Improving the Information Disclosure in Mobility-on-Demand Systems

Yue Yang
yyan2437@uni.sydney.edu.au
University of Sydney
Sydney, Australia

Yuan Shi
shiyuan404@gmail.com
Intelligent Logistics Group, Huolala
Shenzhen, PR. China

Dejian Wang
wdj5433@gmail.com
Intelligent Logistics Group, Huolala
Shenzhen, PR. China

Qisheng Chen
qc2261@columbia.edu
Intelligent Logistics Group, Huolala
Shenzhen, PR. China

Lei Xu
xulei@sribd.cn
Shenzhen Research Institute of Big Data
Shenzhen, PR. China

Hanqian Li
saberleo2016@gmail.com
Intelligent Logistics Group, Huolala
Shenzhen, PR. China

Zhouyu Fu
joey.fu@huolala.cn
Intelligent Logistics Group, Huolala
Shenzhen, PR. China

Xin Li
xin2.li@huolala.cn
Intelligent Logistics Group, Huolala
Shenzhen, PR. China

Hao Zhang
jason.hao.zhang@huolala.cn
Intelligent Logistics Group, Huolala
Shenzhen, PR. China

ABSTRACT

Nowadays, the ubiquity of sharing economy and the booming of ride-sharing services prompt Mobility-on-Demand (MoD) platforms to explore and develop new business modes. Different from forcing full-time drivers to serve the dispatched orders, these modes usually aim to attract part-time drivers to share their vehicles and employ a 'driver-choose-order' pattern by displaying a sequence of orders to drivers as a candidate set. A key issue here is to determine which orders should be displayed to each driver. In this work, we propose a novel framework to tackle this issue, known as the Information Disclosure problem in MoD systems. The problem is solved in two steps combining estimation with optimization: 1) in the estimation step, we investigate the drivers' choice behavior and estimate the probability of choosing an order or ignoring the displayed candidate set. 2) in the optimization step, we transform the problem into determining the optimal edge configuration in a bipartite graph, then we develop a Minimal-Loss Edge Cutting (MLEC) algorithm to solve it. Through extensive experiments on both the simulation and the real-world data from Huolala business, the proposed method remarkably improves users experience and platform efficiency. Based on these promising results, the proposed framework has been successfully deployed in the real-world MoD system in Huolala.

KDD '21, August 14–18, 2021, Virtual Event, Singapore

ACM ISBN 978-1-4503-8332-5/21/08…$15.00
https://doi.org/10.1145/3447548.3467062

CCS CONCEPTS

• **Applied computing** → **Transportation**; • **Computing methodologies** → *Multi-agent planning*.

KEYWORDS

Mobility On-Demand; Information Disclosure; Dynamic Planning

ACM Reference Format:
Yue Yang, Yuan Shi, Dejian Wang, Qisheng Chen, Lei Xu, Hanqian Li, Zhouyu Fu, Xin Li, and Hao Zhang. 2021. Improving the Information Disclosure in Mobility-on-Demand Systems. In *Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery and Data Mining (KDD '21), August 14–18, 2021, Virtual Event, Singapore.* ACM, New York, NY, USA, 11 pages. https://doi.org/10.1145/3447548.3467062

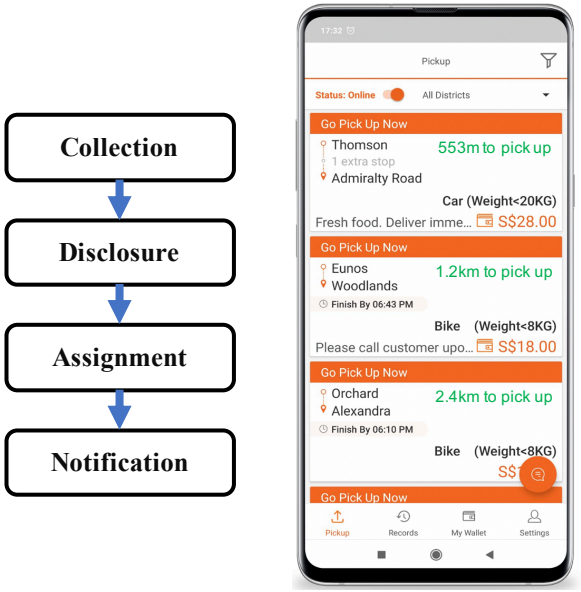
1 INTRODUCTION

The rapid growth of internet-based Mobility-On-demand (MoD) platforms such as Uber, Lyft, and Didi Chuxing brought significant impacts and radical revolutions upon transportation systems in both urban and suburban areas. It is reported that Uber served 5.22 billion trips worldwide in 2018, up from 140 million trips in 2014 [21], and Didi also provided services for over 25 million trips on each day of 2018 in China [24]. The primary goal of these platforms is to deliver value for users and improve service efficiency. Thus, several critical modules such as order dispatching [24, 27], fleet management [14, 25], and demand predicting [9, 28] should be developed in building MoD systems.

In a traditional setting of MoD systems, all the drivers work as full-time employees of the platform and should follow the order dispatching instructions given by the platform. However, the prosperity of sharing economy and ride-sharing services leads to a quiet shift of the role of MoD platforms [1]. To take great use of vehicle capacity, some MoD platforms begin to launch new service modes -such as UberPool, Didi Piggy, Huolala Smart Freight- for recruiting part-time drivers to fulfill the overwhelming trip demands. In these modes, part-time drivers only serve for their self-interest and are

reluctant to follow the mandatory dispatching instructions from the platform. Instead, the platform often employs a 'driver-choose-order' pattern to connect these drivers and orders. As depicted in Figure 1.A, the basic procedure of these modes takes the following steps:

(1) **Collection:** The platform collects all idle drivers, recently arriving orders, and unserved orders in a given time horizon.
(2) **Disclosure:** With displaying prices, origins, destinations, and pick-up distances, the orders collected in the previous step continuously pop up on the screen of drivers' app (as shown in Figure 1.B). If drivers are interested in serving an order, they can click the order on the screen, and then their willingness will be collected by the platform. In particular, they can select at most one order at a time and then just wait for assignment result from the platform.
(3) **Assignment:** If an order receives willingness from more than one driver, the platform ranks these drivers by pick-up distance (other rules can also be adopted for ranking purposes), and then assigns the order to its first-rank driver.
(4) **Notification:** The platform notifies the drivers whether they are successfully matched. The matched drivers will follow the platform's routing guide to serve the assigned orders, while the unmatched drivers will wait in the system till the next time horizon to continually hunt for desired orders.



(A) (B)

Figure 1: We summarize a basic procedure of 'driver-choose-order' pattern in MoD systems in (A), and we also offer an interaction design of displaying a sequence of orders for a driver in (B)

Aiming to enhance users experience for both drivers and passengers and to boost platform efficiency, a critical question needs to be addressed in designing the 'driver-choose-order' pattern: which orders should be displayed to each driver? This question is well known as the Information Disclosure problem [6, 20] in MoD systems, and there are two threads of approaches widely used in both research works and industrial applications:

- **Sequential Recommendation:** Sequential recommendation aims at predicting the driver's next clicked order from the historical clicking sequence. Early works on sequential recommendation were mainly based on Markov Chain (MC) assumption to model sequential dynamics [10, 11, 19]. Thanks to the recent advance of the recurrent neural network (RNN) [5] in natural language processing (NLP), there were many works equipped with these same powerful techniques to achieve significant breakthroughs in sequential recommendation problems [12, 26]. Other deep learning models, -i.e. the convolutional neural network, the self-attention network, were also adopted in sequential recommendation tasks and achieved excellent performance [12, 22].
- **Combinatorial Optimization:** To coordinate multi-driver decisions and optimize a global objective, combinatorial optimization algorithms are commonly leveraged in MoD systems. The matching problems between orders and drivers often fall into the category of bipartite graph matching, in which lots of well-known combinatorial optimization methods have been developed in both academic researches [2, 16, 29] and industrial applications [13, 24, 30]. In general form of these methods, deterministic costs/rewards between orders and drivers were generated from mathematics definition (i.e. Pick-up distance in [29]), machine learning (i.e. Conversion Rate in [30]), or reinforcement learning (i.e. Advantage Function in [24]) and then fed in combinatorial optimization approaches (i.e. Kuhn-Munkres (KM) algorithm [18], and Column Generation algorithm [8]).

However, there are still two key challenges associated with the Information Disclosure problem in MoD systems:

(1) **Competition among multiple drivers:** Information Disclosure in MoD systems is quite different from traditional recommendation engines where the same item can be recommended as many times. For example, in TikTok recommendation, the same video can be recommended to an infinite number of users. In our setting, the exposure of an order should be limited by a finite number of drivers since the order can be assigned to only one driver. Consequently, traditional sequential recommendation methods are not perfectly suitable for our problem because they overlook the competition and interaction among multiple drivers.
(2) **Coupling between choosing probability and disclosure solution:** Driver's preference or willingness to an order, quantified by the probability of the driver chooses the order (we call it 'choosing probability'), is not a fixed value and will be strongly affected by the set of orders displayed to the driver. As a result, choosing probability is coupled with the solution of the Information Disclosure problem, and combinatorial optimization approaches are not tractable: it is impossible to set choosing probability as a fixed cost/reward term for the optimization problem.

To address these challenges and design an approach that can be successfully adopted in the real-world MoD platform, we propose a novel Information Disclosure method that optimizes users experience and platform efficiency simultaneously. Our major contributions are listed as follows:

- We propose the Nested Multinomial Logit (NMNL) model to predict choosing probability of a driver chooses an order. The model considers both of scenarios for the driver that selecting an order and ignoring the displayed candidate set.
- By modeling Information Disclosure as a many-to-many matching problem, we transform it into determining the optimal edge configuration in a bipartite graph. To dynamically capture the change of choosing probability and efficiently achieve the approximately optimal solution, we develop a Minimal-Loss Edge Cutting (MLEC) algorithm and demonstrate its efficiency and optimality in large-scale cases.
- We consider several practical issues such as model evaluation and real-world implementation of the proposed method. In the simulator, our proposed optimization method significantly outperforms several benchmarks with regard to users experience and platform efficiency. Concurrently, the proposed framework has been deployed in the real-world MoD system and demonstrates to improve platform efficiency (up to 4.8 p.p. increase in response rate), platform profit (up to 6.7% improvement in revenue), and driver efficiency (up to 11.3 p.p. increase in occupied rate) in three major cities of China. To our best knowledge, this is the first industry-level solution for Information Disclosure problem in MoD systems.

The remainder of the paper is structured as follows. Section 2 describes an estimation step to predict drivers' choice behavior that considers both selecting an order and ignoring choice candidates at a same time. After the estimation step, we propose an optimization step in Section 3 by introducing an edge configuration problem in a bipartite graph and designing a Minimal-Loss Edge Cutting algorithm to solve it. We evaluate the performance of the proposed approach and provide quantitative observations in Section 4 and then close the paper with some conclusive insights in Section 5.

2 ESTIMATION

The estimation step aims to understand drivers' behavior and predict drivers' choices among their displayed orders. From the perspective of the MoD platform, driver d makes a choice from a candidate set of orders $O_d = \{o_1, ..., o_i, ..., o_n\}$ appearing on his app screen. Alternatively, the driver can select a 'No-choice' option, denoted by s, in anticipating upcoming orders in the next round of displaying. The choice-making behavior of driver d can be illustrated in Figure 2.

2.1 Nested Multinomial Logit Model

To model the drivers' choice behavior by specifying choosing probability for each option, we first introduce the Multinomial Logit (MNL) model [15] widely used in economics. In general, the random utility model (RUM) [17] is the theoretical foundation of the MNL model, and decision-makers in the MNL aim to maximize the utility relative to their choices. Given a choice set $O_d \cup \{s\}$, utility between

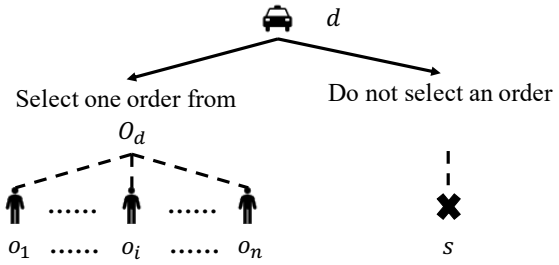


Figure 2: Driver's discrete choices

driver d and choice $o, \forall o \in O_d$ is assumed to be partitioned into two components: a deterministic utility, $U_{o,d}$, and a random component, $\epsilon_{o,d}$, which captures the uncertainty coming from the MNL model:

$$\tilde{U}_{o,d} = U_{o,d} + \epsilon_{o,d}, \forall o \in O_d \cup \{s\}. \tag{1}$$

With the display of several attributes including fare, origin, destination and pick-up distance, driver d faces a sequence of orders on his app screen and then makes a choice among these orders. For simplicity, we assume the deterministic utility $U_{o,d}$ for every order o in O_d is linear-in-parameter:

$$U_{o,d} = \beta_0 + \beta_1 \cdot f_o + \beta_2 \cdot \tau_{o,d}, \forall o \in O_d \tag{2}$$

where f_o represents o's fare (including the information of travel distance and travel time), $\tau_{o,d}$ indicates the pick-up distance from d's location to o's origin, and β_0, β_1, β_2 are preference parameters to be estimated.

In the general assumption of MNL, $\epsilon_{o,d}$ is independent and identically distributed (I.I.D) and follows the Gumbel distribution. A standard MNL model also holds the assumption of Independence from Irrelevant Alternatives (I.I.A) that choosing probability for an arbitrary item is independent of all other items contained in the choice set. However, such a restrictive assumption is inadequate for our problem because whether or not driver d chooses order o in O_d can be strongly affected by other orders in O_d.

To tackle this issue, the Nested Multinomial Logit (NMNL) model [23] is introduced to relax the independence assumption by partitioning the choice set into two mutually exclusive nests (subsets) O_d and $\{s\}$, then we can detail the definition of NMNL below:

- **Choosing probability** denotes the probability that item $o, \forall o \in O_d \cup \{s\}$ is chosen by driver d:

$$p_{o,d} = \begin{cases} p(o|O_d) \cdot p(O_d) & \text{If } o \in O_d \\ p(s|\{s\}) \cdot p(\{s\}) & \text{If } o = s \end{cases}. \tag{3}$$

- **Nest-choosing probability** denotes the probability of choosing nest O_d or $\{s\}$ for driver d:

$$p(O_d) = \frac{exp(\alpha \cdot V_{O_d})}{exp(U_d^0) + exp(\alpha \cdot V_{O_d})} \tag{4}$$

$$p(\{s\}) = \frac{exp(U_d^0)}{exp(U_d^0) + exp(\alpha \cdot V_{O_d})} \tag{5}$$

where U_d^0 is the initial utility expectation of driver d, which can be estimated as the average fare of selected orders for driver d from his historical records.

In addition, α is called the logsum parameter [3] to be estimated and underlies the correlation between the unobserved components for pairs of orders in nest O_d. The value of α is bounded by zero and one to ensure consistency with RUM principles. Specifically, $\alpha = 1$ implies zero correlation among orders in nest O_d, so the NMNL model can be reduced to the standard MNL model.
Besides, V_{O_d} is the inclusive utility for nest O_d:

$$V_{O_d} = ln(\sum_{o}^{O_d} exp(\frac{U_{o,d}}{\alpha})). \tag{6}$$

- **Nest-based choosing probability** denotes the probability that an item is chosen in a given nest O_d or $\{s\}$, which is in a standard softmax formula:

$$p(o|O_d) = \frac{exp(U_{o,d})}{\sum_{o_i}^{O_d} exp(U_{o_i,d})} \tag{7}$$

$$p(s|\{s\}) = 1. \tag{8}$$

The preference parameters β and logsum parameter α are typically estimated by maximizing the log-likelihood function given by:

$$Log\mathcal{L}(y|\alpha,\beta) = \sum_{d\in D}\sum_{o\in O_d\cup\{s\}} y_{o,d} log(p_{o,d}) \tag{9}$$

where $y_{o,d}$ is the observed choice variable (or true label) and is equal to 1 if driver d chooses item o, and to 0 otherwise.

Here, a classical Python tool 'PyLogit' [4] is utilized to perform maximum likelihood estimation of NMNL model, and estimated parameters $\hat{\beta}$ and $\hat{\alpha}$ will be directly leveraged to compute choosing probabilities $p_{o,d}$ to optimize the Information Disclosure problem.

2.2 Discussion

Equation 3 describes the probability that driver d chooses an specific order o in O_d or ignores all the orders displayed to him. However, it is still unclear how this formula reveals the influence of varying choice set brought to the driver's choice. To this end, we take a close look at the form of Equations 4,5,6,7,8:

(1) Whether or not driver d chooses an order in O_d mainly depends on the order with the maximum utility $U_{o,d}$ in O_d.
Proof: Considering the monotonicity of logarithms, the value of V_{O_d} is determined by the order with the maximal utility $U_{o,d}$, and then the value of $p(O_d)$ is also dominated by this order in O_d.
(2) With more orders in choice set O_d, driver d is more likely to select an order in O_d.
Proof: As an order o is added into O_d, the inclusive utility $V_{O_d} \uparrow$, thus, $p(O_d) \uparrow$.
(3) Adding (removing) order o into (from) O_d results in a probable change of the choosing probability $p_{o',d}$ for any other orders o's in O_d.
Proof: When order o is added (removed) into (from) O_d, for any other orders o's in O_d, we can observe that $p(O_d) \uparrow (\downarrow)$ and $p(o'|O_d) \downarrow (\uparrow)$, so $p_{o',d}$ probably changes according to Equation 3.

When making a choice in practice, drivers are often attracted by the most valuable order in their choice sets. Once the highest value exceeds their initial expectation, they prefer to select this order immediately. Meanwhile, as the number of displayed orders increases, they could observe enough information and be aware of the value distribution of the upcoming orders, in this scenario, they also have more chance to select the most valuable order since they are afraid of missing this order. Also, the drivers' preference for a specific order is not cast in stone and will vary from the choice set displayed to them.

3 OPTIMIZATION

The Information Disclosure between multiple drivers and multiple orders can be modeled as a many-to-many matching problem. In this section, the optimization step takes choosing probability $p_{o,d}$ obtained from the estimation step as an input and determines the optimal disclosure matching between drivers and orders.

3.1 Objective Function

A MoD platform may have various goals, depending on what the platform is designed for. A platform running for business purposes may pursue maximizing the number of served orders to increase its profitability. By contrast, a platform running for public service might aim to minimize system-wide vehicle-miles for social welfare, so as to reduce air pollution and relieve traffic congestion. In this paper, we focus on the objective of maximizing the number of orders to be responded, while the proposed algorithm is general in nature and can be modified for fitting other objectives.

Suppose there are two finite sets of unserved orders O and idle drivers D collected in each planning horizon, let $x_{o,d}$ be a binary decision variable that equals 1 if order $o, \forall o \in O$ is displayed to driver $d, \forall d \in D$, and 0 if not. In reality, once order o is chosen by at least one driver, it will surely be assigned to a driver in the following assignment procedure. Thus, we can define the probability $\mathbb{P}_o$ of order o that is responded by at least one driver:

$$\mathbb{P}_o = 1 - \prod_{d\in D}[1 - p_{o,d}(\{x_{o',d}|\forall o' \in O\}) \cdot x_{o,d}] \tag{10}$$

where $p_{o,d}(\{x_{o',d}|\forall o' \in O\})$ (abbreviated as $p_{o,d}$ in the following parts) is the choosing probability that d chooses o if the disclosure solution for d is known as $\{x_{o',d}|\forall o' \in O\}$, and $\prod_{d\in D}[1 - p_{o,d}(\{x_{o',d}|\forall o' \in O\}) \cdot x_{o,d}]$ represents the probability that none of the drivers choose order o.

In this paper, we aim to maximize the number of orders being responded, so the objective of our problem can be formulated to maximize the sum of $\mathbb{P}_o$ over all orders in O:

$$\max \sum_{o\in O} \mathbb{P}_o = \max \sum_{o\in O}[1 - \prod_{d\in D}(1 - p_{o,d}x_{o,d})] \tag{11}$$

3.2 Basic Solutions

To achieve the above goal, we abstract idle drivers $d, \forall d \in D$ and unserved orders $o, \forall o \in O$ as two sets of vertices, and all valid disclosure pairs (if $x_{o,d} = 1, \forall o \in O, \forall d \in D$) as the set of edges with a weight $U_{o,d}$. After that, the optimization problem can be

transformed into determining the optimal edge configuration in the bipartite graph.

Based on the bipartite graph, we can summarize two basic methods widely used in MoD platforms to solve the Information Disclosure problem:

- **Global-Information Disclosure**: An order is displayed to every available driver in D, in turn, a driver can browse every order in O on his screen, so the solution can be represented as $X = \{x_{o,d} = 1 | \forall o \in O, \forall d \in D\}$.
- **Local-Information Disclosure**: This method sets a circle centered at the origin of order with a radius equal to R and only displays the order to the drivers who are in the circle. By introducing this pick-up distance constraint, the solution can be written as $X = \{x_{o,d} = 1 | \tau_{o,d} \leq R; \forall o \in O, \forall d \in D\}$.

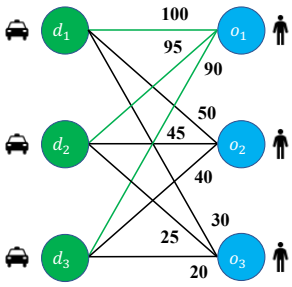
The obvious shortcoming of aforementioned two solutions is that only a few orders can be responded after collecting drivers' choices. Both of strategies are displaying all orders to all qualified drivers. As a result, all drivers have high willingness to serve a few valuable orders and low willingness to serve most cheap orders. In this case, the maximal $p_{o,d}$ for each driver d tends to focus on a few high value orders. From the optimization perspective, both of methods result in a sub-optimal solution for the objective in Equation 11 and undermine platform efficiency to fulfill trip demands.

Figure 3 illustrates such an example for Global-Information Disclosure method. Suppose there is a fully-connected bipartite graph to depict the disclosure relationships between $\{d_1, d_2, d_3\}$ and $\{o_1, o_2, o_3\}$. From the perspective of orders, the edges between drivers $\{d_1, d_2, d_3\}$ and order o_1 have the highest utilities 100, 95, 90, respectively, while the edges between drivers $\{d_1, d_2, d_3\}$ and order o_3 have the lowest utilities 30, 25, 20, respectively. Then the distribution of $p_{o,d}$ for each driver is calculated by the estimation step and shown in the table of Figure 3. Unsurprisingly, all three drivers have the highest willingness to serve order o_1 and the lowest willingness to serve order o_3. As a result, it is highly possible that only o_1 can be responded in this scenario.

3.3 Minimal-Loss Edge Cutting

To deal with the dilemma in above two solutions, an optimization method is needed to avoid the concentration of drivers' willingness and improve the solution for the problem defined in Equation 11. At the same time, it should be noticed that $p_{o,d}$ is not a fixed parameter and changes with the disclosure solution for driver d. As a result, existing methods [7] are unable to solve the unconstrained nonlinear optimization problem in Equation 11. Therefore, given a fully-connected graph $X = \{x_{o,d} = 1 | \forall o \in O, \forall d \in D\}$, we introduce iterative edge cutting algorithms to achieve the best edge configuration in the bipartite graph, and the philosophy of the algorithms is to cut off edges with the maximal gain for our objective function in Equation 11.

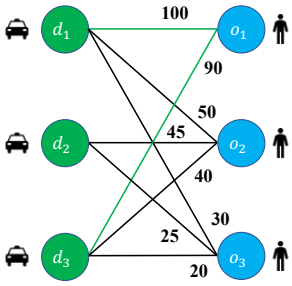
3.3.1 Evaluate the gain of cutting off an edge: Recall the Discussion (3) in Section 2.2, once we cut off the edge between o and d and remove o from driver d's displayed orders O_d, $p_{o',d}$ for every existing o' in O_d should be updated. We can define $\delta_{o',d}$ to capture the change of $p_{o',d}$:



The distribution of $p_{o,d}$ and $\mathbb{P}_o$

	o_1	o_2	o_3	s
d_1	0.990	0.009	0.001	0.000
d_2	0.990	0.008	0.002	0.000
d_3	0.990	0.008	0.002	0.000
$\mathbb{P}_o$	0.999	0.024	0.005	

Figure 3: Example of a Fully-Connected Graph. The estimated parameters of NMNL model is presented in Table 1.



The distribution of $p_{o,d}$ and $\mathbb{P}_o$

	o_1	o_2	o_3	s
d_1	0.990	0.009	0.001	0.000
d_2	0.000	0.980	0.014	0.006
d_3	0.990	0.008	0.002	0.000
$\mathbb{P}_o$	0.999	0.980	0.016	

Figure 4: Example of cutting off the edge between o_1 and d_2 from Figure 3.

$$\delta_{o',d} = p^{\text{update}}_{o',d} - p_{o',d}; \forall o' \in O_d / \{o\} \tag{12}$$

where $p^{\text{update}}_{o',d}$ represents a probability that driver d chooses o' in the new graph without the edge between o and d ($x_{o,d} = 0$).

Furthermore, $\delta_{o',d}$ will lead to a change of $\mathbb{P}_{o'}$ (the detailed derivations are elucidated in Appendix A:

$$\mathbb{P}^{\text{update}}_{o'} - \mathbb{P}_{o'} = \delta_{o',d} \cdot \prod_{d' \in D/\{d\}} (1 - p_{o',d'} x_{o',d'}); \forall o' \in O_d / \{o\} \tag{13}$$

where $\mathbb{P}^{\text{update}}_{o'}$ indicates the updated value of $\mathbb{P}_{o'}$ in the new graph without the edge between o and d ($x_{o,d} = 0$).

On the other hand, $\mathbb{P}_o$ also changes after cutting off the edge between o and d (the detailed derivations are also elucidated in Appendix A:

$$\mathbb{P}^{\text{update}}_{o} - \mathbb{P}_{o} = -p_{o,d} \cdot \prod_{d' \in D/\{d\}} (1 - p_{o,d'} x_{o,d'}). \tag{14}$$

Therefore, we can obtain the overall gain Δ of our objective function when cutting off the edge between o and d:

$$\Delta = \sum_{o' \in O_d / \{o\}} (\mathbb{P}^{\text{update}}_{o'} - \mathbb{P}_{o'}) + \mathbb{P}^{\text{update}}_{o} - \mathbb{P}_{o} \tag{15}$$

, and a positive Δ indicates an improvement of our objective function by cutting off the edge between o and d (let $x_{o,d} = 0$).

To explain the impact of cutting off an edge from a more intuitive view, we show an example of removing the edge between o_1 and d_2 from Figure 3 in Figure 4. We can observe that there exists positive increments for p_{o_2,d_2} and p_{o_3,d_2}: $\delta_{o_2,d_2} = 0.972$ and $\delta_{o_3,d_2} = 0.012$. Furthermore, the overall gain of our objective is $\Delta = (0.980 - 0.024) + (0.016 - 0.005) + (0.999 - 0.999) \approx 0.97$. As a consequence, once we cut off the edge between o_1 and d_2 from Figure 3, we achieve nearly 0.97 improvement for our objective function.

3.3.2 Find the 'best' edge and cut it off: To achieve the best edge configuration in the graph, the most straightforward way is to compute Δ for all the edges in the graph, and then cut off the 'best' edge (let $x_{o^*,d^*} = 0$) with the maximal gain Δ^{max} for the next iteration. The above process will be repeated until there is no edge with a positive gain for the objective. We call the above method Intuitive Edge Cutting (IEC) algorithm and detail the procedure in Algorithm 1.

Algorithm 1 Intuitive Edge Cutting

Input: Unserved orders O, idle drivers D.
Output: Information Disclosure solution X^*.
Initialize a graph $X = \{x_{o,d} = 1 | \forall o \in O, \forall d \in D\}$.
Find out the 'best' edge x_{o^*,d^*} with the maximal Δ^{max}.
while $\Delta^{max} > 0$ **do**
  Cut off the edge between o^* and d^* (let $x_{o^*,d^*} = 0$).
  Update $p_{o,d}, \forall o \in O, \forall d \in D$ in the graph.
  Find out the 'best' edge x_{o^*,d^*} with the maximal Δ^{max}.
end while
$X^* \leftarrow X$
return X^*

Suppose there are N orders and M drivers in the graph, the time complexity of the above solution in the worst case is $O(M^2 \cdot N^3)$. With increasing number of drivers and orders, the computational cost is prohibitive and it is necessary to develop an efficient algorithm for this problem without compromising quality of the solution. To this end, we introduce a *Minimal-Loss Criteria* in the graph:

Minimal-Loss Criteria. Given an order o, if we cut off edge between o and d (let $x_{o,d} = 0$), then $\mathbb{P}_o$ decreases by a Loss function:

$$L = \mathbb{P}_o - \mathbb{P}_o^{\text{update}} \tag{16}$$

Furthermore, if we cut off edge between o and d^* (let $x_{o,d^*} = 0$) with the minimal choosing probability for order o,

$$d^* = argmin_d(\{p_{o,d} | x_{o,d} = 1, \forall d \in D\}) \tag{17}$$

we can achieve the minimal Loss L^{min} among all cutting scenarios for order o (the detailed proofs are elucidated in Appendix A.

Hence, we develop a Minimal-Loss Edge Cutting (MLEC) algorithm detailed in Algorithm 2, and present a numerical example solved by this algorithm in Appendix B. In each iteration, we can just collect edge x_{o,d^*} with the minimal loss L^{min} for every order o into a candidate cutting set X_c, and then cut off the 'best' edge (let $x_{o^*,d^*} = 0$) with the maximal gain Δ^{max} in X_c. The above process will be repeated until $\Delta^{max} \leq 0$.

Since the searching space in each iteration has been limited into X_c rather than all the edges, the number of evaluations in each iteration is reduced from $O(MN)$ to $O(N)$, the time complexity of MLEC algorithm in the worst case can be scaled down to $O(M \cdot N^3)$. By leveraging other heuristics like 'adding pick-up distance constraints' and 'limiting the number of edges linked to valuable orders' to initialize the bipartite graph, we can further accelerate the computational speed of MLEC algorithm.

Algorithm 2 Minimal-Loss Edge Cutting

Input: Unserved orders O, idle drivers D.
Output: Information Disclosure solution X^*.
Initialize a graph $X = \{x_{o,d} = 1 | \forall o \in O, \forall d \in D\}$.
Collect the edge x_{o,d^*} for every order o into X_c based on *Minimal-Loss Criteria.*
Find out the 'best' edge x_{o^*,d^*} in X_c with the maximal Δ^{max}.
while $\Delta^{max} > 0$ **do**
  Cut off the edge between o^* and d^* (let $x_{o^*,d^*} = 0$).
  Update $p_{o,d}, \forall o \in O, \forall d \in D$ in the graph.
  Collect the edge x_{o,d^*} for every order o into X_c based on *Minimal-Loss Criteria.*
  Find out the 'best' edge x_{o^*,d^*} in X_c with the maximal Δ^{max}.
end while
$X^* \leftarrow X$
return X^*

4 NUMERICAL STUDY

To offer a more comprehensive understanding of the effectiveness of the proposed framework, we conduct extensive evaluations on three tasks including toy examples, a MoD simulator and a real-world environment in Huolala [1]. For the first two tasks, we mainly evaluate the performance of the MLEC algorithm in the optimization step. Subsequently, the real-world experiment demonstrates end-to-end performance of the whole pipeline.

4.1 Toy Examples

For the first experiment, we design toy examples to gain a complete comparison of the MLEC algorithm in terms of computational cost and quality of the solution.

Experimental Setup: We consider two cases with 20 orders, 20 drivers and 50 orders, 50 drivers for the Information Disclosure problem. The utility $U_{o,d}$ between o and d will be normalized to the form of dollars and generated from truncated Gaussian with mean \$20.0 and standard deviation \$10.0 in the range from \$5.0 to \$40.0, and the initial utility expectation is set to \$15 for every driver in this phase. Then choosing probability $p_{o,d}$ can be computed from the estimation step by setting $\alpha = 1$. After that, we construct fully-connected graphs for two cases and each graph will be solved by both the IEC algorithm and the MLEC algorithm.

Comparison Results: The quantitative results of two cases are averaged over ten runs and shown in Figure 5. Intuitively, the MLEC algorithm achieves better solution than IEC algorithm in a short time. Although the result of IEC algorithm can be better if given enough time, the computational cost can be extremely expensive because we need to enumerate edges in each iteration.

[1] Huolala is the largest technology company providing on-demand freight delivery services in Southeast Asia and China Mainland.

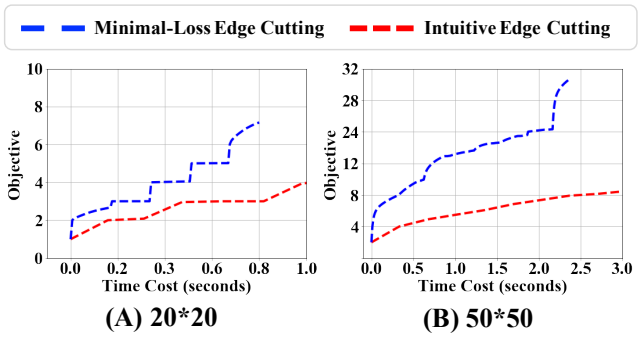


Figure 5: Comparison between the two algorithms

Table 1: The estimated parameters of NMNL model in the simulator, β is estimated by referring to the oil price per kilometre in New York and α is simplified to 1.

Parameter	β_0	β_1	β_2	α	U_d^0
Value	0	\$1	-\$0.7 per km	1.0	\$15

After demonstrating the efficiency and the optimality of the MLEC algorithm, we will utilize it to solve the Information Disclosure problem in large-scale and real-world cases.

4.2 Simulator

Beyond the toy examples, we further test the MLEC algorithm on a MoD simulator. A rough estimation for the parameters of the estimation step is shown in Table 1.

Data Source: We use large-scale taxi trip data collected in Manhattan [2] in the morning peak (7:00AM to 10:00AM) of the first week of December 2019. The trip data includes order location (origin and destination), fare, and arriving time, etc. The geographical information of both orders and drivers is mapped into nodes of the real-world road network, which is comprised of 6,533 nodes and 10,206 directed links, including streets, highways, bridges and tunnels. Afterwards, the shortest paths and the travel time among nodes are pre-calculated and stored in a look-up table.

Simulator Details: We rewind the orders between 07:00AM and 10:00AM and initially generate 3000 vehicles randomly distributed in the Manhattan at 07:00AM. In every ten seconds, we determine the idle drivers' choices based on the choosing probability computed from the estimation step, and then manipulate idle drivers into randomly cruising in the network. Concurrently, we continuously collect the drivers' decisions and assign the nearest responded driver to each order. In particular, an order will be canceled if not being responded for a long time, with the maximum waiting time following truncated Gaussian in the range 0 min to 5 mins with mean of 2.5 mins and standard deviation of 2 mins.

Comparison Baselines: The performance of the following benchmark methods are evaluated by the simulator:

[2]https://www1.nyc.gov/site/tlc/about/tlc-trip-record-data.page

(1) **One-to-One Dispatching**: This method is commonly employed in 'Order Dispatching' or 'Ride Matching' scenarios, the similar idea can be also used to determine the disclosure solution between drivers and orders (the detailed procedures are shown in Appendix C.
(2) **Global-Information Disclosure**: This method has been described in Section 3.2.
(3) **Local-Information Disclosure**: This method has been described in Section 3.2. The 'pick-up distance limitation' R is set to 2 kilometers in the simulator.

Evaluation Metrics: We adopt four metrics to evaluate the effectiveness of our method:

(1) **Avg. Response:** The average number of orders which are successfully responded per day.
(2) **Avg. GMV:** The average quantity of Gross Merchandise Volume (GMV) which records the total payment of all served orders per day.
(3) **Avg. Response time:** The average response time per order from creation to being responded.
(4) **Avg. Occupied rate:** The average occupied rate per driver, defined as the ratio of the time spent on serving to the total operating time of the taxi.

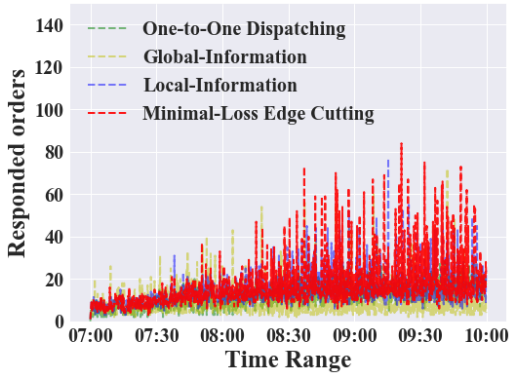


Figure 6: The number of responded orders every ten seconds.

Experimental Results: Table 2 summarizes the outcomes of different methods implemented in the simulator. Compared to the baseline One-to-One Dispatching, Global-Information Disclosure leads to a worse platform efficiency due to the concentration of the drivers' willingness on several attractive orders. As a contrast, Local-Information Disclosure and the MLEC algorithm can achieve more promising results. It is worth noticing that though the MLEC algorithm is designed for optimizing average response in our problem, it achieves the best performance on all evaluation metrics with 42.11% higher average response, 25.07% larger average GMV, 20.94% lower average response time and 22.12% greater average occupied rate compared to the baseline. Concurrently, Figure 6 presents the number of responded orders every ten seconds, the MLEC algorithm dominates other benchmarks and achieves the best results in nearly all rounds.

The main reason for the significant improvements is that orders will be achieved more proper and reasonable exposure to drivers by

Table 2: The matching results in Manhattan simulation under different Information Disclosure strategies.

Strategy	Avg. Response	Avg. GMV (Thousand Dollars)	Avg. Response time (Second)	Avg. Occupied rate
One-to-One Dispatching	12689	278.96	18.63	43.09%
Global-Information	9750 (-23.16 %)	241.98 (-13.26 %)	20.38 (+9.39 %)	37.55% (-12.86 %)
Local-Information	15390 (+21.29 %)	318.23 (+14.07 %)	16.06 (-13.78 %)	48.50% (+12.50 %)
Minimal-Loss Edge Cutting	18032 (+42.11 %)	348.92 (+25.07 %)	14.73 (-20.94 %)	52.62% (+22.12 %)

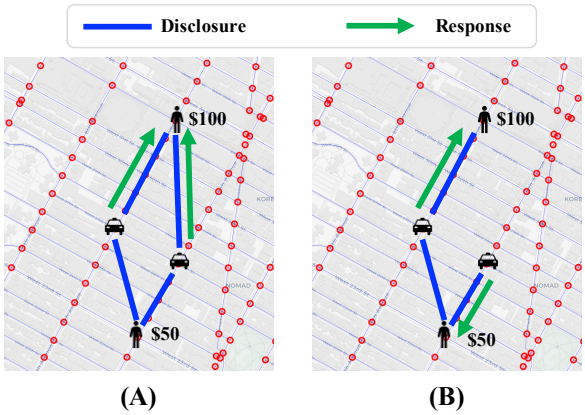


Figure 7: Information Disclosure strategies generated by the Local-Information Disclosure (A) and Minimal-Loss Edge Cutting (B). In (A), both of drivers select $U = \$100$ order and ignore $U = \$50$ order, while in (B), both of orders get responses since we cut one edge based on our optimization.

the MLEC algorithm. Take an example in Figure 7, only the most attractive order is responded in the Local-Information Disclosure. However, both of orders can be responded after cutting a specific edge based on our optimization. In summary, the MLEC algorithm brings remarkable improvements on both users experience (reducing passenger's waiting time and increasing driver's occupied rate) and platform efficiency (fulfilling more orders and stimulating the growth of GMV).

4.3 Real-world Experiments

After proving encouraging results of the optimization method on the simulator, we finally perform real-world experiments in MoD systems to demonstrate the effectiveness of the whole pipeline.

Experimental Setup: To calibrate NMNL model, we gather the historical datasets in three Chinese cities from August to September 2020 on Huolala and take them into the estimation step. Then, we adopt a customized A/B testing design that splits traffic into six-hour slice. The variants A (Original disclosure strategy in Huolala) and B (MLEC algorithm) are implemented separately on first three hours and the next three hours in Day 1. The order is then reversed for Day 2. Specifically, the proposed A/B testings have been lasting for four weeks in December 2020 to offset the daily difference. Consequently, we collect the operation results and then observe the performance of the proposed framework.

Table 3: Estimations of the MNL in the Real-World Application

City	Samples	β_0	β_1	β_2	α
City A	4.7 million	0.00**	0.78**	-7.24**	0.74**
City B	3.3 million	0.02**	0.94**	-8.95**	0.84**
City C	3.8 million	0.01**	0.87**	-8.17**	0.93**

**: P-value is below 0.01

Table 4: Comparison of quantitative results for three cities

City	Response rate	GMV	Occupied rate
City A	+3.7p.p.	+5.8 %	+9.6p.p.
City B	+4.3p.p.	+6.7 %	+8.7p.p.
City C	+4.8p.p.	+6.3 %	+11.3p.p.

NMNL Estimation: Table 3 shows the estimation results of the NMNL model in the estimation step. Intuitively, the results suggests that β_0, β_1, β_2 differ across different cities, and the p-values suggest that these parameters are significant. Table 3 also shows that the estimation of α are different from 1 for the NMNL model, indicating there is a distinction between the NMNL model of the estimation step and the classical MNL model.

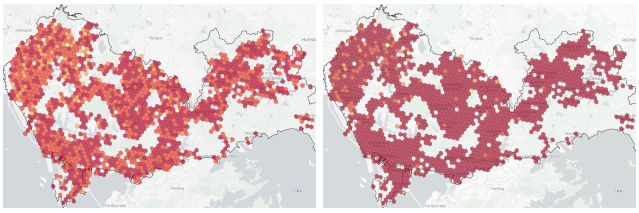


Figure 8: Comparison of the response rate in city A at evening peak 07:00PM-08:00PM. The darker color denotes the higher response rate.

Experimental Results: Comparison results against the control group is demonstrated in Table 4. The proposed method delivers a significant increase in response rate ranging from 3.7 p.p. to 4.8 p.p., brings about gains in global GMV ranging from 5.8% to 6.7%,

and also boosts growth of occupied rate from 8.7 p.p. to 11.3 p.p.. According to these encouraging results, our proposed framework has been successfully deployed in the Huolala online MoD system for millions of orders and drivers in China in a daily basis.

Qualitative study: Figure 8 presents a comparison of response rate during the evening peek in city A. Not surprisingly, our proposed framework achieves a higher rate in nearly everywhere in city A. This phenomenon can be explained from the fact that the proper information disclosure leads to profound improvements in the responding efficiency for each order. Similar observations have also been obtained in many other cities and hours from our online experiments.

5 CONCLUSION AND FUTURE WORKS

In this paper, we propose a novel framework of the Information Disclosure problem to provide drivers with effective exposure of orders in MoD systems. To this end, we propose the estimation step to understand drivers' choice behavior and predict the probability of choosing an order or ignoring the displayed orders. Subsequently, we develop the optimization step to transform the problem into the form of a bipartite graph, and then introduce a Minimal-Loss Edge Cutting (MLEC) algorithm to solve it in a reasonable time. Through extensive experiments on both the simulator and the real-world environment, the proposed method remarkably improves users experience and platform efficiency. Based on these results, the proposed framework has been deployed in the real-world MoD system in Huolala.

Various extensions of the proposed framework will be explored in the future. Typically, we can investigate deep learning approaches for estimating drivers' choice behavior in a varying choice set. Another interesting direction is using multi-agent reinforcement learning to model decision-making process among multiple drivers.

Finally, it is also meaningful to investigate the application of the proposed framework in Information Disclose problems that are common in cases of recommendation with resource constraints. Take e-commerce recommendation as an example, recommending a production with low inventory to customers shares similar challenges to our problems. We are delighted to pursue potential solutions for these problems.

ACKNOWLEDGMENT

This work was done when Yue was a research intern at Huolala. The authors would like to appreciate Mr. Lian Wu and Mr. Zehua Zhou from the marketing intelligence group in Didi Chuxing, for their kind help and comprehensive advice.

REFERENCES

[1] Niels Agatz, Alan Erera, Martin Savelsbergh, and Xing Wang. 2012. Optimization for dynamic ride-sharing: A review. *European Journal of Operational Research* 223, 2 (2012), 295–303.
[2] Roberto Baldacci, Vittorio Maniezzo, and Aristide Mingozzi. 2004. An exact method for the car pooling problem based on lagrangean column generation. *Operations Research* 52, 3 (2004), 422–439.
[3] Axel Börsch-Supan. 1990. On the compatibility of nested logit models with utility maximization. *Journal of Econometrics* 43, 3 (1990), 373–388.
[4] Timothy A. Brathwaite. 2016. *PyLogit Python Package.* https://pypi.org/project/pylogit/
[5] Kyunghyun Cho, Bart Van Merriënboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2014. Learning phrase representations using RNN encoder-decoder for statistical machine translation. *arXiv preprint arXiv:1406.1078* (2014).
[6] Leon Yang Chu, Zhixi Wan, and Dongyuan Zhan. 2018. Harnessing the double-edged sword via routing: Information provision on ride-hailing platforms. *Available at SSRN 3266250* (2018).
[7] John E Dennis Jr and Robert B Schnabel. 1996. *Numerical methods for unconstrained optimization and nonlinear equations.* SIAM.
[8] Paul C Gilmore and Ralph E Gomory. 1961. A linear programming approach to the cutting-stock problem. *Operations research* 9, 6 (1961), 849–859.
[9] Shengnan Guo, Youfang Lin, Ning Feng, Chao Song, and Huaiyu Wan. 2019. Attention based spatial-temporal graph convolutional networks for traffic flow forecasting. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 33. 922–929.
[10] Ruining He, Chen Fang, Zhaowen Wang, and Julian McAuley. 2016. Vista: A visually, socially, and temporally-aware model for artistic recommendation. In *Proceedings of the 10th ACM Conference on Recommender Systems.* 309–316.
[11] Ruining He and Julian McAuley. 2016. Fusing similarity models with markov chains for sparse sequential recommendation. In *2016 IEEE 16th International Conference on Data Mining (ICDM).* IEEE, 191–200.
[12] Wang-Cheng Kang and Julian McAuley. 2018. Self-attentive sequential recommendation. In *2018 IEEE International Conference on Data Mining (ICDM).* IEEE, 197–206.
[13] Ziqi Liao. 2003. Real-time taxi dispatching using global positioning systems. *Commun. ACM* 46, 5 (2003), 81–83.
[14] Kaixiang Lin, Renyu Zhao, Zhe Xu, and Jiayu Zhou. 2018. Efficient large-scale fleet management via multi-agent deep reinforcement learning. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining.* 1774–1783.
[15] R Duncan Luce. 2012. *Individual choice behavior: A theoretical analysis.* Courier Corporation.
[16] Guodong Lyu, Wang Chi Cheung, Chung-Piaw Teo, and Hai Wang. 2019. Multi-objective online ride-matching. *Available at SSRN 3356823* (2019).
[17] Daniel McFadden et al. 1973. Conditional logit analysis of qualitative choice behavior. (1973).
[18] James Munkres. 1957. Algorithms for the assignment and transportation problems. *Journal of the society for industrial and applied mathematics* 5, 1 (1957), 32–38.
[19] Steffen Rendle, Christoph Freudenthaler, and Lars Schmidt-Thieme. 2010. Factorizing personalized markov chains for next-basket recommendation. In *Proceedings of the 19th international conference on World wide web.* 811–820.
[20] Alex Rosenblat and Luke Stark. 2016. Algorithmic labor and information asymmetries: A case study of Uber's drivers. *International Journal of Communication* 10 (2016), 27.
[21] C Smith. 2018. 110 Amazing Uber stats and facts (2017) by the numbers.
[22] Fei Sun, Jun Liu, Jian Wu, Changhua Pei, Xiao Lin, Wenwu Ou, and Peng Jiang. 2019. BERT4Rec: Sequential recommendation with bidirectional encoder representations from transformer. In *Proceedings of the 28th ACM International Conference on Information and Knowledge Management.* 1441–1450.
[23] Huw CWL Williams. 1977. On the formation of travel demand models and economic evaluation measures of user benefit. *Environment and planning A* 9, 3 (1977), 285–344.
[24] Zhe Xu, Zhixin Li, Qingwen Guan, Dingshui Zhang, Qiang Li, Junxiao Nan, Chunyang Liu, Wei Bian, and Jieping Ye. 2018. Large-scale order dispatch in on-demand ride-hailing platforms: A learning and planning approach. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining.* 905–913.
[25] Zhe Xu, Chang Men, Peng Li, Bicheng Jin, Ge Li, Yue Yang, Chunyang Liu, Ben Wang, and Xiaohu Qie. 2020. When Recommender Systems Meet Fleet Management: Practical Study in Online Driver Repositioning System. In *Proceedings of The Web Conference 2020.* 2220–2229.
[26] An Yan, Shuo Cheng, Wang-Cheng Kang, Mengting Wan, and Julian McAuley. 2019. CosRec: 2D convolutional neural networks for sequential recommendation. In *Proceedings of the 28th ACM International Conference on Information and Knowledge Management.* 2173–2176.
[27] Yue Yang, Qiong Tian, and Yuqing Wang. 2020. Who is more likely to get a ride and where is easier to be picked up in ride-sharing mode? *Journal of Management Science and Engineering* (2020).
[28] Junchen Ye, Leilei Sun, Bowen Du, Yanjie Fu, Xinran Tong, and Hui Xiong. 2019. Co-prediction of multiple transportation demands based on deep spatio-temporal neural network. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining.* 305–313.
[29] Xianyuan Zhan, Xinwu Qian, and Satish V Ukkusuri. 2016. A graph-based approach to measuring the efficiency of an urban taxi service system. *IEEE Transactions on Intelligent Transportation Systems* 17, 9 (2016), 2479–2489.
[30] Lingyu Zhang, Tao Hu, Yue Min, Guobin Wu, Junying Zhang, Pengcheng Feng, Pinghua Gong, and Jieping Ye. 2017. A taxi order dispatch model based on combinatorial optimization. In *Proceedings of the 23rd ACM SIGKDD international conference on knowledge discovery and data mining.* 2151–2159.

APPENDIX

A

We provide the derivations for the equations proposed in Section 3.3 below:

Derivation 1

For every order $o^{'}$ in driver d's new choices candidates $O_d/\{o\}$, we can compute a change of $\mathbb{P}_{o'}$:

$$\begin{aligned}\mathbb{P}_{o'}^{\text{update}} - \mathbb{P}_{o'} &= [1-(1-p_{o',d}^{\text{update}})\cdot\prod_{d'\in D/\{d\}}(1-p_{o',d'}x_{o',d'})] \\ &\quad -[1-(1-p_{o',d})\cdot\prod_{d'\in D/\{d\}}(1-p_{o',d'}x_{o',d'})] \\ &= (p_{o',d'}^{\text{update}} - p_{o',d})\cdot\prod_{d'\in D/\{d\}}(1-p_{o',d'}x_{o',d'}) \\ &= \delta_{o',d}\cdot\prod_{d'\in D/\{d\}}(1-p_{o',d'}x_{o',d'})\end{aligned} \tag{18}$$

Derivation 2

Once we cut an edge $x_{o,d}$, the number of drivers browsing order o is reduced by one, the change of $\mathbb{P}_o$ can be:

$$\begin{aligned}&\mathbb{P}_o^{\text{update}} - \mathbb{P}_o \\ &= [1-\prod_{d'\in D/\{d\}}(1-p_{o,d'}x_{o,d'})] - [1-\prod_{d'\in D}(1-p_{o,d'}x_{o,d'})] \\ &= -p_{o,d}x_{o,d}\cdot\prod_{d'\in D/\{d\}}(1-p_{o,d'}x_{o,d'}) \\ &= -p_{o,d}\cdot\prod_{d'\in D/\{d\}}(1-p_{o,d'}x_{o,d'})\end{aligned} \tag{19}$$

Proof of Minimal-Loss Criteria

According to the Equation 14,

$$\begin{aligned}L &= \mathbb{P}_o - \mathbb{P}_o^{\text{update}} \\ &= p_{o,d}\cdot\prod_{d'\in D/\{d\}}(1-p_{o,d'}x_{o,d'}) \\ &= p_{o,d}\cdot(1-\mathbb{P}_o^{\text{update}})\end{aligned} \tag{20}$$

Given an order o, if we cut off the edge x_{o,d^*} with the minimal probability $p_{o,d^*} = \min(\{p_{o,d}|x_{o,d}=1,\forall d\in D\})$, then we can get a loss:

$$L = p_{o,d^*}\cdot(1-\mathbb{P}_o^{\text{update}}) \tag{21}$$

In particular, we can get the maximal $\mathbb{P}_o^{\text{update}}$ compared to cutting off other edges linked to o.

Therefore, we can arrive at the minimal loss L^{min} among all the cutting scenarios of order o.

B

To explain our optimization framework from a more intuitive view, we construct the small numerical example following the problem in Figure 4.

First iteration

As shown in the Figure 9, we can collect x_{o_1,d_3}, x_{o_2,d_3}, x_{o_3,d_1} with the minimal loss L^{min} for o_1, o_2, o_3 into X^c. Furthermore, we compute Δ for every $x_{o,d}$ in X^c and then pick out x_{o_1,d_3} with Δ^{max} in this case.

After cutting off x_{o_1,d_3}, we can arrive at a new graph by cutting off x_{o_1,d_3} and update $p_{o,d}$ in Figure 10.

Second iteration

In the case of Figure 10, we will collect x_{o_1,d_1}, x_{o_2,d_1}, x_{o_3,d_1} with the minimal loss L^{min} for o_1, o_2, o_3 into X^c. Furthermore, we compute Δ for every $x_{o,d}$ in X^c and then pick out x_{o_2,d_1} with Δ^{max} in this case.

After cutting off x_{o_2,d_1}, we can arrive at a new graph by cutting off x_{o_2,d_1} and update $p_{o,d}$ in Figure 11.

Third iteration

In the case of Figure 11, we will collect x_{o_1,d_1}, x_{o_2,d_3}, x_{o_3,d_1} with the minimal loss L^{min} for o_1, o_2, o_3 into X^c. Furthermore, we compute Δ for every $x_{o,d}$ in X^c and then pick out x_{o_2,d_3} with Δ^{max} in this case.

After cutting off x_{o_2,d_3}, we can arrive at a new graph by cutting off x_{o_2,d_1} and update $p_{o,d}$ in Figure 12.

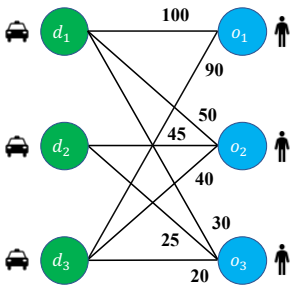
Fourth iteration

In the case of Figure 12, we will collect x_{o_1,d_1}, x_{o_2,d_2}, x_{o_3,d_1} with the minimal loss L^{min} for o_1, o_2, o_3 into X^c. Furthermore, we compute Δ for every $x_{o,d}$ in X^c.

Then we can observe that $\Delta^{max} < 0$, so we terminate our iteration and output the graph as the final solution X^*:

$$X^* = \{x_{o_1,d_1}=1, x_{o_2,d_1}=1, x_{o_3,d_1}=1, x_{o_3,d_2}=1, x_{o_3,d_3}=1\} \tag{22}$$

Delightedly, we can illustrate that it is highly possible that all of three orders can be responded in this scenario.



The distribution of $p_{o,d}$ and $\mathbb{P}_o$

	o_1	o_2	o_3	s
d_1	0.990	0.009	0.001	0.000
d_2	0.000	0.980	0.014	0.006
d_3	0.990	0.008	0.002	0.000
$\mathbb{P}_o$	0.999	0.980	0.016	

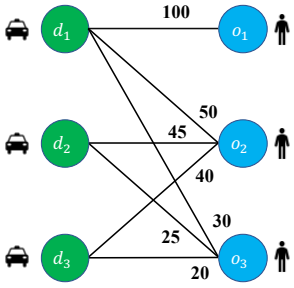
Δ for cutting each edge in X^c

X^c	x_{o_1,d_3}	x_{o_2,d_3}	x_{o_3,d_1}
Δ	0.289	0.001	0.001

Figure 9: First iteration

C

In order dispatching process, the objective of the the platform is to determine the best matching between drivers and orders, and we suppose that the system attempts to minimize the total pick-up distance or the waiting time of passengers, in this way, we set $W_{o,d}$ to depict the reciprocal of pick-up distance between order o and driver d:



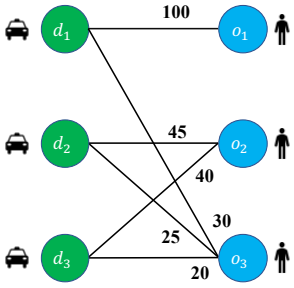
The distribution of $p_{o,d}$ and $\mathbb{p}_o$

	o_1	o_2	o_3	s
d_1	0.990	0.009	0.001	0.000
d_2	0.000	0.980	0.014	0.006
d_3	0.000	0.970	0.022	0.008
$\mathbb{p}_o$	0.990	0.999	0.036	

Δ for cutting each edge in X^c

X^c	x_{o_1,d_1}	x_{o_2,d_1}	x_{o_3,d_1}
Δ	-0.976	0.002	0.000

Figure 10: Second iteration



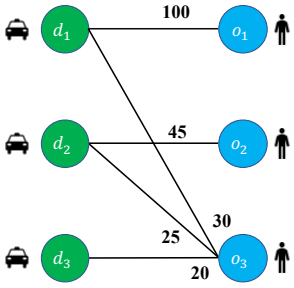
The distribution of $p_{o,d}$ and $\mathbb{p}_o$

	o_1	o_2	o_3	s
d_1	0.990	0.000	0.004	0.006
d_2	0.000	0.980	0.014	0.006
d_3	0.000	0.970	0.022	0.008
$\mathbb{p}_o$	0.990	0.999	0.039	

Δ for cutting each edge in X^c

X^c	x_{o_1,d_1}	x_{o_2,d_3}	x_{o_3,d_1}
Δ	-0.961	0.401	0.000

Figure 11: Third iteration



The distribution of $p_{o,d}$ and $\mathbb{p}_o$

	o_1	o_2	o_3	s
d_1	0.990	0.000	0.004	0.006
d_2	0.000	0.980	0.014	0.006
d_3	0.000	0.000	0.450	0.550
$\mathbb{p}_o$	0.990	0.980	0.452	

Δ for cutting each edge in X^c

X^c	x_{o_1,d_1}	x_{o_2,d_2}	x_{o_3,d_1}
Δ	-0.973	-0.966	-0.002

Figure 12: Fourth iteration

$$W_{o,d} = \frac{1}{\tau_{o,d}} \tag{23}$$

Therefore, the dispatching problem can be modeled as follows:

$$\max \sum_{d \in D} \sum_{o \in O} x_{o,d} \cdot W_{o,d} \tag{24}$$

$$s.t. \sum_{o \in O} x_{o,d} \leq 1; \forall d \in D \tag{25}$$

$$\sum_{d \in D} x_{o,d} \leq 1; \forall o \in O \tag{26}$$

$$x_{o,d} \in \{0, 1\}; \forall o \in O, \forall d \in D \tag{27}$$

where the first constraint guarantee that each driver browses at most one order or views nothing at this point, and the second constraint ensures that each order is displayed to at most one driver or waits until next decision round.

If we also abstract drivers set D and orders set O as two sets of vertices, and valid matching pairs as the set of edges, the ILP problem can be represented as a bipartite graph matching problem. In general, the initial bipartite graph is a fully connected graph where every possible edge between drivers and orders exists. To reduce computational complexity, we further introduce a similar 'matching radius' to eliminate parts of edges whose pick-up distance exceeds the dispatching radius. Consequently, we can arrive at an exceptionally compact graph and employ the Kuhn-Munkres (KM) algorithm [18] to solve it.