2021 22nd IEEE International Conference on Mobile Data Management (MDM) | 978-1-6654-2845-3/20/$31.00 ©2021 IEEE | DOI: 10.1109/MDM52706.2021.00015

A Differentially Private Task Planning Framework for Spatial Crowdsourcing

Qian Tao †, Yongxin Tong †, Shuyuan Li †, Yuxiang Zeng ‡, Zimu Zhou #, Ke Xu †
†BDBC, SKLSDE Lab and IRI, Beihang University, China
‡The Hong Kong University of Science and Technology, Hong Kong SAR, China
#Singapore Management University, Singapore
†{qiantao, yxtong, lishuyuan, kexu}@buaa.edu.cn, ‡yzengal@cse.ust.hk, #zimuzhou@smu.edu.sg

***Abstract*—Spatial crowdsourcing has stimulated various new applications such as taxi calling and food delivery. A key enabler for these spatial crowdsourcing based applications is to plan routes for crowd workers to execute tasks given diverse requirements of workers and the spatial crowdsourcing platform. Despite extensive studies on task planning in spatial crowdsourcing, few have accounted for the location privacy of tasks, which may be misused by an untrustworthy platform. In this paper, we explore efficient task planning for workers while protecting the locations of tasks. Specifically, we define the *Privacy-Preserving Task Planning (PPTP)* problem, which aims at both total revenue maximization of the platform and differential privacy of task locations. We first apply the Laplacian mechanism to protect location privacy, and analyze its impact on the total revenue. Then we propose an effective and efficient task planning algorithm for the PPTP problem. Extensive experiments on both synthetic and real datasets validate the advantages of our algorithm in terms of total revenue and time cost.**

***Index Terms*—Spatial Crowdsourcing, Privacy Preserving, Task Planning**

I. Introduction

Spatial crowdsourcing, a prevailing category of crowdsourcing where tasks are spatiotemporal [1], [2], has triggered a wide range of novel applications such as taxi calling [3], food delivery [4], and so on. In spatial crowdsourcing applications, crowd workers must reach specific locations under certain time constraints to complete tasks. A common example is on-demand taxi calling services where taxi drivers (workers) should arrive at the pickup locations in time and reach the given destination to complete a taxi calling request (task).

A core functionality that enables such spatial crowdsourcing applications is *task planning*, which makes a route plan for each worker to perform a sequence of tasks with different optimization objectives *e.g.* maximizing the total revenue of the platform [5], minimizing the total traveling distance of workers [6] *etc.* and the planning can be either static [7] or dynamic [8].

Despite recent efforts on task planning for spatial crowdsourcing [7], [9], [10], [8], [5], they are still impractical for widespread real-world adoption, due to their ignorance on data privacy protection. In particular, *location privacy* is critical in spatial crowdsourcing applications since location information can reveal movement patterns or habits of individuals [11], [12]. Yet the spatial crowdsourcing platform may not always be trustworthy and there is a trend to enforce data privacy by new regulations *e.g.* General Data Protection Regulation[1].

In this paper, we study the *Privacy-Preserving Task Planning (PPTP)* problem in spatial crowdsourcing. It aims to not only maximize the total revenue as in previous studies [5], [6], but also protect the privacy of task locations. Although solutions to location privacy in spatial crowdsourcing exist [13], [14], [15], directly applying them to PPTP impairs the effectiveness (*i.e.* total revenue maximization) of task planning. This is because location privacy protection mechanisms typically add noises to the locations [16], and it is challenging to precisely calculate the utility of a route composed of obfuscated locations. To this end, we protect location privacy of tasks in the context of Geo-Indistinguishability [17], a differential privacy metric for locations. We analyze the impact of ensuring geo-indistinguishability on the total revenue of task planning, and devise an effective and efficient algorithm that approximately maximizes the total revenue under the location privacy constraint. The main contributions of this paper are as follows.

- We formulate the PPTP problem and propose an Efficient Private Task Planning (EPTP) solution framework, which consists of a privacy mechanism to ensure geo-indistinguishability and a task planning algorithm to maximize total revenue.
- We quantify the impact of a geo-indistinguishable privacy mechanism on the effectiveness (*i.e.* revenue maximization) of any task planning algorithm.
- On basis of the theoretical analysis, we design a novel task planning algorithm on obfuscated locations to approximately maximize the total revenue at low time cost.
- Extensive experiments on both synthetic and real datasets validate the effectiveness and efficiency of our methods.

In the rest of this paper, we define the PPTP problem in Sec. II, introduce our privacy-preserving task planning framework is in Sec. III, and present the evaluations in Sec. IV. Sec. V reviews related work and we conclude in Sec. VI.

II. Preliminaries

In this section, we formally define the *Privacy-Preserving Task Planning* (PPTP) problem and present the analysis model

[1]https://gdpr-info.eu/

978-1-6654-2845-3/21/$31.00 ©2021 IEEE
DOI 10.1109/MDM52706.2021.00015

TABLE I: Summary of major notations.

Notation	Description
t	A crowd task
loc_t	The location of the task t
rel_t	The release time of the task t
exp_t	The expiration time of the task t
rev_t	The revenue of t
ϵ_t	The privacy budget of t
R	Radius that a task can be accomplished
w	A crowd worker
$sLoc_w$	The initial location of w
$dLoc_w$	The destination of w
$cLoc_w$	The current location of w
rel_w	The release time of w
exp_w	The expiration time of w

to assess the effectiveness of task planning under the location privacy constraint. Major notations are listed in Table I.

A. Problem Definition

Definition 1 (Task)**.** A task t from a requester is a quintuple $(loc_t, rel_t, exp_t, rev_t, \epsilon_t)$, where loc_t, rel_t, and exp_t are the location, release time and expiration time of t. rev_t is the revenue contributed to the spatial crowdsourcing platform if t is accomplished, and ϵ_t represents the privacy budget of t.

We consider the *online* setting for task planning [7], [8], [5], *i.e.* the appearance of a task t is unknown before its release time rel_t. A task t is *completed* if a worker w arrives at a circle range centered at loc_t with a radius R before the task's expiration time exp_t. The radius R is predefined and unified in this paper, which depends on the applications. The privacy budget is specified by the requester.

Definition 2 (Worker)**.** A worker w is a quadruple $w = (sLoc_w, dLoc_w, rel_w, exp_w)$, where w can be observed after the release time rel_w with initial location $sLoc_w$, and must arrive at the destination $dLoc_w$ before expiration time exp_w.

In many spatial crowdsourcing applications, workers tend to move constantly. For convenience, we use $cLoc_w$ to represent the current location of w. A worker can complete a task t if he/she reaches t's location in the circle range centered at loc_t with radius R within t's valid time interval $[rel_t, exp_t]$, and will increase a revenue rev_t for the platform. Furthermore, we use $dis(.,.)$ to denote the distance between two locations, and assume workers move at a unit speed.

Definition 3 (Task Plan)**.** A task plan (plan for short) p_w for a worker w is a sequence of locations $\langle l_1, l_2, ..., l_{|p_w|}\rangle$ that w should visit *in order* starting from his/her current location $cLoc_w$. A plan p_w is *valid* if w can arrive at the destination $dLoc_w$ before his/her expiration time exp_w, *i.e.*

$$curT + dis(cLoc_w, l_1) + \sum_{i=1}^{|p_w|-1} dis(l_i, l_{i+1}) \leq exp_w,$$

where $curT$ is the current time.

Suppose P is the set of plans for all workers, *i.e.* $P = \{p_w | w \in W\}$. Further define the accomplished tasks of a plan p_w, denoted by $AT(p_w)$, as tasks completed by w if w follows the plan p_w. For simplicity, we use $AT(P)$ as the set of tasks accomplished by all plans in P, *i.e.* $AT(P) = \cup_{p\in P} AT(p)$. We assume plans are generated by a task planning algorithm and are guaranteed to be valid.

In this work, we are interested in making plans for workers while protecting location privacy of tasks. Particularly, the location privacy is protected by a *privacy mechanism* that achieves *geo-indistinguishability*.

Definition 4 (Privacy Mechanism)**.** Given an exact location space $\mathcal{L}$ and an obfuscated location space $\mathcal{L}'$, a privacy mechanism $\mathcal{M}$ is a probability function that maps each exact location x in $\mathcal{L}$ into an obfuscated location x' in $\mathcal{L}'$ with some probability. Specifically, the probability of mapping $x \in \mathcal{L}$ into $x' \in \mathcal{L}'$ is denoted as $Pr(x, x')$.

Definition 5 (Geo-Indistinguishability [17])**.** A privacy mechanism $\mathcal{M}$ is said to satisfy ϵ-Geo-Indistinguishability if for any $x_1, x_2 \in \mathcal{L}$ and $x' \in \mathcal{L}'$,

$$\frac{Pr(x_1, x')}{Pr(x_2, x')} \leq e^{\epsilon \cdot dis(x_1, x_2)}. \tag{1}$$

Geo-indistinguishability is extended from differential privacy [18] and is a widely used metric for *location* privacy [19], [13], [20].

Now we can formally define the *Privacy-Preserving Task Planning* (PPTP) problem.

Definition 6 (Privacy-Preserving Task Planning Problem)**.** Given a set of workers W and a set of tasks T which appear dynamically, the problem is to design a privacy mechanism $\mathcal{M}$ for tasks, and make plans P for workers such that:

- The mechanism $\mathcal{M}$ takes as input the location loc_t of a task t, outputs the obfuscated location loc'_t, and satisfies geo-indistinguishability.
- Plans are made on the obfuscated locations of tasks, and the total revenue of the accomplished tasks is maximized.

$$Rev(P) = \sum_{t \in AT(P)} rev_t \tag{2}$$

We illustrate the PPTP problem via the following example.

Example 1. Suppose currently there are 2 workers and 3 tasks. Their release time and expire time are shown in Table II and their corresponding locations are shown in Fig. 1. Suppose t_1 locates at $(1.5, 4)$ and is perturbed to $(2, 4)$ by a privacy mechanism (see the blue arrow in Fig. 1). For simplicity we omit the exact locations of other tasks. The platform can only observe the obfuscated locations of the tasks, *i.e.* loc'_{t_1}, loc'_{t_2} and loc'_{t_3}, after their release time. Further set the radius R as 1 and assume t_1 is assigned to w_1. Since $dis(t_1, t'_1) = 0.5 \leq 1$, w_1 can accomplish the task by reaching the obfuscated location $(2, 4)$.

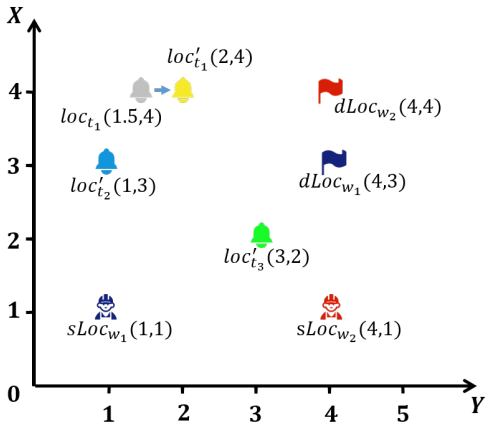


Fig. 1: Initial locations of tasks and workers.

TABLE II: Information of tasks and workers.

Task/Worker	Release time	Expire time	Revenue
t_1	1	5.5	2
t_2	2	4.5	3
w_1	2	7.8	-
w_2	3	7.5	-
t_3	4	8	2

B. Analysis Model

Before introducing our solution to the PPTP problem, we present the analysis model to assess the effectiveness of task planning in the privacy-preserving setting. Specifically, we define a new competitive ratio (CR) as follows.

Definition 7 (Competitive Ratio (CR))**.** Suppose ALG is the algorithm of interest which is based on the obfuscated locations and OPT the optimal algorithm based on the exact locations of tasks. The competitive ratio of ALG is the ratio between the expected revenue on the distribution of the privacy mechanism and the revenue obtained by OPT, *i.e.*

$$CR_p(ALG) = \min_{I \in \mathcal{I}} \frac{\mathbb{E}_{\mathcal{M}}[Rev(ALG(I))]}{Rev(OPT(I))} \tag{3}$$

where $\mathcal{I}$ is the space of all instances, and $ALG(I)$ and $OPT(I)$ are the plans generated by ALG and OPT.

We argue the above defined CR is more suited to study the effectiveness of privacy-preserving task planning algorithms than prior definitions [21], [5], [22]. This is because previous CR definitions [21], [5], [22] are all based on the *exact* locations. However, when algorithms are executed on *obfuscated* locations, as in privacy-preserving task planning, it may produce an extremely bad plan compared with that produced by the optimal algorithm with *the exact locations*. This makes the CR arbitrarily bad since we can always find a bad enough case in the probability function of a privacy mechanism. Hence the expected revenue on the distribution of the privacy mechanism is more reasonable.

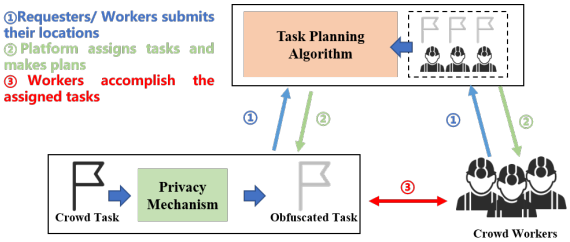


Fig. 2: Framework overview.

III. Efficient Private Task Planning Framework

This section presents our Efficient Private Task Planning (EPTP) framework. We start with its overview, followed by the detailed algorithms and analysis on privacy and effectiveness.

A. Framework Overview

Fig. 2 illustrates the overview of the EPTP framework. It works as follows.

- The platform collects information about workers and tasks. Task locations are obfuscated by the privacy mechanism before submitting to the platform.
- The platform executes the task planning algorithm based on the obfuscated locations to make plans for workers.
- Workers finish the assigned tasks according to the plans.

To solve the PPTP problem, the EPTP framework applies a privacy mechanism to guarantee the geo-indistinguishability of task locations. The framework then utilizes an efficient and effective task planning algorithm executed on obfuscated tasks locations to generate valid plans with high total revenue.

B. Privacy Mechanism and Analysis

We first present an ϵ_t-geo-indistinguishable privacy mechanism for task t with privacy budget ϵ_t and then analyze its impact on task planning.

Since the task locations are in 2D coordinates, hereinafter we will focus on the mechanisms that the exact space and the obfuscated space are both Euclidean spaces. Given a location loc_t with privacy budget ϵ_t, previous work [17] has shown that the planar Laplacian distribution satisfies geo-indistinguishability. The mechanism maps $loc_t \in \mathcal{X}$ to $loc'_t \in \mathcal{X}$ with probability

$$Pr(loc_t, loc'_t) = \frac{\epsilon_t^2}{2\pi} \cdot e^{-\epsilon_t \cdot dis(loc_t, loc'_t)}. \tag{4}$$

We also utilize the planar Laplacian distribution to guarantee the geo-indistinguishability as in [23], [13], [24]. However, we are more interested in how the mechanism affects the revenue of task plans, as explained below.

The planar Laplacian distribution determines the probability to map the exact task location into the obfuscated one. Note that the platform only sees the obfuscated locations. To analyze the effect of the privacy mechanism on task planning, we examine the posterior probability of the exact task location given its obfuscated location, which is given by Lemma 1:

Lemma 1. Given an obfuscated location loc'_t of a task, the probability distribution of t's exact location loc_t also follows the Laplacian distribution, *i.e.*

$$Pr[loc_t|loc'_t] = Pr(loc_t, loc'_t) = \frac{\epsilon_t^2}{2\pi} \cdot e^{-\epsilon_t \cdot dis(loc_t, loc'_t)}. \quad (5)$$

Proof. Suppose the exact and obfuscated locations are derived from the Euclidean space $\mathcal{X}$. By the Bayesian formula,

$$Pr[loc_t|loc'_t] = \frac{Pr[loc_t] \cdot Pr[loc'_t|loc_t]}{\int_{x\in\mathcal{X}} Pr[x]Pr[loc'_t|x]}. \quad (6)$$

Assume the exact location appears uniformly in $\mathcal{X}$, *i.e.* $Pr[x] = Pr[y]$ for any $x, y \in \mathcal{X}$. Hence we can remove the probability $Pr[loc_t]$ and $Pr[x]$ in Eq. 6 by reduction, *i.e.*

$$Pr[loc_t|loc'_t] = \frac{Pr[loc'_t|loc_t]}{\int_{x\in\mathcal{X}} Pr[loc'_t|x]}. \quad (7)$$

Note that $Pr[loc'_t|loc_t]$ is exactly the probability that the privacy mechanism maps loc_t into loc'_t, *i.e.* $Pr[loc'_t|loc_t] = Pr(loc_t, loc'_t)$. Similarly, $\int_{x\in\mathcal{X}} Pr[loc'_t|x] = \int_{x\in\mathcal{X}} Pr(x, loc'_t) = \int_{x\in\mathcal{X}} Pr(loc'_t, x) = 1$. Substituting the above deduction into Eq. 7, $Pr[loc_t|loc'_t] = Pr(loc_t, loc'_t)$. □

After obtaining the posterior probability, we analyze the probability to complete a task. This is because workers should arrive within the range centered at the *exact* task location loc_t with radius R to accomplish the task t, and if an *obfuscated* location is given, the task may not be accomplished. The probability that a task can be accomplished is given by Lemma 2:

Lemma 2. Suppose a worker w is planned to arrive at an obfuscated location loc'_t of the task t and t will be accomplished if loc'_t is in the range of the exact location loc_t with radius R. Then t will be accomplished with a probability $1 - (1 + \epsilon_t R e^{-\epsilon_t R})$.

Proof. w will reach the obfuscated location loc'_t, and t can be accomplished if the distance between loc_t and loc'_t is no greater than R. Hence the probability of the task t being accomplished equals to the probability that the exact location loc_t locates within the range centered at the obfuscated location loc'_t with radius R. Previous work [17] has shown that the probability is $1 - (1 + \epsilon_t R e^{-\epsilon_t R})$. We restate the conclusion with a detailed deduction.

Use an equal form of the planar Laplacian distribution, *i.e.* the polar Laplacian distribution [17], to prove the theorem. Given $d \in [0, \infty]$, $\theta \in [0, 2\pi)$, the distance between loc_t, loc'_t, the angle between the line formed by loc_t, loc'_t and the horizontal axis, the polar Laplacian distribution is defined as

$$Lap(d, \theta) = \frac{\epsilon_t^2}{2\pi} r \cdot e^{-\epsilon_t r}. \quad (8)$$

From Lemma 1, the probability can be calculated by

$$\begin{aligned} &Pr[t \text{ is accomplished}] = Pr[dis(loc_t, loc'_t) \le R|loc'_t] \\ &= \int_0^{2\pi}\int_0^R \frac{\epsilon_t^2}{2\pi} r \cdot e^{-\epsilon_t r} \mathrm{d}r\mathrm{d}\theta \\ &= \int_0^R \epsilon_t^2 r \cdot e^{-\epsilon_t r}\mathrm{d}r = -\epsilon_t \int_0^R r\mathrm{d}e^{-\epsilon_t r} \\ &= 1 - (1 + \epsilon_t R)e^{-\epsilon_t R}. \end{aligned} \quad (9)$$

□

Lemma 2 shows how the privacy mechanism influences task completion. The following theorem further states the effect of the privacy mechanism on the PPTP problem.

Theorem 1. Suppose the locations of the tasks are perturbed independently. Given a task planning algorithm ALG executed on the exact locations with competitive ratio δ, then ALG executed on the obfuscated locations by the Laplacian mechanism $\mathcal{M}$ has a competitive ratio $(1 - (1 + \epsilon_{min} R)e^{-\epsilon_{min} R})\delta$, where ϵ_{min} is the minimum privacy budget among all tasks, *i.e.* $\epsilon_{min} = \min_{t\in T} \epsilon_t$.

Proof. For each instance $I \in \mathcal{I}$,

$$\begin{aligned} \frac{\mathbb{E}_{\mathcal{M}}[Rev(ALG(I))]}{Rev(OPT(I))} &= \frac{\mathbb{E}_{\mathcal{M}}[Rev(ALG(I))]}{Rev(ALG(I))} \frac{Rev(ALG(I))}{Rev(OPT(I))} \\ &\ge \delta \frac{\mathbb{E}_{\mathcal{M}}[Rev(ALG(I))]}{Rev(ALG(I))}. \end{aligned} \quad (10)$$

Since the locations are obfuscated independently, the result if a task t contributes to the total revenue is only determined by the random process of the mechanism. Use α to represent $1 - (1 + \epsilon_{min} R)e^{-\epsilon_{min} R}$, then

$$\begin{aligned} &\mathbb{E}_{\mathcal{M}}[Rev(ALG(I))] = \sum_{t\in ALG(I)} \mathbb{E}_{\mathcal{M}}[Rev_t] \\ &= \sum_{t\in ALG(I)} (1 - (1 + \epsilon_t R)e^{-\epsilon_t R}) Rev_t \ge \sum_{t\in ALG(I)} \alpha Rev_t \\ &= \alpha \sum_{t\in ALG(I)} Rev_t = \alpha Rev(ALG(I)) \end{aligned} \quad (11)$$

where the second line in Eq. 11 is because the function $f(x) = \frac{1+x}{e^x}$ monotonically decreases as $x \ge 0$. □

Theorem 1 shows how the competitive ratio of a *general* task planning algorithm is affected under the privacy mechanism. Next, we introduce an efficient task planning algorithm.

C. Task Planning Algorithm

After tasks with obfuscated locations are submitted to the platform, a task planning algorithm is invoked. Similar to [5], we use a *dynamic insertion approach* to make plans for workers. Specifically, we account for the long-term effect when planning for a single worker and apply dynamic programming to improve the efficiency of the algorithm.

Alg. 1 illustrates the dynamic insertion approach. Two sets, W_{act} and T_{avai}, are first initialized as the set of the workers

Algorithm 1: Dynamic Insertion Architecture

input : A set of dynamically arriving workers W, a set of dynamically arriving tasks T
output: Plans p_w for $w \in W$

1 Initialize W_{act} and T_{avai} as $\emptyset$;
2 **while** *a worker* $w \in W$ *or a task* $t \in T$ *arriving* **do**
3 **if** *a worker* w *arrives* **then**
4 $p_w \leftarrow \texttt{TaskPlanSingleWorker}(T_{avai}, w)$;
5 $W_{act} \leftarrow W_{act} \cup \{w\}$;
6 **else** a task t arrives
7 $w_{best}, p_{best} \leftarrow \texttt{InsertTask}(W_{act}, t)$;
8 Update w_{best}'s plan with p_{best};

who have arrived in the platform and the set of available tasks respectively (line 1). Whenever a worker w appears, the algorithm invokes the $TaskPlanSingleWorker$ function to makes plans for w from the available task set T_{avai} (lines 3-5). Whenever a task t appears, function $InsertTask$ is executed to insert t into the existing plan of a worker that has arrived in the platform (*i.e.* $w_{best} \in W_{act}$) with minimum extra distance, while keeping the order of other tasks unchanged (lines 6-8). The details of the two functions are as below.

1) Task Planning for a Single Worker: When making plans for a newly appeared worker, we choose tasks greedily. Previous work [5] utilizes the strategy of the average revenue based on the empty moving distance to determine the priority of the available tasks. However, this strategy ignores the long-term effect caused by the corresponding insertion. Instead, we consider both the current average revenue and the long-term effect by the following two definitions.

Definition 8 (Revenue per Empty Moving Distance (REMD) [5])**.** Given an available task t and a newly appeared worker w, the REMD between w and t is defined as

$$RE(w,t) = \frac{rev_t}{dis(cLoc_w, loc'_t)}. \tag{12}$$

A task with larger REMD has a higher priority when making plan for a newly appeared worker.

As for the long-term effect, we have the following:

- If the task t has a smaller spare time (*i.e.* $exp_t - curT$), it should be assigned a higher priority because it will expire with a higher probability.
- If the task t locates nearer to w's destination $dLoc_w$, it should be assigned a lower priority since the task can be accomplished more efficiently later (*i.e.* accomplished when the worker heads to the destination *along* the way).

To measure such long-term effect, we define the spare time per heading destination distance:

Definition 9 (Spare Time per Heading Destination Distance (STHDD))**.** Given a worker w and the current time $curT$, the STHDD between w and t is defined as

$$ST(w,t) = \frac{exp_t - curT}{dis(loc'_t, dLoc_w)}. \tag{13}$$

Algorithm 2: TaskPlanSingleWorker

input : Worker w, the set of available tasks T_{avai}
output: Plan p_w of the worker w

1 $p_w \leftarrow \emptyset$;
2 Sort t in T_{avai} according to $\frac{Rev(w,t)}{ST(w,t)}$ in descending order;
3 **foreach** *task* $t \in T_{avai}$ **do**
4 **if** *appending* t *to* p_w *does not violate the expiration time of* t *and* w **then**
5 Append loc'_t to p_w;
6 **if** $p_w = \emptyset$ **then**
7 $p_w \leftarrow \langle dLoc_w \rangle$;
8 **return** p_w;

Unlike REMD, a task with larger STHDD means a *lower priority* because it can be completed more efficiently later.

Alg. 2 shows the task planning algorithm for a single worker by considering the two aspects above. The algorithm first sorts the available tasks in T_{avai} by the ratio of the Revenue per Empty Moving Distance to the Spare Time per Heading Destination Distance, *i.e.* $\frac{RE(w,t)}{ST(w,t)}$ (line 1). Tasks are then appended to the plan p_w in such an order as long as the task does not violate the expiration time of t and w. If no task is assigned to w, w moves to his/her destination, *i.e.* $p_w = \langle dLoc_w \rangle$ (lines 6-7). We further illustrate $TaskPlanSingleWorker$ via the following example.

TABLE III: REMD and STHDD of t_1 and t_2 for w_1.

Task	REMD	STHDD	REMD/STHDD
t_1	0.63	1.57	0.40
t_2	1.5	0.83	1.81

Example 2. At time 2, the worker w_1 appears and there are two available tasks t_1, t_2, *i.e.* $T_{avai} = \{t_1, t_2\}$. Then the values of REMD, STHDD, and their ratios are shown in Table III. For instance, the REMD of t_1 can be obtained by $\frac{rev_{t_1}}{dis(cLoc_{w_1}, loc'_{t_1})} = 0.63$. Based on the ratio between REMD and STHDD, we append t_2 and t_1 one by one, determine if the tasks violate the expiration time, and make plan $p_{w_1} = \{loc'_{t_2}, loc'_{t_1}, dLoc_{w_1}\}$.

2) Dynamic Programming based Insertion: Previous work [5] applies an $O(nm^3)$ algorithm to make plans. In this subsection, we propose a Dynamic Programming based technique to quickly find the best insertion location.

We first introduce some notations. Given a worker w with his/her unaccomplished plan $p_w = \langle l_1, l_2, ..., l_{m_w} \rangle$, we denote $D_w[i]$ as the traveling time along p_w from w's current location $cLoc_w$ to the i-th location l_i. The last location in p_w is also the destination of w, *i.e.* $l_{m_w} = dLoc_w$. For $i = 1, ..., m_w$, $E_w[i]$ represents the extra traveling time caused by inserting

Algorithm 3: InsertTask

input : The active worker set W_{act} and a task t
output: The chosen worker w_{best} and updated plan $p_{w_{best}}$

1 $w_{best} \leftarrow None, bestComPos \leftarrow -1, minCost \leftarrow \infty$;
2 **foreach** $w_a \in W_{act}$ **do**
3 Compute $MD_{w_a}[.]$ and $E_{w_a}[.]$ by Eq. 14 and Eq. 15;
4 **foreach** *insertion position* k *in* p_w **do**
5 **if** $E_{w_a}[k] \leq \min\{MD_{w_a}[k], minCost\}$ **then**
6 $minCost \leftarrow E_{w_a}[k]$;
7 $bestComPos \leftarrow k$;
8 $w_{best} \leftarrow w_a$;
9 Insert t into the k-th location of w_{best}'s plan $p_{w_{best}}$;
return w_{best},$p_{w_{best}}$;

a task t^* before the i-th location l_i, *i.e.*

$$E_w[i] = \begin{cases} dis(cLoc_w, loc'_{t^*}) + dis(loc'_{t^*}, l_i) - dis(cLoc_w, l_i), i = 1 \\ dis(l_{i-1}, loc'_{t^*}) + dis(loc'_{t^*}, l_i) - dis(l_{i-1}, l_i), 2 \leq i \leq m_w. \end{cases} \quad (14)$$

Basic Idea. If we insert the task t^* before the location l_k in the plan p_w, then *(i)* for $1 \leq i < k$, the traveling time from $cLoc_w$ to l_i, *i.e.* $D_w[i]$, remains unchanged; *(ii)* for $k \leq i \leq m_w$, the distance from $cLoc_w$ to l_i will increase by $E_w[k]$. The observation indicates that while finding the best insertion position of a task t^* on a plan p_w, only $E_w[k]$ relates to the task t^*, while other variables can be pre-calculated.

Specifically, we use an array of variables to represent the maximum tolerant extra traveling time if we insert a task t^* at each position in the plan p_w, and use these variables to determine the validity of an insertion position. Let $MD_w[k]$ be the maximum tolerant extra traveling time if a task is inserted before l_k and it guarantees that all tasks in p_w do not expire. Then the value $MD_w[k]$ is either the maximum tolerant extra traveling time for the task at l_k (denoted by e_k), formally $e_k - D_w[k]$, or *the maximum tolerant extra traveling time for tasks in* $\{l_{k+1}, l_{k+2}, ..., l_{m_w}\}$. Hence, we can calculate $MD_w[k]$ by

$$MD_w[k] = \begin{cases} exp_w - D_w[m_w], & k = m_w \\ min(e_k - D_w[k], MD_w[k+1]), & 1 \leq k < m_w. \end{cases} \quad (15)$$

After calculating $MD_w[k]$ for $1 \leq k \leq m_w$, we can confirm if an insertion before t_k of a task t^* is valid (*i.e.* does not cause expiration) by judging if $E_w[k] \leq MD_w[k]$. To this end, we can find the best insertion location by iterating k from m_w to 1, calculating $MD_w[k]$ and $E_w[k]$, and comparing if $E_w[k] \leq MD_w[k]$.

Algorithm Details. Alg. 3 shows the $InsertTask$ function based on dynamic programming. To find the best insertion position, for each worker $w_a \in W_{act}$ we first compute the arrays $E_{w_a}[.]$ and $MD_{w_a}[.]$ in line 3 based on Eq. 14 and Eq. 15 respectively. Then for each insertion position i, we compare the values of $E_{w_a}[i]$ and $MD_{w_a}[i]$ to see if the plan after insertion is valid and has a minimum extra distance (line 5), and update the current best insertion location (lines 6-8). The following example illustrates the dynamic programming based solution.

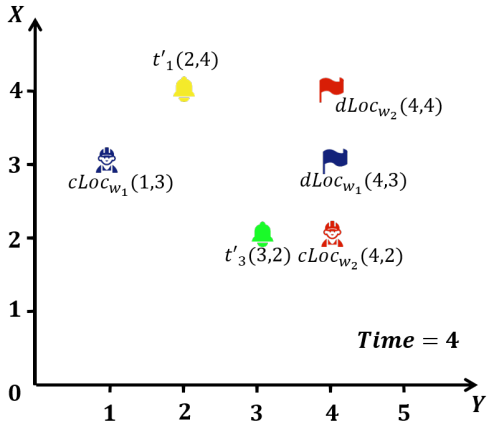


Fig. 3: Locations of workers and tasks at time 4.

TABLE IV: $E_{w_1}[.]$ and $MD_{w_1}[.]$ in p_{w_1}.

Index	1	2
Insertion Location	Before loc_{t_2}	Before $dLoc_{w_1}$
$D_{w_1}[i]$	5.41	7.65
$E_{w_1}[i]$	3.06	1.41
$MD_{w_1}[i]$	0.09	0.15

Example 3. At time $t = 4$, t_3 arrives in the platform. The locations of the tasks and workers are shown in Fig. 3. At this time, w_1 arrives at $(1, 3)$ and has accomplished t_2. Her current plan is $p_{w_1} = \{loc'_{t_1}, dLoc_{w_1}\}$. w_2 has a plan of heading to his/her destination $(4, 4)$, *i.e.* $p_{w_2} = \{dLoc_{w_2}\}$, and locates at $(4, 2)$. We could obtain the values of $D_{w_1}[.]$, $E_{w_1}[.]$ and $MD_{w_1}[.]$ by Eq. 14 and Eq. 15, as shown in Table IV. For example, $D_{w_1}[1] = 4$(current time)$+dis(cLoc_{w_1}, loc'_{t_1}) = 5.41$. As for $MD_{w_1}[.]$, we first calculate $MD_{w_1}[2] = exp_{w_1} - D_{w_1}[2] = 0.15$. Then we use $MD_{w_1}[2]$ to obtain $MD_{w_1}[1] = \min\{exp_{t_1} - D_{w_1}[1], MD_{w_1}[2]\} = 0.09$. After obtaining $MD_{w_1}[.]$, we could judge by $E_{w_1}[i] > MD_{w_1}[i]$ that t_3 cannot be inserted into w_1's plan. Hence it is assigned to w_2, and w_2's plan now is $p_{w_2} = \langle loc'_{t_3}, dLoc_{w_2} \rangle$.

3) Complexity Analysis: We analyze the time complexity of the proposed technique. Suppose currently there are m_{w_a} tasks for a given worker $w_a \in W$. Line 2 in Alg. 3 takes a time complexity of $O(m_{w_a} + 1)$, as $E_{w_a}[i]$ and $MD_{w_a}[i]$ can be obtained according to Eq. 14 and Eq. 15 by $O(1)$ time. It takes $O(m_{w_a} + 1)$ time to find the best insertion position (lines 4-8), because for each insertion position i, we just compare the values $E_{w_a}[k]$ and $MD_{w_a}[k]$. Hence the time complexity of lines 3-8 is $O(m_{w_a} + 1)$. Considering the enumeration of

TABLE V: Experimental settings.

Parameters	Settings
$\|W\|$	$100, 200, \mathbf{300}, 400, 500$
$\|T\|$	$1000, 2000, \mathbf{3000}, 4000, 5000$
ts_t	$\sigma = \mathbf{10}, \mu = 60, 120, \mathbf{180}, 240, 300$
ts_w	$\sigma = \mathbf{10}, \mu = 30, 60, \mathbf{90}, 120, 150$
Rev_{max}	$2, 4, \mathbf{6}, 8, 10$
R	$1, 1.5, \mathbf{2}, 2.5, 3$

the workers in W, the total time complexity of Alg. 3 is

$$\sum_{w_a \in W} O(m_{w_a} + 1) = \quad (16)$$
$$O(|W| + \sum_{w_a \in W} m_{w_a}) = O(n + m)$$

where the last deduction is derived from the truth that $|W| = n$ and $\sum_{w_a \in W} m_{w_a} \leq m$. Noticing that Alg. 3 is executed each time a task appears, its total time complexity in the framework is $O(m(n+m))$. Alg. 2 still takes $O(nm \log m)$ time to make plan for a single worker. Hence the total time complexity of our framework is $O(mn \log m + m^2)$.

IV. Experimental Study

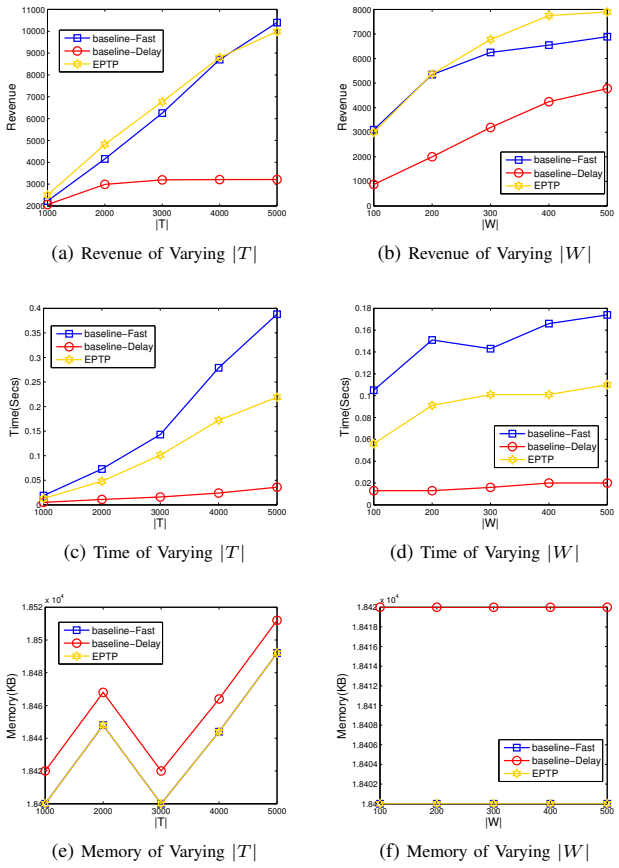
A. Experimental Setup

Datasets. We evaluate the performance of different algorithms on both synthetic and real datasets. Synthetic datasets are generated in a similar setting as [5], and the parameters are shown in Table V. Default parameters are marked in bold. Tasks and workers are generated randomly on a 600×600 square, and the radius of the tasks being accomplished is set to $R = 1$. Following [25], [5], the spare time (*i.e.* the expiration time minuses the release time) of the tasks and workers is randomly generated following the Gaussian distribution. In Table V, we denote the parameters of the spare time of tasks and workers as ts_t and ts_w, respectively. The release time of tasks and workers is randomly generated following the Poisson distribution. Since there is more tasks than workers, the release time is generated with $\lambda_t = 2/min$ and $\lambda_w = 20/min$ for tasks and workers, respectively. Finally, the privacy budgets of the tasks are uniformly generated from $[1, 2]$, and we generate the revenue of the tasks by uniformly sampling from the interval $[1, Rev_{max}]$, as shown in Table V.

As for real datasets, we use the same dataset as [5]. The dataset includes the taxi orders from a taxi-calling service platform. Each order consists of a pickup location and drop-off location, and is randomly chosen as the location information of a task or a worker. We use the pickup location as the location of a task, and use the pickup and drop-off locations as the start location and destination of a worker respectively. Other parameters are generated with the same parameters as the synthetic datasets.

Compared Algorithms. To the best of our knowledge, there is no study on task planning in spatial crowdsourcing under differential privacy. Hence we compare our framework with two state-of-the-art algorithms on task planning with the same privacy mechanism (*i.e.* the Laplacian mechanism).

- baseline-Delay (Delay Planning algorithm in [5]): it makes a plan for each newly arrived worker, and keeps the plan unchangeable until the worker finishes the plan.
- baseline-Fast (Fast Planning algorithm in [5]) it keeps updating plans with the arrival of the tasks.
- EPTP: our proposed framework.



(a) Revenue of Varying $|T|$ (b) Revenue of Varying $|W|$

(c) Time of Varying $|T|$ (d) Time of Varying $|W|$

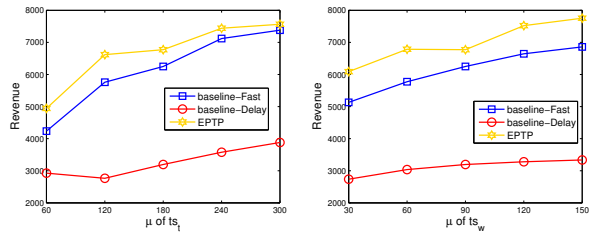
(e) Memory of Varying $|T|$ (f) Memory of Varying $|W|$

Fig. 4: Results on Varying $|T|$ and $|W|$

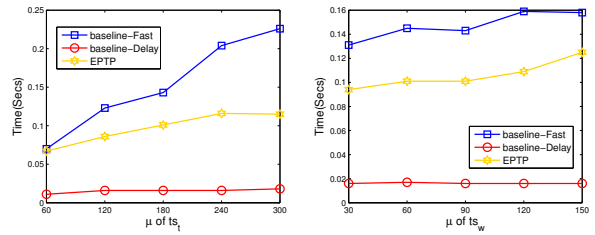
Implementation. All algorithms are implemented in C++, and the experiments were performed on a machine with Intel(R) Xeon(R) Gold 5118 2.30Hz CPUs and 256GB memory. Each case is executed 10 times and average results are reported.

B. Experiment Results

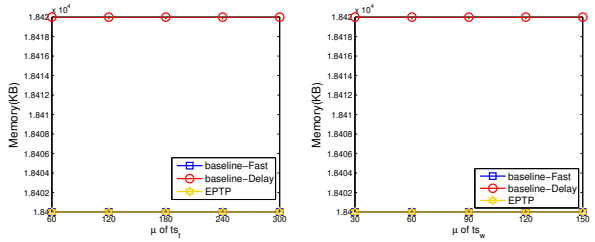
Effect of $|T|$. The first column of Fig. 4 shows the results of varying the number of tasks. The total revenues of baseline-Fast and EPTP are always better than that of baseline-Delay, while EPTP performs the best in most cases. This is reasonable as we consider both the current revenue and the probability of the tasks being accomplished in long-term effect. Another observation is that, with the increase of the number of tasks $|T|$, the revenues of baseline-Fast and EPTP keep increasing, while that of baseline-Delay stays stable. This



(a) Revenue of Varying ts_t (b) Revenue of Varying ts_w

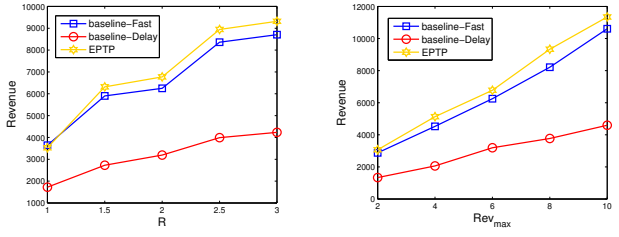


(c) Time of Varying ts_t (d) Time of Varying ts_w

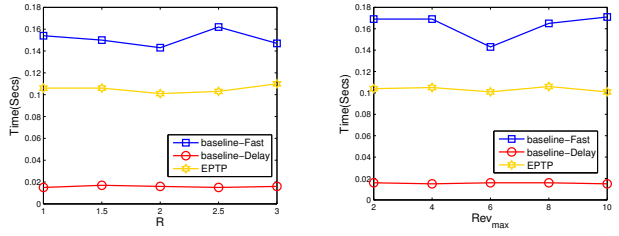


(e) Memory of Varying ts_t (f) Memory of Varying ts_w

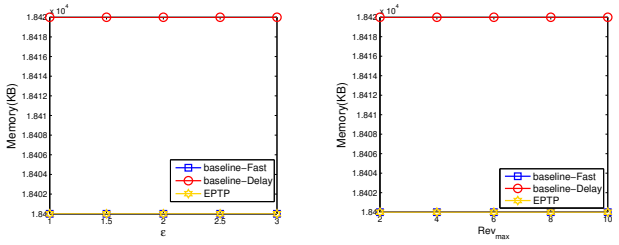
Fig. 5: Results on Varying ts_t and ts_w



(a) Revenue of Varying R (b) Revenue of Varying Rev_{max}



(c) Time of Varying R (d) Time of Varying Rev_{max}



(e) Memory of Varying R (f) Memory of Varying Rev_{max}

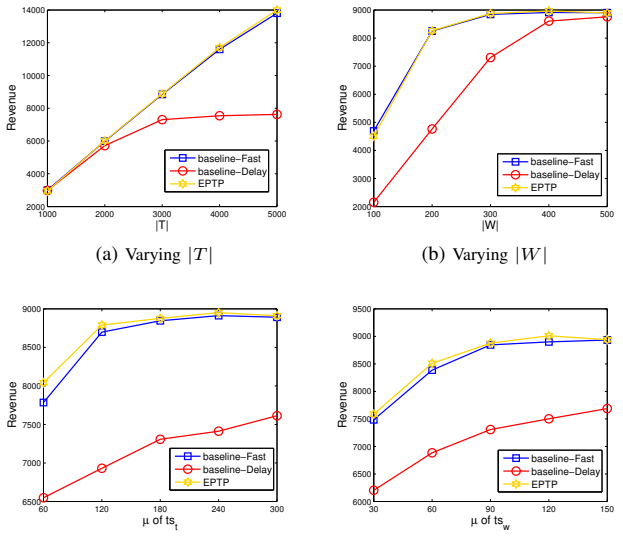
Fig. 6: Results on Varying R and Rev_{max}

has also been observed in [5]. baseline-Delay is the fastest, and EPTP consumes an allowed time cost (less than 0.4s). Specifically, EPTP performs better than baseline-Fast and is up to 34% faster, which validates the power of the dynamic programming technique. Finally, all of the three algorithms consume a similar memory, and the gap is negligible (20KB to 18MB).

Effect of $|W|$. The second column of Fig. 4 reports the results of varying the number of workers $|W|$. With the increase of $|W|$, the revenues of all the three algorithms increase, which is reasonable as more workers can accomplish more tasks. Our EPTP performs the best except when $|W| = 100$, with 39%-70% improvement in revenue compared with the baselines. As for the time cost, EPTP is faster than baseline-Fast, while baseline-Delay is the fastest. With the increase of the number of workers, the time cost of baseline-Fast and EPTP increases, while that of baseline-Delay stays stable. Finally, EPTP consumes a similar size of memory compared with baseline-Fast, and both of them are more efficient in terms of the memory cost.

Effect of ts_t. The first column of Fig. 5 shows the results of varying the mean of the tasks' spare time. EPTP still performs the best. With the increase of ts_t, EPTP, baseline-Fast and baseline-Delay gain more revenues, while the improvement of baseline-Delay is less. This is because the long spare time of the tasks means that the tasks have more opportunity to be accomplished, but baseline-Delay only benefits when new workers appear. As for the time cost, both baseline-Fast and EPTP increase while baseline-Delay stays stable. This is reasonable as a longer spare time leads to a larger search space in baseline-Fast and EPTP, and this has little effect on baseline-Delay. The memory cost of the three algorithms has similar trends to the previous experiments.

Effect of ts_w. The second column of Fig. 5 illustrates the results of varying the mean of the workers' spare time ts_w. Our algorithm EPTP again performs the best in terms of the revenue, and all three algorithms increase as ts_w increases. The extent of the increase is less than the experiments on varying ts_t. This might be because workers may not be able to accomplish more tasks with an increased spare time due to factors like the locations of the tasks, the detour to accomplish the tasks, *etc.* In terms of the time cost, it could be observed that in most cases the time cost of all algorithms stays stable (except for baseline-Fast when $\mu = 30$ and EPTP when $\mu = 120, 150$). This is because no matter how long spare time a worker has, the algorithms should always iterate all tasks,



(a) Varying $|T|$ (b) Varying $|W|$

(c) Varying ts_t (d) Varying ts_w

Fig. 7: Revenue Results on Real Datasets

which leads to a stable time cost. In terms of the memory cost, all of the three algorithms consume an allowable memory (less than 18MB) and their curves keep stable.

Effect of the radius R. The first column of Fig. 6 shows the results of varying radius R of workers. EPTP still performs the best in most cases. With the increase of R, the revenues of all three algorithms increase. This is because the larger the radius, the more possible the tasks are accomplished, as Theorem 1 shows. As for time cost, EPTP still outperforms baseline-Fast, and baseline-Delay performs the best. All of the three algorithms stay stable as R varies since the time complexities of these algorithms are irrelevant to the radius R. In terms of memory cost, we observe a similar trend to previous experiments.

Effect of Rev_{max}. The second column of Fig. 6 reports the results of varying Rev_{max}. EPTP still outperforms baseline-Delay and baseline-Fast. Unsurprisingly, the revenues of all the algorithms increase as Rev_{max} increase. The revenues of baseline-Fast and EPTP increase greater than baseline-Delay when Rev_{max} increases. In terms of time and memory cost, baseline-Fast, baseline-Delay and EPTP perform a stable and allowable cost.

Performance on Real Datasets. Fig. 7 shows partial experimental results on real datasets. We observe a similar trend of the algorithms on real datasets in terms of the revenue, time cost and memory cost. Hence only the revenue results of varying $|T|$, $|W|$, ts_t and ts_w are reported. EPTP outperforms the baselines in most cases and has a reasonable time and memory cost. This accords with our conclusions in the experiments on synthetic datasets.

Summary of Results. In terms of the revenue, EPTP benefits from the long-term effect strategy, and outperforms the baselines by up to 70% in most cases. In terms of the time cost, our EPTP, though not the most efficient, still has an allowable time cost (less than 0.4s with 5000 workers). All of the algorithms have a stable memory cost (roughly 18MB), which validate the memory efficiency of EPTP. Our EPTP obtains the best revenue with an allowable time cost and stable memory cost.

V. Related Work

Our work is related to the domains of **Task Planning** and **Location Privacy Protection** in spatial crowdsourcing.

A. Task Planning in Spatial Crowdsourcing

Task assignment and task planning have been widely considered as the core challenge of spatial crowdsourcing. Despite of many works [26], [27], [28], [29], [22], [30] that focus on task assignment, task planning has attracted attention in recent years because of its practicalness and hardness. The task planning problem in spatial crowdsourcing can trace back to the orienteering problem [31] and the orienteering problem has generated many variants since then. The main difference between the orienteering problem (as well as its variants) and our work is that our problem is defined in a dynamic scenario, and the information of the workers and tasks can only be observed after they appear.

More recently, the task planning problem in spatial crowdsourcing also gets attention in database community. Deng *et al.* [7] designs approximate algorithms to make task planning for a single worker with previous known tasks such that the number of accomplished tasks is maximized. Deng *et al.* [9], [10] further extends the problem to make planning for multiple workers. On the contrary, [32] focuses on the exact solution to the task planning problem. A worker dependency partition is proposed to prune and reduce the searching space.

Another research line in this part is the task planning problem in dynamic scenario, which comforts more to the practice. In dynamic scenario, the information of tasks or workers can be observed after its appearing. This makes the problem harder be bo solved. Li *et al.* [8] first studies the task planning problem in dynamic scenario for single worker. Two kinds of algorithms are proposed to heuristically make plans for workers. Tao *et al.* [5] further extends the problem with multiple workers, and assumes that both workers and tasks dynamically arrive. Two heuristic but efficient and effective algorithms are proposed to solve the problem. More recent works also pay attention to the task planning problems in specific spatial crowdsourcing applications like last-mile delivery [33] and ridesharing services [34]. To accord with the applications, each task is associated with a pickup location and a delivery location in these works, which is quite different from our work. Comparing with these works, we also consider the location privacy of the tasks, which is more different and harder.

B. Location Privacy Protection in Spatial Crowdsourcing

Differential Privacy [18] has been widely developed in recent years. It has also been utilized to protect the privacy of

the locations in spatial crowdsourcing or other applications. For example, To *et al.* [14] studies how to protect the number of workers in differential regions, and makes task assignment such that the acceptance rate is maximized. However, this work still treats the locations as aggregation values and utilizes the general privacy mechanisms to protect the values.

On the other hand, Andrés *et al.* [17] propose Geo-Indistinguishability to protect the differential privacy of the locations, which has been utilized in spatial crowdsourcing [19][13][15]. Wang *et al.* [19] study how to protect the Geo-Indistinguishability of the workers' locations and give a Mixed-Integer-Programming based algorithm to minimize the expected traveling distance. They still assume the platform knows the information of the workers and tasks in advance. More recently, [13] and [15] further propose algorithms to make task assignment for dynamically arriving workers and tasks with objectives of maximizing match size and minimizing total distance, respectively. However, to the best of our knowledge, no previous work has pay attention to the task planning algorithm with location privacy protected. Our focus is on the task planning algorithms with the location privacy, which has essential difference from the above works.

VI. Conclusion

In this paper, we study privacy-preserving task planning in spatial crowdsourcing which aims to ensure differential privacy of task locations while the total revenue of the platform is maximized. To protect location privacy of tasks, we apply the Laplacian mechanism to guarantee geo-indistinguishability, and analyze how the obfuscated locations affect task planning. To make plans over obfuscated locations with large revenue, we consider both the current revenue and long-term effect. We further design a dynamic programming based technique to accelerate task planning. Evaluations both real and synthetic datasets valid the effectiveness and efficiency of our methods.

Acknowledgment

We are grateful to anonymous reviewers for their constructive comments. Qian Tao, Yongxin Tong, Shuyuan Li and Ke Xu's works are partially supported by the National Key R&D Program of China under Grant 2018AAA0101100, National Science Foundation of China (NSFC) under Grant No. 61822201, 62076017 and U1811463.

References

[1] Yongxin Tong, Lei Chen, and Cyrus Shahabi. Spatial crowdsourcing: Challenges, techniques, and applications. *PVLDB*, 2017.
[2] Yongxin Tong, Zimu Zhou, Yuxiang Zeng, Lei Chen, and Cyrus Shahabi. Spatial crowdsourcing: a survey. *The VLDB Journal*, 2020.
[3] Didi Chuxing, http://www.didichuxing.com/.
[4] Meituan, http://www.meituan.com/.
[5] Qian Tao, Yuxiang Zeng, Zimu Zhou, Yongxin Tong, Lei Chen, and Ke Xu. Multi-worker-aware task planning in real-time spatial crowdsourcing. In *DASFAA*, 2018.
[6] Yongxin Tong, Yuxiang Zeng, Zimu Zhou, Lei Chen, Jieping Ye, and Ke Xu. A unified approach to route planning for shared mobility. *PVLDB*, 2018.
[7] Dingxiong Deng, Cyrus Shahabi, and Ugur Demiryurek. Maximizing the number of worker's self-selected tasks in spatial crowdsourcing. In *GIS*, 2013.
[8] Yu Li, Man Lung Yiu, and Wenjian Xu. Oriented online route recommendation for spatial crowdsourcing task workers. In *SSTD*, 2015.
[9] Dingxiong Deng, Cyrus Shahabi, and Linhong Zhu. Task matching and scheduling for multiple workers in spatial crowdsourcing. In *GIS*, 2015.
[10] Dingxiong Deng, Cyrus Shahabi, Ugur Demiryurek, and Linhong Zhu. Task selection in spatial crowdsourcing from worker's perspective. *GeoInformatica*, 2016.
[11] Huaxin Li, Haojin Zhu, Suguo Du, Xiaohui Liang, and Xuemin Sherman Shen. Privacy leakage of location sharing in mobile social networks: Attacks and defense. *IEEE Transactions on Dependable and Secure Computing*, 2018.
[12] Natalia Marmasse and Chris Schmandt. Location-aware information delivery with commotion. In *HUC*, 2000.
[13] Hien To, Cyrus Shahabi, and Li Xiong. Privacy-preserving online task assignment in spatial crowdsourcing with untrusted server. In *ICDE*, 2018.
[14] Hien To, Gabriel Ghinita, and Cyrus Shahabi. A framework for protecting worker location privacy in spatial crowdsourcing. *PVLDB*, 2014.
[15] Qian Tao, Yongxin Tong, Zimu Zhou, Yexuan Shi, Lei Chen, and Ke Xu. Differentially private online task assignment in spatial crowdsourcing: A tree-based approach. In *ICDE*, 2020.
[16] John Krumm. A survey of computational location privacy. *Personal and Ubiquitous Computing*, 2009.
[17] Miguel E. Andrés, Nicolás Emilio Bordenabe, Konstantinos Chatzikokolakis, and Catuscia Palamidessi. Geo-indistinguishability: differential privacy for location-based systems. In *CCS*, 2013.
[18] Cynthia Dwork. Differential privacy. In *ICALP*, 2006.
[19] Leye Wang, Dingqi Yang, Xiao Han, Tianben Wang, Daqing Zhang, and Xiaojuan Ma. Location privacy-preserving task allocation for mobile crowdsensing with differential geo-obfuscation. In *WWW*, 2017.
[20] Yuxiang Zeng, Yongxin Tong, Lei Chen, and Zimu Zhou. Latency-oriented task completion via spatial crowdsourcing. In *ICDE*, 2018.
[21] Yajun Wang and Sam Chiu-wai Wong. Two-sided online bipartite matching and vertex cover: Beating the greedy algorithm. In *ICALP*, 2015.
[22] Yongxin Tong, Jieying She, Bolin Ding, Libin Wang, and Lei Chen. Online mobile micro-task allocation in spatial crowdsourcing. In *ICDE*, 2016.
[23] Konstantinos Chatzikokolakis, Catuscia Palamidessi, and Marco Stronati. Location privacy via geo-indistinguishability. *ACM SIGLOG News*, 2015.
[24] Changsha Ma and Chang Wen Chen. Nearby friend discovery with geo-indistinguishability to stalkers. In *FNC*, 2014.
[25] Yongxin Tong, Libin Wang, Zimu Zhou, Bolin Ding, Lei Chen, Jieping Ye, and Ke Xu. Flexible online task assignment in real-time spatial data. *PVLDB*, 2017.
[26] Leyla Kazemi and Cyrus Shahabi. Geocrowd: enabling query answering with spatial crowdsourcing. In *GIS*, 2012.
[27] Leyla Kazemi, Cyrus Shahabi, and Lei Chen. Geotrucrowd: trustworthy query answering with spatial crowdsourcing. In *GIS*, 2013.
[28] Hien To, Cyrus Shahabi, and Leyla Kazemi. A server-assigned spatial crowdsourcing framework. *ACM Transactions on Spatial Algorithms and Systems*, 2015.
[29] An Liu, Weiqi Wang, Shuo Shang, Qing Li, and Xiangliang Zhang. Efficient task assignment in spatial crowdsourcing with worker and task privacy protection. *GeoInformatica*, 2017.
[30] Yongxin Tong, Yuxiang Zeng, Bolin Ding, Libin Wang, and Lei Chen. Two-sided online micro-task assignment in spatial crowdsourcing. *IEEE Transactions on Knowledge and Data Engineering*, 2019.
[31] Bruce L Golden, Larry Levy, and Rakesh Vohra. The orienteering problem. *Naval research logistics*, 1987.
[32] Yan Zhao, Yang Li, Yu Wang, Han Su, and Kai Zheng. Destination-aware task assignment in spatial crowdsourcing. In *CIKM*, 2017.
[33] Yuxiang Zeng, Yongxin Tong, and Lei Chen. Last-mile delivery made practical: An efficient route planning framework with theoretical guarantees. *PVLDB*, 2019.
[34] Yuxiang Zeng, Yongxin Tong, Yuguang Song, and Lei Chen. The simpler the better: An indexing approach for shared-route planning queries. *PVLDB*, 2020.