A Taxi Order Dispatch Model based On Combinatorial Optimization

Lingyu Zhang, Tao Hu, Yue Min, Guobin Wu
Junying Zhang, Pengcheng Feng, Pinghua Gong, Jieping Ye
Didi Research Institute, Didi Chuxing
Beijing, China 100085
{zhanglingyu,hutao,minyueluna,wuguobin,zhangjunying,fengpengcheng,gongpinghua,yejieping}@didichuxing.com

ABSTRACT

Taxi-booking apps have been very popular all over the world as they provide convenience such as fast response time to the users. The key component of a taxi-booking app is the dispatch system which aims to provide optimal matches between drivers and riders. Traditional dispatch systems sequentially dispatch taxis to riders and aim to maximize the driver acceptance rate for each individual order. However, the traditional systems may lead to a low global success rate, which degrades the rider experience when using the app. In this paper, we propose a novel system that attempts to optimally dispatch taxis to serve multiple bookings. The proposed system aims to maximize the global success rate, thus it optimizes the overall travel efficiency, leading to enhanced user experience. To further enhance users' experience, we also propose a method to predict destinations of a user once the taxi-booking APP is started. The proposed method employs the Bayesian framework to model the distribution of a user's destination based on his/her travel histories.

We use rigorous A/B tests to compare our new taxi dispatch method with state-of-the-art models using data collected in Beijing. Experimental results show that the proposed method is significantly better than other state-of-the art models in terms of global success rate (increased from 80% to 84%). Moreover, we have also achieved significant improvement on other metrics such as user's waiting-time and pick-up distance. For our destination prediction algorithm, we show that our proposed model is superior to the baseline model by improving the top-3 accuracy from 89% to 93%. The proposed taxi dispatch and destination prediction algorithms are both deployed in our online systems and serve tens of millions of users everyday.

CCS CONCEPTS

•Information systems →Mobile information processing systems;

KEYWORDS

Taxi dispatch; destination prediction; combinatorial optimization; circular distribution

1 INTRODUCTION

Recent advances in GPS and 4G networks have made mobile taxi APPs more and more popular. These APPs collect a large amount of individual trajectories on a daily basis. The collected data provide us an unprecedented opportunity to automatically discover knowledge on user behavior, which can be used to build real time intelligent decision making systems in different applications, such as passenger finding [7, 12, 13, 24], taxi demand predicting [16, 17], route planning [14, 22] and taxi order dispatching [8, 11].

The quality of order dispatching can directly influence the user experience of riders as well as taxi operating efficiency. Thus, how to dispatch orders efficiently is a central task. Some previous work [3, 10] on order dispatching focused on how to find a nearest driver or a shortest-travel-time driver for each individual order. When an order comes in, such a system chooses one of the nearest drivers, without judging whether these drivers were more suitable for other orders. Therefore these methods cannot guarantee the global shortest-travel-time for all orders. The authors in [19] proposed a novel model, based on a multi-agent architecture called NTuCab. In order to minimize the waiting-time or the pick-up distance globally, this model considers each agent as a computation unit. Each computation unit processes N order/driver pairs and each order is dispatched to only one driver. An order will be dispatched to another driver if the matched driver does not accept it.

A drawback of the methods mentioned above is the long dispatch time and low success rate, because the methods do not optimize the total success rate. At Didi Chuxing[1], millions of drivers provide transportation services for over ten million passengers every day. In rush hours, Didi Chuxing needs to match over a hundred thousand passengers to drivers every second. Therefore the total success rate of these orders becomes the main metric to evaluate the performance of the underlying order dispatch system.

In this paper, we propose a novel combinatorial optimization model to solve the order dispatch problem at Didi Chuxing. In this model, we dispatch one order to several drivers with the goal of maximizing the total success rate of these orders. When multiple drivers receive the same order, the first one to accept gets the order. If an order is not accepted, it enters the next round of dispatching until it is accepted or canceled.

[1] http://www.didichuxing.com

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than the author(s) must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org.
KDD'17, August 13–17, 2017, Halifax, NS, Canada.
© 2017 Copyright held by the owner/author(s). Publication rights licensed to ACM.
ISBN 978-1-4503-4887-4/17/08…$15.00
DOI: http://dx.doi.org/10.1145/3097983.3098138

An order in the dispatching system typically has three important elements - departure time, origin and destination of the order. Generally, we simply choose the current time and rider location as the departure time and departure place of an order, which is suitable for most taxi-calling situations. However, different people tend to go to very different destinations. Even for the same person, the destinations may be different at different departure times and locations. However, it brings extra work for the users to enter the full name of the destination. Therefore, the user experience can be greatly enhanced if the intended destination can be accurately predicted when a user opens the APP. Most of the current methods use the trajectory data of all users to train a model, and then use the information of the current trip, combined with some auxiliary information, such as time, traffic condition etc., to predict the intended destination [9, 21, 23].

A typical approach was given in [20], which proposed a neural network based on multi-layer perceptions that takes input data such as the user's initial trajectory data and other meta-data such as driver id, user information, travel departure time, etc. After training, it is possible to predict most of the itineraries. However, the model does not rely much on problem specific information that can be derived from the data. It heavily depends on the initial continuous trajectory data of the itinerary. Once this part of the data is discarded, the prediction accuracy is greatly reduced. This was also shown in the experiments of the paper. The work in [4] employed the driver's own historical trip data to train an HMM model. The model divides the time of day into a few subjective segments such as morning, noon, and evening, and this treatment destroys the continuity of time.

Previous approaches are not applicable for the destination prediction problem at Didi Chuxing, because the prediction is required as soon as a passenger opens the APP, but the trajectory data of the current trip cannot be obtained immediately. Moreover, unlike most previous methods that try to minimize the distance between the predicted destination and the actual destination, we aim to identity the exact destination that the user wants to go. In fact, even if the predicted destination is an alias of the true destination, it is highly likely that the passenger will regard it as an unfamiliar location, and simply inputs the address manually. Therefore, our system takes the set of each user's historical travel destinations as the candidate set for destination prediction. Personal historical trip statistics is very different among different people. Take year 2015 as an example, the annual taxi booking usage per user is about twenty for users who have used Didi Chuxing at least once. However, it is unevenly distributed as high-frequency users opened the APP daily while low-frequency users opened the APP less than ten times in one year. It is a major challenge to derive accurate personal statistics from such sparse data.

By analyzing taxi booking behavior of a huge amount of passengers, we obtain some interesting observations. Based on theses observations, we propose a Bayesian destination prediction model, which considers departure longitude, latitude and departure time under the ternary Gaussian distribution. The model is trained using personal historical trip data. The trained model can calculate the probability of users' historical destinations based on departure longitude, latitude and departure time. It then provides a list of the predicted destinations ranked by the probability.

The rest of this paper is as follows: Section 2 presents the taxi dispatch system and Section 3 introduces the destination prediction system, Section 4 provides experiments to show the effectiveness of the two models and we conclude this paper in Section 5.

2 ORDER DISPATCH SYSTEM

We first introduce some notations. The goal of our order dispatch system is to maximize the success rate, denoted as E_{SR}.

If there are N orders to be dispatched to M drivers, we represent the dispatch result as a matrix

$$\begin{pmatrix} a_{11} & \cdots & a_{1M} \\ \vdots & a_{ij} & \vdots \\ a_{N1} & \cdots & a_{NM} \end{pmatrix}, \text{where } 1 \le i \le N, 1 \le j \le M,$$

$$\text{and } a_{ij} = \begin{cases} 1 & \text{order } i \text{ is dispatched to driver } j, \\ 0 & \text{order } i \text{ is not dispatched to driver } j. \end{cases}$$

In this scenario, a driver receives only one order at each round, while one order can be dispatched to several drivers. This imposes the following constraint: $\forall j, \sum_{i=1}^{N} a_{ij} \le 1$.

In Didi Chuxing's business scenario, an order is dispatched to a number of drivers, and each driver decides whether or not to accept it according to his or her own preference. For each order, whether it is accepted by one of the drivers is directly related to each driver's probability of acceptance. Thus, the key problem for order dispatching is to estimate the probability of each driver's acceptance of an order. If we can estimate the matrix with its elements indicating the probability of each driver accepting each order, then we can estimate the probability of an order to be accepted by one of the drivers. Therefore, we divide the order dispatch model into two sub-models. One model predicts each driver's action, in which we estimate the probability of a driver accepting an order. Another model formulates an optimization problem for maximizing the target E_{SR} using the estimated acceptance probabilities, and then solves the underlying optimization problem.

2.1 The model of Driver's Action Prediction

Since the driver's action takes two values: accept or reject (not accept), we use a 0-1 valued binary variable y to denote the action outcome, where 1 stands for accept and 0 stands for reject. We assume that the driver's action is subject to an independent and identical probability distribution.

We use p_{ij} to denote the probability of order o_i accepted by driver d_j. The probability depends on many factors, such as the monetary value of the order, the driving distance and direction, etc. Such information can be encoded into a feature vector $\mathbf{x}_{ij}$, which is associated with the order o_i, the driver d_j, and the interactions between them. Given $\mathbf{x}_{ij}$, we would like to estimate the acceptance probability as follows:

$$p_{ij} = p(y = 1|\mathbf{x}_{ij}).$$

This formulates the problem of predicting driver's action as a typical binary classification problem, and the classifier can be trained using features produced from historical driver-order pairs $\langle o_i, d_j \rangle$ with the associated outcomes.

Table 1: Results of LR and GBDT.

Beijing				
Model	ACC	AUC	Sensitivity	Specificity
LR	0.7822	0.8680	0.7372	0.8257
GBDT	0.7571	0.8549	0.8579	0.6378
Shanghai				
Model	ACC	AUC	Sensitivity	Specificity
LR	0.7632	0.8470	0.7332	0.7933
GBDT	0.7443	0.8438	0.8444	0.6258

In this work, we tried two popular models: linear logistic regression (LR) [5] and gradient boosted decision tree (GBDT) [6, 15]. We train separate models for different cities and evaluate both methods in terms of Accuracy (ACC) and Area under the Curve of ROC (AUC). Experimental results for Beijing and Shanghai are shown in Table 1.

We note that both LR and GBDT are widely used. Our experiments show that for our data, LR is slightly more accurate. Therefore, we choose LR as the prediction model in our system, where the probability p_{ij} can be written as

$$p_{ij} = p(y = 1|o_i, d_j) = \frac{1}{\exp(-\mathbf{w^T}\mathbf{x}_{ij})}. \quad (1)$$

Our system uses SGD (Stochastic Gradient Descent) to train the model parameters [2, 25] . The prediction model considers various factors, which can be summarized as follows:

- Order-Driver related features: the pick-up distance, the broadcasting counts of the order to the driver, whether the order is in front of or behind the driver's current driving direction.
- Order related features: the distance and the estimated time arrival (ETA) between the origin and the destination, the destination category (airport, hospital, school, business district, etc.), traffic situation in the route, historical order frequency at the destination.
- Driver related features: Long-term behaviors (include historical acceptance rate of a driver, active locations of a driver, preference of different broadcast distances of a driver, etc.) and short-term interests of a driver such as orders recently accepted or not, etc.
- Supplemental features, such as day of the week, hour of the day, number of drivers and orders nearby.

2.2 The Combinatorial Optimization model

In our system, one order can be dispatched to several drivers, thus all these drivers contribute to the probability of order acceptance. Assume that there are N orders to be dispatched to M drivers, then the probability of an order o_i to be accepted is:

$$E_i = 1 - \prod_{j=1}^{M}(1 - p_{ij})^{a_{ij}}, \quad (2)$$

where p_{ij} is defined in Eq. (1)

$$\text{and } a_{ij} = \begin{cases} 1 & \text{order } i \text{ is dispatched to driver } j, \\ 0 & \text{order } i \text{ is not dispatched to driver } j. \end{cases}$$

Based on the notations above, we can formulate the success rate E_{SR} as

$$E_{SR} = \frac{\sum_{i=1}^{N}[1 - \prod_{j=1}^{M}(1 - p_{ij})^{a_{ij}}]}{N}, \quad (3)$$

As we mention before, there is a constraint in our model:

$$\forall j, \sum_{i=1}^{N} a_{ij} \leq 1,$$

which means that each driver can receive at most one order at a time, leading to the following order dispatch problem:

$$\begin{cases} \max\limits_{a_{ij}} E_{SR} = \dfrac{\sum_{i=1}^{N}[1 - \prod_{j=1}^{M}(1 - p_{ij})^{a_{ij}}]}{N}, \\ \text{s.t. } \forall j, \ \sum\limits_{i=1}^{N} a_{ij} \leq 1, a_{ij} \in \{0, 1\}. \end{cases}$$

This is a constrained combinatorial optimization problem [18]. We next show how to solve this problem.

Many combinatorial optimization problems are NP hard, and there is no efficient general algorithm to solve this class of problems in polynomial time. A typical approach is to use a heuristic algorithm to find an approximate solution. Commonly used methods include hill-climbing methods, genetic algorithms, simulated annealing algorithms, etc. By balancing the accuracy and the performance, we choose a hill-climbing method to solve the problem. Algorithm 1 below describes the detailed procedure.

Algorithm 1 Proposed HillClimbing Algorithm

```
procedure HillClimbing(A, P)
    for i ← 1, M do
        D[i] ← j with the maximum probability P[i][j]
    end for
    for i ← 1, N do
        E[i] ← success rate of order i with D
        E0 ← average(E[i])
    end for
    for i ← 1, N do
        U ← drivers that are not assigned order i
        for j ← 1, len(U) do
            k ← U(j)
            if replace D[k] with order i, E0 increases then
                replace D[k] with order i
                E[i] ← success rate of order i with D
                E0 ← average(E[i])
            end if
        end for
    end for
end procedure
```

3 DESTINATION PREDICTION

This section describes our proposed destination prediction system. By analyzing a large number of Didi Chuxing users' taxi booking

Table 2: Notations used in this paper.

Symbol	Meaning	Range
T	time of day	[0, 24)
D	Day of the week	workday, holiday
Lng	Departure Longitude	[-180, 180]
Lat	Departure Latitude	[-90, 90]
Y	Description	Set of User Historic Destinations, $\{y_1,y_2,...,y_i,...,y_n\}$

behaviors, we have discovered some interesting patterns: (1) The same user tends to go to the same destination at similar times. Specifically, the departure time (time of day) is the most important factor for predicting a user's intended destination, followed by the departure latitude and longitude. Interestingly, the date variable (workday or holiday) can separate the data into two groups with different characteristics: workday destinations are concentrated on home and workplace; holiday destinations are concentrated on shopping centers, and entertainment places, etc. (2) The same user tends to go to a fixed set of locations even for shopping in the weekends, except for occasional emergencies such as doctor appointments, business travel, etc. (3) The order's location provides useful information for destination prediction. Other information such as the driver information, traffic situation, driving speed, etc. have weak correlations with the destination.

Based on the above observations, we propose to model the probability distribution of a user's destination using Bayesian rule, in which the user's historical data such as departure time, departure latitude and longitude are utilized.

3.1 Model Description

We first introduce some notations in Table 2.

We observe that D has only two categories which can be used to divide the user's historical data into two clusters with very different behaviors, so we preprocess the data by clustering using the feature D. In addition, we use each order's date in the historical data as a timeliness factor to weight the importance of training data (We omit the details here due to the limited space of the paper). The aim of our proposed method is to model the probability distribution of a user's destination. To achieve this goal, we utilize Bayesian formula to express the conditional probability of the destinations $\{y_1,y_2,...,y_i,...,y_n\}$, as follows:

$$p(Y = y_i|X) = \frac{p(X|Y = y_i)p(Y = y_i)}{\sum_{j=1}^{n} p(X|Y = y_j)p(Y = y_j)},$$

where $X = (T, Lng, Lat)$ denotes the departure time, departure longitude and latitude.

We can estimate $p(Y = y_i)$ using the user's historical trip data as follows:

$$p(Y = y_i) = \frac{\text{freq}(y_i)}{\sum_{j=1}^{n} \text{freq}(y_j)}.$$

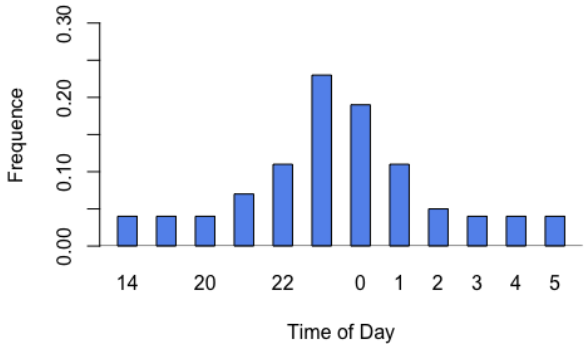

Figure 1: Departure time histogram of Sanlitun (a place in Beijing).

The remaining issue is to estimate the joint distribution of the departure time and location (longitude and latitude) given the destination, that is: $p(T, Lng, Lat|Y = y_i)$. We next show how to achieve this goal step by step. We first study the conditional probability distribution $p(T|Y = y_i)$ of departure time T, given the destination y_i. We will then include longitude and latitude to obtain a joint conditional probability distribution for (T, Lng, Lat), given the destination y_i.

3.2 Conditional distribution of departure time given destination

Figure 1 shows the histogram of a sample user's departure time corresponding to the destination Sanlitun. The shape of the histogram is very similar to that of a Gaussian distribution. By studying many other users' cases, we have similar observations. Therefore, we use Gaussian distribution to estimate the conditional probability of the departure time T:

$$T|Y = y_i \sim N(\mu_i, \sigma_i^2).$$

We note that variable T takes circular values from hour 0 to 23, and then repeat. Therefore the mean μ_i and variance σ_i^2 can not be estimated using traditional methods. For example, if times are 8:00, 9:00, 10:00, then the average is 9:00, which can be calculated as $(8 + 9 + 10)/3$; but if they are 3:00, 15:00, 21:00, the average is 21:00, not $(3 + 15 + 21)/3 = 13$. We next show how to estimate the mean μ_i and the variance σ_i^2.

Notice that the time variable is a circular quantity. A widely used method to compute the mean of circular quantities is to transform all circular variables into unit vectors, and compute the average of the vectors, then transform the result back to the original circular representation. Given the times $t_1, t_2, ..., t_m$, the mean μ can be expressed as:

$$\mu = \frac{24}{2\pi} \cdot \arctan[\frac{1}{m}\sum_{k=1}^{m} \sin(\frac{2\pi}{24} \cdot t_k), \frac{1}{m}\sum_{k=1}^{m} \cos(\frac{2\pi}{24} \cdot t_k)].$$

This method is simple and intuitive, but in some cases the results obtained deviate from the true value (e.g. mean of the three times 00:00:00, 00:00:00, 03:00:00 is 01:00:00, but the vector mean is 00:58:33). Moreover, in certain cases there are no results (e.g. when $\sum_{k=1}^{m} \sin(\frac{2\pi}{24} \cdot t_k) = 0$ and $\sum_{k=1}^{m} \cos(\frac{2\pi}{24} \cdot t_k) = 0$). To remedy the

above mentioned problems, we propose another method to calculate the mean of departure time, and this method is also applicable to the mean of arbitrary circular numerical variables.

Our method is based on the following observation. The mean of the departure times can be obtained by solving the following constrained quadratic optimization problem:

$$\begin{cases} \min\limits_{\mu} \sum\limits_{k=1}^{m} [\text{distance}(t_k, \mu)]^2, \\ \text{s.t. } \mu \in [0, 24), \end{cases} \tag{4}$$

where $\text{distance}(t_1, t_2)$ denotes the distance of two circular variables t_1 and t_2, which is defined as follows:

$$\text{distance}(t_1, t_2) = \begin{cases} |t_1 - t_2| & \text{if } |t_1 - t_2| \le 12, \\ 24 - |t_1 - t_2| & \text{if } |t_1 - t_2| > 12. \end{cases}$$

More concisely, we rewrite the above formula as

$$\text{distance}(t_1, t_2) = -|(|t_1 - t_2| - 12)| + 12.$$

Substituting the above equation into Eq. (4), we obtain

$$\begin{cases} \min\limits_{\mu} \sum\limits_{k=1}^{m} [|(|t_k - \mu| - 12)| - 12)]^2, \\ \text{s.t. } \mu \in [0, 24). \end{cases}$$

By solving the above optimization problem, we obtain the desired mean μ. Similar ideas can also be applied to estimate the variance:

$$\sigma^2 = \frac{1}{m-1} \sum_{k=1}^{m} [|(|t_k - \mu| - 12)| - 12]^2.$$

3.3 The joint conditional distribution of departure time, departure latitude and longitude for a given destination

For most users, one can obtain highly accurate prediction of intended destination based on departure time alone. Figure 2 shows the probability distribution of the departure time for two different destinations of a passenger. We find that the departure time of the passenger to Zhongguancun is concentrated around 11:00, and the departure time to Zhicunlu is around 19:00. If the user opens the APP in the morning, then there is a high probability of going to Zhongguancun; in the evening, there is a high probability of going to Zhicunlu.

For some other users, it is necessary to integrate other information to obtain high-precision prediction. Figure 3 presents a typical example which shows that the departure time distributions of Digital Manor and Libao Plaza have a considerable amount of overlap. Therefore, for this user, it is impossible to give an accurate prediction of the destinations based on the departure time alone.

Figure 4 shows the three-dimensional distributions of the departure time, departure longitude and latitude (T, Lng, Lat) for the two different destinations Digital Manor and Libao Square of this user. We can see that the three-dimensional distributions can easily distinguish the two different destinations.

In order to find the appropriate model to describe this three-dimensional joint distribution, we analyze the joint distribution scatter plot of the departure time and the departure longitude and

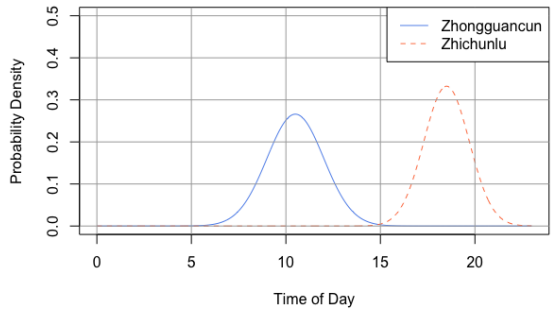

Figure 2: Departure time distributions of Zhongguancun and Zhicunlu (places in Beijing).

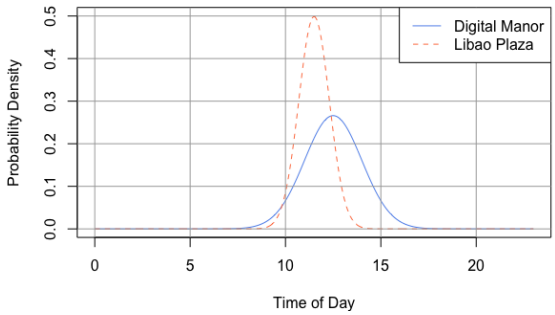

Figure 3: Departure time distributions of Digit Manor and Libao Plaza (places in Beijing).

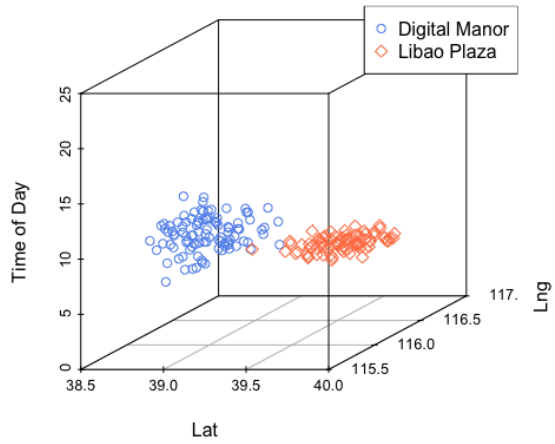

Figure 4: Departure time, longitude and latitude distributions of Digit Manor and Libao Plaza (places in Beijing).

latitude (T, Lng, Lat) for each destination of each user. The distribution is consistent with the shape of a three-dimensional Gaussian distribution. Therefore, we can assume that given a user and an intended destination,

$$Lat, Lng, T | Y = y_i \sim N_3(\boldsymbol{\mu}_i, \Sigma_i). \tag{5}$$

In order to verify this hypothesis, we employ the idea of goodness of fit test from classical statistics for normality test. We still use the user's own historical data to estimate the distribution parameters (mean vector and covariance matrix). It should be noted that,

Lat, Lng, T are all circular quantities.The quadratic programming method described earlier can thus be used to estimate the mean vector $\boldsymbol{\mu}$ and the covariance matrix Σ .

3.4 The complete steps of destination prediction based on departure time, departure latitude and longitude.

The complete steps of destination prediction are:

Step 1. Estimate $\boldsymbol{\mu}_i, \Sigma_i$ for each destination of a user.

Step 2. Calculate $p(Y = y_i)$:

$$p(Y = y_i) = \frac{\text{freq}(y_i))}{\sum_{j=1}^{n} \text{freq}(y_j)},$$

and compute $p(T, Lat, Lng|Y = y_i)$ using Eq. (5).

Setp 3. Calculate $p(Y = y_i|T, Lat, Lng)$ using Bayesian rule:

$$p(Y = y_i|T, Lat, Lng) =$$

$$\frac{p(T, Lat, Lng|Y = y_i)p(Y = y_i)}{\sum_{j=1}^{n} p(T, Lat, Lng|Y = y_j)p(Y = y_j)}.$$

Step 4. Rank the destinations by $p(Y = y_i|T, Lat, Lng)$ and provide them as a list.

4 EXPERIMENT

In this section, we evaluate the proposed order dispatch and destination prediction models. For the order dispatch model, we compare our model with two other methods using multiple evaluation criteria. For the destination prediction model, there is no published methods fitting our task. Thus, we compare our proposed method with a simple method based on KNN which was used in Didi Chuxing's online systems before the proposed system was deployed online.

4.1 Experiments of order dispatch system

We perform experiments to evaluate the order dispatch system based on the following considerations. First, to design the evaluation metrics, we take the general taxi order dispatch business into consideration. We then compare our model with two other models widely used in the industry [10, 19]. We split the data randomly into three random partitions, and compare the performance in terms of the metrics in Table 3. In addition, we study the detailed behaviors of different models and present some interesting observations.

4.1.1 Evaluating order dispatch models. The order dispatch system affects many core metrics, and among these metrics, we choose some key business metrics used in [1], which are similar to what we have used in our system. We list them in Table 3 [1].

4.1.2 Experimental methods. We compare our proposed method with two models that are widely used in the industry: one is to dispatch each order to the driver most suitable for the order [3, 10]. The other is to dispatch a batch of orders to drivers, so that the overall waiting time is minimized [19].

Model 1: [10] proposed a method of computing the shortest driving time for drivers in real-time traffic environment. [3] proposed a method of order dispatch based on the A* algorithm for optimal route planning. Based on these studies, we used a learning-to-rank based method, which uses historical data to obtain a formula to calculate the matching scores of drivers and orders. We then dispatch an order to the driver who has the highest matching score (Note that this method was used at Didi Chuxing until the method of this paper was proposed).

Model 2: [19] proposed a multi-agent based order dispatch system, called NTuCab. The system is superior to most other systems in the sense that it optimizes the overall waiting time.

We conduct the experiments using one day's data in Beijing (the largest city in China). The data contain 1 million orders and 80 thousand drivers. We randomly divide the data evenly into three parts (Flow 1, Flow 2, and Flow 3) based on mobile phone numbers. The number of drivers and the number of orders are summarized in Table 4. We run Model 1, Model 2 and our proposed model in Flow 1, Flow 2 and Flow 3, respectively.

4.1.3 Experimental results and analysis. After running the systems online for one day, we report the following evaluation metrics for the three models in Table 5. The results show that our model is significantly better than the other two in terms of the key evaluation metric SR. On the metric of average-of-pick-up-time (APT), our model is almost identical to model 2 because model 2 tries to optimize APT, while in our model, APT is an important factor in determining SR. On the average-of-dispatch-time (ADT) metric, model 2 is even worse than model 1 because in [19], there is a key constraint on the model 2: each dispatch unit of the model can only handle the same number of drivers and orders. This implies that many orders or drivers can only wait for the next round of dispatch, leading to a longer dispatch-time.

Table 3: Measurements used in our comparison.

Measurement	Abbrev.
Percentage of served calls (Success Rate)	SR
Averaged pick up time(the time from the order accepted by a driver to the passenger getting on the car)	APT
Averaged dispatch time (the time from the passenger make the order to one driver accepted the order)	ADT
Percentage of cancelled calls (Cancellation Rate)	CR
Average total number of calls served by each cab (Fleet Utilization)	FU

Table 4: Number of drivers and orders.

	Driver Number	Order Number
Flow 1	27268	345631
Flow 2	27310	345683
Flow 3	27297	345711

Table 5: Results of three models.

	SR(%)	APT (min)	ADT (sec)	CR(%)	FU
Model 1	79.8	4.538	87	24.1	4.32
Model 2	80.1	4.163	106	22.8	4.47
Our Model	84.4	4.169	89	23.2	4.71

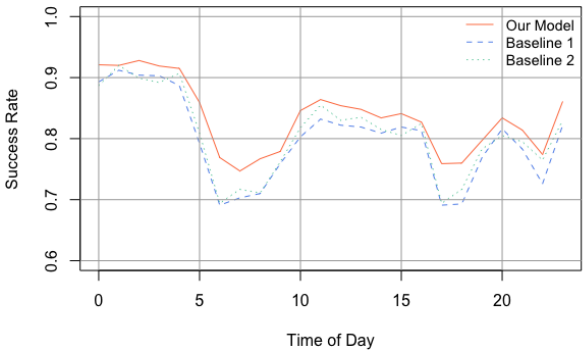

Figure 5: Comparison of success rates of three models.

Table 6: Match score between each order/driver pair.

Driver \ Order	1	2	3	4	5
1	2.85	1.21	1.80	3.27	3.53
2	3.00	3.89	3.88	1.46	4.02
3	3.32	3.91	3.68	2.77	3.94
4	3.83	2.86	1.36	3.75	2.16
5	3.35	3.12	3.22	2.75	1.66
6	3.76	1.61	3.74	3.09	2.12
7	1.07	2.53	1.05	2.17	2.65
8	1.34	2.87	1.72	3.01	3.30
9	2.01	0.97	1.24	0.64	2.33
10	3.35	3.25	3.14	2.18	2.40

Figure 5 plots the success rates of the three models versus time. It is clear that at night (23:00pm to 5:00am the next day), when there are relatively few car bookings, all three models had higher success rates. This is because there are plentiful of free vehicles for relatively few bookings, and with sufficient supplies, it does not matter what order dispatch model to use. With the increase in the number of bookings, especially in the rush hours (7am - 9am and 17pm -19 pm) and evening time after dinner (21pm to 22pm), the success rate is relatively low because the number of car bookings increases significantly, together with a reduction of the actual available free vehicles, resulting in an imbalance of supply and demand. In this case, our model yields a significantly higher success rate than the other two models.

4.1.4 Case Study. The above experimental results show that the success rate of our model is significantly better than those of the other two models. In order to illustrate the differences of the three models, we constructed a synthetic data set containing 10 drivers and 5 orders, and analyze how these three models dispatch the 5 orders to the 10 drivers.

Model 1 first calculates the matching score of all candidate orders for each driver, and then assigns the highest scored order to the driver. As can be seen from Table 6, order (5) achieves the highest scores for drivers (1, 2, 3, 7, 8, 9), while order (1) achieves the highest score for drivers (4, 5, 6, 10). Thus, in accordance with the principle of assigning the highest order for each driver, order (5) will be assigned to the drivers (1, 2, 3, 7, 8, 9) and order (1) will be assigned to the drivers (4, 5, 6, 10). The orders (2, 3, 4) can only be dispatched after the ten drivers have made choices (accept or reject) for orders (1) and (5), leading to relatively low efficiency. This is precisely because this order dispatch model considers each driver and order separately, without taking the overall situation into consideration. The dispatch result for Model 1 is shown in Table 7.

Table 7: Dispatch results for Model 1.

Driver	Dispatched order	Match Score
1	5	3.53
2	5	4.02
3	5	3.94
4	1	3.83
5	1	3.35
6	1	3.76
7	5	2.65
8	5	3.30
9	5	2.33
10	1	3.35

Table 8: Pick-up distance(meter) between each order/driver pair.

Driver \ Order	1	2	3	4	5
1	737	1836	1451	464	381
2	649	166	154	1605	83
3	464	168	290	796	143
4	125	803	1726	167	1238
5	427	648	565	797	1556
6	160	1581	226	569	1252
7	1844	1003	1910	1145	924
8	1700	816	1514	648	542
9	1258	1980	1789	2100	1120
10	415	549	597	1131	1073

Table 9: Dispatch results for Model 2.

Driver	Dispatched Order	Pick-up Distance
1	4	464
2	5	83
3	2	168
4	1	125
5	3	565
6	1	160
7	2	1003
8	4	648
9	5	1258
10	3	597

The dispatch results of Model 2 are shown in Table 9 according to the goal of minimizing the overall pick-up distance (in Table 8). We see that different from model 1, model 2 considers all drivers and orders as a whole, and thus each order can be dispatched to the same number of drivers. Because the model tries to minimize the overall pick-up distance, the total pick-up distance of the model's output is the shortest among all three models.

Table 10: Accept probability between each order/driver pair.

Driver \ Order	1	2	3	4	5
1	0.019	0.007	0.01	0.025	0.029
2	0.021	0.037	0.037	0.008	0.04
3	0.026	0.037	0.032	0.018	0.038
4	0.035	0.019	0.007	0.034	0.012
5	0.026	0.023	0.024	0.018	0.009
6	0.034	0.009	0.034	0.022	0.012
7	0.006	0.016	0.006	0.013	0.017
8	0.007	0.019	0.009	0.021	0.025
9	0.011	0.006	0.007	0.005	0.014
10	0.026	0.025	0.023	0.013	0.014

Table 11: Dispatch results for the proposed model.

Driver	Dispatched Order	Accept Probability
1	5	0.029
2	5	0.040
3	2	0.037
4	4	0.034
5	1	0.026
6	3	0.034
7	2	0.016
8	5	0.025
9	5	0.014
10	1	0.026

In the proposed model, we first estimate the probability of each driver accepting each order by Eq. (1) (see Table 10 for detailed results). Then, according to the acceptance probability matrix, we dispatch orders to drivers such that the overall success rate is maximized by following Eq. (3) (See detailed results in Table 11).

Next, we analyze the differences between the expected results for each order and the overall success probability from the three models. We use Eq. (2) and the acceptation probability in Table 10 to estimate the success probability of each order in the three models (results are shown in Table 12). For model 1, we see that the orders (1, 5) have higher acceptance probabilities, and the orders (2, 3, 4) are not sent to any driver, resulting in a probability of 0. Thus the overall probability of success is low, (0.116 + 0 + 0 + 0 + 0.138) / 5 = 5.08%. Model 2 considers the orders and drivers as a whole, and optimizes the overall pick-up distances to obtain the dispatch outcome. However, minimizing the pick-up distance is not exactly equivalent to maximizing the success rate, since the driver's acceptance of an order depends not only on the pick-up distance but also on other factors such as order revenue, etc. Its overall success probability is (0.068 + 0.052 + 0.046 + 0.045 + 0.053) / 5=5.28%. For our model, the success rate is the optimization criterion, and the dispatch result achieves the highest success probability of (0.051 + 0.052 + 0.034 + 0.034 + 0.104) / 5 = 5.5%.

It should be noted that the success probabilities above are significantly different from the success rates in Table 5. The success rates in Table 5 are the actual results produced by different models from the online system at Didi Chuxing, while the success probabilities above are the simulation results of different models on the same data (include only five orders and ten drivers), jointly considering the estimated driver acceptance probability.

Table 12: Success probability of each order by three models.

Model \ Order	1	2	3	4	5
Model 1	0.116	0	0	0	0.138
Model 2	0.068	0.052	0.046	0.045	0.053
Proposed Model	0.051	0.052	0.034	0.034	0.104

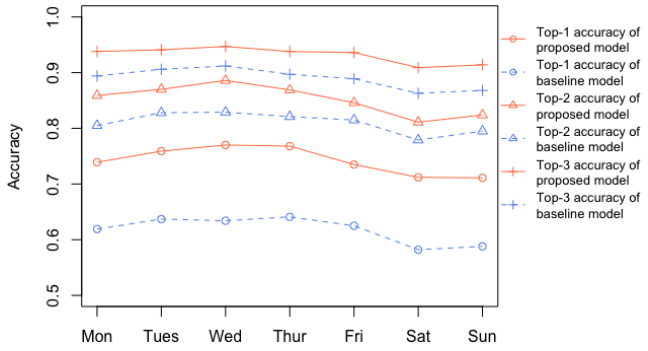

Figure 6: Top-1, top-2 and top-3 accuracies of our proposed model and the baseline model.

4.2 Experiments for destination prediction

As mentioned before, most of the existing destination prediction models were based on some meta-data and the ongoing trip itinerary data. However, at Didi Chuxing the destination prediction happens before a passenger creates an order. Specifically, the prediction system is called as soon as the Didi Chuxing APP is started. At that time, only the passenger's current location, current time, passenger ID and other meta-data can be obtained. Thus, existing methods [4, 9, 20, 21, 23] are not applicable to the scenario at Didi Chuxing.

Before the proposed destination prediction model was deployed in our online system, there is a baseline model used in Didi Chuxing online system, which employs the K-nearest neighbor method (K=100) based on two features: departure time and location. We train both models on 3 months' taxi data in Beijing and compare our proposed model with the baseline model in terms of top-1, top-2 and top-3 accuracies. Experimental results on the test data set including data for one week are presented in Figure 6 . We observe that our proposed model outperforms the baseline model by a large margin.

We can observe from the figure that our proposed destination prediction model achieves a high accuracy about 93%, which is 4% higher than the baseline model on top-3 accuracy. The performance is reasonable because for any user, there will always be some unpredictable events such as seeing a doctor, travel, etc., moreover, there are many new users and for whom the historical destination statistics is not sufficiently high.

We further analyze the results from different days of the week and find the following patterns: (1) The prediction performance on weekends (Saturdays and Sundays) is lower than that on work days

(Monday to Friday). This is because on work days, user's travel patterns are relatively regular, mostly between work and home; on weekends, travel patterns will be irregular, such as shopping, entertainment etc. (2) On work days, the highest accuracy is achieved on Wednesday, while accuracy is relatively low on Monday and Friday. From the data, we found that for many users, private events and other irregular travels occurred on Friday night and Monday morning. (3) On weekends, Saturday's trip is slightly more uncertain relative to Sunday. Our statistics shows that many personal travels, such as visiting relatives and friends, happen on Saturday mornings. The return trips are often on Sunday afternoons. For these users, the Saturday travel destination is relatively random (friends and relatives, transit stations, etc.), and Sunday's destination is relatively focused (home).

5 CONCLUSION AND FUTURE WORK

In this paper, we have proposed a novel order dispatch model which has been deployed in the online system at Didi Chuxing. The proposed model aims to maximize the global success rate and thus optimizes the overall traffic efficiency and delivers the best user experience. We formulate the order dispatch model as a combinatorial optimization problem, in which a key ingredient is to estimate the probability of a driver accepting an order. We employ logistic regression to estimate the probability. To further enhance users' experience, we develop a destination prediction model to predict a destination list when a user starts the app. We formulate the problem using the Bayesian framework based on users' historical data including departure time, latitude and longitude of the departure place. Experimental results produced from Didi Chuxing App show that our proposed order dispatch model is significantly better than the state-of-the-art models in terms of the success rate which is the most important metric in the order dispatch system. Moreover, the proposed model performs much better in terms of many other metrics such as users' waiting-time, pick-up distance, averaged dispatching time, cancellation rate and fleet utilization. Experimental results also show that our destination prediction method is superior to the baseline model.

In the future, we will further investigate some interesting problems in the following aspects: (1) The non-convex problem in the proposed order dispatch model makes it difficult to find a globally optimal solution. We plan to identity a convex surrogate and develop a fast optimization algorithm to solve the corresponding optimization problem. (2) We will further improve the destination prediction model such that the model is able to discover a new destination accurately even if the true destination has never appeared in the user's historical data.

REFERENCES

[1] Aamena Alshamsi, Sherief Abdallah, and Iyad Rahwan. 2009. Multiagent self-organization for a taxi dispatch system. In *8th international conference on autonomous agents and multiagent systems*. Citeseer, 21–28.

[2] Léon Bottou. 2010. Large-scale machine learning with stochastic gradient descent. In *Proceedings of COMPSTAT'2010*. Springer, 177–186.

[3] Lee Chean Chung. 2005. *GPS taxi dispatch system based on A* shortest path algorithm*. Ph.D. Dissertation. Masterfis thesis, Submitted to the Department of Transportation and Logistics at Malausia University of Science and Technology (MUST) in partial fulfillment of the requirements for the degree of Master of Science in Transportation and Logistics.

[4] Alexandre de Brébisson, Étienne Simon, Alex Auvolat, Pascal Vincent, and Yoshua Bengio. 2015. Artificial neural networks applied to taxi destination prediction. *arXiv preprint arXiv:1508.00021* (2015).

[5] Jerome Friedman, Trevor Hastie, and Robert Tibshirani. 2001. *The elements of statistical learning*. Vol. 1. Springer series in statistics Springer, Berlin.

[6] Jerome H Friedman. 2001. Greedy function approximation: a gradient boosting machine. *Annals of statistics* (2001), 1189–1232.

[7] Yong Ge, Hui Xiong, Alexander Tuzhilin, Keli Xiao, Marco Gruteser, and Michael Pazzani. 2010. An energy-efficient mobile recommender system. In *Proceedings of the 16th ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM, 899–908.

[8] Andrey Glaschenko, Anton Ivaschenko, George Rzevski, and Petr Skobelev. 2009. Multi-agent real time scheduling system for taxi companies. In *8th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2009), Budapest, Hungary*. 29–36.

[9] Brendan Guillouet, Loubes Jean-Michel, Philippe Besse, and Royer François. 2016. Destination Prediction by Trajectory Distribution Based Model. (2016).

[10] Der-Horng Lee, Hao Wang, Ruey Cheu, and Siew Teo. 2004. Taxi dispatch system based on current demands and real-time traffic conditions. *Transportation Research Record: Journal of the Transportation Research Board* 1882 (2004), 193–200.

[11] Junghoon Lee, Gyung-Leen Park, Hanil Kim, Young-Kyu Yang, Pankoo Kim, and Sang-Wook Kim. 2007. A telematics service system based on the Linux cluster. In *International Conference on Computational Science*. Springer, 660–667.

[12] Junghoon Lee, Inhye Shin, and Gyung-Leen Park. 2008. Analysis of the passenger pick-up pattern for taxi location recommendation. In *Networked Computing and Advanced Information Management, 2008. NCM'08. Fourth International Conference on*, Vol. 1. IEEE, 199–204.

[13] Bin Li, Daqing Zhang, Lin Sun, Chao Chen, Shijian Li, Guande Qi, and Qiang Yang. 2011. Hunting or waiting? Discovering passenger-finding strategies from a large-scale real-world taxi dataset. In *Pervasive Computing and Communications Workshops (PERCOM Workshops), 2011 IEEE International Conference on*. IEEE, 63–68.

[14] Qingquan Li, Zhe Zeng, Bisheng Yang, and Tong Zhang. 2009. Hierarchical route planning based on taxi GPS-trajectories. In *Geoinformatics, 2009 17th International Conference on*. IEEE, 1–5.

[15] Llew Mason, Jonathan Baxter, Peter L Bartlett, and Marcus R Frean. 1999. Boosting Algorithms as Gradient Descent.. In *NIPS*. 512–518.

[16] Luis Moreira-Matias, Joao Gama, Michel Ferreira, Joao Mendes-Moreira, and Luis Damas. 2013. On predicting the taxi-passenger demand: A real-time approach. In *Portuguese Conference on Artificial Intelligence*. Springer, 54–65.

[17] Luis Moreira-Matias, Joao Gama, Michel Ferreira, Joao Mendes-Moreira, and Luis Damas. 2013. Predicting taxi–passenger demand using streaming data. *IEEE Transactions on Intelligent Transportation Systems* 14, 3 (2013), 1393–1402.

[18] Christos H Papadimitriou and Kenneth Steiglitz. 1982. *Combinatorial optimization: algorithms and complexity*. Courier Corporation.

[19] Kiam Tian Seow, Nam Hai Dang, and Der-Horng Lee. 2010. A collaborative multiagent taxi-dispatch system. *IEEE Transactions on Automation Science and Engineering* 7, 3 (2010), 607–616.

[20] Reid Simmons, Brett Browning, Yilu Zhang, and Varsha Sadekar. 2006. Learning to predict driver route and destination intent. In *Intelligent Transportation Systems Conference, 2006. ITSC'06. IEEE*. IEEE, 127–132.

[21] Kohei Tanaka, Yasue Kishino, Tsutomu Terada, and Shojiro Nishio. 2009. A destination prediction method using driving contexts and trajectory for car navigation systems. In *Proceedings of the 2009 ACM symposium on Applied Computing*. ACM, 190–195.

[22] Tang Xin-min, Wang Yu-ting, and Han Song-chen. 2010. Aircraft Taxi Route Planning for A-SMGCS Based on Discrete Event Dynamic System modeling. In *Computer Modeling and Simulation, 2010. ICCMS'10. Second International Conference on*, Vol. 1. IEEE, 224–228.

[23] Andy Yuan Xue, Rui Zhang, Yu Zheng, Xing Xie, Jin Huang, and Zhenghua Xu. 2013. Destination prediction by sub-trajectory synthesis and privacy protection against such prediction. In *Data Engineering (ICDE), 2013 IEEE 29th International Conference on*. IEEE, 254–265.

[24] Jing Yuan, Yu Zheng, Liuhang Zhang, XIng Xie, and Guangzhong Sun. 2011. Where to find my next passenger. In *Proceedings of the 13th international conference on Ubiquitous computing*. ACM, 109–118.

[25] Tong Zhang. 2004. Solving large scale linear prediction problems using stochastic gradient descent algorithms. In *Proceedings of the twenty-first international conference on Machine learning*. ACM, 116.