Differentially Private Online Task Assignment in Spatial Crowdsourcing: A Tree-based Approach

Qian Tao [†], Yongxin Tong [†], Zimu Zhou [‡], Yexuan Shi [†], Lei Chen [#], Ke Xu [†]
[†]BDBC, SKLSDE Lab and IRI, Beihang University, China [‡]ETH Zurich, Zurich, Switzerland
[#]The Hong Kong University of Science and Technology, Hong Kong SAR, China
[†]{qiantao, yxtong, skyxuan, kexu}@buaa.edu.cn, [‡]zzhou@tik.ee.ethz.ch, [#]leichen@cse.ust.hk

***Abstract*—With spatial crowdsourcing applications such as Uber and Waze deeply penetrated into everyday life, there is a growing concern to protect user privacy in spatial crowdsourcing. Particularly, locations of workers and tasks should be properly processed via certain privacy mechanism before reporting to the untrusted spatial crowdsourcing server for task assignment. Privacy mechanisms typically permute the location information, which tends to make task assignment ineffective. Prior studies only provide guarantees on privacy protection without assuring the effectiveness of task assignment. In this paper, we investigate privacy protection for online task assignment with the objective of minimizing the total distance, an important task assignment formulation in spatial crowdsourcing. We design a novel privacy mechanism based on Hierarchically Well-Separated Trees (HSTs). We prove that the mechanism is ϵ-Geo-Indistinguishable and show that there is a task assignment algorithm with a competitive ratio of $O(\frac{1}{\epsilon^4}\log N \log^2 k)$, where ϵ is the privacy budget, N is the number of predefined points on the HST, and k is the matching size. Extensive experiments on synthetic and real datasets show that online task assignment under our privacy mechanism is notably more effective in terms of total distance than under prior differentially private mechanisms.**

I. Introduction

With the rapid development of mobile Internet and sharing economy, spatial crowdsourcing has deeply penetrated into everyday life [1], [2], [3], [4]. Many core functions in these applications, *e.g.* task assignment, require users (*i.e.,* workers and tasks) to report their physical locations to the spatial crowdsourcing server. For example, Uber drivers and passengers have to report their real-time locations to the Uber server for enabling effective dispatching to the passengers. Since the spatial crowdsourcing server may not be trustworthy, it raises severe privacy concerns if the location information of workers and tasks is leaked or misused by the server. Furthermore, it may even be illegal to directly communicate the true location data to the server under new regulations, *e.g.* the General Data Protection Regulation (GDPR).

Privacy-preserving task assignment arises as a generic solution framework to protect location privacy of crowdsourcing users while still enabling the server to perform task assignment [5], [6], [7]. A privacy mechanism is typically designed to permute the locations of tasks and workers before they are reported to the untrusted server for task assignment. Two characteristics are crucial in a privacy mechanism for task assignment: *(i)* It is desirable that the privacy mechanism satisfies Geo-Indistinguishability [8], a widely acknowledged differential privacy metric to support single location query (as is the case of task assignment). *(ii)* The privacy mechanism should still allow effective task assignment on the permuted location data. In other words, there should exist a task assignment algorithm with a guaranteed approximation ratio. This is especially important in large-scale real-time spatial crowdsourcing applications [1], [9], [10], [11], [12], [13].

Previous privacy mechanisms only provide guarantees on privacy protection without assuring the effectiveness of task assignment. For example, in [5], a differentially private mechanism is proposed to protect the location privacy of workers and a heuristic algorithm is designed for offline task assignment on the protected data. In [7], the authors propose an ϵ-Geo-Indistinguishable privacy mechanism to protect the location privacy of both workers and tasks, and they further apply a heuristic greedy algorithm for online task assignment on the permuted data. However, neither of them has guarantees on the competitive ratio of task assignment.

In this paper, we make the first attempt at privacy protection for task assignment in spatial crowdsourcing that *(i)* is Geo-Indistinguishable and *(ii)* provides theoretical guarantees on task assignment. Particularly, we focus on Online Minimum Bipartite Matching (OMBM), a task assignment formulation with both growing research interests [10],[14],[15] and practical adoption (*e.g.* ride-sharing, food delivery and last-mile delivery). To this end, we devise a novel tree-based privacy mechanism leveraging Hierarchically Well-Separated Trees (HSTs). We prove that our mechanism satisfies Geo-Indistinguishability, and further propose a fast implementation of the mechanism to be fit for large-scale spatial crowdsourcing applications. More importantly, we prove that, under our tree-based privacy mechanism, there exists an online task assignment algorithm that achieves a competitive ratio of $O(\frac{1}{\epsilon^4}\log N \log^2 k)$, where N is the number of predefined points on the HST, and k is the matching size. Experiments show that online minimum bipartite matching on data permuted by our privacy mechanism is notable more effective than by the state-of-the-art differential private mechanisms.

Our main contributions are summarized as follows.

- We study the problem of location privacy protection for online task assignment with the objective of minimizing the total distance, an increasingly important problem in practical spatial crowdsourcing.
- We propose a novel privacy mechanism based on HSTs, which is ϵ-Geo-Indistinguishable, and allows online task

assignment that achieves a competitive ratio (approximation ratio for online algorithms) of $O(\frac{1}{\epsilon^4}\log N \log^2 k)$, where ϵ is the privacy budget, N is the number of predefined points for constructing the HST, and k is the matching size. To the best of our knowledge, this is the first privacy mechanism that provides guarantees for both privacy protection and task assignment.
- Extensive experiments on synthetic and real datasets show that online task assignment under our privacy mechanism is notably more effective than under the state-of-the-art differentially private mechanisms.

In the rest of this paper, we define our problem in Sec. II. We design a tree-based solution and prove its effectiveness in Sec. III. We then present the evaluations in Sec. IV, review related work in Sec. V and finally conclude in Sec. VI.

II. Problem Definition

This section formally defines the privacy protection desired for online task assignment in spatial crowdsourcing.

A. Interaction Model

We first introduce the three main parties in typical spatial crowdsourcing and their interactions.

Definition 1 (Crowd Worker)**.** *A crowd worker (worker for short) w is a tuple (x_w, y_w), which denotes the coordinates of w in the Euclidean space.*

Definition 2 (Spatial Task)**.** *A spatial task (task for short) t is a tuple (x_t, y_t), which denotes the coordinates of t in the Euclidean space.*

Definition 3 (Server)**.** *A crowdsourcing server (server for short) S is an untrusted platform to perform major functionalities of spatial crowdsourcing,* e.g. *task assignment.*

As in many spatial crowdsourcing applications [16], [10], [7], [17], [18], workers, tasks and the server interact as follows. Workers register to the server beforehand and their availability to perform tasks is known to the server. Tasks are dynamically posted to the server and need to be immediately assigned to workers. Finally, the worker who is assigned to the task will travel to the location of the task and complete it. We assume the locations of workers and tasks are indirectly communicated to the server in a metric space $\mathcal{X}$, *i.e.,* their coordinates are transformed into the points in $\mathcal{X}$. Our focus is to provide dedicated privacy protection on these points while allowing efficient task assignment on the protected data.

B. Problem Formulation

Definition 4 (Privacy Mechanism)**.** *A privacy mechanism (mechanism for short) $\mathcal{M}$ is a function that maps a point x in a metric space $\mathcal{X}$ into an obfuscated point z in another metric space $\mathcal{Z}$ with a probability of $\mathcal{M}(x)(z)$.*

We focus on the mechanisms where $\mathcal{X}$ and $\mathcal{Z}$ are the same metric spaces. In our context, x is a point transformed from the coordinate of a worker/task. z is the point that is reported by the worker/task to the server for task assignment. Particularly, we are interested in a popular category of task assignment called online minimum bipartite matching [10], [19], [1].

Definition 5 (Online Minimum Bipartite Matching [10])**.** *Given a set of workers $W = \{w_1, w_2, ..., w_n\}$ and a set of tasks $T = \{t_1, t_2, ..., t_m\}$ that appear dynamically, Online Minimum Bipartite Matching (OMBM) aims to assign for each task a worker immediately when the task appears such that the total travel distance of the assigned worker-task pairs is minimized.*

Now we can define our Privacy-preserving Online Minimum Bipartite Matching (POMBM) Problem as follows.

Definition 6 (Privacy-preserving Online Minimum Bipartite Matching)**.** *Given a set of workers $W = \{w_1, w_2, ..., w_n\}$, a set of tasks $T = \{t_1, t_2, ..., t_m\}$, and an untrusted server S, the Privacy-preserving Online Minimum Bipartite Matching (POMBM) problem aims to design a privacy mechanism $\mathcal{M}$ for the transformed locations of workers and tasks such that (i) the mechanism is **differential private** in the metric space $\mathcal{X}$; and (ii) the server can perform **effective** online minimum bipartite matching on the privacy-protected data.*

Quantitatively, $\mathcal{M}$ should *(i)* be *Geo-Indistinguishability* [8], the widely used differential privacy for location data; and *(ii)* allow an online matching algorithm $\mathcal{A}$ with a guaranteed *competitive ratio*, a metric to assess the effectiveness of online task assignment. We introduce these two criteria below.

C. Evaluation Criteria

We first present the criterion for privacy protection.

Definition 7 (Geo-Indistinguishability [8])**.** *A mechanism $\mathcal{M}$ operating on a metric space $\mathcal{X}$ is ϵ-Geo-Indistinguishable (ϵ-Geo-I for short) if for any $x_1, x_2 \in \mathcal{X}$ and $z \in \mathcal{Z}$, the following inequality holds:*

$$\mathcal{M}(x_1)(z) \le e^{\epsilon d_{\mathcal{X}}(x_1,x_2)}\mathcal{M}(x_2)(z) \tag{1}$$

where $d_{\mathcal{X}}(.,.)$ is the distance between two points in space $\mathcal{X}$.

Geo-Indistinguishability defines the indistinguishability of two points x_1 and x_2 (transformed from two locations in the Euclidean space) when they are obfuscated to the same point z. Hence if a worker/task reports a point z to the server, the server cannot decide whether the actual point is x_1 or x_2, not to mention the corresponding location in the Euclidean space.

Now we introduce competitive ratio, a widely used criterion to assess the effectiveness of online task assignment [9], [10], [20]. Denote $d_{\mathcal{X}}(M)$ as the total travel distance in $\mathcal{X}$ of all pairs in the matching M, *i.e.,* $d_{\mathcal{X}}(M) = \sum_{(t,w)\in M} d_{\mathcal{X}}(t,w)$. We focus on the effectiveness of online matching in random order model, *i.e.,* the theoretical guarantee in the average performance of the privacy mechanism and online algorithm.

Definition 8 (Competitive Ratio in Random Order Model)**.** *The competitive ratio of an online matching algorithm $\mathcal{A}$ in random order model for our POMBM problem is defined as*

$$CR = \max_{\forall T,W} \frac{\mathbb{E}_{\mathcal{M},\mathcal{O}}[d(M_{\mathcal{A}})]}{d(M_{\mathcal{OPT}})} \tag{2}$$

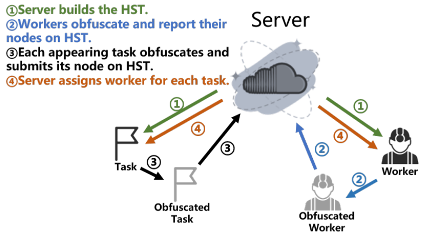

①Server builds the HST.
②Workers obfuscate and report their nodes on HST.
③Each appearing task obfuscates and submits its node on HST.
④Server assigns worker for each task.
Server
①
④
③
①
②
④
Task
③
Obfuscated Task
②
Worker
Obfuscated Worker

Fig. 1: Workflow of our solution.

where $\mathbb{E}_{\mathcal{M},\mathcal{O}}[\cdot]$ represents the expectation of a variable over the distribution of $\mathcal{M}$ and all random orders, and $M_{\mathcal{OPT}}$ is the optimal matching with the minimum total distance given that both T and W are foreknown.

III. A Tree-Based Solution

This section presents our tree-based solution to the POMBM problem. Specifically, we devise a novel privacy mechanism that is ϵ-Geo-I on a dedicated tree structure called the Hierarchically Well-Separated Tree (HST), and then show that a greedy matching algorithm on the obfuscated tree nodes has a guaranteed competitive ratio. As next, we first present the overview solution (Sec. III-A). Then we introduce the basics of HST (Sec. III-B), present our privacy mechanism (Sec. III-C), and further devise a random walk method to accelerate the mechanism (Sec. III-D). Finally, we show the existence of efficient online minimum bipartite matching algorithms on the data protected by our mechanism (Sec. III-E).

A. *Overview*

The workflow of our solution follows the interaction model among workers, tasks and the server introduced in Sec. II-A, but is operated on a dedicated metric space embedded by a Hierarchically Well-Separated Tree (HST) [21]. Fig. 1 illustrates the workflow of our solution. It consists of four steps.

- The server constructs an HST upon a predefined set of points and publishes the tree as well as the set of points.
- Each worker w maps his/her location to a node on the HST, which is then transformed to an obfuscated node on the tree via a privacy mechanism $\mathcal{M}$. The obfuscated nodes on the HST from workers are reported to the server.
- When a new task t appears, its location is mapped to a node on the HST and then transformed to an obfuscated node on the tree via $\mathcal{M}$. The task with the obfuscated node on the HST is then submitted to the server.
- Upon receiving the task with the protected location information, the server runs an online matching algorithm $\mathcal{A}$ to assign a worker to the task. We show that there is an algorithm that achieves a guaranteed competitive ratio on these obfuscated data.

We make the following discussions on the above workflow.

- Our intuition to use HST for our solution are two-fold. *(i)* HSTs are widely used for optimizing distance-related objectives in matching since the distance in the metric space can be upper and lower bounded by the distance on HSTs. HST-based solutions prove effective to the OMBM problem [15], [19], [10], which is part of our requirement on the privacy mechanism. *(ii)* HSTs are tree structures and the edges in the same level have the same length. These properties are crucial to design a mechanism that satisfies ϵ-Geo-I, as will be explained in Sec. III-B.
- As with existing studies [7], we assume that after task assignment, workers can obtain the exact locations of the *assigned tasks* via an extra privacy channel (*e.g.* smartphones). Note that we mainly focus on privacy mechanisms against the untrusted server. Security risks from malicious workers are out of the scope of this work.

Algorithm 1: Construction of a complete HST.

```
input  : A metric space $(V, d)$.
output: A tree space $(V_{\mathcal{T}}, d_{\mathcal{T}})$.
$\pi \leftarrow$ a random permutation of $V$,
  $D \leftarrow \lceil \log_2(2 \cdot \max_{a,b \in V} d(a,b)) \rceil$, $\beta \leftarrow$ uniformly
  generated from $[\frac{1}{2}, 1]$;
$\mathcal{S}_D \leftarrow \{V\}$;
for $i \leftarrow D-1$ to $0$ do
    $r_i \leftarrow \beta \cdot 2^i$;
    Let $\mathcal{S}_i$ be an empty set;
    for $S \in \mathcal{S}_{i+1}$ do
        $T \leftarrow S$;
        for $j \leftarrow 1$ to $k$ do
            // The set of those points in $T$ whose
              distance to $\pi(j)$ closer than $r_i$
              $U \leftarrow \{u \in V | d(u, \pi(j)) \leq r_i\} \cap T$;
            if $U$ is not empty then
                Add $U$ to $\mathcal{S}_i$;
                Make $U$ a child node of $S$ with
                  distance $2^{i+1}$;
                $T \leftarrow T - U$;
$c \leftarrow$ maximum number of branches in the tree;
For each intermediate node $w$ we add fake nodes until
  it has $c$ number of child nodes;
return the HST;
```

B. *Construction of HST*

A Hierarchically Well-separated Tree (HST) [21] can be considered as a space embedding $\mathcal{T} = (V_T, d_T)$ of arbitrary metric space (V, d) such that each leaf node located at level 0 corresponds to a point in V and the distance on the tree from a node at level i to its parent is 2^{i+1}. An important property of the HST is that $d(u,v) \leq \mathbb{E}[d_T(u,v)] \leq O(\log|V|)d(u,v)$.

Before designing our privacy mechanism, we first need to construct an HST. The construction of the HST has two features. *(i)* We construct the HST on a fixed set of predefined points. This is because according to our interaction model (Sec. II-A), the server has no clue about the exact locations of workers and tasks. It also saves communication cost because otherwise the structure of the HST will change according to the locations of tasks or workers as new workers or tasks appear dynamically. *(ii)* We construct a complete HST by adding fake nodes to simplify the information about the HST that needs to be communicated to workers and tasks so as to further save the communication overhead when publishing the HST.

Alg. 1 illustrates the procedure to construct a complete HST, where an HST is first built and then made into a complete one. Initially, we calculate the number of levels, and randomly generate a permutation of V and the factor r of the radius of the levels in line 1. The root node includes V at the beginning. Lines 4-13 then construct the tree from top to bottom. For each cluster in each level (*i.e.,* S), we see if any point in S locates in the ball centered at $\pi(j)$ with radius r_i in the order of π (lines 9-10). If yes, these points are set to be a child node of S (lines 11-12) and removed from S (line 13). Then for the HST, we simply fill each intermediate nodes up such that the HST becomes a complete c-ary HST, as shown in lines 14-15.

Example 1. *Fig. 2 shows an example of building a complete HST from the set of nodes* $V = \{o_1(1,1),\ o_2(2,3),\ o_3(5,3),\ o_4(4,4)\}$. *We know that* $D = \lceil \log_2(2 \cdot d(o_1, o_3)) \rceil = 4$. *Assume we randomly choose the permutation* $\pi =< o_1, o_2, o_3, o_4 >$ *and* $\beta = \frac{1}{2}$. *For the first iteration we have* $r_3 = 4$. *We split* V *in order of* π *into* $\{o_1, o_2\}$ *(located in the circle centered at* o_1 *with radius* r_3*) and* $\{o_3, o_4\}$ *(located in the circle centered at* o_2 *with radius* r_3*), as the red circles show in Fig. 2a. The corresponding tree at this time expands to level* 3 *in Fig. 2b. Then when* $i = 2$*, as shown in Fig. 2a, we draw the blue circle centered at each point in order of* π *with radius* $r_2 = 2$*, and see if these circles have intersection with two obtained subsets* $\{o_1, o_2\}$ *and* $\{o_3, o_4\}$ *when* $i = 3$*. The subset* $\{o_1, o_2\}$ *is split into* $\{o_1\}$ *and* $\{o_2\}$*, and we split the node* $\{o_1, o_2\}$ *at level* 3 *into* $\{o_1\}$ *and* $\{o_2\}$*. A same procedure goes at level* 2 *and level* 1*. After constructing the HST, we find its maximum number of branches is* 2*, and add fake nodes to make it complete. Finally, the complete HST is shown in Fig. 3.*

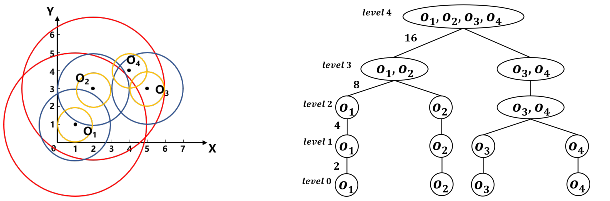
Complexity Analysis. Denote D as the level of the HST. The construction of an HST takes $O(N^2 \cdot D)$ time. To further build a complete HST, the algorithm needs to traverse the complete HST, which takes c^D time. Hence the total time to construct a complete HST is $O(N^2 \cdot D + c^D)$.

Once the HST is constructed and published (together with the predefined set of points), each worker/task will choose the node on the HST whose corresponding predefined point in the Euclidean space is nearest to his/her actual location. These nodes are then fed into our privacy mechanism to generate obfuscated ones, which are finally reported to the server.

C. Privacy Mechanism on HST

This subsection presents our privacy mechanism on the HST, and proves that it is ϵ-Geo-Indistinguishable. We assume a complete c-ary HST.

Our Mechanism. Given a leaf node x transformed from the location of a task/worker, we first partition the whole leaf nodes (the set of which is denoted as L) based on the level of the least common ancestor (LCA) between x and the leaf nodes. Specifically, define *sibling node set at level* i, denoted by $L_i(x)$, as the set of nodes whose LCA with x is located exactly at level i. Let $L_0(x) = \{x\}$. Then $\cup_{i=0}^{D} L_i(x) = L$ and $|L_i(x)| = (c-1)c^{i-1}$ for $i \geq 1$. And the distance on the tree between x and a node $a \in L_i(x)$ is $2^{i+2} - 4$.



(a) Circles in Example 1. (b) The HST in Example 1.

Fig. 2: The circles and the HST in Example 1.

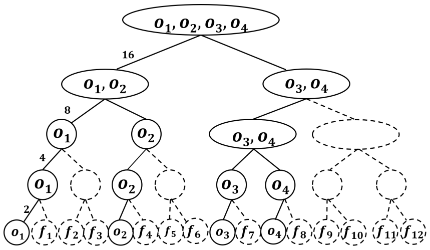


Fig. 3: The complete HST in Example 1.

Now we obfuscate x. For each leaf node in $L_i(x)$, we assign it a weight wt_i, which represents the portion that the node is chosen as the obfuscated node. Specifically, since there is exactly one leaf node in $L_0(x)$ and $c^{i-1}(c-1)$ nodes in $L_i(x)$ for $i \geq 1$, the total weight of all leaf nodes is $WT = wt_0 + \sum_{i=1}^{D} c^{i-1}(c-1)wt_i$. And a leaf node in $L_i(x)$, denoted by z, will be chosen as the obfuscated node with a probability

$$\mathcal{M}(x)(z) = \frac{wt_i}{WT}. \tag{3}$$

Next we determine the values of wt_i. To satisfy ϵ-Geo-I, wt_0 must be no greater than $e^{(2^{i+2}-4)\epsilon} wt_i$ for any $i \geq 1$. To obtain a smaller distance, we choose $wt_0 = 1$ and $wt_i = e^{-(2^{i+2}-4)\epsilon} wt_0 = e^{(4-2^{i+2})\epsilon}$. Hence we have

$$WT = 1 + \sum_{i=1}^{D} c^{i-1}(c-1)e^{(4-2^{i+2})\epsilon}. \tag{4}$$

Now we obtain a mechanism, and we will show that the proposed mechanism satisfies ϵ-Geo-I in Theorem. 1.

Alg. 2 shows how our privacy mechanism $\mathcal{M}$ obfuscates a node x on the HST. For each leaf node in T, we compute its probability of being the obfuscated node according to Eq.(3), as shown in line 1. The obfuscated node is then chosen based on the probability and reported to the server.

Example 2. *Back to our settings in Example 1. Suppose we want to obfuscate node* o_1 *in Fig. 3 with* $\epsilon = 0.1$*. For a leaf node whose LCA with* x *is at level* i*, the weight and the probability of the node being the obfuscated one are listed in Table I. Take* $i = 1$ *as an example. The weight of the leaf node* f_1 *is* $e^{-4\epsilon} = 0.670$*, and the probability of* f_1 *being the obfuscated node is* $wt_1/(w_0 + \sum_{i=1}^{D} 2^{i-1} wt_i) = 0.264$*. Our*

Algorithm 2: Privacy mechanism $\mathcal{M}$ on HST.

```
input : A complete $c$-ary HST $T$, a leaf node $x$ on $T$,
        and a privacy budget $\epsilon$.
output: The obfuscated leaf node on the tree.
Compute for each leaf node $a$ in $T$ the probability
  $\mathcal{M}(x)(a)$ according to Eq.(3);
Sample $x$ to the obfuscated leaf node $x'$;
return $x'$;
```

TABLE I: Probability of leaf nodes being the obfuscated nodes.

Level i	$L_i(o_1)$	wt_i	Probability
0	o_1	1	0.394
1	f_1	0.670	0.264
2	f_2, f_3	0.301	0.119
3	$o_2, f_4 - f_6$	0.061	0.024
4	$o_3 - o_4, f_7 - f_{12}$	0.002	0.001

mechanism then randomly chooses a leaf node among all these leaf nodes with the probability in Table I.

Geo-Indistinguishability. $\mathcal{M}$ is ϵ-Geo-I, which is ensured by the following theorem.

Theorem 1. *The mechanism $\mathcal{M}$ (Alg. 2) is ϵ-Geo-I. That is, given two leaf nodes x_1 and x_2 on the HST, $\mathcal{M}$ satisfies*

$$\mathcal{M}(x_1)(z) \leq e^{\epsilon d_T(x_1,x_2)} \mathcal{M}(x_2)(z) \tag{5}$$

for leaf node $z \in L$, where $\mathcal{M}(x_1)(z)$ is the probability that node x_1 is obfuscated to node z, and $d_T(x_1, x_2)$ is the distance between x_1 and x_2 on the HST.

Proof. We use $lca(u, v)$ and $lvl(u, v)$ to represent LCA of u and v and the level of LCA of u and v, respectively. We prove the theorem in two cases.

Case 1: $lvl(x_1, z) > lvl(x_1, x_2)$. In this case z is located outside the subtree rooted at $lca(x_1, x_2)$ (which we denote as $T_{lca(x_1,x_2)}$). A first observation is that $lvl(x_2, z) > lvl(x_1, x_2)$. In this case the LCA of x_1 and z, *i.e.,* $lca(x_1, z)$, and that of x_2 and z, *i.e.,* $lca(x_2, z)$, coincide, and they both have a higher level than $lca(x_1, x_2)$. This means that for either x_1 or x_2, the weight assigned to z is the same, *i.e.,* $wt_{lvl(x_1,z)} = wt_{lvl(x_2,z)}$. Hence it has the same probability for x_1 and x_2 being obfuscated to z. Since $\epsilon d_T(x_1, x_2) \geq 0$ always holds, 5 holds if $lvl(x_1, z) \geq lvl(x_1, x_2)$.

Case 2: $lvl(x_1, z) \leq lvl(x_1, x_2)$. In this case, we have $lvl(x_2, z) \leq lvl(x_1, x_2)$, since otherwise $lca(x_1, z)$ will also be greater than $lca(x_1, x_2)$, which contradict to our assumption. This means x_1, x_2 and z are located in a same subtree rooted at $lca(x_1, x_2)$. Also note that $lvl(x_1, z) \geq 0$. Hence,

$$\begin{aligned}\frac{\mathcal{M}(x_1)(z)}{\mathcal{M}(x_2)(z)} =& \frac{wt_{lvl(x_1,z)}}{wt_{lvl(x_2,z)}} \\ =& exp\{\epsilon(2^{lvl(x_2,z)+2} - 2^{lvl(x_1,z)+2})\} \\ \leq& exp\{\epsilon(2^{lvl(x_1,x_2)+2} - 4)\} \\ =& e^{\epsilon d_T(x_1,x_2)}.\end{aligned} \tag{6}$$

Thus the theorem also holds if $lvl(x_1, z) \leq lvl(x_1, x_2)$. □

Complexity Analysis. $\mathcal{M}$ enumerates all the leaf nodes in the complete HST and each node has c branches. Hence its time complexity is $O(c^D)$ where D is the height of the tree.

D. Random Walk Based Acceleration

Since the naive implementation (Alg. 2) of our privacy mechanism takes $O(c^D)$, we propose a random walk based alternative, which still generates the same distribution as Alg. 2, but reduces the time complexity to only $O(D)$.

Random Walk Based Implementation of $\mathcal{M}$. Recall that $L_i(x)$ is the set of leaf nodes whose LCA with x is located at level i, and wt_i is the weight for a node in $L_i(x)$ being sampled. Our key observation is that given the exact node x and any level i, each node in $L_i(x)$ has the same probability being sampled.

Concretely, we define tw_k as the total weight of leaf nodes whose level of LCA with x is equal to or greater than k:

$$tw_k = \begin{cases} \sum_{i \geq k}^{D} c^{i-1}(c-1)wt_i, & \text{if } k > 0 \\ w_0 + \sum_{i=1}^{D} c^{i-1}(c-1)wt_i = WT, & \text{if } k = 0. \end{cases} \tag{7}$$

The random walk method first walks upward along the tree from the exact leaf node. In each passed intermediate node located at level i, we continue to walk upward with probability $pu_i = \frac{tw_{i+1}}{tw_i}$ and with probability $1 - pu_i = \frac{c^{i-1}(c-1)wt_i}{tw_i}$ change to walk downward. Once we decide to walk downward at an intermediate node u located at level i_{dw}, among each child node of u except for the node which is the ancestor of the exact node x, we randomly choose a node to walk downward, each of which with probability $\frac{1}{c-1}$. Note that we ignore the ancestor of x located at level $i_{dw} - 1$, say $anc_{i_{dw}-1}(x)$, because those leaf nodes located at the subtree rooted at $anc_{i_{dw}-1}(x)$ do not belong to $L_i(x)$. After we turn our direction to walk downward, at each passed intermediate node u we uniformly choose a child node of u (*i.e.,* each child node with probability $\frac{1}{c}$) to go downward, until we reach a leaf node. The leaf node is then chosen to be the obfuscated node of the exact leaf node x.

Alg. 3 shows the pseudocode of the random walk based implementation. We use u to represent the current node we are going through, and I_{upward} the direction (line 2). We first walk upward and in each passed intermediate node choose whether to change the direction or not (lines 4-7). For the first time of walking downward (now $i = i_{dw}$), we uniformly choose one of the child of u except for $anc_{i-1}(u)$ (line 10). Finally we uniformly choose a child node of the current intermediate node until we reach a leaf node (lines 11-12).

Example 3. *Back to our settings in Example 2. Fig. 4 shows the probability for random walking in each level and one possible path starting from o_1 on the tree. The path is marked in red arrows. Note that we change the notations of intermediate nodes in Fig. 4 for a clear description. We start at o_i and go on walking upward with probability $pu_0 = 0.606$ at node o_1 and with probability $pu_1 = 0.564$ at node $o_{1,1}$. Now we reach the node $o_{2,1}$ at level 2. Suppose the sampling at level 2 changes our direction to downward. Since node $o_{2,1}$ has only*

Algorithm 3: Random walk based implementation of $\mathcal{M}$.

```
input : a complete HST $T$, a leaf node $x$ on $T$, and a
        privacy budget $\epsilon$.
output: An obfuscated leaf node on the tree.
$u \leftarrow x$; // the node we are going
$I_{upward} \leftarrow 1$;
while True do
    $i \leftarrow$ level of $u$;
    $I_{upward} \leftarrow 1$ with probability $pu_i$ or 0 with
      probability $1 - pu_i$;
    if $I_{upward}$ is 1 then
        $u \leftarrow$ parent of $u$;
    else
        break;
$u \leftarrow$ uniformly choose one of the child nodes of $u$
  except for $anc_{i-1}(u)$ ;
while $u$ is not a leaf node do
    $u \leftarrow$ uniformly choose one of the child nodes of $u$;
return $u$;
```

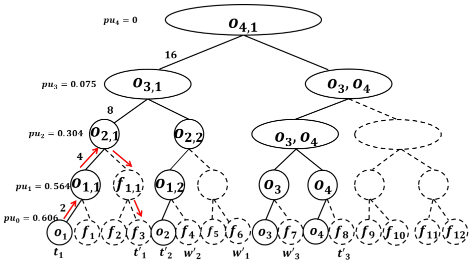


Fig. 4: The path of random walk in Example 3 and the nodes of tasks and workers in Example 4.

two child nodes and $o_{1,1}$ is where the leaf node o_1 comes from. Hence with probability 1 we walk downward to node $f_{1,1}$. At node $f_{1,1}$ we reach the fake node f_3 with probability 0.5 and choose f_3 as the obfuscated node. The probability of o_1 being obfuscated to f_3 is $pu_0 \times pu_1 \times (1 - pu_2) \times 1 \times 0.5 = 0.119$.

Correctness of Random Walk Based Method. The random walk method is still ϵ-Geo-I based on the following theorem.

Theorem 2. *The random walk method in Alg. 3 generates the same distribution as that in Alg. 2.*

Proof. Suppose the exact node is x. For a leaf node a whose LCA with x is located at level $lvl(x, a)$, *i.e.,* $a \in L_i(x)$, the random walk method finishes at a if and only if

- we walk upward until reaching $lca(x, a)$ and
- in each time of downward walking we choose the child node which is the ancestor of a (or a itself).

If $lvl(x, a) = 0$, *i.e.,* x and a coincide, x is chosen to be the obfuscated node when we change the direction immediately at x. The probability is

$$\mathcal{M}(x)(a) = 1 - pu_0 = \frac{wt_0}{WT}.$$

When $lvl(x, a) > 0$, the probability that a is chosen to be the obfuscated node, *i.e.,* $\mathcal{M}(x)(a)$, is

$$\left(\prod_{i=0}^{lvl(x,a)-1} pu_i\right) \cdot \frac{1 - pu_{lvl(x,a)}}{c^{lvl(x,a)-1}(c-1)} = \frac{wt_{lvl(x,a)}}{WT},$$

which is exactly the probability in Alg. 2. □

Complexity Analysis. The random walk traverses each level on the tree at most twice. Hence its time complexity is $O(D)$.

E. Effectiveness of Task Assignment on Obfuscated Nodes

Recall that a privacy mechanism for our POMBM problem should not only be ϵ-Geo-Indistinguishable but also allow online task assignment with a bounded competitive ratio. As next, we present an HST-based greedy algorithm which operates on the obfuscated nodes and achieves a competitive ratio of $O(\frac{1}{\epsilon^4} \log N \log^2 k)$, where N is the size of the predefined point set and $k = \min\{n, m\}$ is the size of the matching result.

Algorithm 4: A greedy algorithm on HST.

```
  input : A complete HST $T$, the set of obfuscated
          points of unassigned workers $W'$.
  output: The matching $M$.
1 $W'_u \leftarrow W'$, $M \leftarrow \emptyset$;
2 for Each appearing task with obfuscated node $t'$ do
3     $w^* \leftarrow$ the closest node in $W'_u$ on $T$ (ties are
        broken arbitrarily);
4     $M \leftarrow M \cup \{(t', w^*)\}$;
5     $W'_u \leftarrow W'_u - w^*$;
6 return $M$;
```

HST-Based Greedy Algorithm. Alg. 4 illustrates the HST-Greedy algorithm for the server to perform online task assignment on the obfuscated nodes. W'_u and M represent the set of unsigned workers and the temporary matching, respectively (line 1). For each new task with the corresponding obfuscated node t', the algorithm assigns the task to the worker that is the closest to t' on the HST and removes the chosen worker from W'_u, as shown in lines 3-5.

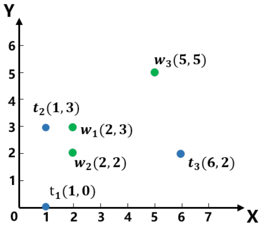


Fig. 5: The locations of tasks and workers.

Example 4. *Back to Example 1. Suppose there are three workers w_1-w_3 and three tasks t_1-t_3, whose locations are shown in Fig. 5. The appearing order of the tasks is t_1, t_2, t_3. The obfuscated nodes t'_1-t'_3 and w'_1-w'_3 are shown in Fig. 4. t'_1 first appears, and w'_1 and w'_2 are the two closest obfuscated nodes to t'_1. Suppose we assign t'_1 to w'_2. After that t'_2 appears* *and is assigned to w_1' as w_1' is closer on the tree. Then t_3' appears and we assign t_3' to w_3'. Finally we obtain the matching $M = \{(t_1', w_2'), (t_2', w_1'), (t_3', w_3')\}$.*

Competitive Ratio Analysis. The competitive ratio analysis of Alg. 4 leverages the observation that the expectation distance between a node v and an obfuscated node u' on the HST is upper and lower bounded, as claimed in the lemmas below.

Lemma 1. *Given a leaf node u, the obfuscated leaf node u' of u by our privacy mechanism, and a leaf node v, the expected distance (on the HST) between u' and v is no less than $\frac{1}{3(2c-1)}$ times the distance between u and v,* i.e., $\mathbb{E}_{\mathcal{M}}[d_T(u', v)] \geq \frac{1}{3(2c-1)} d_T(u, v)$.

Proof. For simplicity, we use $l_{u,v}$ to represent the level of the LCA of u and v. Then the distance between u and v is $d_T(u, v) = 2^{l_{u,v}+2} - 4$. Denote $T_{l_{u,v}-1}(v)$ as the subtree which contains v and locates at level $l_{u,v} - 1$. In the following we assume $l_{u,v} \geq 1$ as the lemma obviously holds when $l_{u,v} = 0$. u' can be any leaf node and the corresponding probability depends on its distance to u. The expectation of $d_T(u', v)$ is

$$\mathbb{E}_{\mathcal{M}}[d_T(u', v)] = \sum_{a \in L} \mathcal{M}(u)(a) \cdot d_T(a, v). \tag{8}$$

Depending on whether a leaf node z is located in $T_{l_{u,v}-1}(v)$, we partition the whole leaf node set L into L_{in} (inside $T_{l_{u,v}-1}(v)$) and L_{out} (outside $T_{l_{u,v}-1}(v)$), and calculate their values in Eq.(8) correspondingly.

We first bound the value of Eq.(8) for those **nodes in** L_{out}, *i.e.,* $\sum_{a \in L_{out}} \mathcal{M}(u)(a) \cdot d_T(a, v)$. The observation is that $d_T(a, v) \leq d_T(u, v)$ for $a \in L_{out}$. Hence we have

$$\begin{aligned}\sum_{a \in L_{out}} \mathcal{M}(u)(a) \cdot d_T(a, v) \geq & \sum_{a \in L_{out}} \mathcal{M}(u)(a) d_T(u, v) \\ =& d_T(u, v)(1 - \sum_{a \in L_{in}} \mathcal{M}(u)(a)) \\ =& d_T(u, v)(1 - \frac{c^{l_{u,v}-1} wt_{l_{u,v}}}{WT}). \end{aligned} \tag{9}$$

When $l_{u,v} = 1$, we have $(1 - \frac{wt_1}{W_u}) \geq 1 - \frac{e^{-4\epsilon}}{1+e^{-4\epsilon}} \geq \frac{1}{3(2c-1)}$, and the lemma holds. Hereafter, we will assume $l_{u,v} \geq 2$.

For those **nodes in** L_{in}, their probability of being the obfuscated node is the same, *i.e.,* $\mathcal{M}(u)(a) = \mathcal{M}(u)(b)$ for $a, b \in L_{in}$. We bound the value of Eq.(8) as follows.

$$\begin{aligned} & \sum_{a \in L_{in}} \mathcal{M}(u)(a) \cdot d_T(u', v) \\ =& \sum_{i=1}^{l_{u,v}-1} c^{i-1}(c-1)(2^{i+2} - 4) \frac{wt_{l_{u,v}}}{WT} \\ =& \frac{wt_{l_{u,v}}}{WT} \{8(c-1) \sum_{i=1}^{l_{u,v}-1} c^{i-1} 2^{i-1} - 4(c-1) \sum_{i=1}^{l_{u,v}-1} c^{i-1}\} \\ \geq& \frac{wt_{l_{u,v}}}{WT} c^{l_{u,v}-1} (\frac{c-1}{2c-1} 2^{l_{u,v}+2} - 4) \\ \geq& \frac{wt_{l_{u,v}}}{WT} c^{l_{u,v}-1} \frac{1}{3(2c-1)} d_T(u, v), \end{aligned} \tag{10}$$

where the last deduction comes from

$$\begin{aligned} & (\frac{c-1}{2c-1} 2^{l_{u,v}+2} - 4)/d_T(u, v) \\ =& \frac{c-1}{2c-1} - (\frac{4c}{2c-1})/(2^{l_{u,v}+2} - 4) \\ \geq& \frac{c-1}{2c-1} - \frac{c}{3(2c-1)} = \frac{2c-3}{3(2c-1)} \geq \frac{1}{3(2c-1)} \end{aligned} \tag{11}$$

when $l_{u,v} \geq 2$ and $c \geq 2$. From Eq.(9) and Eq.(10) we have

$$\begin{aligned} & \sum_{a \in L} \mathcal{M}(u)(a) \cdot d_T(u', v) \\ =& \sum_{a \in L_{out}} \mathcal{M}(u)(a) \cdot d_T(u', v) + \sum_{a \in L_{in}} \mathcal{M}(u)(a) \cdot d_T(u', v) \\ \geq& \frac{1}{3(2c-1)} d_T(u, v). \end{aligned} \tag{12}$$

□

Lemma 2. *Given a leaf node u, the obfuscated leaf node u' of u by our tree-based mechanism, and a leaf node v,* $\mathbb{E}_{\mathcal{M}}[d_T(u', v)] \leq O((\frac{\ln 2c}{\epsilon})^{\log_2 2c}) d_T(u, v)$.

Proof. We calculate the upper bound of the expectation of $d_T(u', v)$ in a similar way to Lemma. 1. Denote $T_{u,v}$ as the subtree rooted at LCA of u and v, and we use L'_{in} and L'_{out} to represent the leaf nodes inside and outside $T_{u,v}$ respectively. For those **leaf nodes in** L'_{in}, we have $\sum_{a \in L'_{in}} \mathcal{M}(u)(a) \cdot d_T(a, v) \leq d_T(u, v)$ as $d_T(a, v) \leq d_T(u, v)$.

For **leaf nodes in** L'_{out},

$$\begin{aligned} & \sum_{a \in L'_{out}} \mathcal{M}(u)(a) \cdot d_T(a, v) \\ \leq & \sum_{i=l_{u,v}}^{D} c^{i-1}(c-1) 2^{i+2} e^{(4-2^{i+2})\epsilon} / WT = \sum_{i=l_{u,v}}^{D} S_i \end{aligned} \tag{13}$$

where we denote $S_i = c^{i-1}(c-1)2^{i+2}e^{(4-2^{i+2})\epsilon}/WT$ as the i-th term in Eq.(13). Notice that $2^{i+2} \leq 2(2^{i+2} - 4)$ always holds. Hence $S_{l_{u,v}} = O(d_T(u, v))$.

Observe that $S_{i+1}/S_i = 2c/e^{2^{i+2}\epsilon} \leq 2c$. Let $i^* = \lceil \log_2 \frac{\ln(2c)}{\epsilon} \rceil - 2$, then $2c/e^{2^{i^*+2}\epsilon} \leq 1$. And for $i > i^*$ we have $\frac{S_{i+1}}{S_i} = 2c/e^{2^{i+2}\epsilon} \leq 2c/e^{2^{i^*+3}\epsilon} \leq \frac{1}{2c}$.

Now we are ready to get the upper bound of Eq.(13). For $l_{u,v} \leq i \leq i^*$ (if any), $S_i \leq (2c)^{i-l_{u,v}} S_{l_{u,v}}$. Then we have

$$\begin{aligned} \sum_{i=l_{u,v}}^{i^*} S_i \leq & \sum_{i=l_{u,v}}^{i^*} (2c)^{i-l_{u,v}} S_{l_{u,v}} = \frac{(2c)^{i^*-l_{u,v}+1} - 1}{2c-1} S_{l_{u,v}} \\ \leq & (2c)^{i^*+1} S_{l_{u,v}} = O((\frac{\ln 2c}{\epsilon})^{\log_2 2c}) d_T(u, v). \end{aligned} \tag{14}$$

For $i^* < i \leq D$ (if any),

$$\begin{aligned} \sum_{i=i^*+1}^{D} S_i \leq & \sum_{i=i^*+1}^{D} (\frac{1}{2c})^{i-i^*} S_{i^*} \leq S_{i^*} \\ \leq & (2c)^{i^*} S_{l_{u,v}} = O((\frac{\ln 2c}{\epsilon})^{\log_2 2c}) d_T(u, v). \end{aligned} \tag{15}$$

Summarizing two parts of leaf nodes in L'_{in} and L'_{out}:

$$\mathbb{E}_{\mathcal{M}}[d_T(u', v)] \leq O((\frac{\ln 2c}{\epsilon})^{\log_2 2c}) \cdot d_T(u, v).$$

□

Based on Lemmas 1-2, we have the following theorem.

Theorem 3. *Alg. 4 has a competitive ratio $O(\frac{1}{\epsilon^4} \cdot \log N \log^2 k)$, where N is the number of predefined points on the HST, and $k = \min\{n, m\}$.*

Proof. An important property of HST is that the distance on the tree can be bounded by $d(u,v) \leq \mathbb{E}[d_T(u,v)] \leq O(\log n) d(u,v)$. From Lemma. 1 we have $\mathbb{E}_{\mathcal{M}}[d_T(t', w')] \geq \frac{1}{3(2c-1)} \mathbb{E}_{\mathcal{M}}[d_T(t, w')] \geq \frac{1}{9(2c-1)^2} d_T(t, w)$. Similarly from Lemma. 2 we have $\mathbb{E}_{\mathcal{M}}[d_T(t', w')] \leq O((\frac{\ln 2c}{\epsilon})^{2\log_2 2c}) \cdot d_T(t, w)$. Let $lb = \frac{1}{9(2c-1)^2}$ and $ub = O((\frac{\ln 2c}{\epsilon})^{2\log_2 2c})$. Further denote $M_{\mathcal{OPT}}$ as the optimal matching in the Euclidean space. Thus

$$\begin{aligned}
\mathbb{E}_{\mathcal{M},\mathcal{O}}[d(M_{\mathcal{A}})] &\leq \mathbb{E}_{\mathcal{M},\mathcal{O}}[\sum_{(t,w)\in M_{\mathcal{A}}} d_T(t,w)] \\
&\leq \frac{1}{lb} \cdot \mathbb{E}_{\mathcal{M},\mathcal{O}}[\sum_{(t,w)\in M_{\mathcal{A}}} d_T(t', w')] \\
&\leq \frac{\log k}{lb} \cdot \mathbb{E}_{\mathcal{M},\mathcal{O}}[\sum_{(t,w)\in M_{\mathcal{OPT}}} d_T(t', w')] \\
&\leq \frac{ub \cdot \log k}{lb} \cdot \mathbb{E}_{\mathcal{O}}[\sum_{(t,w)\in M_{\mathcal{OPT}}} d_T(t, w)] \\
&\leq O(\frac{ub \cdot \log^2 k \log N}{lb}) \cdot d(M_{\mathcal{OPT}}) \qquad (16)
\end{aligned}$$

where we use the fact that the HST-Greedy algorithm has a competitive ratio of $O(\log N \log^2 k)$ when the HST is constructed on the predefined points [15]. Substituting the values of lb and ub, we have

$$\mathbb{E}_{\mathcal{M},\mathcal{O}}[d(M_{\mathcal{A}})] \leq d(M_{\mathcal{OPT}}) O((\frac{\ln 2c}{\epsilon})^{2\log_2 2c} \cdot \log N \log^2 k).$$

Since an arbitrary HST can be transformed to a binary HST [20], we assume $c = 2$. Hence the competitive ratio of Alg. 4 in our tree-based mechanism is $O(\frac{1}{\epsilon^4} \log N \log^2 k)$. □

Complexity Analysis. Computing the distance between two leaf nodes on the HST takes $O(D)$ time. For each appearing task, the algorithm enumerates all available workers and finds the nearest one on the tree. Hence the time complexity of Alg. 4 is $O(Dnm)$.

IV. Experiments

This section presents the evaluations of our methods.

A. Experimental Setup

Synthetic Datasets. Table II shows the parameter settings for synthetic datasets. The default settings are marked in bold. Specifically, task set (denoted by T) and worker set (denoted by W) are generated in a 200×200 Euclidean space under the Normal distribution with different mean μ and standard deviation σ. Inspired by [10], the mean μ varies from 50 to 150 and the standard deviation σ varies from 10 to 30. We also vary the number of tasks T, the number of workers W and the value of ϵ. For scalability tests we vary the number of tasks and workers at the same time from 2×10^4 to 10×10^4.

TABLE II: Experimental settings for synthetic data.

Parameters	Settings
$\lvert T\rvert$	1000, 2000, **3000**, 4000, 5000
$\lvert W\rvert$	3000, 4000, **5000**, 6000, 7000
mean μ	50, 75, **100**, 125, 150
standard deviation σ	10, 15, **20**, 25, 30
privacy budget ϵ	0.2, 0.4, **0.6**, 0.8, 1
scalability ($\lvert T\rvert = \lvert W\rvert$)	$2 \times 10^4, 4 \times 10^4, 6 \times 10^4, 8 \times 10^4, 10 \times 10^4$

TABLE III: Experimental settings for real data.

Parameters	Settings
collected date	2016/11/01, $\cdots$, 2016/11/30
$\lvert T\rvert$	range from 4245 to 5034
$\lvert W\rvert$	6000, 7000, **8000**, 9000, 10000
ϵ	0.2, 0.4, **0.6**, 0.8, 1

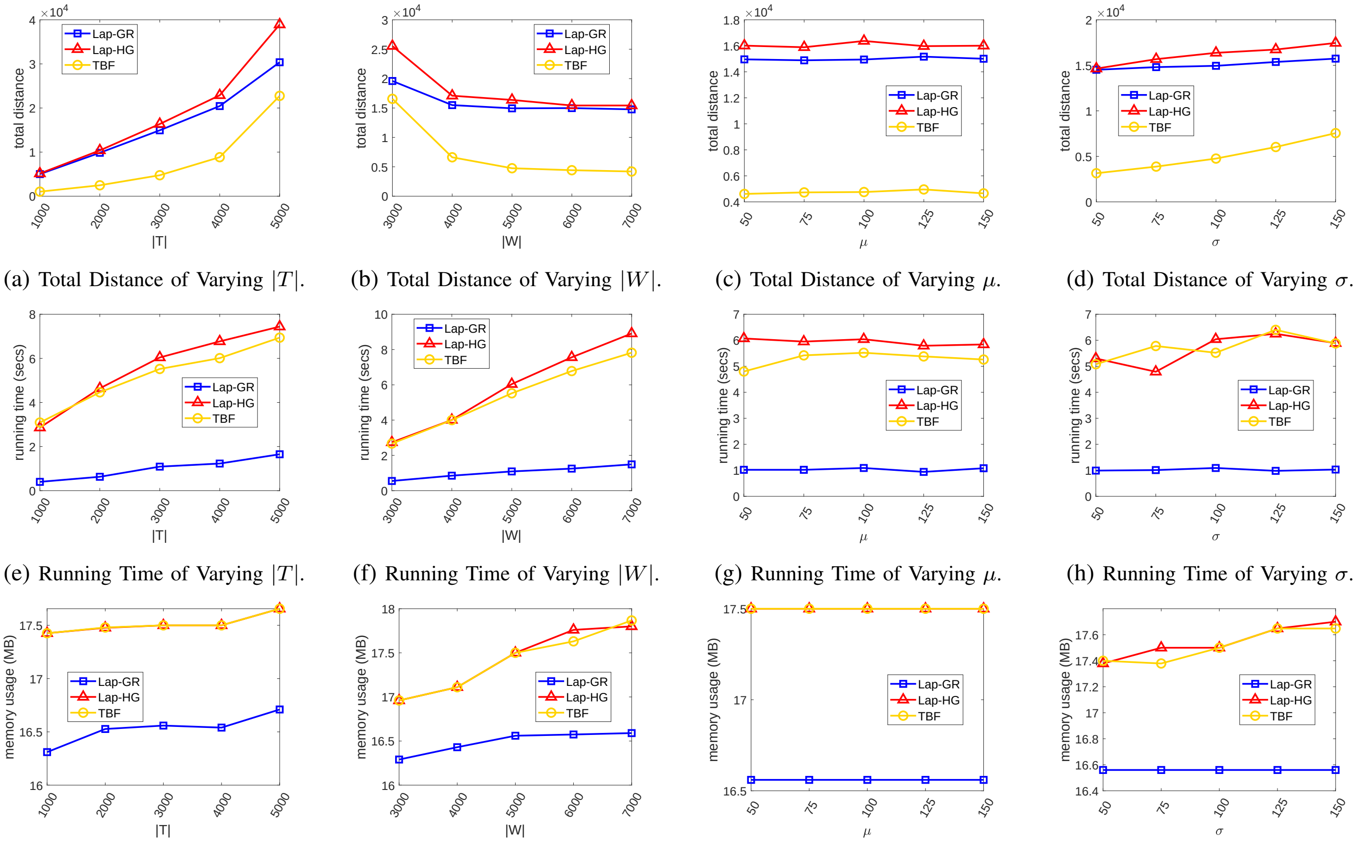
Real Datasets. Table III shows the parameter settings for the real datasets, which are collected by Didi Chuxing [22] and published through the GAIA initiative [23]. The dataset includes 7,065,937 trip records of passengers in Chengdu, China in November, 2016. We choose the records in a 10km×10km region during the peak-hour period 14:00-14:30. The location of each task is extracted by the origin of a passenger in the trip records. Finally, the real datasets include 4,245 to 5,034 tasks in the peak-hour periods of thirty days. We test on the tasks in each day and report the average value of metrics. As the information of workers and the privacy budget are not given in the real datasets, we vary their values based on the same settings as in synthetic datasets.

Compared Algorithms. Since no previous work has studies the POMBM problem before, we combine the widely used privacy mechanism (*i.e.,* the planar Laplacian distribution [8]) and two popular algorithms for OMBM (*i.e.,* greedy [10] and HST-Greedy [15]) as the baselines.

- Lap-GR (Laplacian+Greedy): This is the first baseline, where we use the planar Laplacian distribution [8] as the privacy mechanism, and the greedy algorithm [10] in the Euclidean space for task assignment.
- Lap-HG (Laplacian+HST-Greedy): This is the second baseline which uses the planar Laplacian distribution [8] as the privacy mechanism, and the HST-Greedy algorithm [15] for task assignment.
- TBF (Our Tree-Based Framework): It uses Alg. 3 as the privacy mechanism and Alg. 4 for task assignment.

Metrics. Note that all the compared algorithms are ϵ-Geo-Indistinguishable. Yet they differ in the effectiveness and efficiency of task assignment due to their differently obfuscated location data of tasks and workers. Hence we mainly compare their performances in terms of the total distance of their output matching as well as the running time and memory usage of their task assignment under the same privacy budget. Here the running time refers to the total time an algorithm takes from receiving a task to the completion of the assignment.

Implementation. All algorithms are implemented in C++. We conducted experiments on a server with 40 Intel(R) Xeon(R) E5 2.30GHz processors and 128GB memory. Each experiment is repeated 10 times. The average results are reported.



(a) Total Distance of Varying $|T|$.

(b) Total Distance of Varying $|W|$.

(c) Total Distance of Varying μ.

(d) Total Distance of Varying σ.

(e) Running Time of Varying $|T|$.

(f) Running Time of Varying $|W|$.

(g) Running Time of Varying μ.

(h) Running Time of Varying σ.

(i) Memory of Varying $|T|$.

(j) Memory of Varying $|W|$.

(k) Memory of Varying μ.

(l) Memory of Varying σ.

Fig. 6: Results of varying $|T|$, $|W|$, μ and σ on synthetic datasets.

B. Experimental Results

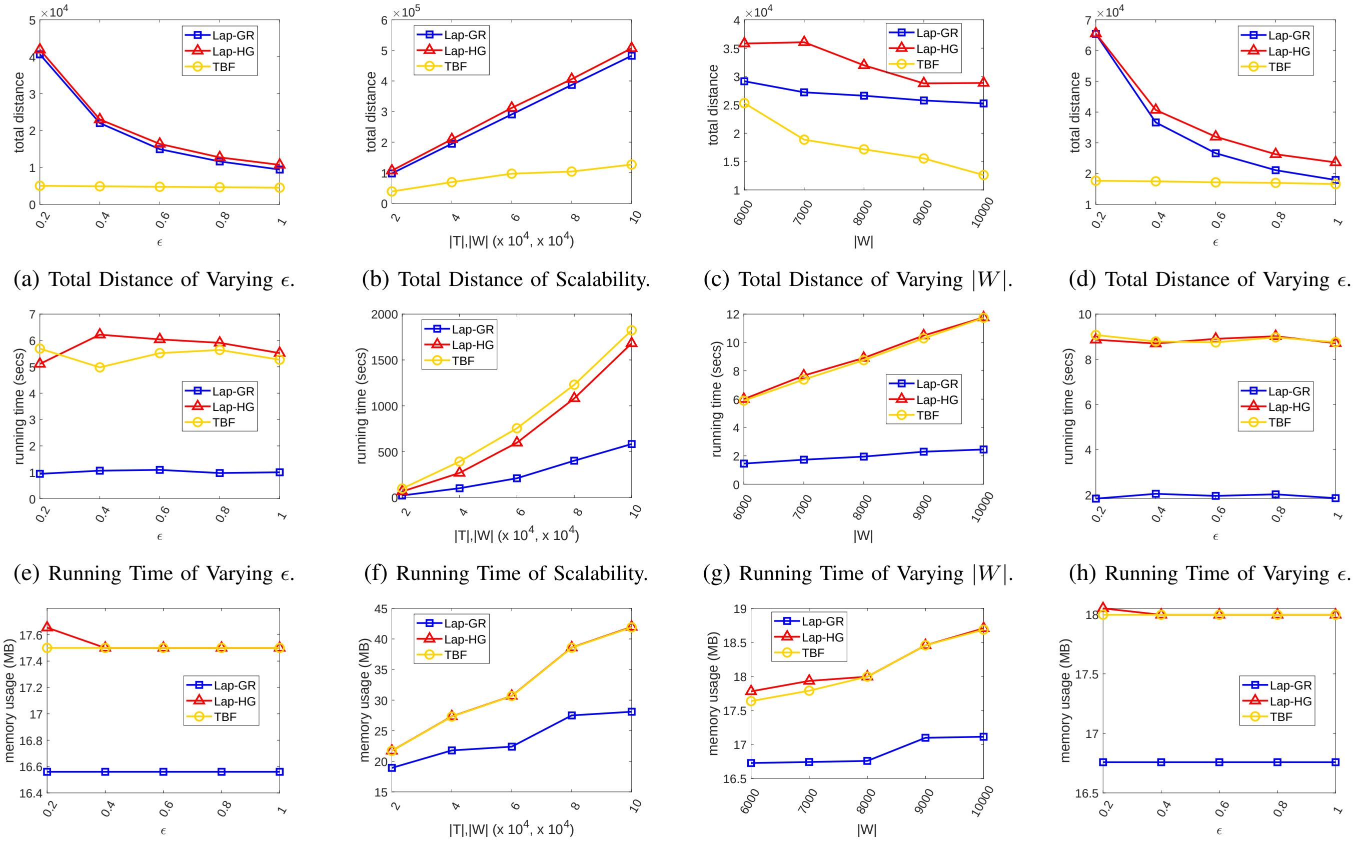
Impact of Number of Tasks. The first column of Fig. 6 shows the results of varying the number of tasks on synthetic datasets. Fig. 6a depicts the total distance of all the algorithms. Our TBF outperforms both Lap-GR and Lap-HG by up to 80.0%. For running time (Fig. 6e), Lap-GR is the most efficient and all algorithms consume more time as $|T|$ increases. This is because TBF and Lap-HG are based on the HST and have a time complexity $O(Dnm)$, while the time complexity of the greedy algorithm Lap-GR is $O(nm)$. The time complexity of all algorithms is proportional to $|T|$. Both TBF and Lap-HG are fast enough for real applications since each task can be responded (*i.e.,* assigned to a worker) in 0.0015 seconds. In terms of memory usage, Lap-GR is still the most efficient while TBF and Lap-HG consume more space of no more than 1.2 MB to construct the HST.

Impact of Number of Workers. The second column of Fig. 6 shows the results of varying the number of workers on synthetic datasets. As shown in Fig. 6b, the total distance decreases with the increase of $|W|$. This is reasonable since the tasks are more likely to be allocated to nearer workers when the number of workers becomes larger. When there are more workers, our TBF saves up to 72.8% total distances than both Lap-GR and Lap-HG. For running time in Fig. 6f, Lap-GR is the most efficient, followed by TBF and Lap-HG, and all algorithms consume more time as $|W|$ increases. This is because the time complexity of the three algorithms is proportional to the number of workers n. For memory cost in Fig. 6j, all algorithms consume no more than 18MB space.

Impact of μ of Locations. The third column of Fig. 6 shows the results of varying the mean of locations on synthetic datasets. TBF achieves the shortest total distance, followed by Lap-GR and Lap-HG (Fig. 6c). In particular, TBF achieves up to 69.2% and 71.2% shorter total distance than Lap-GR and Lap-HG, respectively. The time costs of all the algorithms are relatively stable in Fig. 6g, because their time complexity is irrelevant to μ. As for memory consumption, all the algorithms need no more than 18MB space (Fig. 6k).

Impact of σ of Locations. The last column of Fig. 6 shows the results under privacy preservation on varying the standard deviation of locations on synthetic datasets. In terms of total distance, TBF is still more effective than Lap-GR and Lap-HG (Fig. 6d). Lap-GR is always the most efficient in terms of running time and memory usage, followed by TBF and Lap-HG (Fig. 6h and Fig. 6l).

Impact of Privacy Budget. The first column of Fig. 7 shows the results of varying the privacy budget ϵ on synthetic datasets. As shown in Fig. 7a, the total distance of both Lap-GR and Lap-HG is notably higher than TBF when ϵ is small (*e.g.* 0.2), *i.e.,* with a stricter privacy protection requirement. In



(a) Total Distance of Varying ϵ.
(b) Total Distance of Scalability.
(c) Total Distance of Varying $|W|$.
(d) Total Distance of Varying ϵ.
(e) Running Time of Varying ϵ.
(f) Running Time of Scalability.
(g) Running Time of Varying $|W|$.
(h) Running Time of Varying ϵ.
(i) Memory of Varying ϵ.
(j) Memory of Scalability.
(k) Memory of Varying $|W|$.
(l) Memory of Varying ϵ.

Fig. 7: Results of varying ϵ and scalability on synthetic datasets and results of varying $|W|$ and ϵ on real datasets.

contrast, our TBF is relatively insensitive when ϵ varies from 0.2 to 1. It shows that our tree-based framework is fit for cases with a high privacy budget. As a result, TBF achieves up to 88.0% shorter total distance than both Lap-GR and Lap-HG. As shown in Fig. 7e and Fig. 7i, Lap-GR is still the most efficient while TBF and Lap-HG are also efficient enough.

Scalability Tests. The second column of Fig. 7 shows the results of scalability tests. TBF always outperforms the others in terms of total distance (Fig. 7b). Both Lap-GR and Lap-HG yield 70.0% times longer total distance than TBF. Lap-GR is the most time-efficient, which conforms to the time complexity analysis in Sec. III. TBF is also efficient, which takes no more than 0.02 seconds to assign each newly arrived task on average. In terms of memory usage, all algorithms are efficient, which consume no more than 43.8 MB space.

Real Datasets. The last two columns of Fig. 7 show the results of real datasets. TBF is always the most effective with up to 56.2% shorter total distance than Lap-GR and Lap-HG (Fig. 7c and Fig. 7d). Lap-GR is again the most time-efficient. The time cost of all the algorithms increases linearly as $|W|$ increases, while stays stable when varying μ. This is because the time complexity of the algorithms is proportional to $|W|$ (*i.e.,* n), but is irrelevant to μ. All the algorithms are memory-efficient, which consume no more than 20MB space.

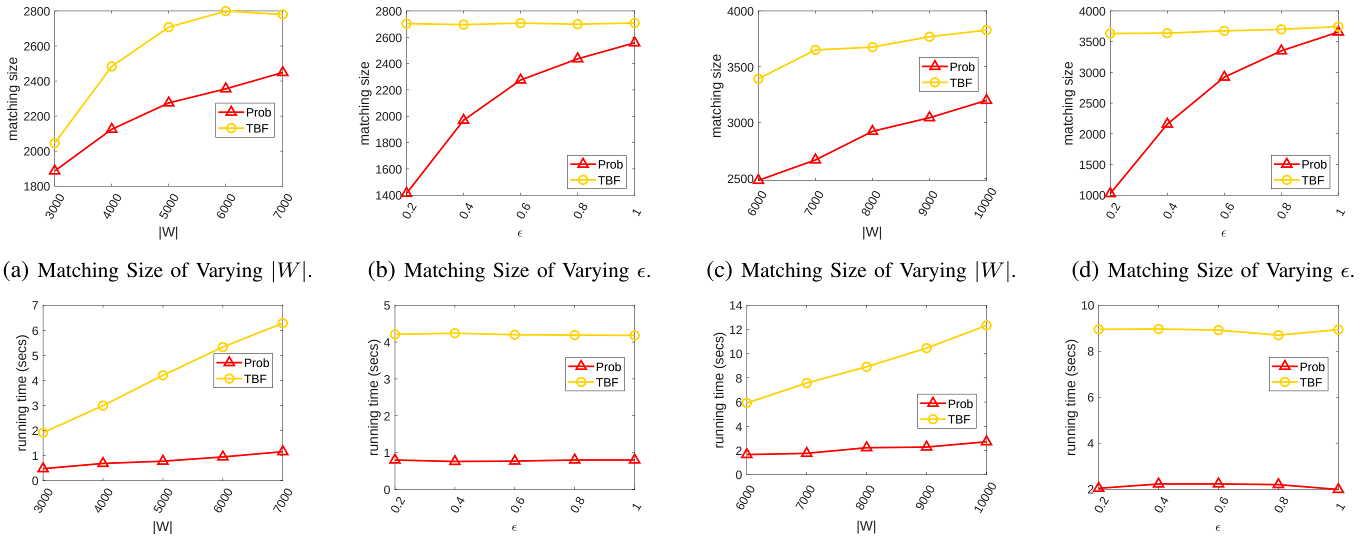
Summary of Results. Our main experimental findings are:

- While all the algorithms are ϵ-Geo-Indistinguishable, our TBF is the most effective on both synthetic datasets and real datasets. It can save up to 79.4% and 80.0% total distance than Lap-GR and Lap-HG.
- Particularly in case of stringent privacy requirements (*i.e.,* with small ϵ), our TBF significantly outperforms the baselines in terms of total distance.
- Our TBF is also efficient for real-time spatial crowdsourcing applications and only consumes small storage.

C. A Case Study on Maximization of Matching Size

In addition to minimizing the total distance, another popular objective in online task assignment in spatial crowdsourcing is maximizing the total number of matching size under incomplete bipartite graph [9]. Hence we conduct a case study on task assignment with this objective to validate our method.

Datasets. For *synthetic datasets*, we only vary the number of workers $|W|$ and the value of privacy budget ϵ with the same parameter settings in Table II due to space limit. For *real datasets*, we use the same procedure in Sec. IV-A to process the raw data and vary the same parameters as in Table III. Since there is no reachable distance for workers, we uniformly generate the reachable distance of workers within $[10, 20]$ for synthetic datasets and $[500, 1000]$ for real datasets.



(a) Matching Size of Varying $|W|$. (b) Matching Size of Varying ϵ. (c) Matching Size of Varying $|W|$. (d) Matching Size of Varying ϵ.

(e) Running Time of Varying $|W|$. (f) Running Time of Varying ϵ. (g) Running Time of Varying $|W|$. (h) Running Time of Varyint ϵ.

Fig. 8: Results of varying $|W|$ and ϵ on synthetic datasets and the results on real datasets.

Compared Algorithms.

- Prob [7]: It uses the Laplacian distribution to protect the privacy and a probability-based algorithm to assign tasks.
- TBF: We use our tree-based privacy mechanism, and for each task find the nearest reachable worker on the HST.

Metrics and Implementation. The implementation is the same as in Sec. IV-A. Due to page limit, we only compare the algorithms in terms of matching size and running time.

Experimental Results.

(1) **Impact of Number of Workers.** The first column of Fig. 8 shows the results of varying the number of workers $|W|$ on synthetic datasets. In terms of matching size, TBF always outperforms the baseline Prob with up to 16.0% more number of assigned tasks (Fig. 8a). In terms of running time, Prob is more efficient while TBF can still response each task within 0.002 seconds (Fig. 8e).

(2) **Impact of Privacy Budget.** The second column of Fig. 8 shows the results of varying the privacy budget ϵ on synthetic datasets. For matching size (see Fig. 8b), TBF achieves 5.6% to 47.7% more number of assigned tasks. As shown in Fig. 8f, the running time of both algorithms are relatively stable with the increase of ϵ.

(3) **Real Datasets.** The last two columns of Fig. 8 show the results of varying the privacy budget ϵ on synthetic datasets. In Fig. 8c and Fig. 8d, TBF consistently obtains the larger matching size by up to 26.8% times larger than Prob. As for running time, Prob is still more efficient while TBF is still able to response each task in no more than 0.003 seconds in Fig. 8g and Fig. 8h.

(4) **Summary of Results.** While both Prob and TBF satisfy Geo-Indistinguishability, our algorithm TBF is notably more effective than Prob by up to 47.7% larger matching size. Though Prob is more efficient, TBF is still efficient enough for real-world applications with no more than 0.003 seconds average response time.

V. Related Work

Online Minimum Bipartite Matching. Online minimum bipartite matching finds a maximal matching on a complete bipartite graph with a minimum total distance. It has been a popular topic in spatial crowdsourcing [10], [19], [15].

Meyerson *et al.* [15] propose to embed the points in the Euclidean space to an HST and apply a greedy algorithm on the HST to assign a worker to each task. They prove that the algorithm has a competitive ratio of $O(\log^3 k)$, where $k = \min\{n, m\}$ is the minimum between n and m. Bansal *et al.* [19] also design an algorithm based on HST. The algorithm successively assigns the task to workers (including those matched ones) until it finds an unmatched worker as the result. Recently, Tong *et al.* [10] conducts an experimental study on the state-of-the-art online minimum bipartite matching algorithms and shows that the heuristic greedy algorithm performs well on many practical settings.

Our work is inspired by these findings, *i.e.,* conducting online matching (*e.g.* using even a greedy algorithm on an HST) may have guaranteed effectiveness for task assignment. However, all the proposals are without privacy protection.

Privacy-Preserving Task Assignment in Spatial Crowdsourcing. There has been extensive research on privacy-preserving task assignment in spatial crowdsourcing [6], [7], [5], [24], [25], [26], [27], [28], [29].

A number of works [6], [30] protect the privacy of tasks or workers by transforming the location into a cloaked region, and task assignment is then executed based on the cloaked regions. In [6] and [30], the authors propose a two-stage approach, where the first stage globally assigns tasks based on the cloaked locations and the second stage locally chooses tasks based on the worker's exact location. However, these schemes are only for offline task assignment, which is unfit for many real-world applications such as ride-sharing.

Differential Privacy [31] is proposed as a stronger alternative of cloaking and has been a new standard for privacy protection. It requires that a data record cannot be distinguished by the aggregation queries (*e.g.* count) over two neighbor datasets. For example, To *et al.* [5] propose the Private Spatial Decomposition [32] to protect the differential privacy of the count of workers in regions and the task is then assigned to all the workers in a chosen region. However, existing studies focus on the privacy of aggregated queries on tasks or workers. They are unfit for queries on individual locations, which are important for task assignment in spatial crowdsourcing.

More recently, Geo-Indistinguishability [8] is proposed as a generalization of differential privacy to protect the privacy of individual location queries. Wang *et al.* [25] explored how to ensure ϵ-Geo-Indistinguishability for workers in spatial crowdsourcing. However, they neglect privacy protection for tasks and only consider offline task assignment.

Our work is most related to [7]. Both [7] and our work are ϵ-Geo-I. Compared with [7], we focus on task assignment with minimum distance, another important objective for task assignment in spatial crowdsourcing. We also propose a novel tree-based privacy mechanism, which, for the first time, allows online task assignment with a guaranteed competitive ratio.

VI. Conclusion

In this paper we explore privacy protection of location data for online task assignment in spatial crowdsourcing which *(i)* is differentially private and *(ii)* allows effective task assignment on the permuted data. We propose a novel privacy mechanism based on Hierarchically Well-Separated Trees (HSTs) and prove the mechanism is ϵ-GEO-Indistinguishable. We further design a faster implementation of the mechanism via random walk. We show that when operating on the data permuted by our mechanism, there exists a task assignment algorithm with a competitive ratio of $O(\frac{1}{\epsilon^4}\log N \log^2 k)$, where ϵ is the privacy budget, N is the number of predefined nodes on the HST, and k is the matching size. Extensive experiments on synthetic and real datasets show that online task assignment under our privacy mechanism is notably more effective than under prior differentially private mechanisms.

Acknowledgment

We are grateful to anonymous reviewers for their constructive comments. Qian Tao, Yongxin Tong and Ke Xu's works are partially supported by the National Science Foundation of China (NSFC) under Grant No. 61822201 and U1811463. Lei Chen's work is partially supported by the Hong Kong RGC GRF Project 16209519, the National Science Foundation of China (NSFC) under Grant No. 61729201, Science and Technology Planning Project of Guangdong Province, China, No. 2015B010110006, Hong Kong ITC ITF Grants ITS/044/18FX and ITS/470/18FX, Didi-HKUST Joint Research Lab Grant, Microsoft Research Asia Collaborative Research Grant and Wechat Research Grant.

References

[1] Y. Tong, L. Chen, and C. Shahabi, "Spatial crowdsourcing: Challenges, techniques, and applications," *PVLDB*, 2017.

[2] Y. Tong, Z. Zhou, Y. Zeng, L. Chen, and C. Shahabi, "Spatial crowdsourcing: a survey," *The VLDB Journal*, 2019.

[3] Q. Tao, Y. Zeng, Z. Zhou, Y. Tong, L. Chen, and K. Xu, "Multi-worker-aware task planning in real-time spatial crowdsourcing," in *DASFAA 2018*.

[4] Y. Tong, Y. Zeng, Z. Zhou, L. Chen, J. Ye, and K. Xu, "A unified approach to route planning for shared mobility," *PVLDB*, 2018.

[5] H. To, G. Ghinita, and C. Shahabi, "A framework for protecting worker location privacy in spatial crowdsourcing," *PVLDB*, 2014.

[6] L. Pournajaf, L. Xiong, V. S. Sunderam, and S. Goryczka, "Spatial task assignment for crowd sensing with cloaked locations," in *MDM 2014*.

[7] H. To, C. Shahabi, and L. Xiong, "Privacy-preserving online task assignment in spatial crowdsourcing with untrusted server," in *ICDE 2018*.

[8] M. E. Andrés, N. E. Bordenabe, K. Chatzikokolakis, and C. Palamidessi, "Geo-indistinguishability: differential privacy for location-based systems," in *CCS 2013*.

[9] Y. Tong, J. She, B. Ding, L. Wang, and L. Chen, "Online mobile micro-task allocation in spatial crowdsourcing," in *ICDE 2016*.

[10] Y. Tong, J. She, B. Ding, L. Chen, T. Wo, and K. Xu, "Online minimum matching in real-time spatial data: Experiments and analysis," *PVLDB*, 2016.

[11] Y. Tong, L. Wang, Z. Zhou, L. Chen, B. Du, and J. Ye, "Dynamic pricing in spatial crowdsourcing: A matching-based approach," in *SIGMOD 2018*.

[12] Y. Wang, Y. Tong, C. Long, P. Xu, K. Xu, and W. Lv, "Adaptive dynamic bipartite graph matching: A reinforcement learning approach," in *ICDE 2019*.

[13] Y. Zeng, Y. Tong, L. Chen, and Z. Zhou, "Latency-oriented task completion via spatial crowdsourcing," in *ICDE 2018*.

[14] A. Alfarrarjeh, T. Emrich, and C. Shahabi, "Scalable spatial crowdsourcing: A study of distributed algorithms," in *MDM 2015*.

[15] A. Meyerson, A. Nanavati, and L. J. Poplawski, "Randomized online algorithms for minimum metric bipartite matching," in *SODA 2006*.

[16] L. Kazemi and C. Shahabi, "Geocrowd: enabling query answering with spatial crowdsourcing," in *GIS 2012*.

[17] T. Song, Y. Tong, L. Wang, J. She, B. Yao, L. Chen, and K. Xu, "Trichromatic online matching in real-time spatial crowdsourcing," in *ICDE 2017*.

[18] Y. Tong, L. Wang, Z. Zhou, B. Ding, L. Chen, J. Ye, and K. Xu, "Flexible online task assignment in real-time spatial data," *PVLDB*, 2017.

[19] N. Bansal, N. Buchbinder, A. Gupta, and J. Naor, "A randomized $o(log^2 k)$-competitive algorithm for metric bipartite matching," *Algorithmica*, 2014.

[20] Y. Emek, S. Kutten, and R. Wattenhofer, "Online matching: haste makes waste!" in *STOC 2016*.

[21] J. Fakcharoenphol, S. Rao, and K. Talwar, "A tight bound on approximating arbitrary metrics by tree metrics," in *STOC 2003*.

[22] Didi Chuxing. [Online]. Available: http://www.didichuxing.com/

[23] GAIA initiative. [Online]. Available: http://gaia.didichuxing.com/

[24] H. To, G. Ghinita, L. Fan, and C. Shahabi, "Differentially private location protection for worker datasets in spatial crowdsourcing," *IEEE Transactions on Mobile Computing*, 2017.

[25] L. Wang, D. Yang, X. Han, T. Wang, D. Zhang, and X. Ma, "Location privacy-preserving task allocation for mobile crowdsensing with differential geo-obfuscation," in *WWW 2017*.

[26] J. Li, A. Liu, W. Wang, Z. Li, G. Liu, L. Zhao, and K. Zheng, "Towards privacy-preserving travel-time-first task assignment in spatial crowdsourcing," in *APWeb-WAIM 2018*.

[27] A. Liu, W. Wang, S. Shang, Q. Li, and X. Zhang, "Efficient task assignment in spatial crowdsourcing with worker and task privacy protection," *GeoInformatica*, 2018.

[28] H. To and C. Shahabi, "Location privacy in spatial crowdsourcing," in *Handbook of Mobile Data Privacy*, 2018.

[29] L. Pournajaf, D. A. Garcia-Ulloa, L. Xiong, and V. S. Sunderam, "Participant privacy in mobile crowd sensing task management: A survey of methods and challenges," *SIGMOD Record*, 2015.

[30] L. Pournajaf, L. Xiong, V. S. Sunderam, and X. Xu, "STAC: spatial task assignment for crowd sensing with cloaked participant locations," in *GIS 2015*.

[31] C. Dwork, "Differential privacy," in *ICALP 2006*.

[32] G. Cormode, C. M. Procopiuc, D. Srivastava, E. Shen, and T. Yu, "Differentially private spatial decompositions," in *ICDE 2012*.