Fed-LTD: Towards Cross-Platform Ride Hailing via Federated Learning to Dispatch

Yansheng Wang
SKLSDE Lab, Beihang University
Beijing, China
arthur_wang@buaa.edu.cn

Yongxin Tong
SKLSDE Lab, Beihang University
Beijing, China
yxtong@buaa.edu.cn

Zimu Zhou
Singapore Management University
Singapore, Singapore
zimuzhou@smu.edu.sg

Ziyao Ren
SKLSDE Lab, Beihang University
Beijing, China
ziyaoren@buaa.edu.cn

Yi Xu
SKLSDE Lab, Beihang University
Beijing, China
xuy@buaa.edu.cn

Guobin Wu
Didi Chuxing Inc.
Beijing, China
wuguobin@didiglobal.com

Weifeng Lv
SKLSDE Lab, Beihang University
Beijing, China
lwf@buaa.edu.cn

ABSTRACT

Learning based order dispatching has witnessed tremendous success in ride hailing. However, the success halts within individual ride hailing platforms because sharing raw order dispatching data across platforms may leak user privacy and business secrets. Such data isolation not only impairs user experience but also decreases the potential revenues of the platforms. In this paper, we advocate federated order dispatching for cross-platform ride hailing, where multiple platforms collaboratively make dispatching decisions without sharing their local data. Realizing this concept calls for new federated learning strategies that tackle the unique challenges on effectiveness, privacy and efficiency in the context of order dispatching. In response, we devise Federated Learning-to-Dispatch (Fed-LTD), a framework that allows effective order dispatching by sharing both dispatching models and decisions while providing privacy protection of raw data and high efficiency. We validate Fed-LTD via large-scale trace-driven experiments with Didi GAIA dataset. Extensive evaluations show that Fed-LTD outperforms single-platform order dispatching by 10.24% to 54.07% in terms of total revenue.

CCS CONCEPTS

• **Applied computing** → **Transportation**.

KEYWORDS

Ride Hailing; Order Dispatching; Federated Learning

KDD '22, August 14–18, 2022, Washington, DC, USA

ACM ISBN 978-1-4503-9385-0/22/08…$15.00
https://doi.org/10.1145/3534678.3539047

ACM Reference Format:
Yansheng Wang, Yongxin Tong, Zimu Zhou, Ziyao Ren, Yi Xu, Guobin Wu, and Weifeng Lv. 2022. Fed-LTD: Towards Cross-Platform Ride Hailing via Federated Learning to Dispatch. In *Proceedings of the 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining (KDD '22), August 14–18, 2022, Washington, DC, USA.* ACM, New York, NY, USA, 11 pages. https://doi.org/10.1145/3534678.3539047

1 INTRODUCTION

Ride hailing has become a prevailing means of transport and there have been increasing numbers of ride hailing platforms including Didi, Uber, Lyft, etc. The central problem for these platforms is how to assign successive taxi orders to appropriate taxi drivers, also known as *order dispatching*. Typically modeled as bipartite graph matching, order dispatching is conventionally solved by combinatorial optimization [8, 18, 19, 23]. More recently, the status quo has shifted to data-driven solutions like reinforcement learning (RL) due to its high adaptability in handling real-world mobility dynamics [14, 15, 20, 24, 28].

Despite the success of learning based order dispatching, we argue that its effectiveness is still restricted by the *data isolation problem* in real-world ride hailing applications. It is common that multiple ride hailing platforms operate in the same city, and a large platform may consist of several affiliated taxi companies. Each company or platform often functions and manages its own data independently. Their data may contain private information of their own customers such as positions and travel records, or business secrets such as their order and driver distributions. Accordingly, these data cannot be aggregated or shared across platforms freely, resulting in the dispatching models trained and the corresponding dispatching decisions made in isolation. For example, a passenger hails a ride on platform A during the rush hour, but there is no driver nearby, and the order is canceled. However, another platform B may have idle drivers at her position. Due to the data isolation problem, the order cannot be taken by B and its excess supply resources are simply wasted. In summary, the data isolation problem impairs

not only the potential revenues of the platforms but also the user experience in ride hailing.

To break the data isolation problem in ride hailing, we introduce *federated order dispatching (FOD)*, a federated learning formulation dedicated to cross-platform ride hailing, where multiple ride hailing platforms collaboratively make dispatching decisions without sharing their raw data. However, realizing this concept faces unique challenges over general FL [12, 29] in terms of effectiveness, privacy, and efficiency.

- *Challenge 1: how to maximize the total utility (*e.g.*, revenue) of order dispatching in cross-platform ride hailing?* As we will show in Sec. 2, the effectiveness of cross-platform ride hailing relies heavily on a global view of dispatching decisions. General FL schemes aim to aggregate local models *e.g.*, gradients to jointly learn a model. It is unknown how to effectively aggregate local dispatching decisions, which are often bipartite graphs, in a federated setting.
- *Challenge 2: how to achieve privacy-preserving and high-efficiency in federated order dispatching?* Although sharing dispatching decisions increases total utility, it also poses extra privacy risks and computation/communication workload, which may impair the user experience in ride hailing. Therefore, we need dedicated privacy mechanisms and efficiency optimization for federated order dispatching.

To address the above challenges, we propose a novel solution framework named Federated Learning-to-Dispatch (Fed-LTD). It tackles the first challenge with a federated learning pipeline that shares not only the dispatching models but also the dispatching decisions, where the decisions are aggregated in the form of residual bipartite graphs. Furthermore, Fed-LTD handles the second challenge with a series of privacy protection and efficiency optimization techniques for both model and decision aggregation such that the distributions of orders and drivers cannot be inferred across platforms and the order requests can be responded in time. Evaluations on real data show that our solution has an improvement of 10.24% ~54.07% in terms of total revenue compared with single-platform order dispatching, and our privacy preservation scheme runs 10x faster than homomorphic encryption based approach. The main contributions of this paper are as follows.

- We are the first to formally define the federated order dispatching (FOD) problem for cross-platform ride hailing.
- We devise the novel solution framework, Fed-LTD. It focuses on aggregating the information from local dispatching models and dispatching decisions simultaneously. Novel privacy preserving and efficiency optimization techniques are designed to make Fed-LTD a practical solution.
- To evaluate our method, we build a simulator based on DiDi's real data and conduct extensive experiments on it. Results validate the effectiveness and efficiency of Fed-LTD.

The rest of the paper is organized as follows. We formally define the federated order dispatching problem in Sec. 2. The details of the proposed solution framework Fed-LTD are presented in Sec. 3. We report our experimental results in Sec. 4. Finally, we review related work in Sec. 5 and conclude in Sec. 6.

2 PROBLEM STATEMENT

In this section, we will first introduce some preliminaries of traditional order dispatching. Then we will introduce the federated order dispatching (FOD) problem.

2.1 Order Dispatching

We will give some general definitions first.

Definition 1 (Driver Set). *U is a set of drivers where each element $u \in U$ represents a driver. $u.loc$ is the location of the driver.*

Definition 2 (Order Set). *V is a set of orders where each element $v \in V$ represents an order. $v.origin$ and $v.destination$ are the current position and destination of the order respectively. $v.reward$ is the revenue of the order.*

The driver set and order set can form a bipartite graph $G = (U \cup V, E)$, where each edge $e = (u, v) \in E$ has edge weight $w(u, v) = v.reward$. The edges will be pruned when the distance between $u.loc$ and $v.origin$ exceeds a threshold R.

Definition 3 (Matching Allocation). *$\mathcal{M}$ is a matching allocation (or order dispatching results) over a bipartite graph $G = (U \cup V, E)$. It is a set of driver-order pairs where each element (u, v) has to satisfy: (1). $u \in U, v \in V$, (2). u and v only appear once in $\mathcal{M}$. We further define the utility function which calculates the sum of edge weights in $\mathcal{M}$,* i.e.*,*

$$SUM(\mathcal{M}(G)) = \sum_{(u,v)\in\mathcal{M}} w(u, v).$$

Given the bipartite graph G, to find a matching allocation $\mathcal{M}$ that can maximize $SUM(\mathcal{M}(G))$ is the classical maximum bipartite matching problem, which can be solved by the Hungarian method [11] in polynomial time. In real order dispatching scenarios, the orders and drivers arrive in an online manner [19]. Batch-based model is commonly used in such scenario [8, 24, 28], and the order dispatching problem can be defined as follow.

Definition 4 (Order Dispatching Problem). *Given a batch sequence $\langle 1, 2, \cdots, T\rangle$, in each batch t, the arrived drivers and orders since the last batch can form a bipartite graph $G^{(t)}$. The order dispatching problem is to decide matching allocations $\mathcal{M}^{(t)}$ for each batch to maximize the sum of utility,* i.e.*,*

$$\max \sum_{t=1}^{T} SUM(\mathcal{M}^{(t)}(G^{(t)})).$$

A simple solution is to conduct maximum bipartite matching in each batch. With the help of large-scale historical data, reinforcement learning (RL) based solutions [14, 15, 20, 24, 28] are commonly applied to achieve better performance. In these approaches, a value function is usually learned as the dispatching model, and decisions are made according to it. We adopt the state-of-the-art order dispatching approach in [20] as the local operator in our solution, and focus on aggregating the local dispatching models and decisions simultaneously in the federated setting. Next, we will define the problem of federated order dispatching.

2.2 Federated Order Dispatching

We first define some basic concepts of federated setting. A federation of ridesharing platforms consists of K platforms (or parties) $P_1, P_2, \cdots, P_K$ and a server S. The server can play as either a coordinator without any data or one of the parties. We assume that each party is semi-honest and the server S is untrusted. Party P_k has a local bipartite graph $G_k = (U_k \cup V_k, E_k)$ where the nodes are its own drivers and orders. We denote G as the global bipartite graph in the non-federated setting. Next, we introduce the concept of *global optimum*.

Definition 5 (Global Optimum). *Given the bipartite graph G and a batch sequence, the global optimum is defined as*

$$\mu(G) = \max_{\langle \mathcal{M}^{(t)} \rangle} \sum_{t=1}^{T} SUM(\mathcal{M}^{(t)}(G^{(t)})).$$

Global optimum can be considered as the optimal matching results in non-federated setting. Correspondingly, we can define the summation of local optimum as below.

Definition 6 (Summation of Local Optimum). *Given a federation of K parties, the summation of local optimum is defined as*

$$\mu(G_{LS}) = \sum_{k=1}^{K} \mu(G_k),$$

where G_{LS} can be considered as the union of all local bipartite graphs.

The purpose of federated order dispatching is to find a point between the summation of local optimum and the global optimum with privacy-preserving information sharing. The information sharing strategy of each party can be defined as a sequence of subgraphs $\langle \mathcal{G}_1, \cdots, \mathcal{G}_K \rangle$ with $\mathcal{G} = \cup_{k=1}^{K} \mathcal{G}_k$. For the global optimum, $\mathcal{G}_k = G_k$ and for the summation of local optimum, $\mathcal{G}_k = \emptyset$. Finally the problem of federated order dispatching can be defined as:

Definition 7 (Federated Order Dispatching Problem (FOD)). *Given a federation with K parties and a batch sequence. Each party has a local bipartite graph G_k . The federated order dispatching problem (FOD) is to find some information sharing strategy $\langle \mathcal{G}_1, \cdots, \mathcal{G}_K \rangle$ with $G_{Fed} = G_{LS} \cup \mathcal{G}$, so that*

$$\mu(G_{Fed}) - \mu(G_{LS}) > \Delta. \quad (1)$$

Our objective is that Δ should be as large as possible, so that federated order dispatching can achieve similar performance to the global optimum. The privacy constraint should also be satisfied, *i.e.*, each party's local data of orders and drivers should not be leaked to others. Meanwhile, we expect that the solution should run with high efficiency to support real-time response.

Next, we will introduce our solution framework, Fed-LTD.

3 FED-LTD FRAMEWORK

In this section, we first give an overview of the proposed Federated Learning-to-Dispatch (Fed-LTD) framework, then introduce the details of each step respectively.

3.1 Overview

The Fed-LTD framework is illustrated in Fig. 1. It takes orders and drivers in each batch as the input and iteratively conducts the learning and dispatching procedures with collaboration between the server and all parties. It mainly has the following 3 steps.

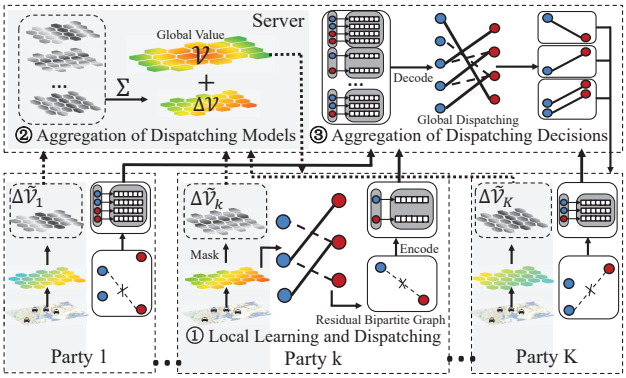


Figure 1: Overview of the Fed-LTD framework.

- *Local learning and dispatching.* It follows the scheme in [20]. Each party will learn a value function locally and make dispatching decisions based on it.
- *Aggregation of dispatching models.* In this step, federated learning is applied to aggregate the dispatching models (*i.e.*, value functions). We also consider privacy preservation and efficiency optimization techniques during this procedure.
- *Aggregation of dispatching decisions.* Only sharing the models is not sufficient as the dispatching decisions play a more important role in order dispatching. They can also be shared via the aggregation of local residual bipartite graphs (*i.e.*, unmatched nodes by each party). In this step we will propose novel algorithms to ensure the privacy and efficiency during the aggregation of decisions.

Next, we will present the technical details of each step respectively. The algorithm for one batch is shown in Algorithm 1.

3.2 Local learning and dispatching

We adopt the state-of-the-art dispatching algorithm in [20] as the local dispatching operator in Fed-LTD. It plays the same role as local SGD in FedAvg [12], which can also be replaced by other RL-based dispatching algorithms [14, 15, 28].

It models the drivers as agents, their geometrical locations (represented by hexagon grids) as states, deciding which order to take or remaining idle as actions, and the value function is the expected accumulated rewards from some state, *i.e.*,

$$\mathcal{V}(s^{(t)}) = \mathbb{E}[\sum_{t} r^{(t)} | s^{(t)}],$$

where $s^{(t)}$ is the state vector and $r^{(t)}$ is the sum of rewards in batch t. The value function is updated by the following Bellman equation:

$$\mathcal{V}(s^{(t)}) \leftarrow \mathcal{V}(s^{(t)}) + \alpha_l \cdot \sum_{u} (r_u^{(t)} + \gamma \mathcal{V}(s_v^{(t+1)}) - \mathcal{V}(s_u^{(t)})), \quad (2)$$

where u and v are the driver and the order, α_l is the learning rate and γ is the discounting factor. Afterwards, dispatching decisions are made by each party based on the learned values. The expected rewards in the future can be encoded into the edge weights of

Algorithm 1: Fed-LTD Algorithm: One Batch

input : Bipartite graphs $G_1, \cdots, G_K$ of this batch;
Global value function $\mathcal{V}$ of this batch;
output : Matching allocations of this batch;

Party k:
  //*Local learning and dispatching*
  Update local value function $\mathcal{V}_k$ according to (2) ;
  $\Delta\mathcal{V}_k \leftarrow \mathcal{V}_k^{(t)} - \mathcal{V}_k^{(t-1)}$;
  $\Delta\tilde{\mathcal{V}}_k \leftarrow Encode(\Delta\mathcal{V}_k)$ according to (4);
  Update the edge weights according to (3) ;
  Conduct matching algorithm and obtain $\mathcal{M}(G_k)$;
  $G_{\Delta_k} \leftarrow G_k - \mathcal{M}(G_k)$;
  $\tilde{G}_{\Delta_k} \leftarrow EncodeRBG(G_{\Delta_k})$;
  Send $\Delta\tilde{\mathcal{V}}_k, \tilde{G}_{\Delta_k}$ to Server;
Server:
  //*Aggregation of dispatching models*
  if *Synchronization happens* **then**
    $\Delta\mathcal{V} \leftarrow \sum_{k=1}^{K} \Delta\tilde{\mathcal{V}}_k$;
    $\mathcal{V} \leftarrow \mathcal{V} + \Delta\mathcal{V}$;
  //*Aggregation of dispatching decisions*
  $G_\Delta \leftarrow DecodeRBG(\tilde{G}_{\Delta_1}, \cdots \tilde{G}_{\Delta_K})$;
  Conduct matching algorithm and obtain $\mathcal{M}'(G_\Delta)$;
  Send $\mathcal{V}$,$\mathcal{M}'(G_\Delta)$ to each party;

bipartite graphs by adding the TD error:

$$w(u,v) = v.reward + \gamma\mathcal{V}(s_v^{(t+1)}) - \mathcal{V}(s_u^{(t)}). \tag{3}$$

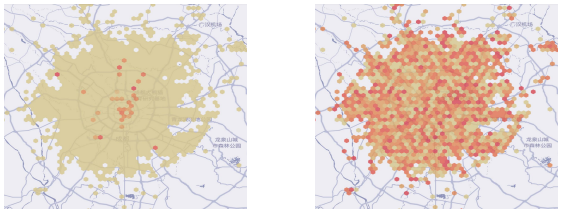
After rebuilding the bipartite graph, a Hungarian method based maximum weighted bipartite matching [11] can be conducted and the local dispatching decisions can be finally made.

Each party can run the local step independently and the server will make a summation of their rewards, which becomes the baseline (*i.e.*, LocalSum) in our experiments. Next we will introduce the key steps of Fed-LTD, *i.e.*, the aggregation of dispatching models and dispatching decisions, which aims to improve the performance in the federated setting via privacy-preserving collaboration.

3.3 Aggregation of Dispatching Models

This step aims to aggregate the dispatching models, *i.e.*, the learned value functions of each party by the server. A very simple aggregating method is to take the average of each party's value table. However, it ignores the following two problems: *(1). The privacy of each party may be leaked during aggregation; (2). The communication cost can be high as the value table is large.* To solve the problems, we design a random masking based private aggregation scheme with delayed synchronization. The details are described as follows.

Private aggregation with random masking. We notice that if all the updating values are uploaded to the server without any preservation, the server can simply infer which grid has been updated. And the location distribution of drivers and orders may be leaked. To obfuscate which grid has been updated, we use random masking [2] to preserve the privacy of value updates. More specifically, suppose $\Delta\mathcal{V}_k$ is the updated value table of party k. The proposed



(a) $\Delta\mathcal{V}_k$ **(without masking)** **(b)** $\Delta\tilde{\mathcal{V}}_k$ **(with masking)**

Figure 2: Example of value updates with/without masking.

random masking approach will perturb values in $\Delta\mathcal{V}_k$ by:

$$\Delta\tilde{\mathcal{V}}_k = \Delta\mathcal{V}_k + \sum_{k'<k} PRG(sd_{k,k'}) - \sum_{k'>k} PRG(sd_{k',k}), \tag{4}$$

where $PRG(\cdot)$ is a pseudorandom generator and $sd_{k,k'}$ is a random seed generated by party k and k' with a key agreement algorithm (*e.g.*, Diffie-Hellman key agreement). The approach is lossless for value aggregation as we have $\sum_k \Delta\tilde{\mathcal{V}}_k = \sum_k \Delta\mathcal{V}_k$. We illustrate the updated values of a party with and without random masking (*i.e.*, $\Delta\tilde{\mathcal{V}}_k$ and $\Delta\mathcal{V}_k$) in Fig. 2. We can observe that without random masking it is easy to infer which grid has been updated. But after the random masking, sensitive information can hardly be inferred.

Delayed synchronization. Another major challenge is the high communication cost between the server and each party. We optimize the communication efficiency by delayed synchronization of values. Although all parties update their local values in each batch, it is unnecessary to aggregate the values with the same frequency, as the values predict future trends approximately and their effects can be in delay. By this intuition, we suppose that the server can make a delayed synchronization of values for every t_d batches, so that the communication cost can be effectively reduced by $1/t_d$.

3.4 Aggregation of Dispatching Decisions

In this part, we further investigate how the parties can collaboratively make the dispatching decisions together under privacy preservation. The basic idea is very simple. After the local dispatching step at the end of a batch, there will be some unmatched nodes in each party, which we call the *Residual Bipartite Graph (RBG)*. We expect to aggregate them by an additional round of global matching and the orders will belong to the party of their matched drivers. Intuitively, the aggregation of decisions can increase the total revenue of each party. But sharing each party's local RBGs directly may result in leaks of privacy like the positions of orders and drivers. Next we will present our solution with two steps: rebuilding global bipartite graph and obfuscating the edge weights.

Rebuilding global bipartite graph with private hashing. We first need to build the global bipartite graph from multiple local RBGs while the location privacy of them should be preserved. Existing works to preserve location privacy in bipartite matching [16, 17] usually apply Geo-indistinguaishability (Geo-I) [1]. As they perturb a node to nearer locations with higher probability, its approximate location can still be inferred. Also the noise will influence the connectivity of the bipartite graph and there is no guarantee on accuracy loss. Another alternative is to apply secure multi-party computation (SMC) to calculate the Euclidean distances [9] between

Algorithm 2: EncodeRBG

input : $G = \{U \cup V\}$: a bipartite graph
h: a family of κ LSH hash functions;
$\mathcal{V}$:the local value function; γ, ϵ_p:parameters;
output: $\tilde{G} = \{\tilde{U} \cup \tilde{V}\}$: the encoded bipartite graph;

$\tilde{U}, \tilde{V}, \tilde{W} \leftarrow \emptyset$;
for $u_i \in U$ **do**
  $\tilde{u}_i.loc \leftarrow SSig(u_i) = MD5(Concat(h(u_i)))$;
  $\tilde{U} \leftarrow \tilde{U} \cup \{\tilde{u}_i\}$;
for $v_i \in V$ **do**
  $\tilde{v}_i.origin \leftarrow SSig(v_i) = MD5(Concat(h(v_i)))$;
  $\tilde{V} \leftarrow \tilde{V} \cup \{\tilde{v}_i\}$;
  $\tilde{v}_i.reward \leftarrow \tilde{w}(v_i)$ from (6);
return $\tilde{G} = \{\tilde{U} \cup \tilde{V}\}$

Algorithm 3: DecodeRBG

input : $\{\tilde{G}_k = \{\tilde{U}_k \cup \tilde{V}_k\}\}$: encoded bipartite graphs from K parties;
output: $G = \{U \cup V, E\}$: the decoded global bipartite graph;

$U, V, E \leftarrow \emptyset$;
for $k = 1, 2, \cdots, K$ **do**
  $U \leftarrow U \cup \tilde{U}_k, V \leftarrow V \cup \tilde{V}_k$;
for $u \in U$ **do**
  for $v \in V$ **do**
    if *isEqual(u.loc, v.origin)* **then**
      $e \leftarrow (u, v)$, $e.weight \leftarrow v.reward$;
      $E \leftarrow E \cup \{e\}$;
return $G = \{U \cup V, E\}$

all pairs of orders and drivers, but the time consumption can be intolerable (which is also validated in Sec. 4).

Therefore we devise a novel approach based on locality sensitive hashing (LSH) [6], which can reach a better trade-off between privacy and efficiency. It maintains the approximate nearest neighbours set after the hashing (we abuse u and v for $u.loc$ and $v.origin$):

- If $dist(u, v) \le R$, then $Pr[h(u) = h(v)] \ge p_1$,
- If $dist(u, v) \ge c \cdot R$, then $Pr[h(u) = h(v)] \le p_2$,

where $dist(\cdot, \cdot)$ is the Euclidean distance, R is the threshold of connectivity (*e.g.*, 3km in real applications), h is the hash function and p_1/p_2 is expected to be large. As the distance metric here is l_2-norm, the L2LSH function [4] can be used: $h(v) = \lfloor \frac{\vec{a} \cdot \vec{v} + b}{r} \rfloor$, where each element a_i in $\vec{a}$ is I.I.D. drawn from the standard Gaussian distribution $\mathcal{N}(0, 1)$ and b is uniformly randomly drawn from $[0, r]$. By using the LSH codes, we can generate the signature of a node v represented by κ hash functions, $Sig(v) = \langle h_1(v), \cdots, h_\kappa(v) \rangle$. We further use MD5 cryptographic hash functions to preserve the relative positions in LSH codes and the secure signature will be $SSig(v) = MD5(Concat(h(v)))$, where the seed $Concat(h(v))$ is the concatenation of the κ binary hash codes.

The algorithms of encoding and decoding local RBGs are shown in Algorithm 2 and Algorithm 3. The encoding algorithm (Algorithm 2) generates the secure signatures of the nodes to be shared locally, and the server can rebuild the bipartite graph by Algorithm 3, which only takes one round of communication.

Obfuscating edge weights with differential privacy. After we recover the global bipartite graph structure, we need to calculate the edge weights which are generated by RL. We only need to consider the edge between a connected pair of order v and driver u, thus $v.origin$ and $u.loc$ are nearby to each other. Due to the continuity of the value function, we have $\mathcal{V}(v.origin) \approx \mathcal{V}(u.loc)$. So we can rewrite (3) as:

$$w(u, v) \approx v.reward + \gamma \mathcal{V}(v.destination) - \mathcal{V}(v.origin), \quad (5)$$

which is only correlated to order v. We apply differential privacy (DP) [5] to perturb $w(u, v)$ so that the server cannot infer the position of orders and drivers through grid values. We inject the Laplace noise to the edge weights ($v.reward$ is taken as constant so that the results will not be sensitive to different grid values) and the sensitivity is calculated by $\Delta_{\mathcal{V}} = \gamma \mathcal{V}_{max} - \mathcal{V}_{min} - (\gamma \mathcal{V}_{min} - \mathcal{V}_{max}) = (1 + \gamma) diam(\mathcal{V})$, where $diam(\mathcal{V}) = \mathcal{V}_{max} - \mathcal{V}_{min}$, so the perturbation is

$$\tilde{w}(v) = w(u, v) + Lap(\frac{(1 + \gamma) diam(\mathcal{V})}{\epsilon_p}), \quad (6)$$

where ϵ_p is the privacy budget. Suppose v and v' are two orders with the same reward but arbitrary origins and destinations. According to the Laplace mechanism, we have that

$$\frac{Pr[\tilde{w}(v) = w]}{Pr[\tilde{w}(v') = w]} \le \exp(\frac{\epsilon_p}{(1 + \gamma) diam(\mathcal{V})} |w(v) - w(v')|) \le e^{\epsilon_p}.$$

It indicates that any two orders are indistinguishable after the perturbation. Therefore, the server cannot infer which order does the edge belong to from the edge weights. After the injection of noise, each party will upload the perturbed edge weights which are binding with the orders. Finally a global matching will be conducted by the server with the greedy algorithm and each party will get additional orders that are matched to their own drivers.

Analysis. We make an analysis to further verify the effectiveness of aggregating the dispatching decisions. Given a bipartite graph $G = \{U \cup V, E\}, |U| = n, |V| = m$, the objective is to prove that the gap between $\mu(G_{Fed})$ and $\mu(G_{LS})$ is large, as (1) indicates. By assuming that all nodes in U have the same degree d with uniformly distributed neighbours and all nodes are uniformly partitioned into K subsets in the federated setting, we have the following conclusion for unweighted bipartite matching.

Theorem 1. *If we set* $d = 1$, $\delta_0 = \frac{1}{1 + m(e^{-\alpha} - e^{-2\alpha})^2}$ *and* $\gamma_0 = \frac{1}{2}(e^{-\alpha} - e^{-2\alpha})$ *where* $\alpha = n/m$, *we have*

$$Pr[\mu(G_{Fed}) - \mu(G_{LS}) \ge \gamma_0 m] \ge 1 - \delta_0.$$

It can further be extended to weighted bipartite matching with any node degree d, and the proof details are in Sec. A.1.

We also explain why we use the greedy algorithm for global matching. In this way the incentive compatibility can be achieved so that all parties are willing to share all of their unmatched nodes. The proof details are in Sec. A.2.

Table 1: Main results of average rewards per day

Method	3 Parties				5 Parties			
	Setting 1	Setting 2	Setting 3	Setting 4	Setting 1	Setting 2	Setting 3	Setting 4
Global	61.1198	55.1719	30.7068	26.3471	61.1198	55.1719	30.7068	26.3471
Greedy	52.2754	44.3217	24.7275	19.6522	48.5105	39.1956	21.8402	16.4044
LocalSum	54.3376	45.8270	25.1841	19.9402	49.7560	40.1255	22.0754	16.6361
V-Only	54.2211	45.8521	25.2039	19.9549	49.7584	40.1233	22.1554	16.6119
Fed-LTD-N	59.9310	54.0201	29.8512	25.7160	59.5628	53.6723	29.6873	25.5830
Fed-LTD	59.9029	54.0068	29.8271	25.6853	59.6439	53.6558	29.6341	25.6320
Accuracy Loss	0.05%	0.02%	0.08%	0.12%	-0.14%	0.03%	0.18%	-0.19%
Δ^+	10.24%	17.85%	18.44%	28.81%	19.87%	33.72%	34.24%	54.07%
Δ^-	1.99%	2.11%	2.86%	2.51%	2.41%	2.75%	3.49%	2.71%

4 EXPERIMENTS

4.1 Experimental Settings

Dataset and Simulation Environment. We use the real dataset from Didichuxing's GAIA initiative [1]. It contains both order request data and driver trajectory data from 11/01/2016 to 11/30/2016 in Chengdu, China. Each day has about 200K order requests in average. We build a simulator based on the dataset. It adopts the batch-based dispatching framework with batch size of 2 seconds. In each batch, the simulator generates the available order requests and drivers and sends them to the dispatching agent. After receiving the dispatching decisions from the agent, it will simulate the dynamics of the system, including the pick-up and delivery behaviours, random walks, log-on and log-off behaviours of the drivers until the next batch. In the federated setting, the simulator generates orders and drivers for each party independently. All parties run a dispatching agent by themselves and return the dispatching results to the simulator iteratively. The number of parties we have set is 3 and 5.

Comparing Methods. We report the experimental results of the following methods.

- Global: it represents the global setting, where the dispatching algorithm is conducted with full data. It uses the SOTA RL-based order dispatching algorithm from [20].
- Greedy: it takes the sum of rewards by single-platform order dispatching in the federated setting. The local dispatching algorithm is the naive greedy algorithm.
- LocalSum: it also takes the sum of rewards by single-platform order dispatching while the local dispatching algorithm is the same as Global. It is also the baseline in (1).
- V-Only: it is a simple extension to LocalSum, where each party can only share their learned values, but the dispatching decisions are not shared.
- Fed-LTD-N: it is the proposed Fed-LTD framework without any privacy preservation measures. We compare this method to show the privacy loss of our approach.
- Fed-LTD-HE: it replaces private hashing by homomorphic encryption (HE) to calculate the Euclidean distances according to [9]. We only show its efficiency as the effectiveness results are the same with Fed-LTD-N.
- Fed-LTD: it is the proposed approach.

Parameter Settings and Implementation. To evaluate the methods with different demand and supply conditions and different graph sparsity, we mainly use the following 4 settings.

- Setting 1: The total number of drivers $n = 6K$ and the maximal radius of picking orders $R = 3km$.
- Setting 2: $n = 6K, R = 1km$. In this setting it will be harder for a driver to take orders so the graph will be more sparse.
- Setting 3: $n = 2K, R = 3km$. In this setting it will be in short supply (*i.e.*, lack of drivers).
- Setting 4: $n = 2k, R = 1km$. It represents the worst situation.

For other common parameters, we set the learning rate and discount factor in RL to 0.025 and 0.9 as [20] does. For LSH, we set $\kappa = r = 3$. For differential privacy, we set $diam(\mathcal{V}) = 10, \epsilon = 1.0$. The synchronization of value is performed every 30 batches (*i.e.*, 60 seconds). We use the data from 11/01 to 11/05 for training and show the testing results from 11/06 to 11/30. We implement all the methods with Python 3.8. The experiments were conducted on five Intel(R) Xeon(R) Platinum 8269CY 3.10GHz CPUs each with 4 cores.

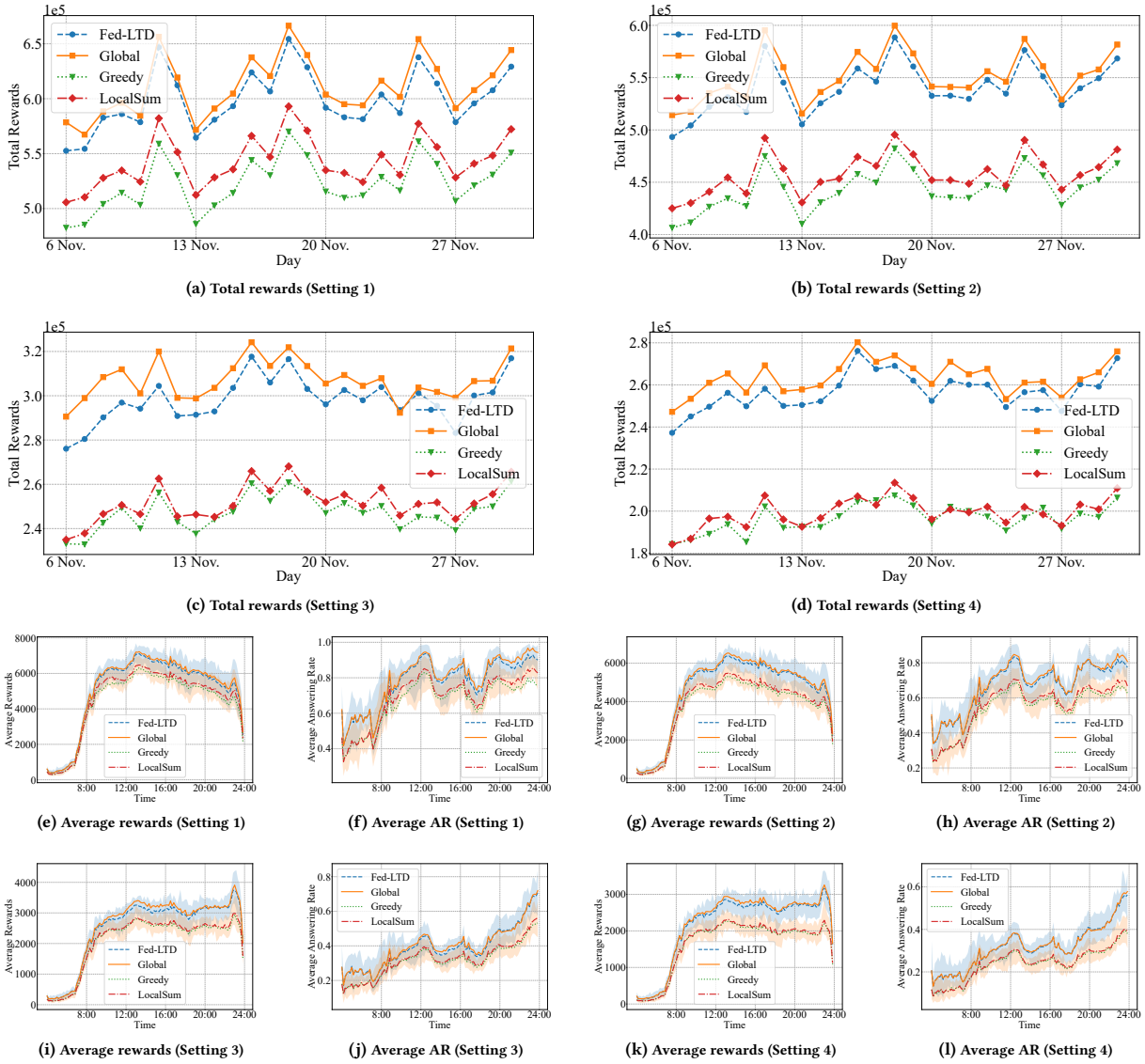
Evaluation Metrics. We compare the performance of all the methods via the following metrics.

- Rewards: the sum of revenue of the completed orders. We also show the average of rewards for the 25 testing days.
- Answering Rate (AR): the ratio between the number of matched orders and the number of all order requests. It can measure the user experience in ride hailing.
- Running Time: the average time consumed for calculating the dispatching results per batch.
- Δ^+: $\frac{R_{Fed}-R_{LS}}{R_{LS}} \times 100\%$, where R_{Fed} and R_{LS} are the average rewards of Fed-LTD and LocalSum.
- Δ^-: $\frac{R_{Global}-R_{Fed}}{R_{Global}} \times 100\%$, where R_{Global} is the average rewards of Global.

4.2 Results

4.2.1 Effectiveness Results. The main results of average rewards per day are shown in Table 1. We vary the 4 settings in a 3-party federation and a 5-party federation respectively. The orders and drivers are evenly partitioned for each party. We can observe that the rewards decrease significantly in federated setting compared

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



(a) Total rewards (Setting 1) (b) Total rewards (Setting 2)

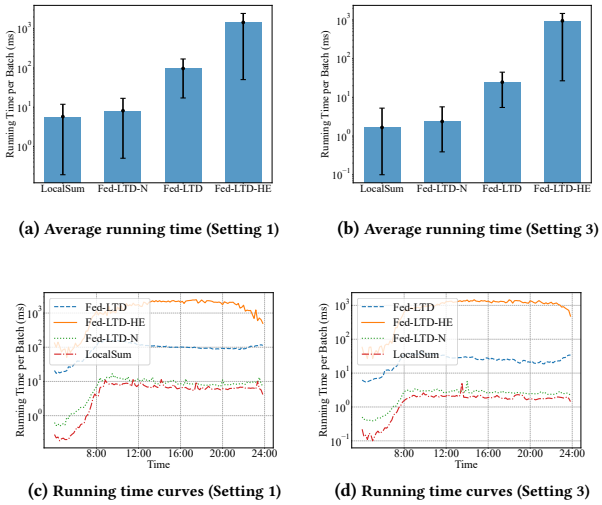
(c) Total rewards (Setting 3) (d) Total rewards (Setting 4)

(e) Average rewards (Setting 1) (f) Average AR (Setting 1) (g) Average rewards (Setting 2) (h) Average AR (Setting 2)

(i) Average rewards (Setting 3) (j) Average AR (Setting 3) (k) Average rewards (Setting 4) (l) Average AR (Setting 4)

Figure 3: Effectiveness results of 3 parties varying the 4 settings. The shaded areas represent the range between the minimal and maximal rewards of LocalSum and Fed-LTD. The results of 5 parties are similar and we move them to Sec. B.

with non-federated setting (*i.e.*, Global). LocalSum can still show advantages over Greedy, which verifies the effectiveness of applying RL. We also notice that V-Only only has subtle improvement compared with LocalSum, which validates that sharing the decisions can be indispensable. We can observe from the results that Fed-LTD has obvious advantages over LocalSum. The increase is from 10.24% to 54.07% while the decrease from Global is within 4%. It proves that the proposed framework can achieve near-optimal results compared with non-federated setting. The row of accuracy loss records the relative error brought by privacy preservation in Fed-LTD. It shows that the proposed privacy-preserving techniques have very slight influence on the effectiveness. We also observe from the different settings that with the bipartite graph becoming more sparse (*i.e.*, Setting 2), the advantage of Fed-LTD becomes larger, which is consistent with the theoretical results. When the market is in short supply (*i.e.*, Setting 3), the leading advantage of Fed-LTD is widened, which proves that our approach suits the imbalanced supply and demand scenarios better. In Fig. 3 we also plot the curves of the rewards and answering rate by day and by hour. We can see that the performances of Fed-LTD and Global are very close to each other while their advantages over LocalSum are still obvious.



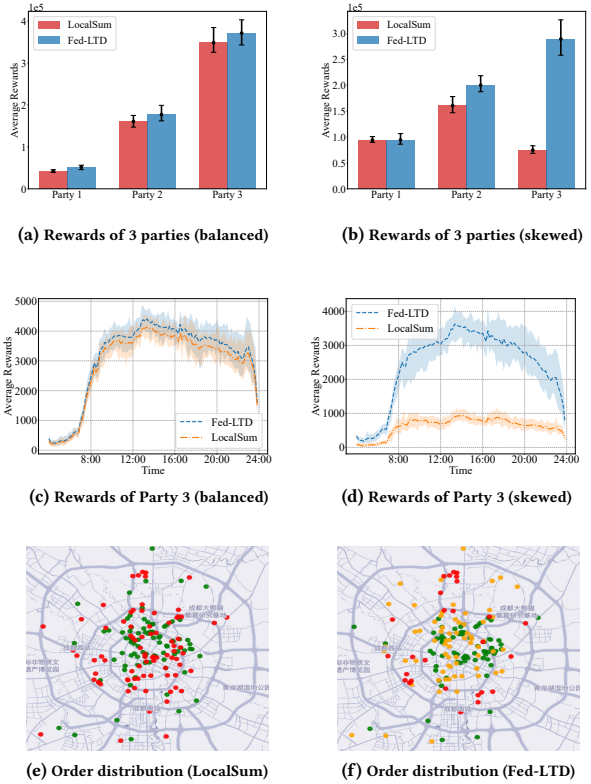
(a) Average running time (Setting 1) (b) Average running time (Setting 3)

(c) Running time curves (Setting 1) (d) Running time curves (Setting 3)

Figure 4: Efficiency Results. (a) and (b) show the average running time per batch, also with the minimal and the maximal running time. (c) and (d) plot the curves of running time of all batches in a day.

4.2.2 Efficiency Results. The efficiency results are shown in Fig. 4. We use logarithmic axis to illustrate the results more clearly. We observe that without privacy preservation, the average running times of Fed-LTD-N and LocalSum are nearly the same, both around 10ms per batch. With private hashing, the efficiency of Fed-LTD will inevitably decrease to about 100ms per batch. However, if homomorphic encryption (HE) is used as a replacement, the running time of Fed-LTD-HE will be 10x larger than Fed-LTD in Setting 1 and 3, reaching the same magnitude of the batch size. It will be unacceptable for the real-time response in order dispatching. The results verify that the proposed privacy preserving techniques can significantly reduce the time consumption and reach a better trade-off between privacy and efficiency.

4.2.3 A Case Study with Skewed Data. We further investigate how the skewed demand and supply distributions can affect the dispatching performance with a case study. We first partition the orders and drivers into 3 parties with the same proportion of $1:3:6$ as the balanced case. We observe from Fig. 5a and Fig. 5c that Fed-LTD can increase the rewards by about 10%, which is similar to the main results. Next, we reverse the distribution of demand (number of orders) with a proportion of $6:3:1$ as the skewed case. In this case, Party 3 has many redundant drivers while Party 1 has too many orders that cannot all be served. With the aggregation of dispatching decisions in Fed-LTD, we find from Fig. 5b and Fig. 5d that the rewards of Party 3 are increased by over 300% and the excess supply resources are not wasted. From the visualization of orders in Fig. 5e and Fig. 5f, we also find that Fed-LTD can take many additional orders (yellow dots) that would have been canceled by LocalSum and raise the answering rate from 0.4 to 0.8, which can significantly improve user experience in ride hailing.



(a) Rewards of 3 parties (balanced) (b) Rewards of 3 parties (skewed)

(c) Rewards of Party 3 (balanced) (d) Rewards of Party 3 (skewed)

(e) Order distribution (LocalSum) (f) Order distribution (Fed-LTD)

Figure 5: A case study of 3 parties with balanced/skewed demand and supply distributions. (a) and (b) show the average rewards of 3 parties in the balanced/skewed case. (c) and (d) plot the reward curves of Party 3 in the two cases. (e) and (f) illustrate the order distributions in the skewed case with LocalSum and Fed-LTD as the dispatching approach, at 10:00 - 10:01, 11/06/2016. The green dots represent the served orders while the red ones are the canceled orders. The yellow dots are the additional orders taken by Fed-LTD.

5 RELATED WORK

We review related work on order dispatching and federated learning respectively.

Order dispatching. Order dispatching is the central problem in ride hailing applications, also a typical task assignment problem [3] in spatial crowdsourcing [8, 23]. The most commonly used model of the problem is online bipartite graph matching [18, 19, 22], with various objectives like maximizing the total utility, minimizing the total waiting time, etc. Existing solutions to order dispatching can be divided into two categories: combinatorial optimization based approaches and reinforcement learning based approaches.

Traditional combinatorial optimization based approaches [8, 19, 21, 27] can perform well in relatively small and simple scenarios. However, they may have efficiency problems and may also lose effectiveness when their assumptions are violated in large-scale

settings. Reinforcement learning (RL) [13] based solutions are becoming popular recently due to the great power on solving sequential decision making problems. For example, Q-learning is applied to estimate the value of decisions in different states, in the form of both tabular values [24, 28] or deep neural networks [14, 15]. A recent work [20] combines combinatorial optimization with reinforcement learning and obtains SOTA results in large-scale order dispatching. We use it as our local dispatching operator and devise novel global aggregation schemes in the federated setting.

Federated learning. Federated learning [10, 12] is a new learning paradigm that collaboratively trains models among multiple parties without sharing their raw data. It has attracted much attention in recent years due to the increasing concerns on data privacy. It has two typical application scenarios [7], the cross-device FL where each party can be an edge device, and the cross-silo FL where each party is an enterprise or organization. We focus on the cross-silo setting as each party is a ride hailing platform or taxi company in our problem.

Privacy preservation is the most widely recognized issue in FL [29]. To preserve the privacy of gradients, some work designs secure multi-party computation protocols based on secret sharing [2] or homomorphic encryption [29]. Differential privacy [5] is another popular technique that can perturb the intermediates in FL by injecting noise [25, 26]. Another major challenge is the efficiency problem. To reduce communication overhead, sampling and gradient compression techniques are usually applied [12]. Some work also uses special data structures (like sketches) to reduce the time cost [26]. However, existing FL techniques to address the challenges above do not fit our problem as they are commonly designed for gradient descent in general supervised learning. But in our problem, the intermediates can be value functions or even the bipartite graphs, which brings novel challenges.

6 CONCLUSION

This paper introduces federated order dispatching for cross-platform ride hailing, where multiple platforms can make dispatching decisions together without sharing their local data. To address the challenges on effectiveness, privacy and efficiency, we propose a novel solution framework named Fed-LTD, which allows sharing of both dispatching models and dispatching decisions. Privacy preservation and efficiency optimization techniques have also been devised to make our solution more practical. Experimental results based on real data show that the proposed solution has obvious advantages in terms of total revenue and running time.

ACKNOWLEDGMENTS

We are grateful to anonymous reviewers for their constructive comments. This work is partially supported by the National Key Research and Development Program of China under Grant No. 2018AAA0101100, the National Science Foundation of China (NSFC) under Grant Nos. U21A20516, U1811463 and 62076017, the State Key Laboratory of Software Development Environment Open Funding No. SKLSDE-2020ZX-07, WeBank Scholars Program, Didi Collaborative Research Program and the Lee Kong Chian Fellowship awarded to Zimu Zhou by Singapore Management University. Yongxin Tong is the corresponding author.

REFERENCES

[1] Miguel E. Andrés, Nicolás Emilio Bordenabe, Konstantinos Chatzikokolakis, and Catuscia Palamidessi. 2013. Geo-indistinguishability: differential privacy for location-based systems. In *CCS*. 901–914.
[2] Kallista A. Bonawitz, Vladimir Ivanov, Ben Kreuter, Antonio Marcedone, H. Brendan McMahan, Sarvar Patel, Daniel Ramage, Aaron Segal, and Karn Seth. 2017. Practical Secure Aggregation for Privacy-Preserving Machine Learning. In *CCS*. ACM, 1175–1191.
[3] Rainer E. Burkard, Mauro Dell'Amico, and Silvano Martello. 2009. *Assignment Problems*. SIAM.
[4] Mayur Datar, Nicole Immorlica, Piotr Indyk, et al. 2004. Locality-sensitive hashing scheme based on p-stable distributions. In *SCG*. 253–262.
[5] Cynthia Dwork. 2006. Differential Privacy. In *ICALP*. 1–12.
[6] Piotr Indyk and Rajeev Motwani. 1998. Approximate Nearest Neighbors: Towards Removing the Curse of Dimensionality. In *STOC*. 604–613.
[7] Peter Kairouz, H. Brendan McMahan, Brendan Avent, et al. 2021. Advances and Open Problems in Federated Learning. *Found. Trends Mach. Learn.* 14, 1-2 (2021), 1–210.
[8] Leyla Kazemi and Cyrus Shahabi. 2012. GeoCrowd: enabling query answering with spatial crowdsourcing. In *SIGSPATIAL*. 189–198.
[9] Vladimir Kolesnikov, Ahmad-Reza Sadeghi, and Thomas Schneider. 2009. Improved Garbled Circuit Building Blocks and Applications to Auctions and Computing Minima. In *CANS*, Vol. 5888. 1–20.
[10] Jakub Konecný, H. Brendan McMahan, Felix X. Yu, et al. 2016. Federated Learning: Strategies for Improving Communication Efficiency. *CoRR* abs/1610.05492 (2016).
[11] Harold W Kuhn. 1955. The Hungarian method for the assignment problem. *Naval research logistics quarterly* 2, 1-2 (1955), 83–97.
[12] Brendan McMahan, Eider Moore, Daniel Ramage, Seth Hampson, and Blaise Agüera y Arcas. 2017. Communication-Efficient Learning of Deep Networks from Decentralized Data. In *AISTATS*. 1273–1282.
[13] Richard S. Sutton and Andrew G. Barto. 1998. *Reinforcement learning - an introduction*. MIT Press.
[14] Xiaocheng Tang, Zhiwei (Tony) Qin, Fan Zhang, Zhaodong Wang, Zhe Xu, Yintai Ma, Hongtu Zhu, and Jieping Ye. 2019. A Deep Value-network Based Approach for Multi-Driver Order Dispatching. In *SIGKDD*. 1780–1790.
[15] 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 *SIGKDD*. 3605–3615.
[16] Qian Tao, Yongxin Tong, Zimu Zhou, et al. 2020. Differentially Private Online Task Assignment in Spatial Crowdsourcing: A Tree-based Approach. In *ICDE*. 517–528.
[17] Hien To, Cyrus Shahabi, and Li Xiong. 2018. Privacy-Preserving Online Task Assignment in Spatial Crowdsourcing with Untrusted Server. In *ICDE*. 833–844.
[18] Yongxin Tong, Jieying She, Bolin Ding, et al. 2016. Online Minimum Matching in Real-Time Spatial Data: Experiments and Analysis. *PVLDB* 9, 12 (2016), 1053–1064.
[19] Yongxin Tong, Jieying She, Bolin Ding, Libin Wang, and Lei Chen. 2016. Online mobile Micro-Task Allocation in spatial crowdsourcing. In *ICDE*. 49–60.
[20] Yongxin Tong, Dingyuan Shi, Yi Xu, Weifeng Lv, Zhiwei (Tony) Qin, and Xiaocheng Tang. 2022. Combinatorial Optimization Meets Reinforcement Learning: Effective Taxi Order Dispatching at Large-Scale. *IEEE Trans. Knowl. Data Eng.* (2022). https://doi.org/10.1109/TKDE.2021.3127077.
[21] Yongxin Tong, Libin Wang, Zimu Zhou, et al. 2017. Flexible Online Task Assignment in Real-Time Spatial Data. *PVLDB* 10, 11 (2017), 1334–1345.
[22] Yongxin Tong, Yuxiang Zeng, Bolin Ding, et al. 2021. Two-Sided Online Micro-Task Assignment in Spatial Crowdsourcing. *IEEE Trans. Knowl. Data Eng.* 33, 5 (2021), 2295–2309.
[23] Yongxin Tong, Zimu Zhou, Yuxiang Zeng, et al. 2020. Spatial crowdsourcing: a survey. *VLDB J.* 29, 1 (2020), 217–250.
[24] Yansheng Wang, Yongxin Tong, Cheng Long, Pan Xu, Ke Xu, and Weifeng Lv. 2019. Adaptive Dynamic Bipartite Graph Matching: A Reinforcement Learning Approach. In *ICDE*. 1478–1489.
[25] Yansheng Wang, Yongxin Tong, and Dingyuan Shi. 2020. Federated Latent Dirichlet Allocation: A Local Differential Privacy Based Framework. In *AAAI*. 6283–6290.
[26] Yansheng Wang, Yongxin Tong, Dingyuan Shi, and Ke Xu. 2021. An Efficient Approach for Cross-Silo Federated Learning to Rank. In *ICDE*. 1128–1139.
[27] Pan Xu, Yexuan Shi, Hao Cheng, et al. 2019. A Unified Approach to Online Matching with Conflict-Aware Constraints. In *AAAI*. 2221–2228.
[28] 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 *SIGKDD*. 905–913.
[29] Qiang Yang, Yang Liu, Tianjian Chen, and Yongxin Tong. 2019. Federated Machine Learning: Concept and Applications. *ACM Trans. Intell. Syst. Technol.* 10, 2 (2019), 12:1–12:19.

A ADDITIONAL PROOFS

A.1 Proof of Theorem 1

We first prove the following lemma.

Lemma 1. *If we set* $d = 1, \gamma = (\frac{1}{e}-\epsilon)^{\alpha K}-(\frac{1}{e})^{\alpha}-\beta$ *and* $\delta = \frac{1}{1+2\beta^2 m}$, *where* $\alpha = n/m$, ϵ *and* β *are small numbers, we have*

$$Pr[\mu(G) - \mu(G_{LS}) \geq \gamma \cdot m] \geq 1 - \delta.$$

Proof. Suppose $S = \{v \in V | deg(v) \geq 1\}$. We have $\mu(G) \leq |S|$. Since the degree of nodes in U is 1, it means every two nodes in S will have distinct neighbours. So every node in S can be matched, *i.e.*, $\mu(G) = |S|$.

The probability that a node $v \in V$ is in S is:

$$Pr[v \in S] = 1 - Pr[deg(v) = 0] = 1 - (1 - \frac{1}{m})^n.$$

Let $n = \alpha \cdot m$. Without loss of generality, we suppose $\alpha \geq 1$, and we get

$$\mathbb{E}[\mu(G)] = \mathbb{E}[|S|] = m - m[(1 - \frac{1}{m})^m]^{\alpha},$$
$$Var[\mu(G)] = m \cdot (1 - (1 - \frac{1}{m})^{\alpha m}) \cdot (1 - \frac{1}{m})^{\alpha m}.$$

So we have $m - m(\frac{1}{e})^{\alpha} \leq \mathbb{E}[\mu(G)] \leq m - m(\frac{1}{e} - \epsilon)^{\alpha}$, and $\epsilon \to 0$ when $m \to +\infty$. We also have $m(e^{-\alpha} - e^{-2\alpha}) \leq Var[\mu(G)] \leq \frac{m}{4}$.

In the federated setting, graph G is partitioned into K subgraphs $G_1, G_2, \cdots, G_K$. Similarly we can define $S_k = \{v \in V_k | deg(v) \geq 1\}$ and get $\mu(G_k) = |S_k|$. We further have $Pr[v \in S_k] = \frac{1}{K} \cdot (1 - (1 - \frac{1}{m})^{n_k}$, where $deg_k(v)$ is the degree of node $v \in V_k$ in graph G_k, and $n_k = |U_k| = \frac{n}{K}$. Therefore, $\mathbb{E}[\mu(G_k)] = \mathbb{E}[|S_k|] = \frac{m}{K} \cdot (1 - (1 - \frac{1}{m})^{\frac{\alpha m}{K}})$, $Var[\mu(G_k)] = m \cdot (1 - (1 - \frac{1}{m})^{\frac{\alpha m}{K}}) \cdot (1 - \frac{1}{m})^{\frac{\alpha m}{K}}$. Since $\mathbb{E}[\mu(G_{LS})] = \sum_{k=1}^{K} \mathbb{E}[\mu(G_k)]$, we obtain

$$m - m(\frac{1}{e})^{\frac{\alpha}{K}} \leq \mathbb{E}[\mu(G_{LS})] \leq m - m(\frac{1}{e} - \epsilon)^{\frac{\alpha}{K}},$$
$$m(e^{-\alpha/K} - e^{-2\alpha/K}) \leq Var[\mu(G_{LS})] \leq \frac{m}{4}.$$

By applying the Cantelli inequality, we have

$$Pr[\mu(G) - \mu(G_{LS}) < \mathbb{E}[\mu(G)] - \mathbb{E}[\mu(G_{LS})] - \beta m]$$
$$\leq \frac{Var[\mu(G)] + Var[\mu(G_{LS})]}{Var[\mu(G)] + Var[\mu(G_{LS})] + \beta^2 m^2} \leq \frac{m/4}{m/4 + 2\beta^2 m^2} = \frac{1}{1 + 2\beta^2 m},$$

$$Pr[\mu(G) - \mu(G_{LS}) < \mathbb{E}[\mu(G)] - \mathbb{E}[\mu(G_{LS})] - \beta m]$$
$$\geq Pr[\mu(G) - \mu(G_{LS}) \leq m(\frac{1}{e} - \epsilon)^{\frac{\alpha}{K}} - m(\frac{1}{e})^{\alpha} - m\beta].$$

We denote $\gamma = (\frac{1}{e} - \epsilon)^{\frac{\alpha}{K}} - (\frac{1}{e})^{\alpha} - \beta$ and $\delta = \frac{1}{1+2\beta^2 m}$, then

$$Pr[\mu(G) - \mu(G_{LS}) \geq \gamma \cdot m] \geq 1 - \delta. \tag{7}$$

□

Next, we will prove Theorem 1.

Proof. Suppose the unmatched nodes in G_{LS} form a new bipartite graph G_Δ while G_{Δ_k} contains the unmatched nodes in G_k. We have $\mu(G_{Fed}) = \mu(G_{LS}) + \mu(G_\Delta) = \mu(G_{LS}) + \sum_{k=1}^{K} \mu(G_{\Delta_k})$. And similarly we have $\mu(G_\Delta) = |S_\Delta| = \sum_{k=1}^{K} |S_{\Delta_k}|$.

According to the maximum cardinality matching in each G_k, there is no edge between U_{Δ_k} and V_{Δ_k}. For the subgraph G_Δ,

$$Pr[v \in S_{\Delta_k} | R_k = i] = 1 - (1 - \frac{1}{m})^i,$$

where $R_k = \sum_{k' \neq k} |U_{\Delta_{k'}}|$. Since $|U_{\Delta_k}| \sim B(\frac{n}{K}, p_0)$ is a random variable following the Binomial distribution with $p_0 = 1 - \frac{1}{\alpha}(1 - (1 - \frac{1}{m})^{\frac{\alpha m}{K}})$, R_k is also binomial, *i.e.*, $R_k \sim B(n_0, p_0)$ with $n_0 = (1 - \frac{1}{K})n$. Suppose $m_0 = m(1 - \frac{1}{m})^{\alpha m}$, we have

$$\begin{aligned}
\mathbb{E}[\mu(G_\Delta)] = \mathbb{E}[|S_\Delta|] &= \sum_{k=1}^{K} \mathbb{E}[|S_{\Delta_k}|] = \sum_{k=1}^{K} \sum_{i=1}^{n} \mathbb{E}[|S_{\Delta_k}| | R_k = i] \cdot Pr[R_k = i] \\
&\geq \frac{m_0}{K} \cdot \sum_{k=1}^{K} \sum_{i=1}^{n} (1 - (1 - \frac{1}{m})^{\alpha m}) Pr[R_k = i] \\
&\geq \frac{m_0}{K}(1 - e^{-\alpha}) \cdot \sum_{k=1}^{K} \sum_{i=1}^{n} Pr[R_k = i] \\
&= m_0(1 - e^{-\alpha}) = m(1 - \frac{1}{m})^{\alpha m}(1 - e^{-\alpha}).
\end{aligned}$$

Since m is large, $(1 - \frac{1}{m})^{\alpha m} \to e^{-\alpha}$, then $\mathbb{E}[\mu(G_\Delta)] \geq m \cdot (e^{-\alpha} - e^{-2\alpha})$. We still have $Var[\mu(G_\Delta)] \leq \frac{m}{4}$, by setting $\gamma_0 = \frac{1}{2}(e^{-\alpha} - e^{-2\alpha})$ and applying the Cantelli inequality again, we have

$$Pr[\mu(G_\Delta) < \gamma_0 m] \leq \frac{m/4}{m/4 + \epsilon_0^2} = \frac{1}{1 + m(e^{-\alpha} - e^{-2\alpha})^2}.$$

With $\delta_0 = \frac{1}{1+m(e^{-\alpha}-e^{-2\alpha})^2}$, we finally obtain

$$Pr[\mu(G_{Fed}) - \mu(G_{LS}) \geq \gamma_0 m] \geq 1 - \delta_0.$$

□

Then we make extensions to general node degree d. According to Hall's theorem, we have $\mu(G) \leq N(U) = |S|$. We define $\bar{\mu}(G) \triangleq |S|$ which is an upper bound of $\mu(G)$. It will become more tight when d is larger. And the following corollary holds for $\bar{\mu}(G)$.

Corollary 1. *With* $\delta_0' = \frac{1}{1+\frac{m}{d}(e^{-\alpha/d}-e^{-2\alpha/d})^2}$ *and* $\gamma_0' = \frac{1}{2}(e^{-\alpha/d} - e^{-2\alpha/d})$, *we have*

$$Pr[\bar{\mu}(G_{Fed}) - \bar{\mu}(G_{LS}) \geq \gamma_0' \frac{m}{d}] \geq 1 - \delta_0'. \tag{8}$$

Extension to weighted case. We make a simple extension to maximum weighted matching. Suppose A is the upper bound of edge weights, and $\bar{\mu}(G) \triangleq A \cdot \bar{\mu}(G)$ which is a loose upper bound for $\mu(G)$ in weighted case. Obviously Corollary 1 holds again by replacing $\bar{\mu}(\cdot)$ with $\bar{\bar{\mu}}(\cdot)$ and replacing m, α with $m \cdot A, \alpha \cdot A$.

A.2 Proof of Incentive Compatibility

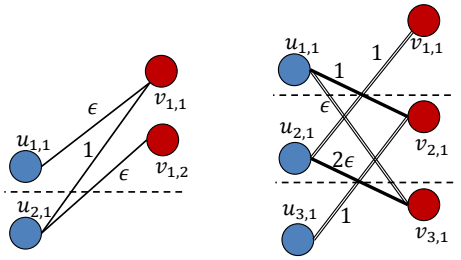
Suppose $\mathcal{G}_k^*$ is a subgraph that contains all the unmatched orders and drivers after a local maximum matching by party k. All the $\mathcal{G}_k$ forms a global graph G_Δ. Let $\mathcal{M}(G_\Delta)_k$ denote the subset of matching allocation on G_Δ where the drivers in the allocation are all from party k. $f(\langle \mathcal{G}_1, \cdots, \mathcal{G}_K \rangle)_k$ is the utility function for party k ($f(\mathcal{G}_k)$ for short). We use $\mathcal{M}_{Hun}$ and $\mathcal{M}_{Gre}$ to denote the Hungarian and greedy matching algorithms respectively.

We expect that Fed-LTD has the properties of incentive compatibility (IC) or individual rationality (IR), which means the parties will have motivation to share the unmatched orders. And we have the following conclusions.

Lemma 2. *If the utility function for party* k *is*

$$f(\langle \mathcal{G}_1, \cdots, \mathcal{G}_K \rangle)_k = \mathcal{M}_{Hun}(G_k - \mathcal{G}_k) + \mathcal{M}_{Hun}(G_\Delta)_k,$$

the proposed Fed-LTD framework satisfies IR but IC cannot be achieved.



(a) Example 1 (b) Example 2

Figure 6: Two counterexamples

Proof. We first show that with $\mathcal{G}'_k \subseteq \mathcal{G}^*_k$ and $\mathcal{G}_k = \emptyset$, $f(\mathcal{G}'_k) \geq f(\mathcal{G}_k)$. Since $\mathcal{G}'_k$ only contains the unmatched nodes in G_k, it will not influence $\mathcal{M}_{Hun}(G_k - \mathcal{G}_k)$. And obviously we have $\mathcal{M}_{Hun}(G_\Delta)_k \geq 0$, so $f(\mathcal{G}'_k) \geq f(\mathcal{G}_k)$. Therefore, with fixed $\mathcal{G}_{i\neq k}$, $\forall k$ we have

$$f(\langle \mathcal{G}_1, \cdots, \mathcal{G}'_k, \cdots, \mathcal{G}_K \rangle)_k \geq f(\langle \mathcal{G}_1, \cdots, \emptyset, \cdots, \mathcal{G}_K \rangle)_k. \quad (9)$$

We further show that the inequality will be broken if $\mathcal{G}'_k$ contains any node that is not in $\mathcal{G}^*_k$. We only need to prove it by an counterexample. In Fig. 6a, there are two parties and the initial local match allocation of party 1 is $(u_{1,1}, v_{1,1})$ with utility ϵ. If party 1 shares the matched nodes *i.e.*, $v_{1,1}$, it will be matched by $u_{2,1}$ from party 2, and the utility of party 1 will be 0, which violates (9). Next, we will show that it cannot satisfy IC by another counterexample. The unmatched nodes of 3 parties are illustrated in Fig. 6b. If party 1 does not share $v_{1,1}$, the matching allocation of Hungarian method will be the bold line, and the utility of party 1 is 1. However, if party 1 shares $v_{1,1}$, the maximum matching allocation will be the double line, and its utility will decrease to ϵ. It indicates that $f(\langle \mathcal{G}_1, \cdots, \mathcal{G}^*_k, \cdots, \mathcal{G}_K \rangle)_k < f(\langle \mathcal{G}_1, \cdots, \mathcal{G}_k, \cdots, \mathcal{G}_K \rangle)_k$ for some $\mathcal{G}_k \subseteq \mathcal{G}^*_k$. Thus IC cannot be satisfied in this case. □

Therefore, with the Hungarian method for global matching, all parties are willing to join in the federation (*i.e.*, IR), but they may not be willing to share all the unmatched nodes (*i.e.*, not IC). Next, we will see that by replacing the global matching method by the greedy algorithm, IC can also be satisfied.

Theorem 2. *If the utility function for party k is*

$$f(\langle \mathcal{G}_1, \cdots, \mathcal{G}_K \rangle)_k = \mathcal{M}_{Hun}(G_k - \mathcal{G}_k) + \mathcal{M}_{Gre}(G_\Delta)_k,$$

the proposed Fed-LTD framework satisfies IC.

Proof. First, IR can still be achieved here regardless of the global matching algorithm. So we will prove that with the Greedy algorithm as the global matching approach, the following inequality holds for any $\mathcal{G}_k \subseteq \mathcal{G}^*_k$.

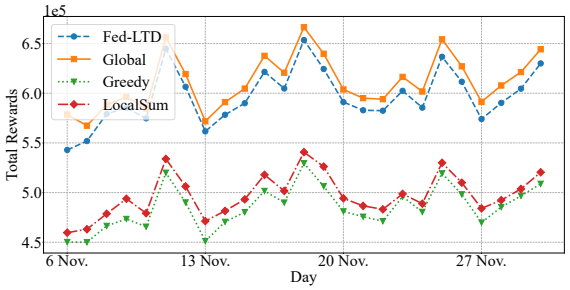
$$f(\langle \mathcal{G}_1, \cdots, \mathcal{G}^*_k, \cdots, \mathcal{G}_K \rangle)_k \geq f(\langle \mathcal{G}_1, \cdots, \mathcal{G}_k, \cdots, \mathcal{G}_K \rangle)_k. \quad (10)$$

We only need to prove that for party k, sharing one more nodes in $\mathcal{G}^*_k$ will not decrease its utility. We will prove it by induction on $|V_\Delta|$ where $G_\Delta = (U_\Delta \cup V_\Delta, E_\Delta)$.

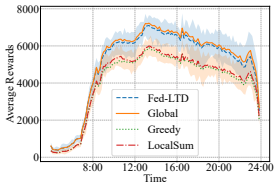
When $|V_\Delta| = 1$, there is only one node v_1 in V_Δ. (1). If v_1 is matched to $u_{i,1}$ by the Greedy algorithm, the utility of party k will be 0. So sharing new nodes $v_{k,1}$ will not decrease the utility. (2). If v_1 is matched to $u_{k,1}$ by the Greedy algorithm, as $v_{k,1}$ and $u_{k,1}$ are not connected, sharing new nodes $v_{k,1}$ will not change the utility of party k. Therefore, (10) holds when $|V_\Delta| = 1$.

We make the hypothesis that (10) holds when $|V_\Delta| = m$ and analyze on the case $|V_\Delta| = m + 1$. (1). v_{m+1} does not appear in the matching allocation of the Greedy algorithm. Then we can remove v_{m+1} from the graph with that the matching allocation on the new graph will not change. According to the hypothesis above, (10) still holds. (2). v_{m+1} appears in the matching allocation of the Greedy algorithm. We suppose the edge with the maximal weight is (u_x, v_1) with edge weight A. (2.1) If $u_x = u_{k,1}$, sharing a new node $v_{k,1}$ will not change the matching result on $u_{k,1}$, as the edge $(u_{k,1}, v_1)$ will always be removed by the Greedy algorithm. (2.2) If $u_x = u_{i,1}$ with $i \neq k$, we suppose the edge weight between $u_{i,1}$ and $v_{k,1}$ is A'. (2.2.1) If $A' \geq A$, according to the Greedy algorithm, $(u_{i,1}, v_{k,1})$ will be removed first. Suppose $u_{k,1}$ is matched to v_m before the appearance of $v_{k,1}$ with edge weight B and the weight between $u_{k,1}$ and v_1 is B'. If $B' \leq B$, $u_{k,1}, v_m$ will be removed no later than $u_{k,1}, v_1$, therefore the utility of $u_{k,1}$ does not decrease. If $B' > B$, the utility of $u_{k,1}$ will remain unchanged or increase to B'. So (10) still holds here. (2.2.2) If $A' < A$, we remove $u_{i,1}, v_1$ from the graph by the Greedy algorithm and we have $|V_\Delta| = m$. According to the hypothesis above, (10) holds. By induction on m we have proved the theorem. □

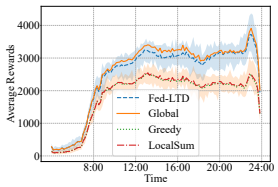
B ADDITIONAL EXPERIMENTAL RESULTS



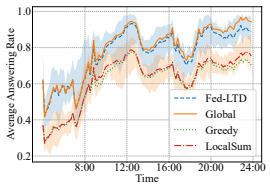
(a) Total rewards (Setting 1)



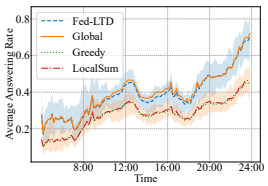
(b) Average rewards (Setting 1)



(c) Average rewards (Setting 3)



(d) Average AR (Setting 1)



(e) Average AR (Setting 3)

Figure 7: Effectiveness results of 5 parties