setting $\rho_r = bid_{d_{opt}}^r$ is that drivers would have the incentive to lower their bids by misreporting their profiles and hence, the framework will not be truthful. Theorem 5.4 shows by adopting a

second-price auction scheme, the drivers do not have an incentive to misreport their profiles.

One of the key features of the second-price auction scheme is truthfulness. In Theorem 5.4 we show how truthfulness is guaranteed in APART by adopting the second price-auction scheme.

Theorem 5.4. *If for every request r assigned to driver d, ρ_r is equivalent to the value of the second highest bid, then the platform is truthful.*

Proof. We assume d_2 is the driver with the second highest bid and $bid^r_{d_2}$ is his corresponding bid. We show that the winning driver d_{opt} cannot increase his utility by misreporting his profile and either increasing or decreasing his bid. We show d_{opt}'s bid based on his actual profile as $bid^r_{d_{opt}}$ and his bid based on a misreported profile as $\overline{bid}^r_{d_{opt}}$.

Case 1: $bid^r_{d_2} < bid^r_{d_{opt}} < \overline{bid}^r_{d_{opt}}$**:** In this case d_{opt} will have the highest bid so he will be selected as the winner and his payment will be $bid^r_{d_2}$.

Case 2: $bid^r_{d_2} < \overline{bid}^r_{d_{opt}} < bid^r_{d_{opt}}$**:** Similar to the first case, here d_{opt} will have the highest bid and will be assigned the request. The second highest bid is still from d_2 and hence, d_{opt} will still pay $bid^r_{d_2}$ to the platform provider.

Case 3: $\overline{bid}^r_{d_{opt}} < bid^r_{d_2} < bid^r_{d_{opt}}$**:** In this case, d_{opt} will no longer have the highest bid and r will not be assigned to him. Therefore, both his *cost* and *payment* will be 0 and there will be no change to his $utility$.

Consequently, in all three cases, d_{opt} cannot increase his utility by misreporting his profile and thus, the framework is truthful. □

With $\rho_r > 0$ for every request r, we can guarantee that the platform is *budget balanced.*

Lemma 5.5. *If for every request r, $\rho_r > 0$, then the ride-sharing platform is budget balanced.*

Asking the drivers to pay an equivalent amount to the second highest bid takes away any incentives for the drivers to misreport their profiles. However, if there is only one driver who can fit the request in his current schedule and thus there will be no second highest bid and the driver will not pay anything to the platform. To avoid situations like this, the platform can set a *reserved price* for every request.

Definition 5.6 (Reserved Price). For every request r, the reserved price bid^r_{server}, is a hidden minimum price the platform sets for the payments it expects from the winning driver.

The server treats the reserved price as another bid. If there is no other bid higher than the reserved price, the server does not assign the request to any driver. Otherwise, the reserved price guarantees the second highest bid is not 0. With APART, for every request, the reserved price is the difference between the requests default fare and the cost of the most expensive driver servicing that request:

$$\forall r,\ bid^r_{server} = F(w_r) - \theta^*(w_r) \quad \text{and} \quad \theta^* = \arg\max_{\theta} \{\theta(w_r) \mid \theta \in \Theta\}$$

where Θ is the set of all possible profiles for the drivers.

6 EXPERIMENTS

6.1 Dataset

We evaluate our algorithms using one month (May, 2013) of New York City's taxi dataset [2], which contains 39,437 drivers and around 500,000 trips per day. Each ride in the dataset has a pick-up latitude/longitude, a drop-off latitude/longitude and request time. We extracted the road network of New York City from Open Street Map (OSM), which is represented as an undirected graph with 55,957 vertices and 78,597 edges. Subsequently, we mapped the source and destination of each trip to the road network. Similar to [9], we maintain a cache for shortest paths between vertices, which means that the shortest path can be found in constant time. Initially, each driver is randomly located on one vertex of the road network. When the vehicle is serving rider requests, we assume it is following the schedule and constantly moving towards the destination.

6.2 Experimental Methodology

To evaluate **LSTM**'s prediction precision, we compare its performance in predicting the transition model to that of **LORE** [23]. LORE models transition probabilities based on their previous locations using Additive Markov Chains.

In our evaluation we used the k-fold cross validation (k=4) method to divide our data into test and train sets and for each fold compared the models built with LSTM and LORE with the actual transition probabilities for the test data. To compare the transition probabilities of the test data to those from the output of the model, we used the *Kullback-Leibler divergence (KLD)* metric [12]. The reason for choosing KLD is that unlike other widely used metrics (e.g. the Jensen-Shanon divergence [18]), KLD is not symmetric and is best used when measuring how a probability distribution (model output) diverges from an expected probability distribution (test data). With KLD, we measure the divergence of probability distribution Q from P as:

$$KLD\,(P\|Q) = \sum_i P(i) \log \frac{P(i)}{Q(i)}$$

Based on the definition of KLD, the lower the KLD score is, the closer Q is to P where a KLD score of 0 means that Q and P are equal.

Subsequently, we compare the proposed second-price auction with reserved price scheme (**SPARP**) with the regular second-price auction scheme (**SPA**) and the first-price auctions scheme where drivers are not truthful and bid competitively (**FPACB**). We compare the generated revenue in different pricing mechanisms and also evaluate the effects of bidding competitively on the workers utility in a first-price auction scheme.

Table 1 shows the different values we used for various parameters we used in our experiments (default values are shown in **bold**).

For the pricing model, the default configurations are:

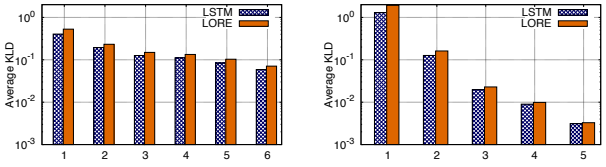
$$F(\phi_r) = 2 \times \phi_r$$
$$\forall r, \lambda_r(\Delta_r) = 1 - \Delta_r^2$$
$$\forall d, \theta_d(l) = 1.5 * l$$

Parameter	Values
Grid Size (km)	1, **2**, 3, 4, 5
Time Slot Size (hour)	1, **2**, 3, 4, 5, 6
Max Wait Time (min)	3, **6**, 9, 12, 15, 20
# of Drivers	1000, 2000, **5000**, 10000, 20000
Max Passengers	2, 3, **4**, 5, 6
Max Allowed Detour	25%, **50%**, 75%, 100%

Table 1: Parameters for Algorithm Comparison

6.3 Prediction Model

In the first set of experiments, we compared the prediction precision of LSTM to that of LORE. Figures 2(a) and 2(b) depict the KLD scores of LSTM and LORE for varying time slots and cell sizes, respectively. The reason LSTM predicts transition probabilities better than LORE is because different topics can capture transition patterns that are observed in the training data. To better illustrate the improvements of LSTM over LORE, in Figure 3 we did a pairwise comparison between the KLD scores of LSTM and LORE. Each bar shows the percentage of improvement we observed for each (*Cell Size*, *Time Slot Size*) pair. As the number of regions and time slots increase (i.e., smaller cell and time slot sizes), more patterns can be observed in the data and yielding better accuracy for LSTM.



(a) Varying Time Slot Size (h) (b) Varying Cell Size (km)

Figure 2: Comparing Precision of LSTM Vs LORE

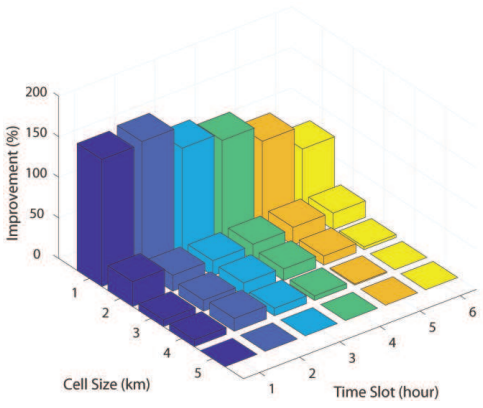
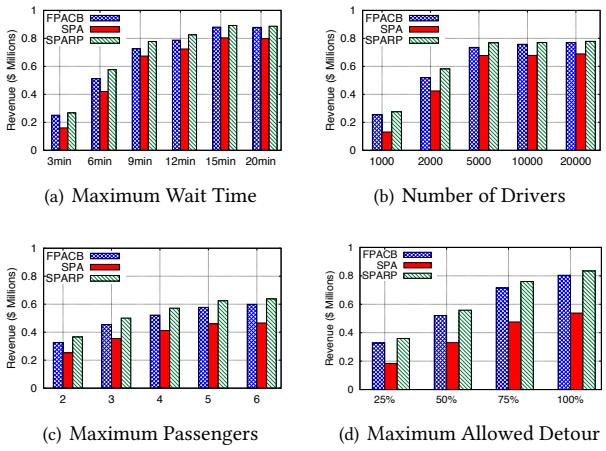


Figure 3: LSTM Vs LORE - Varying Cell & Time Slot Size

6.4 Pricing Mechanism

In the next set of experiments, we compare the generated revenue of each pricing mechanism. Towards this end, we compute the fares and cost of the drivers based on Equations (15) and (16). We compare the generated revenue under different scenarios by varying the parameters in Table 1.

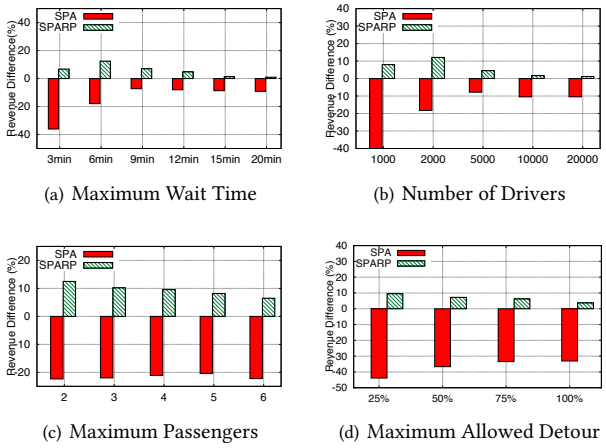
As depicted in Figure 4, SPA generates less revenue than FPACB since the platform will only receive a profit for each request that's equal to the second highest bid. However, by introducing the *reserved price* in SPARP, we manage the losses in SPA. Furthermore, because the drivers do not bid truthfully in FPACB, in most cases, SPARP slightly generates more revenue than FPACB. To better show the difference between the revenue of each mechanism under different configurations, in Figure 5 we show the relative difference of both SPA and SPARP when compared to FPACB.



(a) Maximum Wait Time (b) Number of Drivers

(c) Maximum Passengers (d) Maximum Allowed Detour

Figure 4: Comparing Revenue of Different Mechanisms



(a) Maximum Wait Time (b) Number of Drivers

(c) Maximum Passengers (d) Maximum Allowed Detour

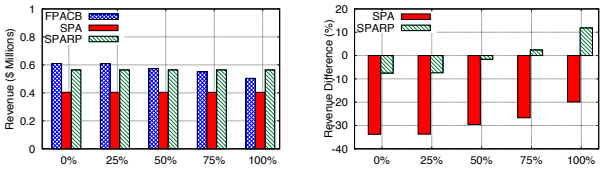
Figure 5: Comparing Relative Revenue of SPA & SPARP Vs FPACB

In our next experiments, we changed the ratio of the drivers who bid untruthfully. As illustrated in Figure 6, when no driver tries to

		Untruthful Drivers (%)		
		25%	50%	75%
Truthful	Assigned Requests (Average)	14.33	14.19	13.04
	Assigned Requests (Median)	8	6	9
	Income per Mile (Average)	$1.00	$1.00	$1.00
	Income per Mile (Median)	$1.00	$1.00	$1.00
Untruthful	Assigned Request (Average)	13.51	13.49	12.67
	Assigned Requests (Median)	7	6	8
	Income per Mile (Average)	$1.24	$1.25	$1.24
	Income per Mile (Median)	$1.21	$1.23	$1.23

Table 2: Effects of Untruthful Bidding

manipulate the framework, FPACB generates more profit as compared to both SPA and SPARP (in this case there is no competitive bidding so FPACB works as the normal first-price auction scheme). However, as the percentage of untruthful drivers increases, the platform makes less revenue and when every driver is bidding competitively in FPACB, we notice that SPARP generates almost 10% more revenue. In Section 3 we proved that in theory, drivers can increase their utility by bidding untruthfully. To show this in practice, we observe in Table 2 that when drivers bid untruthfully, their average and median income per mile increases 20-25%. At the same time, bidding untruthfully does not cost them losing many requests. We can see in Table 2 that the assigned requests for untruthful drivers is on average only 1 request less than truthful bidders.



(a) Untruthfull Drivers (b) Untruthfull Drivers

Figure 6: Comparing Relative Revenue of SPA & SPARP Vs FPACB

The last set of our experiments focuses on how the accuracy of the prediction model can affect the utility of the drivers. For this experiment we compared the drivers' utilities when they predicted the number of drivers using LSTM with when they used LORE. To better explain the observed results, we distinguish between the cases where LORE overestimates and when it underestimates the number of drivers. As depicted in in Table 3, when drivers overestimate, it causes them to take less risk in bidding. However, while this does not increase their chances of winning an auction, it does affect their average income per mile. With LSTM the drivers earn ~25% more by bidding competitively while when they overestimate the number of drivers, they only gain ~15% more than a driver who bids truthfully. On the other hand, when the drivers underestimate, their income per mile is up to 10% higher as compared to using LSTM. However, this is the result of bidding much lower than their true valuation and as such they do not win many auctions. As shown in Table 3, when drivers underestimate, they win about 50% less auctions than when they predict more accurately.

		Untruthful Drivers (%)		
		25%	50%	75%
LSTM	Assigned Request (Average)	13.51	13.49	12.67
	Assigned Requests (Median)	7	6	8
	Income per Mile (Average)	$1.24	$1.25	$1.24
	Income per Mile (Median)	$1.21	$1.23	$1.23
Over Est.	Assigned Requests (Average)	14.01	13.98	12.91
	Assigned Requests (Median)	7	6	8
	Income per Mile (Average)	$1.14	$1.14	$1.15
	Income per Mile (Median)	$1.11	$1.12	$1.11
Under Est.	Assigned Request (Average)	7.51	7.39	8.81
	Assigned Requests (Median)	5	4	5
	Income per Mile (Average)	$1.34	$1.33	$1.37
	Income per Mile (Median)	$1.32	$1.31	$1.32

Table 3: Effects of Prediction Accuracy

7 RELATED WORK

Following, we review the related works in two separate categories.

7.1 Location Prediction

With the rapid growth of GPS enabled devices and the availability of fine-grained location data, many location prediction approaches have been studied in the past few years. This research includes location recommendation systems [17, 20–23], human mobility modeling [6, 7, 14], etc.

In location recommendation systems, the goal is to predict the user's next check-in location based on their social network. In [21] and [22], the next location prediction is enhanced by incorporating the spatial and temporal correlation of users' data, respectively. In [17], in addition to spatiotemporal features, several social and behavioral features are proposed and the effect of each feature on predicting the user's next location is studied separately. Furthermore, in [20, 23] the sequence of users' check-ins are exploited in order to predict the next check-in location. Compared with these studies, rather than studying an *individual's* next check-in *venue*, our model predicts the transition pattern of the *population* between different *regions*. In our work, instead of having personalized trajectories, we only know the point to point trips of anonymous individuals so it is not possible to look at the movement pattern of a single user. Furthermore, previous studies focus on the correlation between the types of venues visited by each user. This requires all venues visited by different users to be labeled appropriately in advance. Unlike the *supervised* nature of these prediction approaches, our model generation process is completely *unsupervised*.

With regard to human mobility, previous studies model the distribution of geolocal events (i.e., check-ins, taxi bookings, etc.) over space and time. Lichman et al. [14] predict the spatial distribution of check-in data by interpolating the spatial distributions of both the individuals and the population. In [7], separate spatial and temporal Gaussian components are exploited to model check-in data. On the other hand, Chiang et al. [6], propose a unified Gaussian mixture model to predict spatiotemporal dynamics of taxi bookings. Similar techniques can be used with our model to predict the number of *new* drivers that are added to the platform in each region at any

point in time. However, another aspect of our model is to predict the number of drivers who have already been in the platform and *relocated* to a certain location. This requires the transition pattern of users within the framework which has not been studied in previous approaches.

7.2 Mechanism design for ride-sharing

Several mechanisms have been introduced to promote ride-sharing in traditional peer-to-peer carpooling platforms [5, 10, 11, 19, 24]. These mechanisms assume passengers have valuations for each driver and assign passengers to drivers based on this valuation. Most previous studies assume all the information regarding passengers and drivers are known a priori and process the ride requests in batch [5, 10, 11, 24]. These mechanism do not work in on-line environments and cannot provide an immediate response to ride requests, which is a key requirement of the current commercial ride-sharing platforms. One exception is the mechanism proposed in [19] that does generate on-line assignments. However, it assumes the drivers are autonomous and comply with any assignment made by the platform and thus the drivers' incentives are ignored. Furthermore, none of these mechanisms consider the platform providers revenue in their pricing model.

8 CONCLUSION AND FUTURE WORK

In this paper we presented an on-line truthful and individually rational pricing mechanism for ride-sharing platforms. We presented a latent space transition model in ride-sharing platforms where the drivers can use to predict the number of available drivers at different locations at each point in time. Subsequently, we showed how the drivers can exploit this information to manipulate their bids to increase their own income and cause the platform to lose profit. Next, we introduced a pricing mechanism based on the second-price auction scheme that is both individually rational and truthful. We also showed how the platform can set a reserved price for each request and manage the losses in revenue which happens in the typical second-price auction scheme.

In the future, we plan to expand the framework by dynamically pricing each ride request where the default fare of a request does not only depend on the shortest path for that request, but also depends on other factors such as drivers' availability. The platform can then provide higher incentives to drivers for accepting less popular requests which allows balancing the supply and demand in different regions. Another interesting direction is to predict the final fare of every request and present it to the requester at the time the request is submitted.

ACKNOWLEDGEMENT

This research has been funded in in part by NSF grants IIS-1320149 and CNS-1461963, METRANS Transportation Center under grants from Caltrans (DTRT12-G-UTC57), and the USC Integrated Media Systems Center. Any opinions, findings and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of any of the sponsors.

REFERENCES

[1] It's a beautiful (pool) day in the neighborhood. https://www.uber.com/blog/los-angeles/its-a-beautiful-pool-day-in-the-neighborhood/. Accessed: 2017-03-30.

[2] Nyc taxi trips. http://www.nyc.gov/html/tlc/html/about/trip_record_data.shtml. Accessed: 2017-03-30.

[3] Asghari, M., Deng, D., Shahabi, C., Demiryurek, U., and Li, Y. Price-aware real-time ride-sharing at scale: An auction-based approach. In *Proceedings of the 24th ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems* (New York, NY, USA, 2016), GIS '16, ACM, pp. 3:1–3:10.

[4] Blei, D. M., Ng, A. Y., and Jordan, M. I. Latent dirichlet allocation. *J. Mach. Learn. Res. 3* (Mar. 2003), 993–1022.

[5] Cheng, S.-F., Nguyen, D. T., and Lau, H. C. Mechanisms for arranging ride sharing and fare splitting for last-mile travel demands. In *Proceedings of the 2014 International Conference on Autonomous Agents and Multi-agent Systems* (Richland, SC, 2014), AAMAS '14, International Foundation for Autonomous Agents and Multiagent Systems, pp. 1505–1506.

[6] Chiang, M.-F., Hoang, T.-A., and Lim, E.-P. Where are the passengers?: A grid-based gaussian mixture model for taxi bookings. In *Proceedings of the 23rd SIGSPATIAL International Conference on Advances in Geographic Information Systems* (New York, NY, USA, 2015), SIGSPATIAL '15, ACM, pp. 32:1–32:10.

[7] Cho, E., Myers, S. A., and Leskovec, J. Friendship and mobility: User movement in location-based social networks. In *Proceedings of the 17th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining* (2011), KDD '11, pp. 1082–1090.

[8] Cici, B., Markopoulou, A., and Laoutaris, N. Designing an on-line ride-sharing system. In *Proceedings of the 23rd SIGSPATIAL International Conference on Advances in Geographic Information Systems* (2015), GIS '15, pp. 60:1–60:4.

[9] Huang, Y., Bastani, F., Jin, R., and Wang, X. S. Large scale real-time ridesharing with service guarantee on road networks. *Proceedings of the VLDB Endowment 7*, 14 (2014), 2017–2028.

[10] Kamar, E., and Horvitz, E. Collaboration and shared plans in the open world: Studies of ridesharing. In *Proceedings of the 21st International Jont Conference on Artifical Intelligence* (San Francisco, CA, USA, 2009), IJCAI'09, Morgan Kaufmann Publishers Inc., pp. 187–194.

[11] Kleiner, A., Nebel, B., and Ziparo, V. A. A mechanism for dynamic ride sharing based on parallel auctions. In *Proceedings of the Twenty-Second International Joint Conference on Artificial Intelligence - Volume Volume One* (2011), IJCAI'11, AAAI Press, pp. 266–272.

[12] Kullback, S., and Leibler, R. A. On information and sufficiency. *Ann. Math. Statist. 22*, 1 (03 1951), 79–86.

[13] Lagoudakis, M., Berhault, M., Koenig, S., Keskinocak, P., and Kleywegt, A. Simple auctions with performance guarantees for multi-robot task allocation. In *Intelligent Robots and Systems, 2004. (IROS 2004). Proceedings. 2004 IEEE/RSJ International Conference on* (Sept 2004), vol. 1, pp. 698–705 vol.1.

[14] Lichman, M., and Smyth, P. Modeling human location data with mixtures of kernel densities. In *Proceedings of the 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining* (2014), KDD '14, pp. 35–44.

[15] Ma, S., Zheng, Y., and Wolfson, O. Real-time city-scale taxi ridesharing. *IEEE Transactions on Knowledge and Data Engineering 27*, 7 (2015), 1782–1795.

[16] Mehta, A., Saberi, A., Vazirani, U., and Vazirani, V. Adwords and generalized on-line matching. In *Foundations of Computer Science, 2005. FOCS 2005. 46th Annual IEEE Symposium on* (Oct 2005), pp. 264–273.

[17] Noulas, A., Scellato, S., Lathia, N., and Mascolo, C. Mining user mobility features for next place prediction in location-based services. In *Proceedings of the 2012 IEEE 12th International Conference on Data Mining* (2012), ICDM '12, pp. 1038–1043.

[18] Rubner, Y., Tomasi, C., and Guibas, L. J. A metric for distributions with applications to image databases. In *Sixth International Conference on Computer Vision (IEEE Cat. No.98CH36271)* (Jan 1998), pp. 59–66.

[19] Shen, W., Lopes, C. V., and Crandall, J. W. An online mechanism for ridesharing in autonomous mobility-on-demand systems. In *Proceedings of the Twenty-Fifth International Joint Conference on Artificial Intelligence* (2016), IJCAI'16, AAAI Press, pp. 475–481.

[20] Wang, W., Yin, H., Sadiq, S., Chen, L., Xie, M., and Zhou, X. Spore: A sequential personalized spatial item recommender system. In *2016 IEEE 32nd International Conference on Data Engineering (ICDE)* (May 2016), pp. 954–965.

[21] Yin, H., Sun, Y., Cui, B., Hu, Z., and Chen, L. Lcars: A location-content-aware recommender system. In *Proceedings of the 19th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining* (2013), KDD '13, pp. 221–229.

[22] Yuan, Q., Cong, G., Ma, Z., Sun, A., and Thalmann, N. M. Time-aware point-of-interest recommendation. In *Proceedings of the 36th International ACM SIGIR Conference on Research and Development in Information Retrieval* (2013), SIGIR '13, pp. 363–372.

[23] Zhang, J.-D., Chow, C.-Y., and Li, Y. Lore: Exploiting sequential influence for location recommendations. In *Proceedings of the 22Nd ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems* (2014), SIGSPATIAL '14, pp. 103–112.

[24] Zhao, D., Ramchurn, S. D., and Jennings, N. R. Incentive design for ridesharing with uncertainty. In *arXiv preprint arXiv:1505.01617* (2015).