The following publication L. Yang, X. Yu, J. Cao, W. Li, Y. Wang and M. Szczecinski, "A Novel Demand Dispatching Model for Autonomous On-Demand Services," in IEEE Transactions on Services Computing, vol. 15, no. 1, pp. 322-333, 1 Jan.-Feb. 2022 is available at https://doi.org/10.1109/TSC.2019.2941680.

A Novel Demand Dispatching Model for Autonomous On-Demand Services

Lei Yang, Xi Yu, Jiannong Cao, *Fellow, IEEE,* Wengen Li, Yuqi Wang, Michal Szczecinski

Abstract—Recent on-demand services, such as Uber and DiDi, provide a platform for users to request services on the spot and for suppliers to meet such demand. In such platforms, demands are dispatched to suppliers round by round, and suppliers have autonomy to decide whether to accept demands or not. Existing approaches dispatch a demand to multiple suppliers in each round, while a supplier can only receive one demand. However, by using these approaches, pended demands can not be fully dispatched in a round specially when suppliers are not sufficient, and thus need to wait for many rounds to be dispatched, leading to long response time. In this paper, we propose a novel demand dispatching model, named by many-to-many model. The novelty of the model is that a supplier could receive multiple demands in a round, such that the demand has high chance to be dispatched and answered within short time. More specifically, we first learn the probability distribution function of the response time of a supplier to a given demand, by considering the features of both the demand and the supplier. Taking the learned results as input, our model generates an optimal matching between the demands and suppliers to minimize the overall response time of the demands via solving an optimization problem. Experiments on real-world datasets show that our model is better than the start-of-art models in terms of successful acceptance rate and response time.

Index Terms—demand dispatching, on-demand services, response time prediction

1 Introduction

Recent on-demand services, such as Uber, Didi Chuxing and GoGoVan in Hong Kong, provide a platform for users to request services on the spot and for suppliers to meet such demand. Didi Chuxing, for example, allows its users to call for taxies on demand through mobile apps. The demand is dispatched by the platform to the drivers, who decide whether or not to accept them. If multiple drivers are willing to serve, the order will be assigned to one of them according to pre-established policies. Future energy systems [1] can also be considered as on-demand services, as multiple suppliers, including power stations, energy stores, and mobile charging stations, compete to satisfy energy demand from their customers.

One fundamental problem for these emerging on-demand services is demand dispatching, i.e., to determine for the demand which suppliers it should be dispatched to. Existing works on demand dispatching can be classified into two modes: one is Server-Assigned Demand mode (SAD), the other is Supplier-Selected Demand mode (SSD). In SAD mode, the sever matches supplier and demand directly in order to achieve some evaluation metrics, e.g., maximizing the number of assigned demands with some given constraints. While in SSD mode, suppliers can select demand from all demands in the on-demand service platform. However, SSD mode always lets unpopular demands unserved for a long time and leads to low efficiency of demand dispatching [2] [5]. Thus, the demand dispatching model focused in this paper is in the scope of SAD mode.

Existing solutions in SAD mode [4] [15] [30] have been proposed for the traditional supply and demand system, which directly assign a request to a particular supplier, and assume that the supplier must accept it. In autonomous on-demand services, in contrast, the suppliers are autonomous in deciding whether to accept the request. The rejection would happen when a supplier is not interested at the assigned demand [5] [6] [29]. We need to develop new approaches for the autonomous on-demand services by considering the supplier's preference to the demand.

Zhang et al. [6] proposed a one-to-many dispatching model based on the estimation of a supplier's acceptance probability towards a particular demand. In this model, demands are dispatched round by round. In each round, a demand will be dispatched to several suppliers, and a supplier can only receive one demand. If a demand is not accepted by any supplier, it will enter the next round of dispatching until it is accepted or canceled. However, the performance of this model becomes poor during peak hours when the suppliers are not sufficient. In this scene, the pended demands could not be fully dispatched in a round, and thus need to wait for many rounds to be dispatched, leading to long response time. Furthermore, since a supplier can only receive one demand in a round, if he/she happens to reject the demand, the supplier will get idle in this round, which results in high empty-loading ratio for the suppliers.

To address aforementioned issues, we propose a novel demand dispatching model, named by many-to-many model. The novelty of our model is allowing a supplier to receive multiple demands in each round of dispatching. As a result, more demands would be dispatched out in a round. The demands have high chance be answered within

- *L. Yang and X. Yu are with the School of Software Engineering, South China University of Technology, China. Email: sely@scut.edu.cn, secruddev@mail.scut.edu.cn*
- *J. Cao, W. Li and Y. Wang are with the Department of Computing, The Hong Kong Polytechnic University, Hong Kong. E-mail: {csjcao, cswgli, csyqwang}@comp.polyu.edu.hk*
- *M. Szczecinski is with GoGoVan, Hong Kong. E-mail: michal@gogotech.hk*

fewer rounds. Our model brings an additional challenge to quantify the supplier's response behaviour when he/she receives multiple demands, i.e., which demand the supplier would accept and how long the response time is. To solve the challenge, we use the probability distribution of the supplier's response time to represent his/her behavioural pattern toward the demand. Note that although ride-sharing [7] can improve the efficiency of on-demand system, to make the problem easy to solve, we will not consider ride-sharing.

Specifically, we first learn the probability distribution of a supplier's response time to a given demand based on the historical data sets. Based on the one-to-one response time pattern, we then infer the probability distribution of the response time in the many-to-many model by considering the competition of the demands as they are dispatched to the same supplier. With the probability distribution, we formulate the demand dispatching into a combinational optimization problem that aims to minimize the overall response time for the demands. However, the exponentially large solution space of the many-to-many model makes it difficult to find the exact optimal solution. We make a constraint on the maximum number of demands that a supplier can receive to reduce the solution space, and propose an efficient heuristic to solve it. Our contributions are listed as follows.

- We first have a 3-dimensional classification of demand dispatching models, and show that this is the first work that uses a many-to-many model to solve the demand dispatching problem for autonomous on-demand services.
- We use the probability distribution of response time to quantify the supplier's behaviour pattern toward the demand, while existing work uses the coarse-grained binary acceptance probability. Accordingly we develop an approach to predict the probability distribution of response time based on the historical data sets.
- We formulate the many-to-many dispatching problem, and develop a heuristic algorithm with high efficiency.
- We conduct extensive experiments using the real-world datasets from a large van-calling platform at Hong Kong. The results show that our proposed many-to-many model outperforms the state-of-art models significantly in both the success rate and response time.

2 Background and Problem Statement

In this section, in order to introduce the background of demand dispatching, we classify existing works firstly. Next, we introduce representative works in these categories and state the problem scope in this paper.

2.1 3D Classification of Demand Dispatching

The works on demand dispatching can be classified according to three dimensions, which have been shown in Fig.1. The first dimension means that the demand is dispatching in real-time mode (R) or batch mode (B). In real-time mode, the dispatching server operates immediately when a new request (demand/supplier) appears [5]. While in batch mode, demands are dispatched periodically every short time interval. The former responds to each demand, and the latter responds to temporal accumulated demands [5]. The second dimension represents that whether a on-demand service has taken the rejection of supplier into consideration (NR and RA). Rejection is common in on-demand service, if a supplier have no interest in a demand, he/she will reject it. Furthermore, we use *autonomous* to represent *rejection aware* in this work.

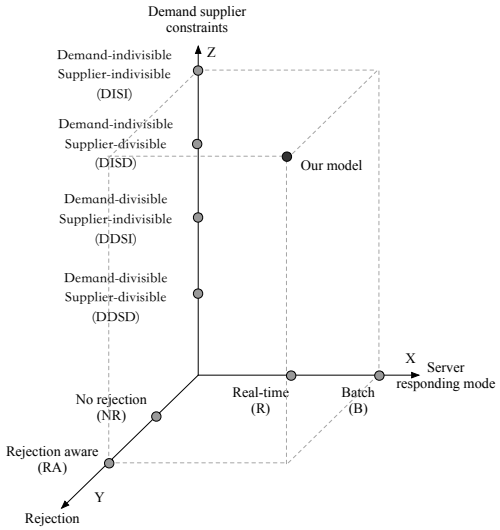


Fig. 1: The classification of works on demand dispatching

The last dimension indicates that whether the demand and supplier are divisible. A demand is divisible only if it can be served by multiple suppliers simultaneously, and a supplier is divisible only if he/she can serve multiple demands at the same time. For example, we consider the non-ride-sharing mode in Uber. In this mode, a demand can only be assigned to one supplier, and a supplier can only serve one demand at the same time. So the demand and supplier in this application are both indivisible. On the other hand, if we consider ride-sharing, the supplier becomes divisible since he/she can serve multiple demands at the same time. In cooperative Electric Vehicle (EV) charging, a discharging EV (supplier) can supply electricity for multiple EVs simultaneously and a charging EV (demand) can also get electricity from multiple EVs simultaneously, so the demand and supplier in this application are both divisible [8].

Because the three dimensions are independent of one another, a wide variety of demand dispatching schemes can be created by combining properties from each. An XX-YY-ZZ string expresses a kind of demand dispatching model, in which XX represents server responding mode (R or B), YY stands for whether the model considers rejection (NR or RA), and ZZ symbolizes the constraints of demand and supplier (DDSD, DDSI, DISD or DISI).

2.2 Representative Works

In R-NR-DISI, the problems can always be considered as online bipartite matching. The majority of them focuses on

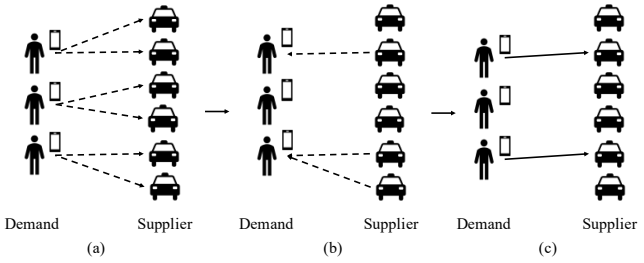


Fig. 2: One-to-many dispatching model

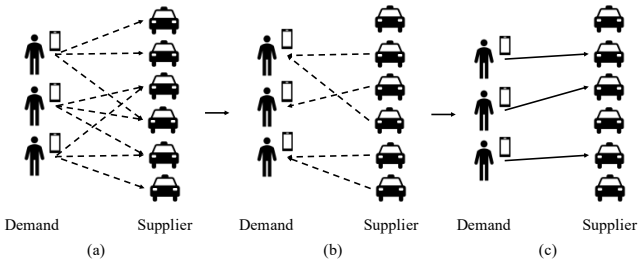


Fig. 3: Many-to-many dispatching model

one-sided online matching, while in [4], the authors considered two-sided online task assignment. In other aspects of R-NR-* (in which * represents those situations with non-fixed attributes and can be any combination of properties along the dimension), many works also appeared. The supplier is divisible if he/she has a capacity, which is the maximum number of demands that can be assigned to him/her [9]. In some works [10], if the platform receives a demand which needs to be performed multiple times or needs multiple suppliers to serve collaboratively, then this demand is divisible.

Problems in R-RA-* mode have been widely studied, in which the method of contextual multi-armed bandit [14] is always used to capture the rejection behaviour of suppliers. In these problems, the features of demand and supplier are considered as the contextual features of multi-armed bandit, and each arm of it represents a supplier. When the platform receives a demand, it needs to select one or multiple suppliers to finish this demand [2]. And in some works, a supplier who is serving a demand may also receive a demand [3], so demands and suppliers in such modes can either be divisible or indivisible.

B-NR-* is also a well studied research field. Most works in this field only consider the information of demands and suppliers in current time slot, and we call them the greedy solutions. In B-NR-DISI, the problem can always be formulated as a weighted bipartite matching problem. To overcome the shortcoming with greedy solution, [11] considered to predict the features of demands and suppliers in next time slot. In B-NR-DISD, [12] proposed a demand assignment model which gave demand with low location entropy high priority in order to improve the global efficiency of model. When problem comes to B-NR-DDSI, the representative work [13] assigns workers to spatial tasks such that the completion reliability and the spatial/temporal diversities of spatial tasks are maximized. In DDSD, [8] proposed a cooperative EV charging framework, in which a charging EV can get electricity from multiple discharging EVs simultaneously, and a discharging EV can also serve multiple charging EVs at the same time.

Only a few works have studied the problems in B-RA-DISI mode [6], and our problem studied in this paper is in the category. Few works study the other aspects of B-RA-* as far as we know. Due to the possible rejection by the supplier to a demand, a demand is always dispatched to multiple suppliers in order to increase the overall successful acceptance rate for the demands. Fig.2 shows the working process of demand dispatching model in [6], which contains three steps. First, the platform collects features of demands and suppliers in this round, and then gets the acceptance probability of each supplier to each demand. After that, the platform performs the one-to-many demand dispatching algorithm and finally dispatches each demand to a set of suppliers (shown in Fig.2a). One-to-many means that a demand can be dispatched to multiple demands, but a supplier can only receive one demand for further selection. In the second step, each supplier has received the demand dispatched to him/her, and either accept or reject this demand (Fig.2b). In the third step, the platform will select one from the suppliers who have accepted the demand according to pre-established policy, and assigns him/her to the demand (Fig.2c).

2.3 Problem Statement

In the above one-to-many model [6], the successful acceptance rate of the demands in a dispatching round would be not satisfactory especially when the suppliers are not sufficient. Some of the demands may need to wait for many rounds until it is successfully accepted by the supplier. So we propose a many-to-many demand dispatching model, where a demand is dispatched to multiple suppliers and each supplier can also receive multiple demands. Fig.3 shows the working process of our dispatching model. In the first step, our model collects the features of demands and suppliers in this round, and then gets the predicted behaviour pattern of each supplier to each demand. Next, our model will make a many-to-many dispatching in order to maximize some defined metrics. In the second step, each supplier can perform two kinds of choices according to the properties of the demand and his/her preference: 1) reject all the demands he/she received; 2) select one demand from all the demands he/she received. In the third step, our model selects a supplier for the demand by a particular filtering policy if multiple suppliers want to accept the service. Normally for fairness among the suppliers, we assume that the supplier who answers the demand at the first time would serve the demand.

In this paper, we study the problem of demand dispatching in the first step (shown in Fig.3a) in order to satisfy the following objectives.

• **Successful acceptance rate**. If a demand is accepted finally by one of the suppliers, we name that the demand has a successful acceptance. If a demand is dispatched to a set of suppliers none of whom want to accept it, the demand would have a unsuccessful acceptance. We hope that as many demands as possible can be successfully accepted by the suppliers.

• **Response time**. Response time is defined as the period from the time when the demand is released to the time when it is accepted by a supplier. The response time reflects the supplier's behaviour towards the demand. It depends on

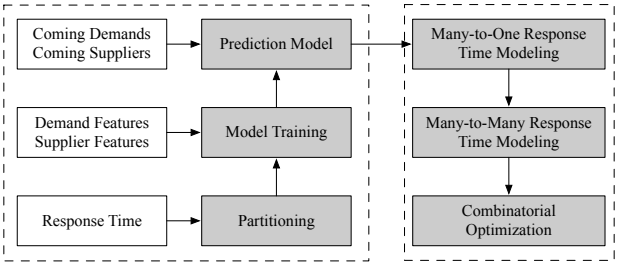


Fig. 4: An overview framework of our solution

the properties of the demand and the supplier's preference. Our objective is to find a dispatching such that the average response time of the accepted demands is as short as possible.

3 Solution

We first present an overview framework of our solution shown in Fig. 4. The framework consists of two parts: prediction of the supplier's response time to the demand, and demand dispatching.

• **Response Time Prediction**. We learn the supplier's behaviour pattern to the demand from the historical data set. Here behaviour pattern means that, given a demand and a supplier, how likely it is for the supplier to accept the demand? How much time it takes for the supplier to accept the demand? In our solution, we use the response time to model the supplier's behaviour. If the response time is extremely large, it means the supplier does not accept the demand within a round of dispatching. The historical data set includes information about the demand, the supplier who accepts the demand, and the response time. Instead of directly using regression models [16], we first divide the continuous response time into several time ranges without intersections [18], and then transfer the prediction problem into a multi-classification problem. In prediction phase, the multi-classifier outputs the response time probability, which is represented by a piecewise function on the time ranges.

Since the prediction model might have changed after the deployment, the response time distribution of each supplier may not fully describe the real response pattern of him/her. To overcome this shortcoming, our prediction model needs to be updated continuously in real applications.

• **Demand Dispatching Models**. The probability distribution from the multi-classifier reflects the supplier's response pattern when the supplier receives one and only one demand, where we call it one-to-one model. We assume that the response time is infinite if a demand is rejected by a supplier. However, if a supplier receives multiple demands, his/her expected response time to a specific demand will be affected by the other received demands. We call it many-to-one model. For the same demand, the expected response time of a supplier in many-to-one model is usually longer than that in the one-to-one model due to two aspects: 1) the supplier can only select one demand to accept, so there will be a competitive game among the demands; 2) the infinite response time of failed demands in this game. In Section 3.2.1, we present how to calculate the probability distribution of the response time in many-to-one model.

Meanwhile, a demand can be dispatched to multiple suppliers in a many-to-many dispatching model. The response time of a demand depends on the number of suppliers who receive it. Normally if more suppliers receive the demand, the expected response time will be shorter. We present how to deduce the probability distribution of the response time of a demand in the many-to-many model in Section 3.2.2.

Note that although the deduced response time may not fully describe the real response time probability distribution in the proposed many-to-many model, it is still an effective way to model it when we have no historical data for many-to-many model. Based on the probability distribution, we formulate the demand dispatching problem as a combinatorial optimization problem. The problem decides for each demand to whom it should be dispatched such that the expected response time of all the demands is minimized.

3.1 Response Time Prediction

3.1.1 Data Sources and Feature Extraction

The data sources are from a van-calling service platform. They include two data sets: one is the demand historical data set, and the other is the supplier/driver's GPS location data set. The demand data set includes the information indicating when the demand was released and accepted, which supplier accepted it, the pick up location and destination, the customer's extra requirements and so on. The supplier location data set includes the GPS trajectory locations of the suppliers during the same time period with the demand data set.

In order to simplify the process of feature selection and make us put main energy into next steps, we use the features in existing work [16] directly, in which the datasets and the main prediction target are the same as us. The features extracted from the two data sets are listed as follows.

- Demand related features: 1) the temporal features including the date, day of week, and the time when the demand was released and accepted; 2) the rating of the customer who made the demand; 3) the price of the demand; 4) personalized requirements including required language, whether the user has pets and whether the demand will be paid by the sender or the receiver; 5) locations including the start and end locations; 6) the total number of demands at the start location and the end location.
- Supplier related features: 1) the average and variance of his/her travel distances in each hour during all days; 2) average and variance of the idle ratio in each hour of all days.
- Demand-Supplier related feature: the pick-up distance, which is the distance between a supplier and the demand he/she receives.

3.1.2 Response Time Partitioning

Since the problem of predicting response time has been transformed into a multi-classification problem, we need to choose a way to partition the entire range of the response time. Note that our dispatching model is round-based, and each round is associated with a time window to dispatch the

TABLE 1: Performance of Prediction Models

Model	3 classes		4 classes		5 classes	
	ACC	AUC	ACC	AUC	ACC	AUC
LR	0.534	0.694	0.447	0.683	0.364	0.677
GBDT	0.572	0.734	0.494	0.724	0.414	0.717
GBDT+LR	0.527	0.702	0.431	0.689	0.351	0.683
RF	0.649	0.809	0.525	0.771	0.430	0.755

pended demands. On the other hand, dividing the response time beyond the time window makes no sense. This is because if the response time of a supplier is over the time window, it means that he/she did not accept the demand in this round, and the demand is dispatched in the next round. Our purpose is to predict the probability distribution of the response time within the time window. So we divide the samples whose response time is longer than the time window into a separate class. The samples whose response time is shorter than the time window are partitioned into k classes using K-means clustering [17] on the response time. The selections of k and the time window RT will be discussed in Section 4. Table 2 shows the mathematical notations in this paper.

3.1.3 Multi-Classification

Using the features of demands and suppliers, we train a $(k+1)-class$ classifier. Let L denote the number of partitioned time ranges, where $L = k+1$, and T_b represents the boundary of each time range. $p_{ij}(t)$ is the probability distribution of the response time of the supplier s_j to the demand d_i. We illustrate the prediction result from the multi-classifier by the following example.

Example 1. *We suppose that the time window in the dispatching model is* $30\ sec$*, and divide the samples whose response time is shorter than* $30sec$ *into* 2 *classes using K-means. The samples whose response time is longer than* $30\ sec$ *is partitioned into the third class. We assume that the corresponding time range of the three classes is* $0-10\ sec$*,* $10-30\ sec$*, and* $30\ sec$ *to infinity. So we have* $L=3$*,* $T_0=0\ sec$*,* $T_1=10\ sec$ *and* $T_2=30\ sec$*. Then we can train a 3-class classifier and it will output* $p_{ij}(t)$*, e.g.,* $p_{ij}(T_0 < t \leq T_1) = 0.3$*,* $p_{ij}(T_1 < t \leq T_2) = 0.3$ *and* $p_{ij}(t > T_3) = 0.4$*.*

We compared four popular models: linear logistic regression (LR) [19], gradient boosted decision tree (GBDT) [20], a model which combines GBDT and LR (GBDT+LR) [21], and random forest (RF) [22]. All models were trained on the van-calling service data sets, which consist of about 170000 demand historical records and corresponding supplier features. We set the time window of the dispatching model as $30\ sec$, and evaluate the performance of the four models in terms of Accuracy (ACC) and Area under the Curve of ROC (AUC). According to the result of K-means clustering on the labels, we have divided training data respectively into 3, 4 and 5 classes. Note that the number of samples of each class is not balanced, so we use micro-AUC to calculate the value of AUC [23]. For all the models, we conduct 5-fold cross-validation for 100 times, which was chosen in favor of [24]. Table 1 shows the performance results of the four models. The performance of RF is the best among the four models on our data sets. For all the models, ACC and AUC get lower with the increase of the number of classes L.

TABLE 2: Mathematical notations in this paper

RT	time window between two dispatching rounds;
k	the number of clusters in history data whose response time is before RT;
L	the number of partitioned response time ranges;
T_b	the border of each response time range;
t	the response time;
N	the number of demands in a round;
M	the number of suppliers in a round;
$p_{ij}(t)$	the probability distribution of t of demand i if it is dispatched to supplier j in one-to-one model;
d_i	the demand i;
s_j	the supplier j;
$P_{ij}(t)$	the probability distribution of t of demand j if it is dispatched to supplier j in many-to-one model;
x_{ij}	if demand i is dispatched to supplier j, $x_{ij}=1$; else, $x_{ij}=0$
$S_i(t)$	the probability distribution of t of demand i in many-to-many model;
$g(t)$	the benefit a demand can get if it is accepted at time t;
G_i	the expected benefit a demand can get in many-to-many model;
UL	the maximum number of demands a supplier can receive in a dispatching round;
$D[j]$	the demand list that supplier j receives in a dispatching round;
TN	the number of trapezoids for calculating $P_{ij}(t)$ in an approximate way;
r	the demand-supplier ratio in a round;
pdf	the probability distribution function.

3.2 Many-to-Many Demand Dispatching Model

Note that $p_{ij}(t)$ from the multi-classifier only represents the response time of supplier s_j to demand d_i in the one-to-one model. In the many-to-many dispatching model, a demand is dispatched to more than one suppliers, and a supplier could receive more than one demands. We need to deduce the probability distribution of the response time in the many-to-many model based on $p_{ij}(t)$.

3.2.1 Response Time in Many-to-One Dispatching Model

We consider the many-to-one model firstly, in which a supplier can receive multiple demands and a demand is dispatched to only one supplier. Suppose there are N demands and M suppliers in a dispatching round. We assume that if multiple demands are dispatched to a supplier, the supplier can respond to each demand simultaneously. That is to say, if a supplier s_1 received 3 demands $\{d_1, d_2, d_3\}$, we suppose that there will be three same suppliers $\{s_1, s_1', s_1''\}$ responding to each demand simultaneously, which means s_1 responds to d_1, s_1' responds to d_2 and s_1'' responds to d_3. The final accepted demand of s_1 is the first responded demand among 3 demands, and the other demands s_1 received get rejection. Let $P_{ij}(t)$ denote the pdf of the supplier s_j's response time to the demand d_i in a many-to-one dispatching model. $P_{ij}(T_b < t \leqslant T_{b+1})$ is the probability demand d_i is responded firstly in time range $(T_b, T_{b+1}]$ among demands supplier s_j received. Then we compute $P_{ij}(t)$ as follows.

$$P_{ij}(T_b < t \leqslant T_{b+1}) = p_{ij}(T_b < t \leqslant T_{b+1}) \times \prod_{\alpha=1,\alpha\neq i}^{N} p_{\alpha j}(t > T_{b+1})^{x_{nj}} \quad (1)$$

where the 0-1 variable x_{ij} indicates whether the demand d_i is dispatched to the supplier s_j, and $\sum_{j=1}^{M} x_{ij} \leq 1, \forall i \in [1, N]$ in many-to-one model. After we get $P_{ij}(T_b < t \leqslant T_{b+1})$, $P_{ij}(rejection)$ is represented as

$$P_{ij}(rejection) = 1 - \sum_{b=0}^{k-1} P_{ij}(T_b < t \leqslant T_{b+1}). \quad (2)$$

Note that $P_{ij}(t > T_k)$ is also a part of $P_{ij}(rejection)$, since demand's being responded in time range $(T_k, +\infty]$ firstly is also a kind of rejection. The calculation process of this probability makes no sense.

Equation (1) presents the probability that the supplier s_j accepts the demand d_i at the time range $(T_b, T_{b+1}]$ if the other demands dispatched to s_j excluding d_i are responded by the supplier beyond the time T_{b+1}. However, if two or more demands including d_i are responded by the supplier s_j in the same time range $(T_b, T_{b+1}]$, we will calculate the probability that d_i is responded at the first among these demands. Suppose that the probability is uniform in each time range, then we can get

$$P_{ij}(T_b < t \leqslant T_{b+1}) = \int_{T_b}^{T_{b+1}} (\frac{p_{ij}(T_b < t \leqslant T_{b+1})}{T_{b+1} - T_b} \times \prod_{\alpha=1, \alpha \neq i}^{N} (1 - p_{\alpha j}(t \leqslant T_b) - \frac{p_{\alpha j}(T_b < t \leqslant T_{b+1})}{T_{b+1} - T_b}(t - T_b))^{x_{\alpha j}}) dt \quad (3)$$

In equation (3), integral function means only demand d_i was responded at second t, and the response time of other demands are all beyond t. And we can get $P_{ij}(t)$ through the method of integral.

3.2.2 Response Time in Many-to-Many Dispatching Model

Based on the response time in the many-to-one model, we extend to consider the many-to-many model. Suppose the demand d_i is dispatched to more than one supplier. The response time of d_i is determined by the supplier who answers the demand at the first among all the suppliers receiving d_i. So the probability distribution of the response time in many-to-many model is calculated by the recursive equation as follows.

$$S_i(T_b < t \leqslant T_{b+1}) = 1 - \prod_{\beta=1}^{M} [1 - P_{i\beta}(t > T_{b+1})]^{x_{i\beta}} - S_i(t \leqslant T_b), \quad (4)$$

where $b = 1, ..., L-2$, $S_i(t \leqslant T_0) = 0$, and

$$S_i(t \leqslant T_b) = S_i(t \leqslant T_{b-1}) + S_i(T_{b-1} < t \leqslant T_b). \quad (5)$$

Note that $x_{i\beta}$ indicates whether the demand d_i is dispatched to the supplier s_β. $P_{i\beta}(t > T_{b+1})$ can be calculated based on Equation (3) by

$$P_{i\beta}(t > T_{b+1}) = 1 - \sum_{\gamma=0}^{b} P_{i\beta}(T_\gamma < t \leqslant T_{\gamma+1}). \quad (6)$$

3.2.3 Model Formulation

In the many-to-many model, it is challenging to determine which suppliers the demand should be dispatched to. If a demand is dispatched to many suppliers, the demand will be answered shortly since more suppliers compete to serve the demand. Meanwhile, if a supplier has received too many demands, his/her response time to a specific demand will be long. So we need to consider both cases to determine the optimal receivers for each demand.

In our dispatching model, we have two goals: one is to have as many demands as possible accepted by the suppliers, the other is to have as low response time as possible for all the accepted demands. The two objectives are in conflict with each other in a dispatching round. Suppose that in an on-demand service platform which uses one-to-many dispatching model, if we want to get a lower average response time of accepted demands, an extreme solution is to dispatch popular demands to all the suppliers. In this case, the average response time of accepted demand is the shortest, but the successful acceptance rate is almost the lowest since unpopular demands get no response in this round. On the other hand, if a higher successful acceptance rate is needed, popular demands and unpopular demands should be dispatched to suppliers together. So it is also challenging to make a trade-off between these two objectives.

In our many-to-many model, in order to get a high success rate, we can firstly sum the probabilities of each response time range for each demand d_i, named as SS_i, and then maximize the sum of SS_i among all demands in this round. However, this solution fails in reducing the average response time of accepted demands, since it considers that the acceptance of a demand in later time range has the same weight as earlier time range. In order to combine the two goals, we introduce a function $g(t)$ to indicate the benefit of the dispatching of a demand to a supplier. $g(t)$ varies depending on the response time t of a demand. If a demand could be answered within a shorter time, it will acquire a greater benefit. Since we use several time ranges to partition the response time of a demand, so $g(t)$ is defined as a piecewise function. The piecewise points of $g(t)$ are the same as T_b, and the value of $g(t)$ in each interval is a constant.

Example 2. *Suppose that the time window of the dispatching is $30sec$. The whole response time range is divided into three classes, with the time border 0, 10 and 30. In this case, $g(t)$ can be defined as (3, 1, 0), which means*

$$g(t) = \begin{cases} 3 & 0 < t \leqslant 10 \\ 1 & 10 < t \leqslant 30 \\ 0 & t > 30 \end{cases} \quad (7)$$

So in our demand dispatching system, the goal is to maximize the sum of all demands' benefits.

$$\arg\max_{x_{ij}} \sum_{i=1}^{N} G_i = g(t_i) \quad (8)$$

where G_i is the benefit of the demand d_i, and t_i is the response time of d_i. The response time obeys a probabilities distribution in Equation (4), so G_i can be calculated by

$$G_i = \sum_{b=0}^{L-1} g(T_b < t \leqslant T_{b+1}) S_i(T_b < t \leqslant T_{b+1}). \quad (9)$$

In the model, we assume that the supplier can respond to each demand concurrently. However, if we dispatch hundreds of demands to a supplier, this assumption will be not realistic. Furthermore, the solution space of the dispatching problem depends on the number of demands that a supplier has received. If we dispatch too many demands to a supplier, the complexity for finding the optimal dispatching will be huge. So we define a variable UL to limit the number of demands that a supplier can receive.

$$\forall j \in M, \sum_{i=1}^{N} x_{ij} \leqslant UL. \tag{10}$$

3.2.4 Demand Dispatching Algorithm

Algorithm 1: Many-to-Many Dispatching Algorithm

Input : The set of demands $\mathcal{D}$ and suppliers $\mathcal{S}$, the pdf of response time $p_{ij}(t)$, benefit function $g(t)$

Output: The dispatching results

1 **for** $j = 1 \rightarrow M$ **do**
2 Select a demand i from $\mathcal{D}$ for the supplier j with the probability distribution $p_{ij}(t)$ which generates the maximum benefit, and add i to $D[j]$;
3 **for** $i = 1 \rightarrow N$ **do**
4 Compute the benefit of the demand $G[i]$ with all the demands considered via Equation (9);
5 $G[0] \leftarrow$ the average of $G[i]$;
6 **for** $i \leftarrow 1, N$ **do**
7 $U \leftarrow$ suppliers that are not assigned to demand i;
8 **for** $j \leftarrow 1, len(U)$ **do**
9 $k \leftarrow U[j]$;
10 $tmp \leftarrow D[k]$;
11 $final \leftarrow D[k]$;
12 **if** $sizeof(D[k]) < UL$ **then**
13 **if** *add demand i to $D[k]$, and $G[0]$ increases* **then**
14 add i to $D[k]$;
15 $final \leftarrow D[k]$;
16 update $G[0]$;
17 $D[k] \leftarrow tmp$;
18 **for** $m = 1 \rightarrow sizeof(D[k])$ **do**
19 **for** *each combination C of m demands from $D[k]$* **do**
20 **if** *replace C with demand i, and $G[0]$ increases* **then**
21 replace C in $D[k]$ with demand i ;
22 $final \leftarrow D[k]$;
23 update $G[0]$;
24 $D[k] \leftarrow tmp$;
25 $D[k] \leftarrow final$;
26 **return** *the dispatching results $D[j]$*

The existing one-to-many dispatching problem in [6] is NP-hard. Our many-to-many dispatching model is a more generic and complex model. Correspondingly the solution space is much larger. We can prove that the many-to-many demand dispatching problem is also NP-hard. We design an efficient and fast heuristic algorithm for solving it.

Algorithm 1 presents the pseudo-code. In lines 1-2, for each supplier j, we first assign a demand i with the probability distribution $p_{ij}(t)$ which will generate the maximum benefit for j. Then we will get an initial dispatching result $D[j]$. In lines 3-5, we calculate the benefit of all the demands using Equation (9) and get the average benefit. Next, in lines 6-7, for each demand i, we find the suppliers who have not received i, and then put them in a set U. In lines 8-25, for each supplier j in U, we utilize two ways to modify $D[j]$ in order to increase the average benefits of all the demands. One way is to add the demand i into $D[j]$, which is called *adding*; the other way is to replace one or more demands in $D[j]$ with the demand i, which is named by *replacing*. In lines 12-16, we illustrate the process of *adding*. If the size of $D[j]$ is not over UL, we then check whether adding the demand to $D[j]$ increases the average benefits of the demands. If yes, we will add demand i into $D[j]$. Lines 18-24 show the process of *replacing*. The main idea of *replacing* is to find whether replacing one or more demands in $D[j]$ with demand i will increase the average benefit. If yes, we find the optimal $replacing$ strategy from all the possible ones which will maximize the average benefit. Finally, we compare the benefit of $adding$ and $replacing$, and choose the better one. By performing the above process for each demand, we get the final dispatching results.

• **Time Complexity.** The time cost of Many-to-Many Dispatching Algorithm depends on two factors: 1) the calculation process of $P_{ij}(t)$; 2) the *replacing* process (line 18 - line 24). In section 3.2.1, we use the method of integration to get $P_{ij}(t)$. In order to reduce the time cost of this process, we use compound trapezoidal formula to get the approximate value of $P_{ij}(t)$. In such a method, the number of trapezoids is an important factor to the accuracy of the result. We use Q to represent this value. The maximum time cost of calculating $P_{ij}(t)$ for one demand is $O(UL \times Q)$. The upper bound of time cost of calculating $P_{ij}(t)$ for UL demands is $O(UL^2 \times Q)$.

In the *replacing* process, the algorithm needs to traverse all the possible combinations of $D[j]$, and the maximum length of $D[j]$ is UL, so the upper bound of traverse space is $2^{UL} - 1$. For each combination traversed, the algorithm needs to calculate $G[0]$, which consists of the calculation of $P_{ij}(t)$ and $S_i(t)$. These two processes are performed sequentially, and in most instances, the time cost of $P_{ij}(t)$ is higher than $S_i(t)$, so we only consider the time cost of $P_{ij}(t)$. As a consequence, the time cost of *replacing* process is $O(2^{UL} \sum_{n=0}^{UL-1}(n+1)^2 \times Q)$. The time complexity of this algorithm is $O(M \times N \times 2^{UL} \times \sum_{n=0}^{UL-1}(n+1)^2 \times Q)$.

4 Evaluations

4.1 Environment Settings

The data sets we are using to conduct our experiment are from GoGoVan, which is a large van-calling platform in Hong Kong. The training of the response time prediction model are based on the whole data set including 170000 demand records, and about 6400 suppliers are involved.

We randomly select the demands and corresponding suppliers from the data set for the simulation of the dispatching model. In our simulations, the dispatching models are round-based. It is important to set a time window RT for a dispatching round. In [25], RT is varied in $\{5\ sec, 10\ sec, 15\ sec, 20\ sec\}$. Considering that the number of demands and suppliers is less than the data using in [25], let RT vary in $\{10\ sec, 20\ sec, 30\ sec\}$ in our experiments.

We set $L = 3$ and use a 3-$class$ classifier based on RF to calculate $p_{ij}(t)$. The reason is that the prediction accuracy of the response time decreases as L increases. If L is 2, the probability distribution of response time has no difference from the binary acceptance probability in existing approaches. We set $g(t) = (3, 1, 0)$ to calculate the benefit of a demand.

To measure the performance, we need to emulate the supplier's actual response behavior when he/she receives multiple demands. For each received demand, we randomly generate a response time of the supplier. The random response time needs to satisfy the probability distribution p_{ij} from the prediction model. The demand with the shortest response time is then finally accepted by the supplier.

We utilize two metrics to evaluate the performance of the dispatching model, which are the key metrics discussed in [6]. One is the **Success Rate (SR)**, which means the percentage of served calls in each round, the other is the **Averaged Response Time (ART)**, which represents the time it takes for a newly created demand to be accepted by a supplier. The benchmark dispatching models we use for performance comparison are as follows.

One-to-One Dispatching Model. We propose a naive demand dispatching model called the one-to-one dispatching model, in which a demand will be dispatched to one supplier, and a supplier can receive one demand at most in a round. If we treat demands and suppliers as nodes of a bigraph, this dispatching problem can be solved using a maximum weighted matching. The weight represents the benefit of a demand when it is answered by a particular supplier. The classic Kuhn-Munkres (KM) method [26] is used to solve this problem.

One-to-Many Dispatching Model. The state-of-art work [6] proposed a round-based one-to-many dispatching model. In this model, a demand will be dispatched to several suppliers, and a supplier can receive one demand at most. Although in [6], the model originally takes the prediction of acceptance probabilities as input, and generate optimal dispatching accordingly. In our experiment, for the fairness of comparison with our many-to-many model, we first evaluate the performance of the model by fitting it with the probability distribution of response time as inputs.

4.2 Single Round Performance Comparison

We first run simulations on a single round of dispatching. As the performance of the dispatching model depends on the ratio of the number of demands to suppliers [6], we set the ratio r to vary in $\{0.2, 0.4, 0.6, 0.8, 1.0, 1.25, 1.67, 2.5, 5\}$. Higher ratio represents the peak hours, in which suppliers are not sufficient. To simulate each value of ratio, we set the number of demands N to vary in $(20, 40, 60, 80, 100, 100, 100, 100, 100)$, and the corresponding number of suppliers M is $(100, 100, 100, 100, 100, 80, 60, 40, 20)$. All the demands and suppliers are randomly selected from the entire data set.

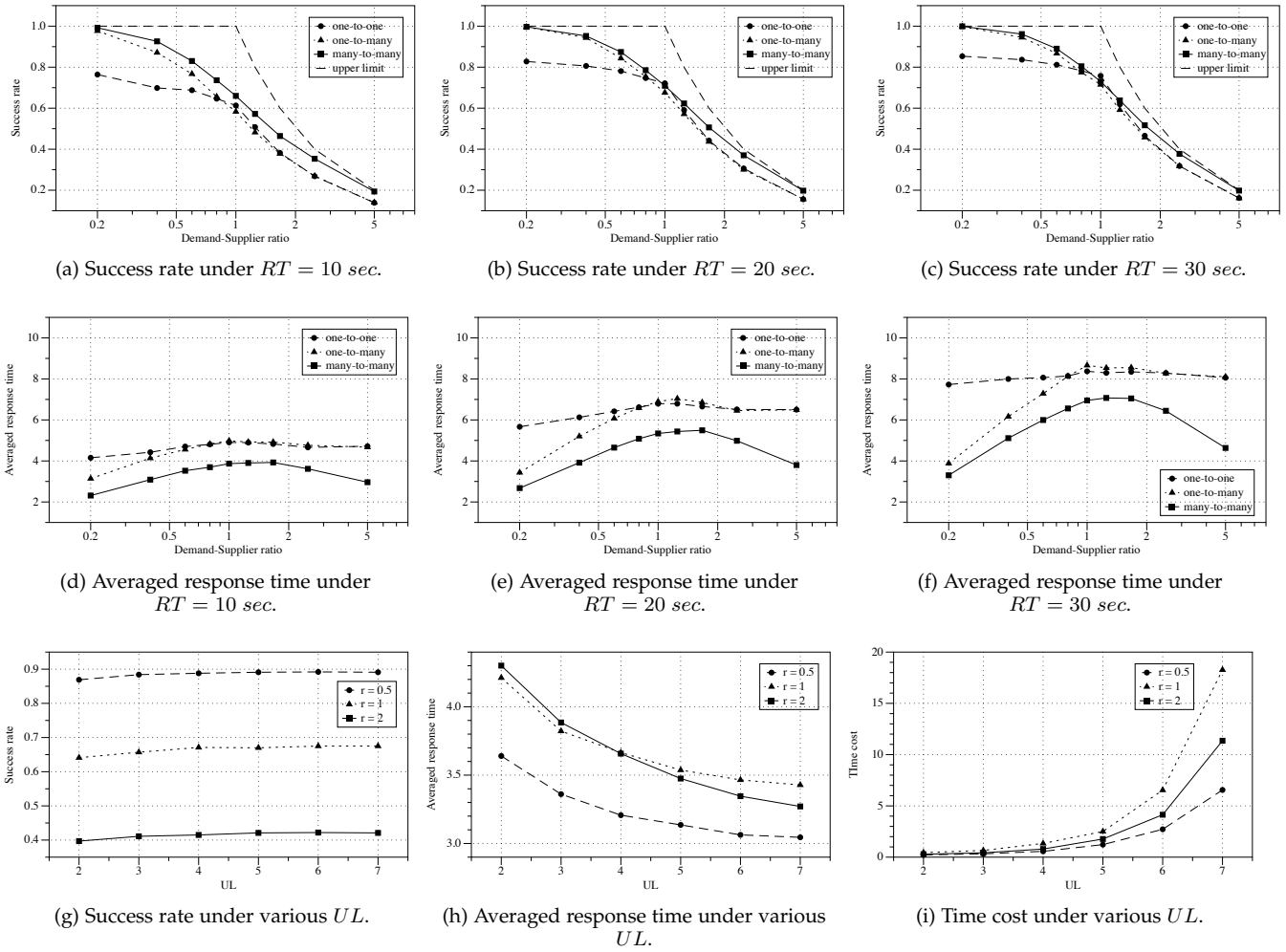
Fig.5a, 5b and 5c show the comparison of the three dispatching models in terms of SR, where the dotted lines represent the upper limits of SR. If $RT = 10\ sec$, we can find that SR of many-to-many model is better than others whatever the value of r is. The reason is that a supplier can receive one demand at most in one-to-one and one-to-many models, and the probability that a supplier does not accept the demand within a short RT is high. However, in the many-to-many model, a supplier could receive more than one demands, so the probability that the supplier rejects all of these demands is relatively low. If $RT = 20\ sec$ or $30\ sec$, the many-to-many model has obviously greater SR than the one-to-many model when r is larger than 1. When the value r is less than 1, the many-to-many model still has slightly better performance than the one-to-many model. This is because when the number of demands exceeds the number of suppliers, the one-to-many model has many demands that could not be dispatched out. Considering various values of RT and r, the many-to-many model has 11.8% higher SR on average than the one-to-many model. Especially when $r > 1$, the increase in SR is 24.6%.

We find that the many-to-many model also has much greater SR than the one-to-one model except when r is around 1. When $r = 1$, SR of the one-to-one model is the best, because the solution found in one-to-one model is optimal using KM algorithm, but the solution found in one-to-many model and many-to-many model is an approximate solution.

In the comparison of ART, we can find from Fig.5d, 5e and 5f that the many-to-many model outperforms the benchmark models regardless of the value of r and RT. Specifically compared with the one-to-many model, our model has a decrease of 24.6% in ART on average. The reason is that a supplier in our model has multiple demands to choose, the supplier is more likely to respond a demand in a shorter time. As the time window RT increases, ART will be longer because suppliers have longer time to accept a demand. SR is higher in all dispatching models if RT gets longer, since $p_{ij}(t > RT)$ becomes lower and the supplier has higher probability to accept the demand successfully.

Trade-off between the Performance and Time Cost. In the many-to-many dispatching algorithm, we use the variable UL to limit of the number of demands that a supplier can receive in order to make a trade off between performance and time cost. So we conduct an experiment to evaluate the effect of UL respectively to SR, ART and the time cost of the dispatching algorithm. We let r vary in $\{0.5, 1, 2\}$, since these values represent three common dispatching situations. To simulate each value of r, N is set to be $(50, 100, 100)$ and correspondingly M is $(100, 100, 50)$. We set $RT = 10\ sec$.

Fig.5g, 5h and 5i show how the performance of our algorithm varies depending on UL. We can see that SR and ART gets better with the increase of UL whatever the value of r is. However, the time cost of the algorithm gets high as UL increases. Since a great UL will lead to a large solution space. Although we can find a better solution in such a space, the time it takes for the algorithm will be long. We



(a) Success rate under $RT = 10\ sec.$ (b) Success rate under $RT = 20\ sec.$ (c) Success rate under $RT = 30\ sec.$

(d) Averaged response time under $RT = 10\ sec.$ (e) Averaged response time under $RT = 20\ sec.$ (f) Averaged response time under $RT = 30\ sec.$

(g) Success rate under various UL. (h) Averaged response time under various UL. (i) Time cost under various UL.

Fig. 5: Performance comparison of dispatching models: (a) (b) (c) the success rate of three dispatching models under various RT; (d) (e) (f) the averaged response time of three dispatching models under various RT; (g) (h) (i) the performance of the many-to-many dispatching algorithm under various UL.

can achieve various trade-off between the performance (SR and ART) and the time cost by choose various values of UL.

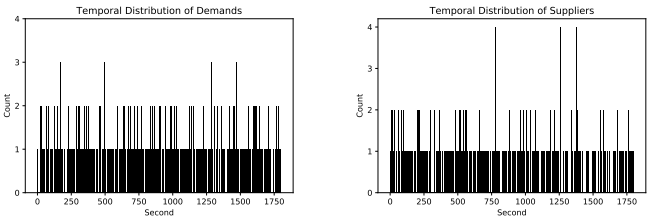
According to the results of algorithm performance, the recommended value of UL is $\lceil \frac{N}{M} \rceil + 3$, where the changes of SR and ART are small enough and the time cost is in an acceptable range. When $\lceil \frac{N}{M} \rceil$ becomes larger, in order to reduce the time cost of the algorithm, it is better to set UL to a small value, although this will make demands not fully dispatched in a round.

Impact of Modeling Approaches for the Supplier's Response Behavior. The many-to-many model uses the probability distribution of response time to represent the supplier's response behavior towards the demands. The model takes the overall response time as the optimization goal. Existing works [5] [6] utilize a supplier's acceptance probability to a demand to model his/her response behavior, and the optimization goal of the dispatching model is to maximize the overall acceptance probability of the demands. We want to evaluate the impact of the two modeling approaches of response behavior.

In our data set, due to the lack of label indicating acceptance or rejection, we use the prediction of response time to approximate the acceptance probability. Normally if we want to get such a acceptance probability, we need to train a binary classifier. So we can partition the whole response time range into 2 classes using RT and train a binary (2-$class$) classifier. As mentioned before, we train a 3-$class$ classifier by default to predict the probability distribution of response time. Actually we want to compare the performance of the dispatching model respectively under the 2-$class$ acceptance classifier and the 3-$class$ response time

TABLE 3: Performance of demand dispatching using different response behavior modeling approaches

Model	$r = 0.4$		$r = 1$		$r = 2.5$	
	SR	**ART**	**SR**	**ART**	**SR**	**ART**
Response time probability distribution						
One-to-One	0.699	4.424	0.613	4.904	0.268	4.661
One-to-Many	0.871	4.136	0.583	4.984	0.269	4.764
Many-to-Many	0.926	3.088	0.660	3.869	0.353	3.616
Acceptance probability						
One-to-One	0.753	5.429	0.646	5.531	0.279	5.253
One-to-Many	0.886	4.558	0.596	5.389	0.277	5.302
Many-to-Many	0.927	3.987	0.671	4.896	0.353	4.619



(a) The demands distribution (b) The suppliers distribution

Fig. 6: Temporal distribution of demands and suppliers between $9:30\ am$ and $10:00\ am$

TABLE 4: Performance of dispatching models in multiple rounds

Model	10 sec		20 sec		30 sec	
	ASR	ART	ASR	ART	ASR	ART
Response time probability distribution						
One-to-one	0.443	26.64	0.576	34.99	0.649	42.14
One-to-many	0.594	23.04	0.664	31.64	0.696	40.38
Many-to-many	0.680	18.33	0.719	27.07	0.749	35.45
Acceptance probability						
One-to-one	0.475	25.87	0.604	34.81	0.657	43.11
One-to-many	0.616	22.85	0.681	32.18	0.699	40.85
Many-to-many	0.681	19.15	0.730	27.99	0.755	36.13

classifier. For fairness of the comparison, the two classifiers should generate the same $p_{ij}(t > RT)$ for a specific pair of demand and supplier, which is difficult to achieve in the experiments. So use the prediction results from the 3-$class$ classifier to emulate the acceptance probability, which is equal to the sum of probabilities of the first 2 time ranges. For example, if $p_{ij} = (0.4, 0.4, 0.2)$, the acceptance rate of s_j to d_i is considered as 0.8.

We conduct an experiment with $RT = 10\ sec$ to compare the performance of the dispatching models under the two behavior modeling approaches. Table 3 shows that the results for three values of r. If we choose the response time probabilities as input to the demand dispatching, ART will be obviously shorter for all the three dispatching models than that with choosing acceptance probability. In term of S-R, the one-to-one and one-to-many models has performance decrease if we replace the acceptance probabilities with the response time probabilities as the input for dispatching. However, our many-to-many model nearly have no performance decrease in SR. The reason is that the probability that the supplier rejects all of the received demands in the many-to-many model is small enough. We can conclude that by using the response time probabilities instead of acceptance probabilities to model the supplier's response behavior, the many-to-many model can decrease ART by 21.8% while achieving the same SR.

4.3 Multi-Round Performance Comparison

We simulate the multi-round demand dispatching based on the data sets between 9:30 am and 10:00 am of a weekday. In each round, the model will dispatch the newly arrived demands and the pended demand from the previous rounds. The time window RT of a round is respectively set to $10sec$, $20sec$, and $30sec$. The value of UL is selected according to the rule mentioned before.

The temporal distributions of demands and suppliers have been shown in Fig.6a and Fig.6b, which contain 469 demands and 370 suppliers. Since the number of demands and suppliers in each round is too few in the real data set, we insert extra demands and suppliers in each round from the data sets with the same time period in a different day. If we simply increase the number of both proportionally, the difference between the number of demands and suppliers will also increase proportionally. This causes more failed demands and suppliers entering next round, and then seriously affects the dispatching in the next rounds.

In order to determine the number of extra demands and suppliers, we use an experiment-driven approach. We first triple the number of demands and suppliers, and then run the multi-round demand dispatching experiment. After the experiment, we find that the difference between demands and suppliers become much higher in the last 60% rounds, where the number of demands is far larger than suppliers. In order to control the difference within a reasonable range, we increase the number of suppliers 4.8 times in the last 60% rounds. The purpose is to ensure that the demand-supplier ratio r yields a uniform distribution over all the rounds. That is, $r < 1$ in the first 33.3% rounds, $r \approx 1$ in the middle 33.3% rounds, and $r > 1$ in the last 33.3% rounds.

With this load traces, we report averaged SR (ASR) and ART for the three models in Table 4. ASR is the average of SR of all the rounds. Results show our many-to-many model is better than the other models in all the settings of RT. With the increase of RT, ART in all settings is getting longer. Because suppliers have longer time to accept a demand and some demands may need more than one round to be accepted. And in most situations, the differences of different settings are like the analysis in section 4.2.

In comparison of the two modeling approaches for response behavior, we also have concluded that the many-to-many model has a shorter ART and almost the same ASR under the probability distribution of response time. For the other models, simply replacing the acceptance probability with response time probabilities can not improve the performance. The reason of this is explained by following two aspects: the one is that using benefits to solve the dispatching problem will cause the decrease of SR in one round, so more failed demands will need more than one round to be accepted, resulting in longer ART; the second is that when $RT = 10\ sec$, the decrease of RT in one round is too small to offset the increase caused by the first reason.

5 Discussion

• **Multi-choice Many-to-Many Dispatching Model**. In our current model, each supplier in each round can select one demand at most, and a demand will be assigned to the supplier with the shortest response time. Such setting can reduce the response time of demands, but the failed suppliers in this round can only wait for next round. In order to reduce the idle ratio of suppliers, in real application, we allow each supplier to select more than one demand in each round. That is to say, if one suppliers rst selected demand has been assigned to the other suppliers, he/she can still continue to select other demands instead of being idle in this round. Although our many-to-many dispatching model

only considers the rst response of each supplier, allowing suppliers multi-choice can still improve the successful acceptance rate and reduce suppliers idle ratio in each round. Researches on the multi-choice many-to-many model will also be a promising direction.

• **Fairness**. In our dispatching model, we mainly focus on the benefits of demands, which means the objective is to make the demands to be accepted as quickly as possible. In this situation, the supplier who answers first gets the demand is optimal. In order to attract more suppliers to join the platform, the platform should also consider the benefit of suppliers. Inspired by [10], we use the exible UL (maximum demands a supplier can received) to guarantee the fairness among to some extent. For example, we give higher UL to suppliers who did one or two trips only, and set UL of the supplier who did 10 trips to a small value. In this case, the supplier with low income today has higher probability to get his/her favorite demands and accepted it more quickly.

6 Related Work

The related works mainly include response time prediction and demand dispatching.

• **Response time prediction**. In terms of response time prediction, two groups of work are closely related. One is predicting the accurate response time, while the other is predicting which time range/period it pertains to. Wang et al proposed a prediction method for the response time based on matrix factorization [16]. The response time in this work is the same as the response time in our dispatching models. However, the response time they predicted is a specific value. It is difficult to utilize such a prediction result to optimize the demand dispatching, because existing demand dispatching models require the probability distribution as input.

Most of the related researches which predict the response time in a certain period are in the Question and Answers (Q&A) communities. Response time in such a domain is defined as the time it takes for a newly post question to be answered [16]. In [17], the whole response time range in Stack Overflow was divided into 25 time ranges using K-means, and the authors predicted in which time range it is. Avrahami et al. [18] generated various features from different settings in instant messaging, and applied a decision tree classifier to predict whether a message will receive a response within a certain period. Burlutskiy et al. formulated the prediction of response time in Stack Exchange as a binary classification task [27], and a fixed time is considered as the separation boundary of the whole response time range. Mahmud et al. [28] developed different approaches for training the model in order to predict the response times of users to questions on Twitter within specific time periods.

• **Demand dispatching**. Representative works on demand dispatching have been classified and described in Section 2. Compared with the most related model [6], our many-to-many model is a more generic dispatching approach that can achieve satisfactory performance no matter what the ratio of demands to the suppliers is. Moreover, our model takes the response time probabilities as input rather than the binary acceptance probability in existing models. This can reduce the average response time of the demands.

7 Conclusion

In this paper, we have proposed a novel many-to-many demand dispatching model for autonomous on-demand service. The model takes the probability distribution of the supplier's response time to the demand as input, and generates an optimal matching between the demands and suppliers via an optimization solver. Experiment results on the real-world data set from a large van-calling platform show that the many-to-many model has 11.8% higher successful acceptance rate than benchmark dispatching model. Especially when the suppliers are not sufficient relatively to the demands, the many-to-many model outperforms the state-of-art one-to-many model by 24.9%. In terms of average response time, the many-to-many model achieves obviously better performance than the benchmark models by 24.6%. Moreover, by replacing the acceptance probabilities with the response time probabilities as the input of demand dispatching, the many-to-many model has remarkable decrease of 21.8% in the response time, while not affecting the successful acceptance rate.

Acknowledgments

This work is supported in part by National Natural Science Foundation of China (No. 61972161), in part by the Fundamental Research Funds for the Central Universities, China (No. 2018MS53), and in part by Hong Kong RGC General Research Fund under Grant PolyU 152199/17E.

References

[1] L. Xie and M. D. Ilic. Model predictive dispatch in electric energy systems with intermittent resources. In *2008 IEEE International Conference on Systems, Man and Cybernetics*, pp.42-47, 2008.

[2] S. Muller, A. Klein and et al. Context-Aware Hierarchical Online Learning for Performance Maximization in Mobile Crowdsourcing. In *IEEE/ACM Transactions on Networking*, vol.26, no.3, pp.1334-1347, 2018.

[3] U. ul Hassan and E. Curry. A multi-armed bandit approach to online spatial task assignment. In *IEEE 11th International Conference on Ubiquitous Intelligence and Computing (UTC)*, pp.212219, 2014.

[4] Y. Tong et al. Flexible online task assignment in real-time spatial data. In *Proceedings of the VLDB Endowment*, vol.10, no.11, pp.1334-1345, 2017.

[5] L. Zheng and L. Chen. Maximizing Acceptance in Rejection-Aware Spatial Crowdsourcing. In *IEEE Transactions on Knowledge and Data Engineering*, vol.29, no.9, pp.1943-1956, 2017.

[6] L. Zhang et al. A taxi order dispatch model based on combinatorial optimization. In *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pp.2151-2159, 2017.

[7] Jung, J., Jayakrishnan, R. and Park, J. Y. Dynamic Shared-Taxi Dispatch Algorithm with Hybrid-Simulated Annealing. In *Computer-Aided Civil and Infrastructure Engineering*, pp.275-291, 2016.

[8] M. Zeng et al. QoE-Aware Power Management in Vehicle-to-Grid Networks: A Matching-Theoretic Approach. In *IEEE Trans. Smart Grid*, vol.9, no.4, pp.2468-2477, July, 2018

[9] Y. Tong, J. She, B. Ding, L. Wang, and L. Chen. Online mobile Micro-Task Allocation in spatial crowdsourcing. In *2016 IEEE 32nd International Conference on Data Engineering (ICDE)*, pp.4960, 2016.

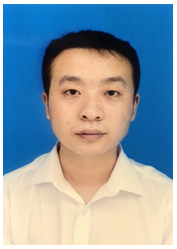
[10] C. Ho and J. Vaughan. Online Task Assignment in Crowdsourcing Markets. In *Twenty-Sixth AAAI Conference on Artificial Intelligence*, pp.4551, 2012.

[11] P. Cheng, X. Lian, L. Chen, and C. Shahabi. Prediction-based task assignment in spatial crowdsourcing. In *33rd IEEE International Conference on Data Engineering*, pp.9971008, 2017.

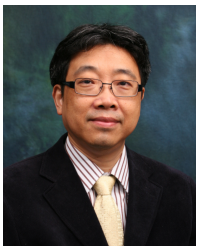
[12] H. To, C. Shahabi, and L. Kazemi. A server-assigned spatial crowdsourcing framework. In *Acm Transactions on Spatial Algorithms and Systems*, vol.1, no.1, pp.128, 2015.

[13] Cheng, Peng and Lian, Xiang and Chen, Zhao and Chen, Lei and Han, Jinsong and Zhao, Jizhong. Reliable Diversity-Based Spatial Crowdsourcing by Moving Workers. In *Proceedings of the Vldb Endowment*, vol.8, no.10, pp.10221033, 2015.
[14] J. Vermorel and M. Mohri. Multi-armed Bandit Algorithms and Empirical Evaluation. In *Proceedings of the 16th European Conference on Machine Learning*, Berlin, Heidelberg, pp. 437448, 2005.
[15] A. Alshamsi, S. Abdallah, and I. Rahwan. Multiagent self-organization for a taxi dispatch system. In *8th international conference on autonomous agents and multiagent systems*, pp.21-28, 2009.
[16] Y. Wang, J. Cao, L. He, W. Li, L. Sun, and P. S. Yu. Coupled sparse matrix factorization for response time prediction in logistics services. In *Proceedings of the 2017 ACM on Conference on Information and Knowledge Management*, pp.939-947, 2017.
[17] P. Arunapuram, J. W. Bartel, and P. Dewan. Distribution, correlation and prediction of response times in stack overflow. In *2014 International Conference on Collaborative Computing: Networking, Applications and Worksharing (CollaborateCom)*, pp.378-387, 2014.
[18] D. Avrahami and S. E. Hudson. Responsiveness in instant messaging: predictive models supporting inter-personal communication. In *Proceedings of the SIGCHI conference on Human Factors in computing systems*, pp.731-740, 2006.
[19] J. Friedman, T. Hastie, and R. Tibshirani. The elements of statistical learning. In *Springer series in statistics Springer*, Berlin, vol.1, 2001.
[20] J. H. Friedman. Greedy function approximation: A gradient boosting machine. In *Annals of Statistics*, vol.29, no.5, pp.1189-1232, 2001.
[21] X. He et al. Practical lessons from predicting clicks on ads at facebook. In *Proceedings of the Eighth International Workshop on Data Mining for Online Advertising*, pp.1-9, 2014.
[22] L. Breiman. Random Forests. In *Machine Learning*, vol.45, no.1, pp.5-32, 2001.
[23] T. Bulmer, L. Montgomery, and D. Damian. Predicting Developers IDE Commands with Machine Learning. In *Proceedings of the 15th International Conference on Mining Software Repositories*, pp.82-85, 2018.
[24] Y. Jung. Multiple predicting K-fold cross-validation for model selection. In *Journal of Nonparametric Statistics*, vol.30, no.1, pp.197-215, 2018.
[25] L. Zheng, L. Chen, and J. Ye. Order dispatch in price-aware ridesharing. In *Proceedings of the VLDB Endowment*, vol.11, no.8, pp.853-865, 2018.
[26] J. Munkres. Algorithms for the assignment and transportation problems. In *Journal of the society for industrial and applied mathematics*, vol.5, no.1, pp.32-38, 1957.
[27] N. Burlutskiy, A. Fish, N. Ali, and M. Petridis. Prediction of Users Response Time in Q&A Communities. In *2015 IEEE 14th International Conference on Machine Learning and Applications (ICMLA)*, pp.618-623, 2015.
[28] J. Mahmud, J. Chen, and J. Nichols. When Will You Answer This? Estimating Response Time in Twitter. In *Seventh International AAAI Conference on Weblogs and Social Media*, 2013.
[29] K. T. Seow, N. H. Dang, and D.-H. Lee. A collaborative multiagent taxi-dispatch system. In *IEEE Transactions on Automation Science and Engineering*, vol.7, no.3, pp.607-616, 2010.
[30] M. Maciejewski. Online taxi dispatching via exact offline optimization In *Logistyka*, vol.3, pp.2133-2142, 2014.

Lei Yang is currently an associate professor at the School of Software Engineering, South China University of Technology, China. He received the BSc degree from Wuhan University, in 2007, the MSc degree from the Institute of Computing Technology, Chinese Academy of Sciences, in 2010, and the PhD degree from the Department of Computing, Hong Kong Polytechnic University, in 2014. He has been a visiting scholar at Technique University Darmstadt, Germany from Nov. 2012 to Mar. 2013. His research interests include big data analytic, edge and cloud computing, Internet of Things with particular focus on task scheduling and dispatching.



Xi Yu is a 2nd-year master student in the School of Software Engineering, South China University of Technology, China. He got his BSc degree from Southern Medical University, China, 2017. Since September 2017 he has been a post-graduate student working in the laboratory of mobile cloud computing. His research interests include cloud computing and data mining.



Jiannong Cao is a Chair Professor of Distributed and Mobile Computing of the Department of Computing at The Hong Kong Polytechnic University. He is also the director of the Internet and Mobile Computing Lab in the department and the director of University Research Facility in Big Data Analytics. He received the B.Sc. degree in computer science from Nanjing University, China, in 1982, and the M.Sc. and Ph.D. degrees in computer science from Washington State University, USA, in 1986 and 1990 respectively. His research interests include parallel and distributed computing, wireless networks and mobile computing, big data and cloud computing, pervasive computing, and fault tolerant computing. He has co-authored 5 books in Mobile Computing and Wireless Sensor Networks, co-edited 9 books, and published over 500 papers in major international journals and conference proceedings. He is a fellow of IEEE, a member of ACM, a senior member of China Computer Federation (CCF)

Wengen Li received the B.Eng. degree and Ph.D. degree in computer science from Tongji University, Shanghai, China, in 2011 and 2017, respectively. In addition, he received a dual Ph.D. degree in computer science from the Hong Kong Polytechnic University in 2018. He is currently a Postdoctoral Fellow of the Department of Computing at the Hong Kong Polytechnic University. His research interests include spatial data management, and big data analytics for human mobility and urban logistics. He is a member of China Computer Federation (CCF) and a member of IEEE.

Yuqi Wang received the B.Sc. degree in computer science and technology from Xiamen University, China, in 2011, the M.Sc. degree in computer science and technology from Zhejiang University, China, in 2014, and the Ph.D. degree in computer science from the Hong Kong Polytechnic University in 2018. Dr. Wang currently works in Fujian Nebula Big Data Application Service Co., Ltd. His research interests include data mining, machine learning and urban computing.

Michal Szczecinski received the Master degree in IT and Econometrics from University of Szczecin in Poland. He is Head of Analytics at GoGoVan. His interest is in building strong analytics capability and using data science to contribute value to the business. Currently he focuses on optimization projects in logistics industry. Previously, he has worked in top professional services, technology and mobile gaming companies.