ACTA: Autonomy and Coordination Task Assignment in Spatial Crowdsourcing Platforms

Boyang Li
Beijing Institute of Technology
Beijing, China
liboyang@bit.edu.cn

Yurong Cheng
Beijing Institute of Technology
Beijing, China
yrcheng@bit.edu.cn

Ye Yuan
Beijing Institute of Technology
Beijing, China
yuan-ye@bit.edu.cn

Yi Yang
Beijing Institute of Technology
Beijing, China
yangyi@bit.edu.cn

QianQian Jin
Beijing Institute of Technology
Beijing, China
1176002998@qq.com

Guoren Wang
Beijing Institute of Technology
Beijing, China
wanggr@bit.edu.cn

ABSTRACT

Spatial platforms have become increasingly important in people's daily lives. Task assignment is a critical problem in these platforms that matches real-time orders to suitable workers. Most studies only focus on independent platforms that are in a competitive relationship. Recently, an emerging service model was proposed, where orders are shared with multiple similar platforms. It aims to solve the imbalance between supply and demand through cooperation. However, it faces the following main challenges: 1) Coordinating independent platforms fairly based on the limited information; 2) Building a task assignment process with personalized algorithms. In this paper, we study real applications and define the Autonomy and Coordination Task Assignment problem (ACTA) to maximize the global revenue and fairness. We propose a framework to solve ACTA that consists of public order sending, local matching, global conflict adjustment and results notification. The framework uses mid-products and public data to train a revenue estimation model to coordinate participants. We further propose dynamic weight task assignment algorithms to guarantee fairness. Through the experiments, we prove that the platforms can obtain higher revenue, which shows the effectiveness and efficiency of our work.

PVLDB Reference Format:
Boyang Li, Yurong Cheng, Ye Yuan, Yi Yang, QianQian Jin, and Guoren Wang. ACTA: Autonomy and Coordination Task Assignment in Spatial Crowdsourcing Platforms. PVLDB, 16(5): 1073 - 1085, 2023.
doi:10.14778/3579075.3579082

PVLDB Artifact Availability:
The source code, data, and/or other artifacts have been made available at https://github.com/liboyang77/ACTA/.

Proceedings of the VLDB Endowment, Vol. 16, No. 5 ISSN 2150-8097.
doi:10.14778/3579075.3579082

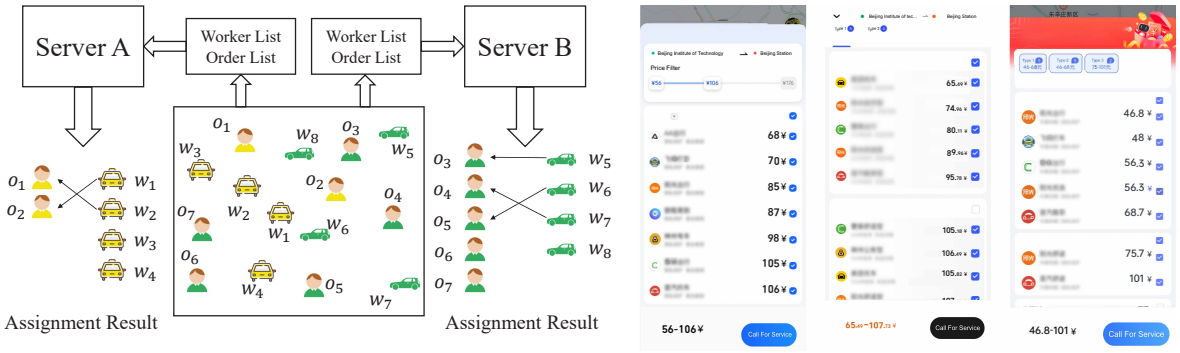
1 INTRODUCTION

Spatial platforms such as Uber[1], Lyft[2] and DiDi[3] have become popular types of spatial crowdsourcing services [15, 38]. Task assignment is a critical problem in these platforms. The goal is to assign dynamic tasks to workers who are moving freely [32, 42]. The assignment results directly influence the satisfaction of users and revenue of the platforms.

Most of the existing studies focus on the task assignment problem for independent platforms (ITA) [7, 9, 29, 34, 44]. The central servers receive real-time data, including users' orders and workers' locations. They use algorithms to find optimal assignment results. Figure 1(a) shows two ride-hailing platforms in the ITA. The central servers receive the respective order and worker data with no intersection of their matching results. Their goals may be maximizing the utility score [44], maximizing the response rate [25], minimizing the latency [10], etc. Bipartite graph matching satisfying spatial and temporal constraints is a classical solution [9]. Orders and workers are modeled as a bipartite graph. An edge will form between a worker and an order if the spatial and temporal constraints are satisfied. In recent years, multi-agent reinforcement learning (MARL) [5] has been successfully applied to task assignment problems. Workers are treated as individual agents, and the central server trains a model for all agents to guide them to serve the orders [20, 24, 30, 31].

However, the imbalance of supply and demand is an urgent problem due to the constantly increasing demands of users. In Figure 1(a), there are 7 orders and 8 workers in total. However, o_6 and o_7 will be rejected because there are no workers of ServerB nearby. Meanwhile, w_3 and w_4 of ServerA remain idle. This leads to lower user satisfaction and waste of workers. If users cannot be served by one platform, they start to carry on multiple platforms until one of them responds. If drivers are idle in one platform, they may start to work for multiple platforms, which may also lead to default caused by simultaneously accepting multiple orders. Obviously, it is very inconvenient. Cooperative task assignment (CTA) [11, 39, 40] is an emerging solution for this problem. In CTA, ride-hailing platforms form a cooperative relationship. Orders will be sent to multiple platforms that provide similar services,

[1]https://www.uber.com/
[2]https://www.lyft.com/
[3]https://www.didiglobal.com/



(a) Independent Task Assignment

(b) Cooperative Task Assignment Applications

Figure 1: An Example of Task Assignment

which increases the probability of the orders being served. Similar applications have been adopted in DiDi[3], AMAP[4], Baidu Map[5] and Meituan[6], as shown in Figure 1(b). In these applications, companies with many users often act as *leaders*. The ride-hailing platforms act as *participants*. They have their own market and can also receive orders from the *leader*. They can assign these orders to idle workers who work for them. However, several challenges must be solved.

- **Challenge 1: fairly coordinating independent platforms based on the limited information.** In real applications, a platform can only observe a part of the global information. From their perspective, they only care about how to maximize their own revenue. It is difficult to coordinate the platforms on the limited information. All existing works [11, 39] and [40] simply aim at maximizing the total utility and ignore the personality of the platforms. If the platforms feel unfair, they may withdraw from the cooperation.
- **Challenge 2: building a task assignment process with personalized algorithms.** ITAs have been widely studied, and platforms have developed their own personalized algorithms. [11, 39, 40] propose task assignment algorithms for platforms, but they require all platforms to use their proposed algorithms, which is unrealistic. The differences among the methods increase the difficulty of coordination among the platforms.

In this paper, we formally define the Autonomy and Coordination Task Assignment problem (ACTA) based on real scenarios. The key to the problem is how to fairly coordinate participants based on their local information and achieve optimal global results.

In summary, our contributions are as follows:

- We define the Autonomy and Coordination Task Assignment problem (ACTA) based on real scenarios. We study how to improve revenue and guarantee fairness by coordinating multiple spatial platforms.
- We propose a framework to solve the problem. The framework consists of four parts: public order sending, local matching, global conflict adjustment and result notification. It does not interfere with the local matching of platforms and allows them to use personalized algorithms.
- We propose a revenue estimation-based algorithm to guide spatial platforms to choose suitable orders based on expected revenue. The expected revenue can significantly reduce the conflicts between platforms. We further propose dynamic weight task assignment algorithms to adjust the parameters of different participants in the public order sending part. The algorithm can better guarantee fairness.
- We conduct experiments on real and synthetic datasets. The experimental results show that our algorithms can improve the revenue of the platforms.

The rest of this paper is organized as following. We introduce the background and related works in Section 2. The basic concepts and ACTA problem are defined in Section 3. The major parts of ACTA framework are introduced in Section 4. We propose the revenue estimation-based task assignment algorithm in Section 5. We further propose the dynamic weight task assignment algorithms in Section6. We conduct the experiments and analyze the results in Section 7. Finally, we concluded the paper and introduce the future work in Section 8.

2 RELATED WORK

In this section, we review the related work on task assignment and federated systems.

Task assignment. Task assignment is an important problem in ride-hailing platforms [12, 43, 46]. In early studies [22], offline scenarios were studied, and the problem was reduced to the bipartite matching problem with complete information of workers and orders. In recent years, the dynamic environment of spatial crowdsourcing platforms has drawn increasing attention. Online algorithms are applied in the task assignment problem. In the problem setting, orders and workers dynamically appear anywhere at any time. Temporal and spatial uncertainty introduces great challenges to task assignment problems. The online task assignment

[4]https://www.amap.com/
[5]https://map.baidu.com/
[6]https://www.meituan.com/

problem can be categorized based on different objectives. One of the most studied objectives is to maximize the total revenue. Previous studies design the online matching algorithm under the adversarial model, where the performance evaluation of the algorithm is based on the worst arriving case [3, 35]. In [3], Aggarwal et al. proposed a perturbed greedy algorithm with an optimal $(1-\frac{1}{e})$-competitive ratio on a one-side scenario where only the information of one side is unknown. Ting et al. [35] presented a randomized algorithm called Greedy-RT with a random threshold based on a two-sided scenario, where both workers and orders arrive in real time. However, the appearance probability of the worst case is very low. Random order models are designed to evaluate the online algorithm under the average arriving case, which is more practical [13, 23, 37]. [23] propose the BOM algorithm motivated by the secretary problem with a competitive ratio of $\frac{1}{e}$ in a one-side scenario. Tong et al. [37] presented the TGOA algorithm with a $\frac{1}{4}-$ competitive ratio in a two-sided scenario. Inspired by the prediction of the arrival of tasks, Dickerson et al. [13] designed an algorithm ADAP with an offline guide and proved that the competitive ratio of ADAP was 0.343. Another well-studied object is minimizing the total travel distance [6, 21, 36]. [41] studied a problem that the profit decay over time.f Kalyanasundaram et al. [21] studied an online minimum matching algorithm with an optimal $(2k-1)$ competitive ratio, where $2k$ is the number of nodes. [6] proposed the HST-reassignment algorithm based on hierarchically separated tree metrics. Tong et al. [36] gave a comprehensive experimental comparison and theoretical proof of the online minimum bipartite matching problem and showed that the competitive ratio of the greedy algorithm was 3.195. Zhao et al. [45] also considered fairness in task assignment. In their problem, they assign task based on Nash Equilibrium.

In these studies, the algorithms focused on solving one objective problem. However, maximizing or minimizing a single objective may incur the loss of other performance. Therefore, it is necessary to ensure the balance of performance in the spatial crowdsourcing problem. Recently, studies based on multi-objective online matching problems have been conducted [2, 9, 28]. [2] studied a bio-bjective online bipartite matching problem that maximized the weight and balance of the matching result. Both deterministic and randomized algorithms were proposed and achieved competitive ratios of 0.343 and 0.573. Zhao et al. [9] proposed an FETA problem with the optimization goal of maximizing the total utility and minimizing the maximum fairness cost. [28] studied a multi-objective task allocation problem in an online spatial crowdsourcing system that maximized both platform utility and worker utility.

Traditional optimization approaches can perform well in simple scenarios. With the development of multiagent reinforcement learning, more platforms have begun to use reinforcement learning to perform task assignment and manage the movement of workers [20, 24, 31, 33, 42]. Zhe et al. [42] modeled order dispatch as a large-scale sequential decision-making problem and implement reinforcement learning to optimize utility. [33] proposed a value-based dynamic learning framework to solve the task assignment problem and vehicle repositioning problem. Li et al. [24] modeled their work as POMDP to maximize total income by considering the cost of each order and potential demand of the destination. [31] proposed an LAF scheme that optimized both utility and fairness.

These studies focus on individual problems that are difficult to extend to cooperative task assignment. Cheng et al. [11] presented the first work in cross online matching. They studied order pricing to encourage partitions to accept public tasks and proposed two matching algorithms to improve the revenue of platforms. However, it faces data leakage, and the orders are processed one by one, which has a low throughput. To protect the data privacy, Wang et al. [40][39] proposed task assignment algorithms based on federated learning. They only considered how to find maximum match results but did not consider the personalization between platforms. In contrast, our problem solves a real application and overcomes the challenges in real life. We define a problem with two objectives that can both maximize the total revenue and consider the platform personalized factors.

Federated System. One of the most challenging problems in most industries is that the data are heterogeneous and are in the form of isolated islands that are difficult to collect. Several studies have designed federated systems to solve this problem. The federated system can collaborate data with multiple independent participants and provide better performance. It is successfully applied in database [16], streaming service [14], etc. Some realistic applications in IoT devices and grounding applications in industry based on federated learning can also be viewed as federated systems [1, 8, 17, 19, 26, 27]. Google [17] adopts federated learning and proposes a CIFG language model for the prediction in a mobile keyboard. It provides an alternative to the server-based data collection and training paradigm in a commercial setting. Webank [26] proposes the federated AI technology enabler (FATE), which is the first open source industrial-level framework of federated learning and has promoted implementation in credit risk control, anti-money laungering, etc. Chen et al. [8] constructed a FedHealth model and performed transfer learning to offer personalized service for health care by gathering data owned by different organizations. Jiang et al. [19] adopted federated learning in sensitive industrial data and propose an iFTM framework to train a topic model with several partitions. Abdel-Basset et al. [1] proposed Fed-TH to detect cyber threats against industrial cyber-physical systems. Meng et al. [27] combined federated learning and a graph neural network to build a spatiotemporal model that could predict the traffic flow in IoT devices.

Cloud federation is another type of federated system. It provides resource migration, resource redundancy and a combination of complementary resources in a single pool aggregated by different providers. Cloud companies in cloud federation can share resources with each [18][4]. However, computing resources are known in cloud federation, and data are transmitted through the internet. Companies do not need to consider spatiotemporal constraints. In contrast, the workers and orders in our problem are uncertain. Platforms must consider spatiotemporal constraints in the real world, which improves the difficulty of the problem.

3 PROBLEM STATEMENT

In this section, we introduce some basic concepts and illustrate the problem definition of ACTA. The notations are summarized in Table 1.

Table 1: Summary of notations

Notation	Description
$\boldsymbol{P}$	the set of participants
$\boldsymbol{T}$	the set of matching rounds
$\boldsymbol{O}$	the set of orders
$\boldsymbol{W}$	the set of workers
p	the participant
o	the order
w	the worker
$\boldsymbol{O}^t_{pub}$	the set of public orders in matching round t
$\boldsymbol{O}^t_i$	the set of inner orders in matching round t of participant p_i
$\boldsymbol{W}_i{}^t$	the set of inner workers in matching round t of participant p_i
s_o	the source location of order o
d_o	the destination location of order o
r_o	the revenue of order o
t_o	the matching round of order o
p_o	the platform of order o
l_w	the location of worker w
t_w	the matching round of worker w
rad_w	the service radius of worker w
p_w	the platform of worker w
$\mathcal{M}^t$	the set of matching results in matching round t
(o, w)	the pair of each matching result
α	the commission rate
β	the platform personalized weight

3.1 Basic Concepts

There are one *leader* and n spatial platforms as *participants* in this problem; each participant receives orders and employs workers to serve orders. Orders and workers are grouped into multiple *matching rounds* $\boldsymbol{T} = \{t_1, t_2 \cdots, t_m\}$ with a short period of time [32][31].

Definition 1 (Order). Orders are submitted by users in real time, and each order o is denoted as $< s_o, d_o, r_o, t_o, p_o >$, where s_o and d_o are the source and destination locations, respectively; r_o is the revenue when it is served; t_o is the matching round to which it belongs; p_o is the platform to which it is submitted.

Definition 2 (Worker). Workers appear in real time and can serve orders within the service radius. Each worker w is denoted as $< l_w, t_w, rad_w, p_w >$, where l_w is the location, t_w is the matching rounds, rad_w is the service radius, and p_w is the platform for which w works.

Leader does not have workers; it is responsible for sending orders to participants and receiving their matching results. Each *participant* can employee workers to serve orders. These workers are named *inner workers*. If a user submits an order to a specific participant, the order is named the *inner order* of the participant, which implies that only the inner workers of that participant can serve him or her. If the order is submitted to the leader, the leader can share it with the participants and select a worker from the participants. Orders submitted to the leader are called *public orders*. The participants can receive their inner orders and the public orders from the leader. The information of inner orders is confidential to one another. In each matching round t, the orders $\boldsymbol{O}^t$ are matched with the suitable workers $\boldsymbol{W}^t$. Public orders $\boldsymbol{O}^t_{pub}$ are sent to the participants and may be served by any idle workers. Inner orders $\boldsymbol{O}^t_i$ can only be served by inner workers W^t_i of platform p_i. $\mathcal{M}^t$ is the matching result, and each element is denoted as a pair (o, w). Thus, w moves from the current position l_w to s_o to pick up the user, and the user pays for o after arriving at d_o.

Definition 3 (Total Revenue). The total revenue is the sum of the revenue of each pair (o, w). If o is an inner order, the participants obtain the entire revenue. If o is a public order, the leader obtains αr_o as the commission, and the participants obtain $(1-\alpha)r_o$ as the return. Therefore, the total revenue is denoted as

$$Rev = \sum_{p_i \in P} rev^{in}_i + rev^{pub}_i = \sum_{t \in \boldsymbol{T}} \sum_{o \in \boldsymbol{M}_t} r_o \tag{1}$$

Our first goal is to maximize the revenue of all platforms, bu public orders' assignment strategies by different participants are unknown. It is possible for participants to reject inner orders and compete for public orders, which may be beneficial to certain participants. However, competition may lead to the decline of other participants' revenue by wasting the workers of other participants. Therefore, we must fairly balance the participants, which is the second goal. Because participants are different in scale, service quality and so on, we improve the temporary earnings fairness [31] as the platform personalized fairness to balance the participants.

Definition 4 (Platform Personalized Fairness). Platform personalized fairness is the logarithm of the average ratio of each participant's revenue to the maximum revenue on the public order.

$$SF_i = \frac{\beta_i rev^{pub}_i}{\max \beta rev^{pub}} \tag{2}$$

$$PPF = -\log \frac{\sum_{p_i \in P} SF_i}{|P|} \tag{3}$$

SF_i is the normalized fairness of a single participant, and PPF is the global fairness. We use β_i as the platform personalized weight to normalize the revenue of a participant. Unlike [31], our function aims to normalize the revenue of the participants according to their scale to ensure that the growth rate of participants is similar. When the growth rate is similar, the value of platform personalized fairness is equal to 0.

3.2 ACTA Problem

The Autonomy and Coordination Task Assignment problem (ACTA) is defined as follows:

Definition 5 (Autonomy and Coordination Task Assignment). Given a leader, participants $\boldsymbol{P}$, real-time orders $\boldsymbol{O}$, workers $\boldsymbol{W}$ and matching rounds $\boldsymbol{T}$ in a day, the goal of ACTA is to find the global matching results $\mathcal{M}$ with the maximal revenue and minimal fairness, which satisfies the following constraints:

- ***Spatial Constrain:** Workers can only serve orders within the service radius, denoted as $dis(l_w, s_o) \leq rad_w$, where $dis(x, y)$ is the distance between two locations.*
- ***Temporal Constrain:** Workers can only serve orders in the same matching round.*
- ***Unique Constrain:** Workers can serve one order, and only one order can be served by one worker in each matching round.*

- ***Belonging Constrain:*** *Inner orders can only be served by inner workers in the same platform, and public orders can be served by any workers.*

To solve ACTA, we first describe the workflow of our framework and introduce the major parts. Next, we propose a revenue estimation-based task assignment algorithm using a deep learning model to help participants estimate the expected revenue of public orders. The model reduces conflicts between participants. Furthermore, we propose dynamic weight task assignment algorithms, which aim to improve fairness while sending public orders.

4 OVERVIEW OF ACTA FRAMEWORK

In this section, we introduce a framework to solve ACTA, as shown in Fig. 2. This framework consists of four parts: public order sending (Step 1), local matching (Steps 2 and 3), global conflict adjustment (Step 4) and result notification (Step 5).

4.1 Major Parts

The major parts of the framework are as follows.

4.1.1 Public order sending. The leader sends public orders to the participants. In this step, the leader only knows the information of public orders without knowing the actual information of inner workers and orders in participants. For participants, they receive the information of public order, their own inner workers and orders.

4.1.2 Local matching. Local matching of participants is a key part of our framework. Participants do not share data with each other; they only know part of the global information. They usually adopt task assignment strategies that can maximize their own revenue. When participants want to serve public orders, they may reject some inner orders. However, there may be conflicts in which participants compete for the same public order. The conflicts are unpredictable, and only one of the participants can win the opportunity. Therefore, the conflicts are a waste of supply and can reduce the revenue of some platforms. It is crucial to reduce losses under the premise of incomplete information.

4.1.3 Global conflict adjustment. When the leader receives the matching result of public orders from participants, it starts to resolve the conflicts of the public orders. Conflicts refer to the competition among different participants for the same public order. In ACTA, the leader takes minimizing the waiting time as the principle. It selects the winner with the shortest pickup time among multiple participants; then, other participants lose.

4.1.4 Results notification. The leader notifies the final results of public orders and the matching round terminates. All participants know the corresponding matching results. For the winner, it will serve the order and earn revenue. For the losers, they only know that they have failed but do not know who is the winner. Workers work for the losers cannot serve any tasks in this round.

In ACTA, the optimal solution is to globally share all information and assign tasks by a third party using the ITA algorithms such as bipartite graph matching or reinforcement learning. However, both leaders and participants are independent companies, and they will not allow information to be shared by third parties. Therefore, each participant in ACTA intelligently makes decisions through local information. The leader and participants can only use public information or some mid-products such as embeddings and network parameters, which do not involve the original data.

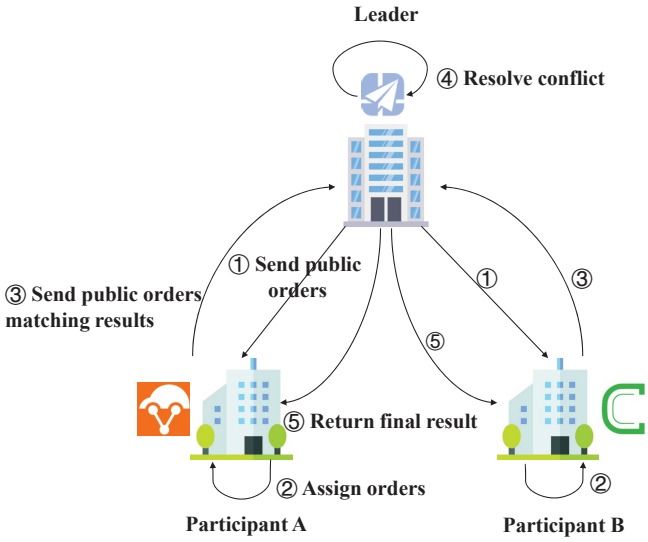


Figure 2: Workflow of ACTA Framework

Algorithm 1: ACTA Framework

Input: P, W, O, T
Output: $\mathcal{M}$
1 $M = \emptyset$
2 **foreach** $t \in T$ **do**
3 Obtain the information of public orders O^t_{pub}
4 Initialize $\mathcal{M}^t = \emptyset$, $\mathcal{M}^t_{pub} = \emptyset$
5 **foreach** $p_i \in P$ **do**
6 Obtain the information of inner orders O^t_i
7 Receive O^t_{pub} with corresponding payment
8 $M_{in}, M_{pub} = OrderDispatching(W_i, O^t_i \cup O^t_{pub})$
9 $\mathcal{M}^t = \mathcal{M}^t \cup M_{in}$
10 $\mathcal{M}^t_{pub} = \mathcal{M}^t_{pub} \cup M_{pub}$
11 **foreach** $o \in O^t_{pub}$ **do**
12 $(o, w) = ConflictAdjuct(\mathcal{M}^t_{pub}, o)$
13 $\mathcal{M}^t = \mathcal{M}^t \cup \{(o, w)\}$
14 Notify the results of o
15 **return** $\mathcal{M}$

4.2 Description of ACTA Framework

In the implementation of the framework, it sends all public orders to all participants with the revenue, which is autonomously handled by the participants. The pseudocode is shown in Algorithm 1. It first initializes the matching results (Lines 1-4). For each matching round t, the information of public orders are sent to the participants. Participants use αr_o as the revenue of each public order and obtain

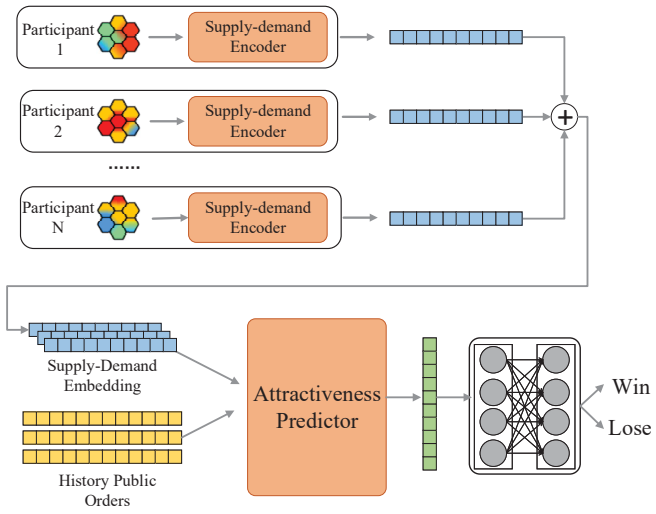


Figure 3: Architecture of Revenue Estimation Model

local assignment results. The results corresponding to public orders are put in $\mathcal{M}^t_{pub}$ (Lines 5-10). In the local matching part, we adopt the classical bipartite graph matching as the assignment algorithm, which can be extended to any self-developed algorithm. The leader resolves the conflicts and finds the most suitable worker who can arrive at the source location earliest (Lines 11-14). Finally, the global matching results are calculated (Line 15).

The matching results completely according to the interests of the participants and decision-making strategies of the leaders. Obviously, a large number of conflicts will inevitably appear in the conflict adjustment part, which will lead to a waste of service resources. The key points are to reduce the number of conflicts to improve the revenue and limit platform competition to ensure fairness. Therefore, we propose the revenue estimation-based and dynamic weight task assignment algorithms to solve the two objectives.

5 REVENUE ESTIMATION-BASED TASK ASSIGNMENT ALGORITHM

In this section, we focus on how to reduce the number of conflicts. We design a revenue estimation model based on deep learning. The model is trained over the historical assignment results. It helps participants estimate the winning probability for the public orders to guide them in the local matching part.

5.1 Revenue Estimation Model

The leader does not intend to interfere with the decision-making process of participants; they can freely process the orders. Since they do not know the decision results of other participants, this leads to conflicts. Once a conflict occurs, there must be only one winner, and other participants fail and waste the supply of the workers. One method to improve the overall revenue is to minimize the number of conflicts. Therefore, it is necessary to suggest whether participants should compete for a public order. While resolving the conflicts, the leader aims to minimize the waiting time of users. A shorter distance corresponds to more favorable workers for most cases. For some unpopular orders, it may be less sensitive to distance because fewer participants want to accept them. Simultaneously, the revenue of orders influences the popularity of orders. Therefore, the existence of conflict is determined by the spatial distribution and the attractiveness of orders.

The main idea of the revenue estimation model is to learn the relationship between distance and winning probability in conflicts for each worker. Each worker can obtain the personalized expected payment as the reward in the assignment phase. It will influence the motivation of participants to compete for public orders, which reduces the number of conflicts. For some participants with a high probability of successful competition, their reward will be closer to the original revenue. For other participants with a lower winning probability, their reward will be much lower. The output of the model is the possibility pr to win the public order for a certain worker. Therefore, participants can use the expected revenue $\mathbb{E} = pr \cdot r_o$ to replace the original revenue of the public order. The architecture of the model is shown in Figure 3. The main features are supply-demand embedding and order attractiveness.

Supply-Demand Embedding. To represent the spatial information, the hexagonal system is used to index the geographic space. Each participant embeds the supply and demand by the supply-demand encoders, and transmits the embeddings to the leader. Supply and demand determine the ability of participants to compete for public orders. Since there is a potential distribution of order prices, whether participants are willing to arrange workers to compete for a specific public order is affected by the relationship between supply and demand.

Order Attractiveness. Because the participants want to maximize their own revenue, the public orders with high payment are more attractive. Since the order revenue distribution of each participant may be different, we try to estimate the attractiveness of orders with different payment to the different participants, and finally estimate the winning probability of workers with different distances in the conflicts.

The model is trained and updated by the leader. In the training process, the participants share the supply-demand embeddings to the leader. The leader stores all historical public orders and the workers who serve them. Because the transmission is the mid-product, the leader cannot know the real difference between supply and demand, which will not lead to data leakage. In the assignment phase, the model and the mid-products are shared with the participants. The participants can calculate the expected revenue in their local servers.

5.2 Description of the Algorithm

The pseudocode of the revenue estimation-based task assignment algorithm is shown in Algorithm 2. Before the matching starts, the leader trains the estimation model and shares the parameters with all participants (Lines 2-3). For each matching round t, the participants calculate the expected revenue of the public orders to different inner workers (Line 9). Based on the expected value, Lines 5-14 of Algorithm 1 are executed in local matching, and the leader resolves the conflicts as Algorithm 1 .

Example 1. Suppose in matching round t, participant p_1 receive public order $\{o_1\}$ and inner orders $\{o_2, o_3\}$. The payment of orders is

Algorithm 2: ReACTA Algorithm

Input: P, W, O, T
Output: $\mathcal{M}$
1 $\mathcal{M} = \emptyset$
2 Train the estimation model $\mathcal{F}$
3 Share $\mathcal{F}$ to the participants
4 **foreach** $t \in T$ **do**
5 Initialize $\mathcal{M}^t = \emptyset$, $\mathcal{M}^t_{pub} = \emptyset$
6 **foreach** $p_i \in P$ **do**
7 Receive O^t_{pub} with corresponding payment
8 **foreach** $o \in O^t_{pub}$ **do**
9 Update the expected payment to each inner workers based on $\mathcal{F}$
10 Execute Lines 5-14 of Algorithm 1
11 **return** $\mathcal{M}$

$\{30, 15, 20\}$. *Worker* w_1 *can serve* o_1 *and* o_3, w_2 *can serve* o_2. *Through the revenue estimation model, the probability of* w_1 *being selected by the leader for service* o_1 *is* 80%. *Therefore, in the local matching process,* p_1 *knows that the payment of* o_1 *for* w_1 *service is 24, the payment of* o_3 *for* w_1 *service is 20, and the payment of* o_2 *for* w_2 *service is 15. The possible matching results are* $\{(o_1, w_1), (o_3, w_2)\}$. *Since* o_1 *is a public order,* (o_1, w_1) *is sent to the leader, who will ultimately decide.*

5.3 Analysis

In each matching round, the complexity of updating the expected revenue of each inner worker is $O(|\boldsymbol{W}_i^t| \times |\boldsymbol{O}^t_{pub}|)$. Then, the local matching complexity of each participant is $O(|\boldsymbol{W}_i^t| \times (|\boldsymbol{O}_i^t| + |\boldsymbol{O}^t_{pub}|))$. The complexity of conflict adjustment is $O(|\boldsymbol{O}^t_{pub}| \times |\boldsymbol{P}|)$. Therefore, the computational complexity is $O(|\boldsymbol{W}| \times |\boldsymbol{O}| + |\boldsymbol{O}| \times |\boldsymbol{P}|)$. The space complexity is $O(|\boldsymbol{W}| + |\boldsymbol{O}|)$.

The revenue estimation-based algorithm can effectively reduce the number of conflicts. However, The increase in revenue of each platform is a random process, which cannot guarantee the fairness objective. In some cases, the gap in revenue improvement of different participants may be very large. Therefore, we further propose algorithms that can both improve revenue and ensure fairness.

6 DYNAMIC WEIGHT TASK ASSIGNMENT ALGORITHM

To overcome the shortcoming of the revenue estimation-based algorithm, we propose the dynamic weight task assignment algorithms in this section. From the perspective of leaders, the algorithms dynamically adjust the commission rate and selectively balances the fairness obtained by different participants. We first introduce a normalized dynamic weight algorithm. On this basis, we propose an expected dynamic weight algorithm based on the expected total revenue.

6.1 Normalized Dynamic Weight Algorithm

To balance the revenue of participants, the main idea of the normalized dynamic weight algorithm is to update the commission rate according to the difference of participants' revenue. Different commission rates can influence the payment that a participant can earn. It normalizes the revenue gap to 0-1. The participants with low revenue use a lower rate to boost the growth of revenue. Participants with high revenue use a higher rate to slow revenue growth. By calculating the platform personalized revenue, we can rank the revenue growth from maximum to minimum. Therefore, the commission rate α_i of different participants is as follows:

$$\begin{aligned} \alpha_i &= Nor(rev_{max}, rev_{min}, rev^i_{pub}) \\ &= \Delta\alpha log(\frac{rev^i_{pub} - rev_{min}}{rev_{max} - rev_{min}} + 1) + \alpha_{min} \end{aligned} \tag{4}$$

where rev is the revenue from public orders, $\Delta\alpha$ is equal to $\alpha_{max} - \alpha_{min}$, and α_i is a dynamic value in the range $[\alpha_{max}, \alpha_{min}]$ related to the real-time revenue.

Algorithm 3: NorACTA Algorithm

Input: $P, W, O, T, \mathcal{F}$
Output: $\mathcal{M}$
1 $\mathcal{M} = \emptyset$
2 Train the estimation model $\mathcal{F}$
3 Share $\mathcal{F}$ to the participants
4 **foreach** $t \in T$ **do**
5 Calculate rev_{min} and rev_{max} according to Eq. 2
6 **foreach** $p_i \in P$ **do**
7 Update α_i according to Eq. 4
8 Execute Lines 5-10 of Algorithm 2
9 **return** $\mathcal{M}$

The pseudocode of the normalized dynamic weight algorithm is shown in Algorithm 3. When each matching round starts, the leader calculates the revenue of each participant according to the platform personalized fairness (Line 5). Later, the leader evaluates the dynamic commission rate according to Eq. 4 (Line 7). Then, the leader sends the public orders with the personalized commission rate and receives the results from participants (Line 8).

Example 2. Suppose three participants $p_1 - p_3$, *the range of* α *is* $[0.1, 0.3]$. *The current revenues of participants are* $\{200, 500, 1000\}$. *Therefore, according to Eq. 4, the* α *values of participants are* $\{0.1, 0.19, 0.3\}$. *This means that when they receive a public order with a revenue of 30, the actual payments they will get are* $\{27, 24.3, 21\}$.

In this algorithm, α dynamically changes. However, the gap of revenue may be shrinking and then enlarging. Participants with higher revenue need to wait for the revenue growth of other participants. This strict fairness seriously affects the improvement of the total revenue. Therefore, we propose an improved algorithm, named the expected dynamic weight algorithm. The algorithm can make the change in α smoother, and realize trade-off between revenue and fairness.

6.2 Expected Dynamic Weight Algorithm

Since the daily revenue of participants can be estimated, it is not necessary to adjust the commission rate frequently when the revenue increases. Based on this idea, we adopt a two-stage dynamic

weight algorithm. First, when the participants' revenue does not reach the daily expected value, we simultaneously use the normalization of the expected revenue and the normalization of the maximum and minimum gap to adjust the commission rate. The normalization of the expected revenue is the main factor, which can make the commission rate smoothly change according to the trend of the revenue. In addition, the normalization of the maximum and minimum gap is the auxiliary factor. It is used to prevent the possible imbalance when the revenue of all participants does not satisfy the expected value. Second, when the platform revenue reaches the expected value, we also adopt maximum and minimum normalization to adjust the weight. The algorithm makes the revenue grow smoothly and prevents the inequality caused by special circumstances.

The main idea is to obtain the expected revenue of each participant in a day by calculating the historical results. To meet the goal of minimum fairness, we make the revenue of each participant as close to the expected value as possible. Therefore, when the platform revenue is lower than the expected value, we use the following equation to change the value of α.

$$\alpha_i = \xi Nor(\mathbb{E}[rev], 0, rev_i) + (1-\xi) Nor(rev_{max}, rev_{min}, rev_i) \quad (5)$$

where ξ is the weight to adjust the proportion of two factors, and $\mathbb{E}[rev]$ is the expected revenue. When the expected value of earnings is overdue, we continue to use the normalized weight in Section 6.1. Thus, the fluctuation of α will be relatively small, and almost all of them will fluctuate toward the expected value, so it will be smoother.

The pseudocode of the expected dynamic weight algorithm is shown in Algorithm 4. Before the matching starts, the leader calculates the expected revenue $\mathbb{E}_i[rev]$ for each participant (Line 4). When each matching round starts, the leader calculates the revenue of each participant according to the platform personalized fairness (Lines 5-6). If the real revenue is less than the expected value, the leader evaluates the dynamic commission rate according to Eq. 5 (Line 9). Otherwise, the leader evaluates the dynamic commission rate according to Eq. 4 (Line 11). Then, the leader sends the public orders with the personalized commission rate and receives the results from participants (Line 12).

Example 3. Suppose three participants $p_1 - p_3$, the range of α is $[0.1, 0.3]$, ξ equals to 0.6. The current revenues of participants are $\{200, 500, 1000\}$. The expected revenues are $400, 800, 1200$. Therefore, according to Eq. 5, the α values of participants are $\{0.1, 0.21, 0.28\}$. This means that when they receive a public order with a revenue of 30, the actual payment they will get are $\{27, 23.7, 21.6\}$.

6.3 Analysis

In each matching round, the complexity of calculating the platform personalized revenue is $O(|\boldsymbol{P}|)$. The complexity of updating the commission rate is also $O(|\boldsymbol{P}|)$. The complexity of updating the expected revenue of each inner worker is $O(|\boldsymbol{W}_i^t| \times |\boldsymbol{O}_{pub}^t|)$, and local matching is still $O(|\boldsymbol{W}_i^t| \times (|\boldsymbol{O}_i^t| + |\boldsymbol{O}_{pub}^t|))$. The complexity of conflict adjustment is $O(|\boldsymbol{O}_{pub}^t| \times |\boldsymbol{P}|)$. Therefore, the computational complexity is $O(|\boldsymbol{W}| \times |\boldsymbol{O}| + |\boldsymbol{O}| \times |\boldsymbol{P}|)$. It must store the expected revenue of each participant; therefore, the space complexity is $O(|\boldsymbol{W}| + |\boldsymbol{O}| + |\boldsymbol{P}|)$.

Algorithm 4: DyACTA Algorithm

Input: $\boldsymbol{P}, \boldsymbol{W}, \boldsymbol{O}, \boldsymbol{T}, \mathcal{F}, \xi$
Output: $\mathcal{M}$

1 $\mathcal{M} = \emptyset$
2 Train the estimation model $\mathcal{F}$
3 Share $\mathcal{F}$ to the participants
4 Calculate the expected revenue $\mathbb{E}_i[rev]$ for each participant
foreach $t \in \boldsymbol{T}$ **do**
5 Calculate the real revenue rev_i for each participant
6 Calculate rev_{min} and rev_{max} according to according to Eq. 2
7 **foreach** $p_i \in \boldsymbol{P}$ **do**
8 **if** $rev_i \leq \mathbb{E}_i[rev]$ **then**
9 Update α_i according to Eq. 5
10 **else**
11 Update α_i according to Eq. 4
12 Execute Lines 5-10 of Algorithm 2
13 **return** $\mathcal{M}$

Table 2: Statistics of Datasets

(a) Real Datasets in Xi'an

	A	B	C	D
$\lvert\boldsymbol{O}\rvert$	–	34483	42087	52454
$\lvert\boldsymbol{W}\rvert$	–	3215	4532	5176
$\lvert\boldsymbol{O}_{pub}\rvert$	31813	–	–	–

(b) Real Datasets in Chengdu

	A	B	C	D
$\lvert\boldsymbol{O}\rvert$	–	68401	51876	52347
$\lvert\boldsymbol{W}\rvert$	–	6741	5732	5121
$\lvert\boldsymbol{O}_{pub}\rvert$	65674	–	–	–

(c) Synthetic Datasets

Parameter	Value
$\lvert\boldsymbol{W}\rvert$	1k, 2k, **5k**, 8k, 10k
$\lvert\boldsymbol{O}\rvert$	10k, 20k, **50k**, 80k, 100k
$\lvert\boldsymbol{O}_{pub}\rvert \setminus \lvert\boldsymbol{O}\rvert$	20%, 30%, **40%**, 50%, 60%
rad_w(km)	0.5, **1.0**, 1.5, 2.0, 2.5
t (sec)	30, **60**, 120, 180, 300

7 EXPERIMENTS

In this section, we verify our framework by analyzing the experimental results on real and synthetic datasets.

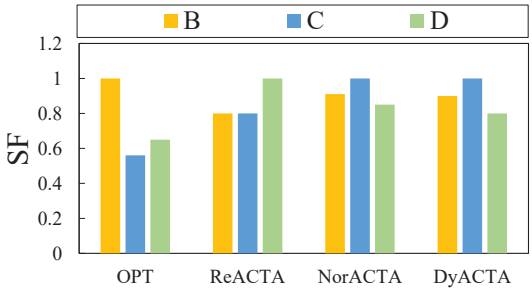
Table 3: Results on Real Datasets

(a) Results on Xi'an

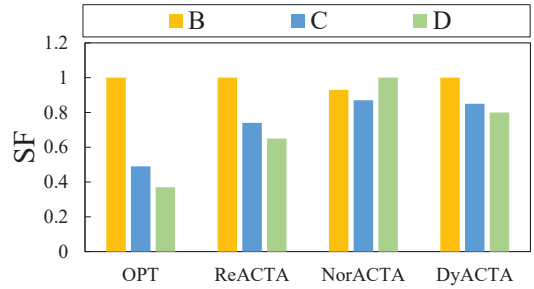
Algorithms	Total Revenue ($\times 10^6$)	PPF	Running Time(s)	Memory Cost(MB)
OPT	5.18	0.31	537.33	516.21
TGOA	3.66	–	97.21	104.42
RamCOM	4.19	–	121.40	143.91
ReACTA	4.83	0.14	1356.98	678.77
NorACTA	4.51	0.08	1347.57	647.43
DyACTA	4.72	0.11	1384.62	655.08

(b) Results on Chengdu

Metrics	Total Revenue ($\times 10^7$)	PPF	Running Time(ms)	Memory Cost(MB)
OPT	6.95	0.47	816.11	768.32
TGOA	4.47	–	134.43	176.59
RamCOM	5.81	–	233.48	250.43
ReACTA	6.52	0.22	1598.55	874.16
NorACTA	6.06	0.06	1599.95	838.58
DyACTA	6.41	0.12	1588.76	890.35



(a) Fairness of participants on Xi'an



(b) Fariness of participants on Chengdu

Figure 4: Revenue of Each Platform on Real Datasets

7.1 Setup

We use two real datasets from DiDi OpenData[7], which include the orders and worker trajectories from two cities in China, Chengdu and Xi'an. There are 4 ride-hailing platforms: Platform A is the leader, and Platforms B-D are the participants. The order data describes the source location, destination location, revenue, submit timestamp and platform. The worker trajectory data describes the location with timestamps, service radius and platform. The time interval of the matching round is set as 60 seconds. The orders and workers are grouped in matching rounds according to the timestamps. The service radius of each worker is $1.0km$. We set the default α as 0.15, and α can vary in $[0.05, 0.3]$. We generate synthetic datasets with different scales of orders and workers. We also vary the service radius and interval of matching round. The synthetic datasets are used to verify the scalability of our framework. For the parameters of the platform personalized fairness, we firstly count the market size and revenue of each platform, then set the parameters according to their scale. The statistics of the datasets are shown in Table 2.

We compare the performance among the following algorithms:

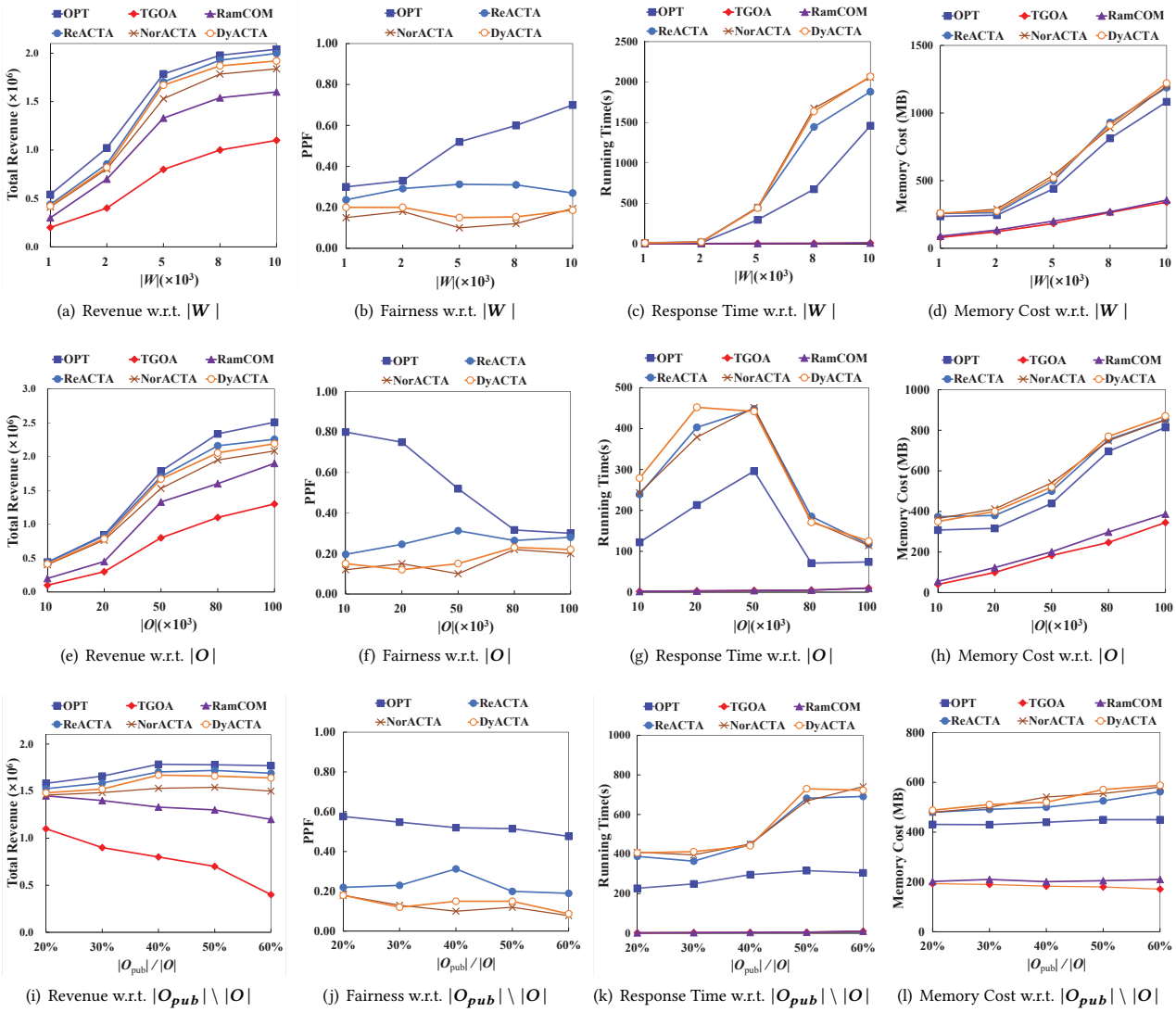
- **OPT**: A classical bipartite matching algorithm that regards all tasks and workers from the same platform.
- **TGOA [37]:** A two-phase assignment algorithm based on greedy. It is proven with a competitive ratio of $\frac{1}{4}$.
- **RamCOM [11]:** It uses a random threshold value and the expected estimated payment to find a trade-off between revenue and accept ratio.
- **ReACTA:** The revenue estimation-based task assignment algorithm proposed in this work.
- **NorACTA:** The normalized dynamic weight algorithm proposed in this work.
- **DyACTA:** The expected dynamic weight algorithm proposed in this work.

To verify the global matching results, we build all workers and orders in a bipartite graph as the **OPT** results. **TGOA** is an ITA algorithm. It is conducted on individual platforms and sums them up as the global results. **RamCOM**, **ReACTA**, **NorACTA** and **DyACTA** are cooperative algorithms. TGOA is adopted as the local matching strategy in these algorithms. We report the following metrics to verify the effectiveness and efficiency:

- **Total revenue:** the sum of the revenue of the leader and participants, as shown in Eq. 1.
- **Platform personalized fairness:** the fairness value (SF and PPF) of all participants as Eq. 2 and 3.
- **Running time:** the total time to match all the tasks.
- **Memory cost:** the peak memory cost of each algorithm.

We report the total revenue and the platform personalized fairness to verify the effectiveness. We report the running time and memory cost to verify the efficiency. Since RamCOM uses a random threshold, we execute it 10 times and report the average results. The running time of the TGOA and RamCOM is the sum of the executing time of each order. For other algorithms, it is the sum of executing time in each matching round. The experiments are conducted on a machine with Intel Xeon Silver 4110 CPU, 128-GB main memory and Nvidia RTX 3090 GPU.

[7]https://outreach.didichuxing.com/research/opendata/



(a) Revenue w.r.t. $|\boldsymbol{W}|$ (b) Fairness w.r.t. $|\boldsymbol{W}|$ (c) Response Time w.r.t. $|\boldsymbol{W}|$ (d) Memory Cost w.r.t. $|\boldsymbol{W}|$

(e) Revenue w.r.t. $|\boldsymbol{O}|$ (f) Fairness w.r.t. $|\boldsymbol{O}|$ (g) Response Time w.r.t. $|\boldsymbol{O}|$ (h) Memory Cost w.r.t. $|\boldsymbol{O}|$

(i) Revenue w.r.t. $|\boldsymbol{O_{pub}}| \setminus |\boldsymbol{O}|$ (j) Fairness w.r.t. $|\boldsymbol{O_{pub}}| \setminus |\boldsymbol{O}|$ (k) Response Time w.r.t. $|\boldsymbol{O_{pub}}| \setminus |\boldsymbol{O}|$ (l) Memory Cost w.r.t. $|\boldsymbol{O_{pub}}| \setminus |\boldsymbol{O}|$

Figure 5: Results on Synthetic Datasets

7.2 Results on Real Datasets

In this section, we conduct experiments on real datasets and analyze the results by the metrics.

Effectiveness *w.r.t* total revenue and fairness. Table 3 shows the total revenue and fairness using different algorithms. The revenue of OPT is the highest of all algorithms, which is 5.18×10^6 in Xi'an and 6.59×10^6 in Chengdu. Although OPT has achieved the best results, it ignores the differences between platforms and is impossible in real-world applications. TGOA is an ITA algorithm, it does not consider public orders, the public orders' revenue cannot be received. Therefore, TGOA is the worst in all cases. RamCOM can achieve a higher revenue because it considers hiring workers of participants to serve the public orders. Compared with TGOA and RamCOM, ACTA algorithms can significantly improve the total revenue. ReACTA can obtain the highest revenue. The revenue of NorACTA is reduced due to the strict fairness limit. The revenue of DyACTA is higher than NorACTA, which is similar to ReACTA. For Fairness, OPT has the highest PPF because it does not consider the differences between the participants. Among the ACTA algorithms, NorACTA has the lowest fairness because it has the most strict constraints on fairness. DyACTA is worse than NorACTA, but better than ReACTA. Furthermore, we analyze the fairness of different participants, and the results are shown in Fig. 4. We can observe that NorACTA's gap among participants is smaller than other algorithms. The smaller the gap, the fairer the algorithm is. According to the above experimental results, revenue is reduced when considering the fairness objective. Considering revenue and fairness at the same time, DyACTA can achieve the trade-off of the two objectives. The above results prove the effectiveness of our proposed framework.

Efficiency *with* running time and memory cost. Because the training phase of the revenue estimation model is asynchronously executed, it does not influence the efficiency in the online assignment phase. TGOA and RamCOM execute orders one by one without considering matching round, so their matching speed is far faster than other algorithms. The OPT and ACTA algorithms are processed by matching rounds. Because the local matching process uses bipartite graph matching as the matching algorithm, the running time is longer than TGOA and RamCOM. ACTA algorithms' running time is longer than that of the OPT algorithm, because the ACTA algorithms calculate the expected revenue of all orders for each worker during each matching round. As for memory, OPT and ACTA algorithms need to build bipartite graphs, and they need more memory than TGOA and RamCOM. In general, although the ACTA algorithms cost more time and memory, the average running time in each round is much shorter compared with the time span of each matching round, and the memory occupation of the algorithms is also acceptable. Therefore, our proposed framework is also sufficiently efficient.

7.3 Results on Synthetic Datasets

In this section, we conduct experiments on synthetic datasets. We analyze the results and verify the scalability, effectiveness and efficiency.

Total revenue *w.r.t* $|W|$. To verify the effect of the number of workers on the total revenue, we vary $|W|$ from $1k$ to $10k$; the results are shown in Figure 5(a). When $|W|$ is small, the number of orders is far greater than the number of workers. With the increase of $|W|$, the gap between supply and demand gradually decreases. In general, the total revenue of all the algorithms is increasing. When $|W| > 5k$, the growth rate starts to decrease. In the experiments, OPT and ACTA algorithms are always better than RamCOM and TGOA. The revenue of DyACTA is higher than NorACTA.

Total revenue *w.r.t* $|O|$. To verify the effect of the number of orders on the total revenue, we vary $|O|$ from $10k$ to $100k$, and the results are shown in Figure 5(e). The change in $|O|$ influences revenue by changing the relationship between supply and demand. The revenue of OPT and ACTA algorithms is still significantly higher than that of other algorithms. With the increase in $|O|$, all workers work at full capacity, and total revenue's growth trend becomes gentle.

Total revenue *w.r.t* $|O_{pub}|\backslash|O|$. To verify the effect of public orders' proportion on the total revenue, we vary $|O_{pub}|$ from $10k(20\%)$ to $30k(60\%)$. The results are shown in Figure 5(i). TGOA ignores public orders from the leader, and the revenue keeps decreasing because the proportion of inner orders decreases. The revenue of RamCOM is also reduced, because participants may reject public orders, resulting in a decrease in the number of orders that are served, thus reducing the revenue. The revenue of other algorithms increases when $|O_{pub}| \setminus |O| < 40\%$ and subsequently starts to decrease. When the number of public orders keeps increasing, the number of candidate orders for each worker increases. Because the number of workers is limited, many public orders cannot be served, and the total revenue begins to decrease. In contrast, if the number of public orders is too small and the number of workers is not sufficient to serve the inner orders, it also decreases the revenue.

Total revenue *w.r.t* rad_w. To verify the effect of the service radius on the total revenue, we vary rad_w from $0.5km$ to $2.5km$; the results are shown in Figure 6(a). With the increase in service radius, workers have more candidate orders, which indicates that more orders may be served, so there is an increase in total revenue.

Total revenue *w.r.t* $|t|$. To verify the effect of the time span of matching rounds on the total revenue, we vary $|t|$ from 30 seconds to 300 seconds, the results are shown in Figure 6(e). TGOA and RamCOM are not influenced by the length of the time span, and their revenue remains unchanged. A longer time span positively affects the algorithms because there are more workers and users with a high throughput. However, when the time span is larger than 120 seconds, it is counterproductive. In real applications, it is impossible to let users wait too long. A suitable time span is the trade-off between user satisfaction and global revenue.

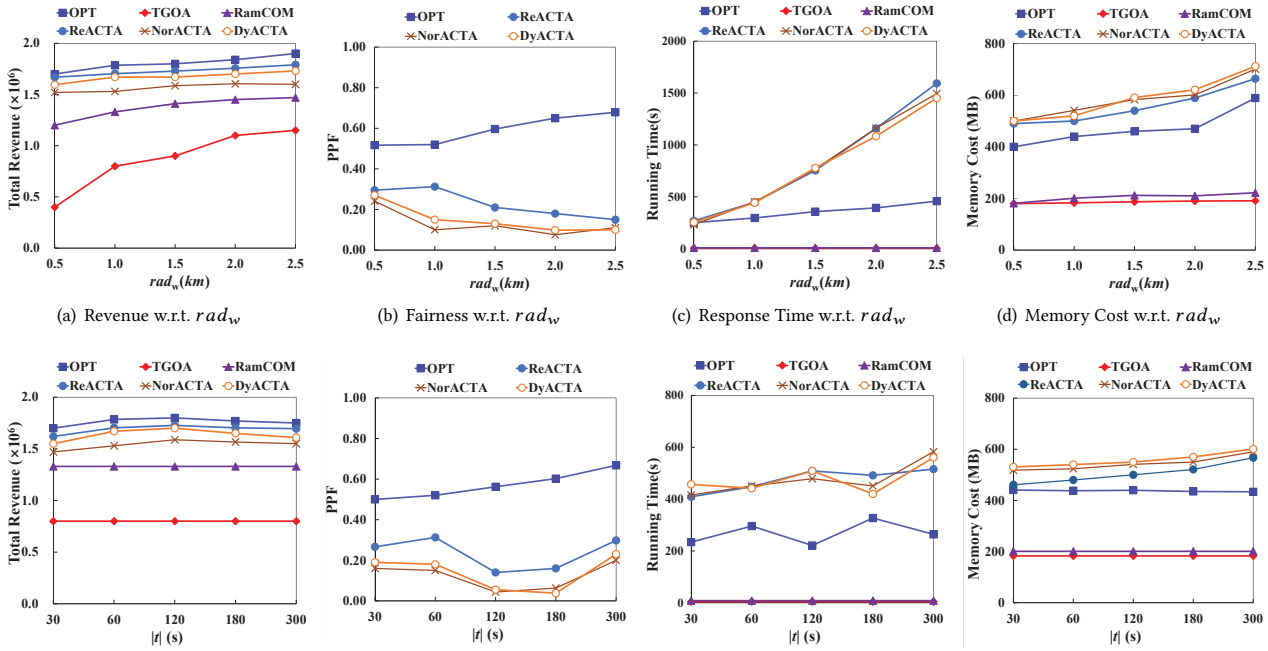
Fairness *w.r.t* $|W|$. To verify the effect of the number of workers on fairness, we vary $|W|$ from $1k$ to $10k$, and the results are shown in Figure 5(b). In this part, we only compare ACTA algorithms and the OPT algorithm. We observe that OPT and ReACTA show a random trend in the results. When $|W| < 5k$, the number of workers is small, it is difficult to strictly guarantee fairness. When $|W|$ is close to $5k$, the number of workers is relatively sufficient, and fairness can be guaranteed. When $|W| > 5k$, the fairness of NorACTA and DyACTA increases. The reason is that the supply exceeds the demand, and many workers may compete for public orders even if the commission rate is adjusted.

Fairness *w.r.t* $|O|$. To verify the effect of the number of orders on fairness, we vary $|O|$ from $10k$ to $100k$, and the results are shown in Figure 5(f). With the increase of O, the fairness of OPT gradually decreases. The reason is that there are almost no idle workers, and the revenue of all platforms increases. In all the cases, DyACTA and NorACTA are better than other algorithms and NorACTA is the best in most cases.

Fairness *w.r.t* $|O_{pub}| \setminus |O|$. To verify the effect of the proportion of public orders on fairness, we vary $|O_{pub}|$ from $10k(20\%)$ to $30k(60\%)$, and the results are shown in Figure 5(j). When the proportion of public order increases, the fairness of DyACTA and NorACTA decreases gradually. The reason is that the number of public orders is gradually increasing, and the number of inner workers who serve public orders is also gradually increasing. All participants have the opportunity to serve public orders, and the growth of their revenue becomes more fair. We still find that NorACTA is always the best among ACTA algorithms.

Fairness *w.r.t* rad_w. To verify the effect of the service radius on fairness, we vary rad_w from $0.5km$ to $2.5km$, and the results are shown in Figure 6(b). With the increase in service radius, the number of candidate workers for each public order increases. Similar with the increase of $|O_{pub}|$, workers have more candidate orders. Therefore, the fairness of ACTA algorithms decreases.

Fairness *w.r.t* $|t|$. To verify the effect of the time span of matching rounds on fairness, we vary $|t|$ from 30 seconds to 300 seconds, and the results are shown in Figure 6(f). With the increase in length of the time span, the value of fairness firstly decreases and subsequently gradually increases. Thus, a suitable time span is beneficial to fairness. Because there are few workers and orders, there is contingency for a time span that is too short. A longer time span



(a) Revenue w.r.t. rad_w (b) Fairness w.r.t. rad_w (c) Response Time w.r.t. rad_w (d) Memory Cost w.r.t. rad_w

(e) Revenue w.r.t. $|t|$ (f) Fairness w.r.t. $|t|$ (g) Response Time w.r.t. $|t|$ (h) Memory Cost w.r.t. $|t|$

Figure 6: Results on Synthetic Datasets

increases the number of workers and orders, which may increase the number of conflicts.

Running time *w.r.t* all parameters. The running time varying different parameters is shown in the third columns in Figure 5 and 6. With the increase in $|\boldsymbol{W}|$, there are more workers in each matching round, which costs more time to find an assignment result. When $|\boldsymbol{O}| < 50k$, the increase of $|\boldsymbol{O}|$ increases the size of bipartite graph, thus consuming more time. When $|\boldsymbol{O}| > 50k$, because most of workers are serving orders and the number of idle workers is small, the speed of bipartite graph matching will be faster, so the running time is reduced. The increase in rad_w, $|\boldsymbol{O_{pub}}| \setminus |\boldsymbol{O}|$ and $|t|$ requires more candidate orders for each worker. Therefore, the running time of most algorithms increases. With the increase in rad_w, the number of idle workers decreases, TGOA and RamCOM become faster. The proportion of public orders has less influence on OPT.

Memory cost *with* all parameters. The change in peak memory cost with different parameters is shown in the last columns in Figures 5 and 6. Similar to the running time, the increases in $|\boldsymbol{W}|$, $|\boldsymbol{O}|$, rad_w and $|t|$ increase the memory cost of all algorithms. When $|\boldsymbol{W}|$ and $|\boldsymbol{O}|$ increase, the algorithms need more memory to store the information. With the increase in rad_w and $|t|$, the algorithms need more memory to store the mid-products such as candidate orders. The proportion of public orders still does not influence OPT. The length of $|t|$ does not influence TGOA or RamCOM. ACTA algorithms use more memory to store the bipartite graphs in the local matching, which increases the memory cost.

8 CONCLUSION

In this paper, we study the Autonomy and Coordination Task Assignment problem (ACTA) from real scenarios. We establish a federation where we can send orders to other ride-hailing platforms. We propose a framework to solve the problem. The framework consists of four phases: public order sending, local matching, global conflict adjustment and result notification. To reduce the number of conflicts, we design a revenue estimation model to coordinate the participants. To realize global revenue and fairness, we propose the normalized dynamic weight algorithm and the expected dynamic weight algorithm. We conduct experiments on real and synthetic datasets, and the experimental results verify the effectiveness, efficiency and scalability of our framework. In future works, we will study how to better coordinate the competition between participants through game theory and find the Nash Equilibrium.

ACKNOWLEDGMENTS

Boyang Li is supposed by the NSFC (Grant No.62202046), the China Postdoctoral Science General Program Foundation (Grant No. 2018M 631358). Yurong Cheng is supposed by the NSFC (Grant Nos.61902023, U21B2007). Ye Yuan is supported by the NSFC (Grant Nos. 61932004, 62225203, U21A20516). Guoren Wang is supported by the NSFC (Grant Nos. 61732003, U2001211). Yurong Cheng is the corresponding author.

REFERENCES

[1] Mohamed Abdel-Basset, Hossam Hawash, and Karam M. Sallam. 2022. Federated Threat-Hunting Approach for Microservice-Based Industrial Cyber-Physical System. *IEEE Trans. Ind. Informatics* 18, 3 (2022), 1905–1917.

[2] Gagan Aggarwal, Yang Cai, Aranyak Mehta, and George Pierrakos. 2014. Biobjective Online Bipartite Matching. In *WINE*, Vol. 8877. 218–231.

[3] Gagan Aggarwal, Gagan Goel, Chinmay Karande, and Aranyak Mehta. 2011. Online Vertex-Weighted Bipartite Matching and Single-bid Budgeted Allocations. In *SODA*. 1253–1264.

[4] Messaouda Ayachi, Hassina Nacer, and Hachem Slimani. 2021. Cooperative game approach to form overlapping cloud federation based on inter-cloud architecture. *Clust. Comput.* 24, 2 (2021), 1551–1577.

[5] Haris Aziz. 2010. Multiagent systems: algorithmic, game-theoretic, and logical foundations by Y. Shoham and K. Leyton-Brown Cambridge University Press, 2008. *SIGACT News* 41, 1 (2010), 34–37.

[6] Nikhil Bansal, Niv Buchbinder, Anupam Gupta, and Joseph Naor. 2014. A Randomized O(log2 k)-Competitive Algorithm for Metric Bipartite Matching. *Algorithmica* 68, 2 (2014), 390–403.

[7] Xiaohui Bei and Shengyu Zhang. 2018. Algorithms for Trip-Vehicle Assignment in Ride-Sharing. In *AAAI*. 3–9.

[8] Yiqiang Chen, Xin Qin, Jindong Wang, Chaohui Yu, and Wen Gao. 2020. FedHealth: A Federated Transfer Learning Framework for Wearable Healthcare. *IEEE Intell. Syst.* 35, 4 (2020), 83–93.

[9] Zhao Chen, Peng Cheng, Lei Chen, Xuemin Lin, and Cyrus Shahabi. 2020. Fair Task Assignment in Spatial Crowdsourcing. *Proc. VLDB Endow.* 13, 11 (2020), 2479–2492.

[10] Zhao Chen, Peng Cheng, Yuxiang Zeng, and Lei Chen. 2019. Minimizing Maximum Delay of Task Assignment in Spatial Crowdsourcing. In *ICDE*. 1454–1465.

[11] Yurong Cheng, Boyang Li, Xiangmin Zhou, Ye Yuan, Guoren Wang, and Lei Chen. 2020. Real-Time Cross Online Matching in Spatial Crowdsourcing. In *ICDE*. 1–12.

[12] Alaa Daoud, Flavien Balbo, Paolo Gianessi, and Gauthier Picard. 2021. A Generic Multi-Agent Model for Resource Allocation Strategies in Online On-Demand Transport with Autonomous Vehicles. In *AAMAS*. 1489–1491.

[13] John P. Dickerson, Karthik Abinav Sankararaman, Aravind Srinivasan, and Pan Xu. 2018. Assigning Tasks to Workers based on Historical Data: Online Task Assignment with Two-sided Arrivals. In *AAMAS*. 318–326.

[14] Jeroen Famaey, Steven Latré, Ray van Brandenburg, M. Oskar van Deventer, and Filip De Turck. 2013. On the Impact of Redirection on HTTP Adaptive Streaming Services in Federated CDNs. In *AIMS*, Vol. 7943. 13–24.

[15] Srinivasa Raghavendra Bhuvan Gummidi, Xike Xie, and Torben Bach Pedersen. 2019. A Survey of Spatial Crowdsourcing. *ACM Trans. Database Syst.* 44, 2 (2019), 8:1–8:46.

[16] Ankush M. Gupta, Vijay Gadepally, and Michael Stonebraker. 2016. Cross-engine query execution in federated database systems. In *HPEC*. 1–6.

[17] Andrew Hard, Kanishka Rao, Rajiv Mathews, Françoise Beaufays, Sean Augenstein, Hubert Eichner, Chloé Kiddon, and Daniel Ramage. 2018. Federated Learning for Mobile Keyboard Prediction. *CoRR* abs/1811.03604 (2018).

[18] H. V. Jagadish, Dawei Jiang, David Maier, Beng Chin Ooi, Kian-Lee Tan, and Wang-Chiew Tan. 2014. Federation in Cloud Data Management: Challenges and Opportunities. *IEEE Trans. Knowl. Data Eng.* 26, 7 (2014), 1670–1678.

[19] Di Jiang, Yongxin Tong, Yuanfeng Song, Xueyang Wu, Weiwei Zhao, Jinhua Peng, Rongzhong Lian, Qian Xu, and Qiang Yang. 2021. Industrial Federated Topic Modeling. *ACM Trans. Intell. Syst. Technol.* 12, 1 (2021), 2:1–2:22.

[20] Jiarui Jin, Ming Zhou, Weinan Zhang, Minne Li, Zilong Guo, Zhiwei (Tony) Qin, Yan Jiao, Xiaocheng Tang, Chenxi Wang, Jun Wang, Guobin Wu, and Jieping Ye. 2019. CoRide: Joint Order Dispatching and Fleet Management for Multi-Scale Ride-Hailing Platforms. In *CIKM*. 1983–1992.

[21] Bala Kalyanasundaram and Kirk Pruhs. 1993. Online Weighted Matching. *J. Algorithms* 14, 3 (1993), 478–488.

[22] Leyla Kazemi and Cyrus Shahabi. 2012. GeoCrowd: enabling query answering with spatial crowdsourcing. In *SIGSPATIAL*. ACM, 189–198.

[23] Thomas Kesselheim, Klaus Radke, Andreas Tönnis, and Berthold Vöcking. 2013. An Optimal Online Algorithm for Weighted Bipartite Matching and Extensions to Combinatorial Auctions. In *ESA*, Vol. 8125. 589–600.

[24] Minne Li, Zhiwei (Tony) Qin, Yan Jiao, Yaodong Yang, Jun Wang, Chenxi Wang, Guobin Wu, and Jieping Ye. 2019. Efficient Ridesharing Order Dispatching with Mean Field Multi-Agent Reinforcement Learning. In *WWW*. 983–994.

[25] Jia-Xu Liu and Ke Xu. 2020. Budget-aware online task assignment in spatial crowdsourcing. *World Wide Web* 23, 1 (2020), 289–311.

[26] Yang Liu, Tao Fan, Tianjian Chen, Qian Xu, and Qiang Yang. 2021. FATE: An Industrial Grade Platform for Collaborative Learning With Data Protection. *J. Mach. Learn. Res.* 22 (2021), 226:1–226:6.

[27] Chuizheng Meng, Sirisha Rambhatla, and Yan Liu. 2021. Cross-Node Federated Graph Neural Network for Spatio-Temporal Data Modeling. In *KDD*. 1202–1211.

[28] Ellen Mitsopoulou, Iouliana Litou, and Vana Kalogeraki. 2020. Multi-Objective Online Task Allocation in Spatial Crowdsourcing Systems. In *ICDCS*. IEEE, 1123–1133.

[29] Kien Nguyen, John Krumm, and Cyrus Shahabi. 2020. Spatial Privacy Pricing: The Interplay between Privacy, Utility and Price in Geo-Marketplaces. In *SIGSPATIAL*. ACM, 263–272.

[30] Zhiwei (Tony) Qin, Xiaocheng Tang, Yan Jiao, Fan Zhang, Zhe Xu, Hongtu Zhu, and Jieping Ye. 2020. Ride-Hailing Order Dispatching at DiDi via Reinforcement Learning. *INFORMS J. Appl. Anal.* 50, 5 (2020), 272–286.

[31] Dingyuan Shi, Yongxin Tong, Zimu Zhou, Bingchen Song, Weifeng Lv, and Qiang Yang. 2021. Learning to Assign: Towards Fair Task Assignment in Large-Scale Ride Hailing. In *KDD*. 3549–3557.

[32] Tom Sühr, Asia J. Biega, Meike Zehlike, Krishna P. Gummadi, and Abhijnan Chakraborty. 2019. Two-Sided Fairness for Repeated Matchings in Two-Sided Markets: A Case Study of a Ride-Hailing Platform. In *KDD*. 3082–3092.

[33] Xiaocheng Tang, Fan Zhang, Zhiwei (Tony) Qin, Yansheng Wang, Dingyuan Shi, Bingchen Song, Yongxin Tong, Hongtu Zhu, and Jieping Ye. 2021. Value Function is All You Need: A Unified Learning Framework for Ride Hailing Platforms. In *KDD*. 3605–3615.

[34] Qian Tao, Yongxin Tong, Zimu Zhou, Yexuan Shi, Lei Chen, and Ke Xu. 2020. Differentially Private Online Task Assignment in Spatial Crowdsourcing: A Tree-based Approach. In *ICDE*. 517–528.

[35] Hing-Fung Ting and Xiangzhong Xiang. 2015. Near optimal algorithms for online maximum edge-weighted b-matching and two-sided vertex-weighted b-matching. *Theor. Comput. Sci.* 607 (2015), 247–256.

[36] Yongxin Tong, Jieying She, Bolin Ding, Lei Chen, Tianyu Wo, and Ke Xu. 2016. Online Minimum Matching in Real-Time Spatial Data: Experiments and Analysis. *Proc. VLDB Endow.* 9, 12 (2016), 1053–1064.

[37] Yongxin Tong, Yuxiang Zeng, Bolin Ding, Libin Wang, and Lei Chen. 2021. Two-Sided Online Micro-Task Assignment in Spatial Crowdsourcing. *IEEE Trans. Knowl. Data Eng.* 33, 5 (2021), 2295–2309.

[38] Yongxin Tong, Zimu Zhou, Yuxiang Zeng, Lei Chen, and Cyrus Shahabi. 2020. Spatial crowdsourcing: a survey. *VLDB J.* 29, 1 (2020), 217–250.

[39] Yansheng Wang, Yongxin Tong, and Dingyuan Shi. 2020. Federated Latent Dirichlet Allocation: A Local Differential Privacy Based Framework. In *AAAI*. 6283–6290.

[40] Yansheng Wang, Yongxin Tong, Zimu Zhou, Ziyao Ren, Yi Xu, and Weifeng Lv. 2022. Fed-LTD: Towards Cross-Platform Ride Hailing via Federated Learning to Dispatch. In *KDD*.

[41] Jinfu Xia, Yan Zhao, Guanfeng Liu, Jiajie Xu, Min Zhang, and Kai Zheng. 2019. Profit-driven Task Assignment in Spatial Crowdsourcing.. In *IJCAI*. 1914–1920.

[42] 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 *KDD*. 905–913.

[43] Boming Zhao, Pan Xu, Yexuan Shi, Yongxin Tong, Zimu Zhou, and Yuxiang Zeng. 2019. Preference-Aware Task Assignment in On-Demand Taxi Dispatching: An Online Stable Matching Approach. In *AAAI*. 2245–2252.

[44] Yan Zhao, Jiannan Guo, Xuanhao Chen, Jianye Hao, Xiaofang Zhou, and Kai Zheng. 2021. Coalition-based Task Assignment in Spatial Crowdsourcing. In *ICDE*. 241–252.

[45] Yan Zhao, Kai Zheng, Jiannan Guo, Bin Yang, Torben Bach Pedersen, and Christian S Jensen. 2021. Fairness-aware Task Assignment in Spatial Crowdsourcing: Game-Theoretic Approaches. In *ICDE*. 265–276.

[46] Yan Zhao, Kai Zheng, Hongzhi Yin, Guanfeng Liu, Junhua Fang, and Xiaofang Zhou. 2022. Preference-aware task assignment in spatial crowdsourcing: from individuals to groups. *TKDE* 34, 7 (2022), 3461–3477.