Fine-Grained Geo-Obfuscation to Protect Workers' Location Privacy in Time-Sensitive Spatial Crowdsourcing

Chenxi Qiu, Sourabh Yadav, Yuede Ji
University of North Texas
Denton, Texas
{chenxi.qiu,sourabhyadav,yuede.ji}@unt.edu

Anna Squicciarini
Pennsylvania State University
University Park, Pennsylvania
acs20@psu.edu

Ramanamurthy Dantu
University of North Texas
Denton, Texas
ram.dantu@unt.edu

Juanjuan Zhao
Shenzhen Institute of Advanced Technology
Shenzhen, Guangdong
jj.zhao@siat.ac.cn

Chengzhong Xu
University of Macau
Macau, Guangdong
czxu@um.edu.mo

ABSTRACT

Geo-obfuscation is a *location privacy protection mechanism* used by mobile users to conceal their precise locations when reporting location data, and it has been widely used to protect the location privacy of workers in *spatial crowdsourcing (SC)*. However, this technique introduces inaccuracies in the reported locations, raising the question of how to control the quality loss that results from obfuscation in SC services. Prior studies have addressed this issue in time-insensitive SC settings, where some degree of quality degradation can be accepted and the locations can be expressed with less precision, which, however, is inadequate for time-sensitive SC.

In this paper, we aim to minimize the quality loss caused by geo-obfuscation in *time-sensitive SC applications*. To this end, we model workers' mobility on a fine-grained location field and constrain each worker's obfuscation range to a set of *peer locations*, which have similar traveling costs to the destination as the actual location. We apply a *linear programming (LP)* framework to minimize the quality loss while satisfying both peer location constraints and *geo-indistinguishability*, a location privacy criterion extended from *differential privacy*. By leveraging the constraint features of the formulated LP, we enhance the time efficiency of solving LP through the geo-indistinguishability constraint reduction and the column generation algorithm. Using both simulation and real-world experiments, we demonstrate that our approach can reduce the quality loss of SC applications while protecting workers' location privacy.

ACM Reference Format:
Chenxi Qiu, Sourabh Yadav, Yuede Ji, Anna Squicciarini, Ramanamurthy Dantu, Juanjuan Zhao, and Chengzhong Xu. 2023. Fine-Grained Geo-Obfuscation to Protect Workers' Location Privacy in Time-Sensitive Spatial Crowdsourcing. In *Proceedings of ACM Conference (Conference'17).* ACM, New York, NY, USA, 15 pages. https://doi.org/10.1145/nnnnnnn.nnnnnnn

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than ACM must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org.
Conference'17, July 2017, Washington, DC, USA
© 2023 Association for Computing Machinery.
ACM ISBN 978-x-xxxx-xxxx-x/YY/MM…$15.00
https://doi.org/10.1145/nnnnnnn.nnnnnnn

1 INTRODUCTION

With the rapid advancement of wireless communication and positioning technologies in mobile devices, *spatial crowdsourcing (SC)* has become increasingly popular and attracted a large number of mobile users to participate in various location-based services (LBS) [1–3]. In SC, workers are required to be physically present at task locations to complete tasks, such as providing rides to passengers [4] or taking photos [5]. To ensure cost-effective services, tasks should be assigned to nearby workers with minimal traveling costs, which requires workers to report their current location to servers. However, such reporting may disclose sensitive personal information like home addresses [6]. Furthermore, in many SC platforms, like PulsePoint [7], workers are volunteers receiving little compensation, and disclosing their location may discourage participation, ultimately leading to a low number of workers available in SC services.

In recent years, location privacy issues in LBS have received significant attention, where a rich body of works has been centered on *geo-obfuscation* [8–14], a *location privacy protection mechanism (LPPM)* that enables workers to report obfuscated locations to servers instead of their exact locations. Recently, Andrés *et al* [10] introduced a formal privacy criterion for geo-obfuscation, called *geo-indistinguishability (Geo-Ind)*. Geo-Ind requires that for each pair of real locations that are geographically close, their obfuscated locations are generated with similar probability distributions, making it difficult for an attacker to distinguish between the two real locations based on their obfuscated representations. Geo-obfuscation has been recognized as a stronger alternative to mobile LBS compared to traditional cryptographic approaches [15], as it places a lower computational demand on mobile devices [14, 16] while effectively protecting data in the case of a data breach on the server side [17].

However, the use of geo-obfuscation inevitably introduces errors to the reported locations of workers, which can cause SC servers to assign tasks to workers with higher traveling costs. Therefore, a key issue of geo-obfuscation techniques is *how to select suitable obfuscated locations that allow SC servers to accurately estimate the traveling costs for the requested tasks.*

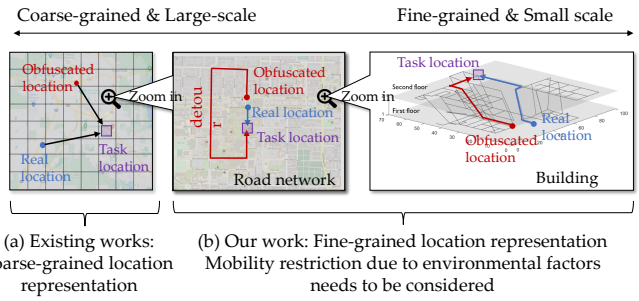


Figure 1: Coarse-grained obfuscation vs. fine-grained obfuscation.

Existing works. One of the most widely used paradigms to tackle the quality issue caused by geo-obfuscation is to employ a *linear programming (LP)* framework, of which the objective is to minimize the quality loss (measured by the estimation error of the workers' traveling costs) while still satisfying the Geo-Ind constraints [18]. Typically, LP-based geo-obfuscation methods discretize the location field into a finite set of discrete locations. Its decision variables determine the probability distributions of obfuscated locations given each possible real discrete location. As such, LP-based methods require K^2 decision variables to derive in LP given K discrete locations in the location set. Such a high computation load makes it difficult for LP-based methods to cover a sufficiently high number of locations. For example, covering thousands of discrete locations within a small town can lead to millions of decision variables in LP [8].

To reduce the computation load, most existing LP-based works limit the design of geo-obfuscation to *low granularity location representation* [12, 14, 19–22], such as the grid map shown in Fig. 1(a) (each grid cell represents a location). Moreover, the existing LP-based methods [8, 12] aim to minimize the expected traveling cost estimation error of all possible obfuscated locations, but with no restriction for the estimation error caused by each single obfuscated location. These methods, clearly, are unsuitable to time-sensitive SC applications, such as *cardiopulmonary resuscitation (CPR) worker assignment*, where every minute of delay in initiating CPR reduces the probability of survival by 7–10% [23].

Our contributions. To address the research gap outlined above, *this paper aims to study the optimization of geo-obfuscation by considering users' fine-grained mobility features (**Obj1**)*, such as within a building or a university campus, as Fig. 1(b) shows. In particular, we consider workers' mobility restrictions caused by diverse environmental factors, e.g., workers cannot move freely in a building due to the building's structure; workers have to traverse stairs to reach different floors. In this case, the location field needs to be discretized at a higher granularity level. Moreover, *we propose a new approach to restrict the obfuscation range of each actual location to a "peer location set", which is composed of the obfuscated locations with similar traveling costs to the destination as the actual location (**Obj2**)*. As such, the estimation error of traveling costs caused by each obfuscated location can be bounded by a predetermined threshold.

The main challenge of achieving **Obj1** and **Obj2** lies in how to efficiently solve the formulated LP to meet the demands of time-sensitive SC. Given the number of fine-grained locations K, formulating the LP problem requires $O(K^2)$ decision variables and $O(K^3)$ Geo-Ind constraints in the worst case. This means that even a few hundred discrete locations can result in tens of thousands of decision variables and millions of Geo-Ind constraints, leading to a significant computation delay if we attempt to solve it using the classic LP algorithms such as the simplex method [24]. Additionally, the process of computing the peer location set for each real location exhibits a $O(K)$ time complexity, leading to an overall time complexity of $O(K^2)$ when calculating the peer location sets for all locations. Finally, adding the peer location constraints to the LP formulation causes a different constraint structure compared to that of the classic geo-obfuscation optimization problems [8, 12, 25], making their solutions hard to apply to our newly formulated problem.

To address the aforementioned challenges, we design the following three methods to improve the computation efficiency of geo-obfuscation:
(1) We design an algorithm to identify the peer locations for all the locations jointly. Specifically, we sort all the locations in the mobility graph based on their traveling costs to the given destination, and then calculate their peer location sets sequentially via two sliding windows, **achieving a $O(K)$ average-case time complexity**.
(2) We perform *Geo-Ind constraint reduction* by exploring the Geo-Ind's *transitivity property on each peer location set*, which **reduces the number of Geo-Ind constraints from $O(K^3)$ to $O(K^2)$ without compromising the optimality of the LP's solutions**.
(3) By leveraging the *angular block* structure of the constraint matrix of the formulated LP, we employ the *column generation algorithm* to **further decrease both the number of decision variables and the number of Geo-Ind constraints in LP from $O(K^2)$ to $O(K)$**.

Finally, we assess the effectiveness of our approach through both trace-driven simulation and real-world experiments. We employ the vehicle trajectory dataset in Shenzhen [26] to simulate the distribution of crowdsourcing workers. We compare the quality loss of our approach against three benchmarks: Grid-based obfuscation [14], Laplacian noise [10], and vehicle-based geo-obfuscation [8]. The simulation results demonstrate that our approach reduces the quality loss by at least 61.76% compared to the benchmarks. Furthermore, using the prototype we developed, we conducted real-world experiments in two relatively smaller target regions: our college building (Discovery Park) and our university campus (UNT Denton campus). The experimental results show that our approach has an average computation time of approximately 2.29 seconds, which is suitable for time-sensitive SC tasks, such as CPR assignments.

The remainder of the paper is organized as follows: We first overview the SC framework in Section 2 and formulate the problem in Section 3. We then introduce our algorithms in Section 4, and evaluate the performance in 5. We discuss related work in Section 6 and conclude in Section 7.

2 FRAMEWORK

Our SC framework is depicted in Fig. 2, which comprises an *SC server*, a *task requester*, and *a pool of workers*. Consider the *CPR assignment* as an example: In the event of an emergency, nearby CPR workers must proceed to the designated patient with the shortest traveling time, aided by a server that provides task assignments. It is worth noting that although our primary focus is on SC applications, this framework can be applied with minimal modifications to

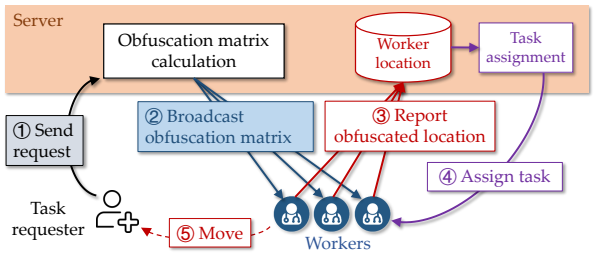


Figure 2: The framework of CPR for worker assignment.

general time-sensitive SC applications. For instance, the SC server can be replaced by a service provisioning entity.

Threat Model. The task requester and the workers need to report the task's location and the workers' locations to the SC server to enable task assignments. We assume that although the server is not malicious, it might suffer from a *passive attack where attackers can eavesdrop on workers' reported locations breached by the server* [9–12]. As we target time-sensitive SC, we consider the case that the *task requester needs to disclose the exact task location to the server* (e.g., when a patient suffers from a heart attack out of the hospital, his/her exact location needs to be disclosed to the server to receive CPR as early as possible). Yet, *workers' exact locations should be hidden from the server*, especially in applications where self-location disclosure might discourage more workers from participating (e.g., volunteer-based scenarios [7]).

SC task assignment. Similar to previous studies [8, 12], our privacy-aware workers conceal their actual location by employing an obfuscation function before transmitting their location to the server. The obfuscation function takes the worker's true location as input and generates a probability distribution of the obfuscated location. The worker can then choose an obfuscated location from this distribution to report. Due to computational feasibility, we consider the workers' location field to be a discrete set $\mathcal{V} = \{v_1, ..., v_K\}$ [8, 12]. In this case, the obfuscation function can be represented by a stochastic matrix called the *obfuscation matrix*, denoted by $\mathbf{Z} = \{z_{i,k}\}_{K \times K}$. Each $z_{i,k}$ represents the probability of selecting v_k as the obfuscated location given the actual location v_i.

The process of a task assignment in our SC framework includes the following steps (as shown in Fig. 2).
① The task requester, such as a patient, sends a request to the SC server, including the task's exact location.
② The server calculates the obfuscation matrix and broadcasts the task request, along with the obfuscation matrix, to nearby workers who have opted in, without disclosing the task's exact location.
③ If a worker accepts the request, they report their obfuscated location to the server using the obfuscation matrix.
④ Based on workers' reported locations, the server estimates their traveling costs to the task location and selects workers with lower traveling costs to complete the task. The server then sends the request, including the task's exact location, to the selected workers.
⑤ Finally, the selected workers move to the task location to complete the task.

Note that although the server generates the obfuscation matrix, the workers' exact locations remain hidden from the server [12]. The obfuscation matrix is designed to satisfy the privacy criterion *Geo-Ind*, which means that even if an attacker obtains the workers' reported obfuscated location and the obfuscation matrix from the SC server, it is still difficult for the attacker to distinguish the workers' exact locations from other nearby locations. More details on the calculation of the geo-obfuscation matrix, necessary for achieving this feature, are presented in Sections 3 and 4.

3 PROBLEM STATEMENT

In this section, we start by introducing the mathematical models in Section 3.1, based on which we then formulate the problem in Section 3.2.

Table 1 lists the main notations and their descriptions used throughout this paper.

Table 1: Main notations and their descriptions.

Symbol	Description
$\mathcal{V}$	$\mathcal{V} = \{v_1, ..., v_K\}$ denotes the discrete location set
$\mathcal{G}$	$\mathcal{G} = (\mathcal{V}, \mathcal{E})$ denotes the mobility graph of workers; where $\mathcal{V}$ and $\mathcal{E}$ are the location set and the edge set
$e_{i,j}$	Edge from v_i to v_j
$c_{i,j}$	Traveling cost from v_i to v_j
$\mathbf{Z}$	Obfuscation matrix $\mathbf{Z}$
$z_{i,k}$	Probability of selecting v_k as the obfuscated location given the real location v_i
$\mathcal{P}_i$	Peer location set of v_i
ϵ	Privacy budget of Geo-Ind
$\Delta(\mathbf{Z})$	Expected quality loss caused by $\mathbf{Z}$
p_i	Prior probability that the worker is at location v_i in $\mathcal{G}$
$\mathcal{T}_i$	Shortest path tree rooted at location v_i in $\mathcal{G}$

3.1 Model

3.1.1 Worker mobility model. Like [8, 11, 14], we consider workers' locations on a discrete location set $\mathcal{V} = \{v_1, ..., v_K\}$. If a worker can travel from v_i to v_j without visiting other locations in $\mathcal{V}$, then we build an edge $e_{i,j}$ from v_i to v_j, and call that v_i is an *in-neighbor* of v_j (or v_j is an *out-neighbor* of v_i), and v_i and v_j are *adjacent* to each other. Each $e_{i,j}$ is assigned a weight $c_{i,j}$ representing the traveling cost from v_i to v_j. Then, we can create a *weighted directed graph* $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, called *mobility graph*, where $\mathcal{V}$ and $\mathcal{E}$ denote the node (location) set and the edge set, respectively. If v_i is not an in-neighbor of v_j, then the traveling cost $c_{i,j}$ from v_i to v_j is equal to the length of the *shortest path* from v_i to v_j in the graph $\mathcal{G}$.

Based on the discrete location set $\mathcal{V}$, the obfuscation matrix $\mathbf{Z} = \{z_{i,k}\}_{K \times K}$ describes the probability distributions of obfuscated locations given any real location, i.e., each $z_{i,k}$ denotes the probability of taking v_k as the obfuscated location given the actual location v_i.

3.1.2 Quality loss bounded by the peer location constraints. As Fig. 3 shows, given the task location v_t, two locations $v_i, v_j \in \mathcal{V}$ are called *peer locations*, written as $v_j \sim v_i$, if and only if the difference between their traveling costs to v_t is no larger than η, i.e.,

$$\left| c_{i,t} - c_{j,t} \right| \leq \eta, \tag{1}$$

where $\eta > 0$ is a pre-determined constant.

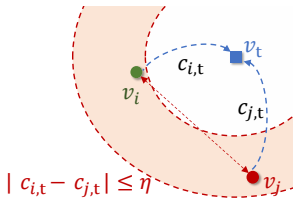


Figure 3: Definition of peer locations.

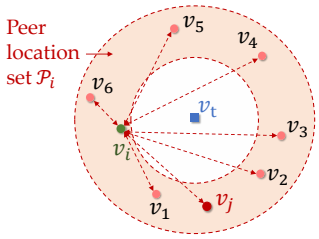


Figure 4: Most peer locations ($\{v_2, v_3, v_4, v_5\}$) are distanced from the real location v_i.

Property 3.1. *(Properties of "$\sim$") According to the definition in Equ. (1), the peer relation "$\sim$" is (a)* reflexive, *i.e., $v_i \sim v_i, \forall v_i \in \mathcal{V}$; (b)* commutative, *i.e., $v_j \sim v_i$ implies $v_i \sim v_j, \forall v_i, v_j \in \mathcal{V}$; (c) but* not transitive, *i.e., $v_j \sim v_i, v_i \sim v_k$ doesn't imply $v_j \sim v_k$.*

For a given task location v_t, we refer to the set of peer locations of each location v_i as $\mathcal{P}_i = \{v_j \in \mathcal{V} | v_i \sim v_j\}$. To ensure the estimation accuracy of traveling cost, we restrict the obfuscation range of a worker to his/her real location v_i's peer location set $\mathcal{P}_i$, called the *peer location constraints*. Specifically, any location v_k outside of v_i's peer location set won't be selected as an obfuscated location for v_i:

$$z_{i,k} = 0, \quad \forall v_k \notin \mathcal{P}_i, \tag{2}$$

In addition, for each real location v_i, the sum probability of selecting the obfuscated locations in $\mathcal{P}_i$ should be 1 (*probability unit measure*), i.e.,

$$\sum_{v_k \in \mathcal{P}_i} z_{i,k} = 1, \ \forall i = 1, ..., K. \tag{3}$$

Note that although limiting the selection of obfuscated locations to the peer location set narrows the obfuscation range, it still upholds users' location privacy. As exemplified in Fig. 4, the peer locations $\mathcal{P}_i$ of v_i are in the red area, and the majority of these peer locations, e.g., $\{v_2, v_3, v_4, v_5\}$, are distanced from the actual location v_i, even though they share a similar traveling cost to the destination as the real location. The substantial distance between the real location and the peer locations leads to a notable degree of inference error in the estimated location made by a potential attacker. This observation is further demonstrated by the findings presented in Fig. 16(a)(b) and Fig. 17(a)(b) in Section 5.

3.1.3 Privacy criterion. We select *geo-indistinguishability (Geo-Ind)* [8, 12, 14] as the privacy criterion for the obfuscated location selection. Intuitively, Geo-Ind enforces that, for any pair of locations v_i and v_j that are geographically close, the probability distributions of their obfuscated locations should be sufficiently close, so that it is hard for attackers to distinguish v_i and v_j based on their obfuscated locations. Formally, ϵ-Geo-Ind is originally defined as, $\forall v_k \in \mathcal{V}$,

$$z_{i,k} - e^{\epsilon c_{i,j}} z_{j,k} \leq 0, \ \forall v_i, v_j \in \mathcal{V}, \tag{4}$$

where ϵ is called *privacy budget*, quantifying how close are v_i and v_j's obfuscated location probability distributions. Higher ϵ implies that the two real locations are more distinguishable and hence a lower privacy level to achieve.

According to the peer location constraint (Equ. (2)) and the peer relation's *commutativity* (*Property 3.1 (b)*), when formulating the Geo-Ind constraint (in Equ. (4)) for each obfuscated location v_k, we only need to consider the real locations v_i and v_j that are v_k's peer locations, i.e., $v_i, v_j \in \mathcal{P}_k$. Therefore, we modify the ϵ-Geo-Ind constraint in Equ. (4) to: $\forall v_k \in \mathcal{V}$,

$$z_{i,k} - e^{\epsilon c_{i,j}} z_{j,k} \leq 0, \ \forall v_i, v_j \in \mathcal{P}_k. \tag{5}$$

3.2 Problem Formulation

Given the task location v_t and the real location v_i, the quality loss caused by an obfuscated location v_k is calculated by $\Delta c_{i,k} = |c_{i,t} - c_{k,t}|$. Our objective is to minimize the expected quality loss of the obfuscation matrix $\Delta(\mathbf{Z})$, defined by

$$\Delta(\mathbf{Z}) = \sum_{i=1}^{K} p_i \sum_{k=1}^{K} z_{i,k} \Delta c_{i,k} = \sum_{k=1}^{K} \mathbf{c}_k^\top \mathbf{z}_k, \tag{6}$$

where p_k ($k = 1, ..., K$) denotes the prior probability that a worker's real location is at v_k, $\mathbf{c}_k = [p_1 \Delta c_{1,k}, ..., p_K \Delta c_{K,k}]^\top$, and $\mathbf{z}_k = [z_{1,k}, ..., z_{K,k}]^\top$. To satisfy the constraints of peer location (Equ. (2)), probability unit measure (Equ. (3)), Geo-Ind (Equ. (5)), and minimize $\Delta(\mathbf{Z})$, we formulate the problem of ***Geo-obfuscation generation in Time-sensitive SC (GTS)*** as the following *linear programming (LP)* problem.

$$\min \quad \Delta(\mathbf{Z}) \tag{7}$$

$$\text{s.t.} \quad \text{Equ. (2) (3) (5) are satisfied.} \tag{8}$$

In general, the computation load of an LP problem depends on the number of decision variables and linear constraints [24]. In the case of GTS, the numbers of decision variables and Geo-Ind constraints are $O(K^2)$ and $O(K^3)$, respectively, leading to an extremely high computation load. Furthermore, altering the Geo-Ind constraints in Equ. (4) to Equ. (5) results in our problem distinct from the classic geo-obfuscation optimization problems [8, 12, 25]. As a consequence, the applicability of their constraint reduction and decomposition techniques to this newly formulated problem becomes challenging. Considering that the optimal $\mathbf{Z}$ should be derived quickly in SC due to the time-sensitive nature of applications, in Section 4, we present the new algorithms that can solve GTS in a high time efficiency.

4 ALGORITHM DESIGN

In this section, we introduce three algorithms to improve the computation efficiency of solving GTS: *peer location searching* (Section 4.1), *Geo-Ind-constraint reduction* (Section 4.2), and the *column generation algorithm* (Section 4.3).

4.1 Peer Location Searching

Upon receiving a task request, the server calculates the peer location set for the actual location v_i based on the task location v_t. Note that as the server lacks information regarding the precise location of the worker, it has to compute the peer location set for all $v_i \in \mathcal{V}$.

To calculate the traveling cost $c_{i,t}$ (or the shortest path distance) from each v_i to v_t, the server first builds the *shortest path tree (SPT)* rooted at v_t using the Dijkstra's algorithm [27]. The server then sorts the locations in $\mathcal{V}$ according to $c_{i,t}$. Without loss of generality, in what follows we assume that $c_{i,t} \leq c_{i+1,t}$ ($i = 1, ..., K-1$).

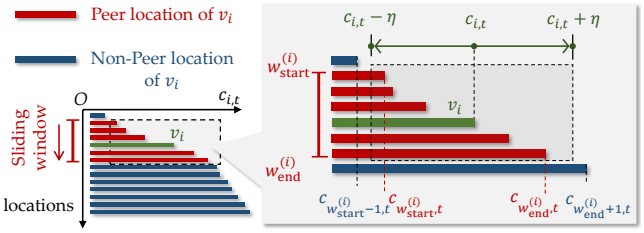


Figure 5: Sliding window of peer location searching (at round i).

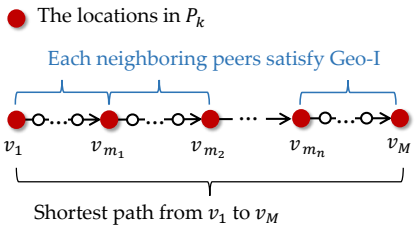


Figure 6: Transitivity property of Geo-Ind in a graph.

Given the sorted locations $v_1, ..., v_K$, the server searches the peer location set of each v_i sequentially, such that in each round i, the peer location set $\mathcal{P}_i$ of v_i can be identified.

As Fig. 5 shows, to avoid using nested loops with $O(K^2)$ time complexity, a sliding window $\left[w_{\text{start}}^{(i)}, w_{\text{end}}^{(i)}\right]$ is maintained by the server so that v_i's peer location set $\mathcal{P}_i$ falls exactly within the window in each round i, i.e., $\mathcal{P}_i = \left\{v_j \in \mathcal{V} \middle| v_j \in \left[w_{\text{start}}^{(i)}, w_{\text{end}}^{(i)}\right]\right\}$. To this end, we locate $w_{\text{start}}^{(i)}$ and $w_{\text{end}}^{(i)}$ to satisfy

$$c_{i,t} - \eta \in \left[c_{w_{\text{start}}^{(i)},t}, c_{w_{\text{start}}^{(i)}+1,t}\right), \; c_{i,t} + \eta \in \left(c_{w_{\text{end}}^{(i)}-1,t}, c_{w_{\text{end}}^{(i)},t}\right]. \quad (9)$$

Here, both $w_{\text{start}}^{(0)}$ and $w_{\text{end}}^{(0)}$ are initialized by 1. Comparing the windows for v_{i-1} and v_i, denoted by $\left[w_{\text{start}}^{(i-1)}, w_{\text{end}}^{(i-1)}\right]$ and $\left[w_{\text{start}}^{(i)}, w_{\text{end}}^{(i)}\right]$, we can find that

$$c_{w_{\text{start}}^{(i-1)},t} \le c_{i-1,t} - \eta \le c_{i,\text{t}} - \eta < c_{w_{\text{start}}^{(i)}+1,t} \quad (10)$$

$$\Rightarrow \quad w_{\text{start}}^{(i-1)} < w_{\text{start}}^{(i)} + 1 \Rightarrow w_{\text{start}}^{(i-1)} \le w_{\text{start}}^{(i)} \quad (11)$$

$$c_{w_{\text{end}}^{(i-1)}-1,t} < c_{i-1,t} + \eta \le c_{i,t} + \eta \le c_{w_{\text{end}}^{(i)},t} \quad (12)$$

$$\Rightarrow \quad w_{\text{end}}^{(i-1)} - 1 < w_{\text{end}}^{(i)} \Rightarrow w_{\text{end}}^{(i-1)} \le w_{\text{end}}^{(i)} \quad (13)$$

indicating that the sliding window never moves backward from round $i-1$ to round i ($i = 1, ..., K$).

Time complexity of peer location searching. The time complexity of the peer location searching can be calculated using *amortized analysis*, which is a worst-case time complexity analysis of a sequence of operations [27]. In each round i, the starting location of the window moves from $w_{\text{start}}^{(i-1)}$ to $w_{\text{start}}^{(i)}$, taking $w_{\text{start}}^{(i)} - w_{\text{start}}^{(i-1)} + 1$ iterations. In each iteration, the server checks whether the current starting location satisfies Equ. (9), and if not, it moves the window's starting location to the next location, which takes $O(1)$ operations. Thus, the time complexity of moving the starting location of the window from round 1 to round K is $\sum_{i=1}^{K}\left(w_{\text{start}}^{(i)} - w_{\text{start}}^{(i-1)} + 1\right) \times O(1) = O(K)$. Similarly, we derive that the time complexity to move the ending location of the window $w_{\text{end}}^{(i)}$ is also $O(K)$. Therefore, the time complexity of the peer location searching is $O(K) + O(K) = O(K)$.

4.2 The Geo-Ind Constraint Reduction

Since each obfuscated location v_k ($k = 1, ..., K$) can have at most $K-1$ peer locations, the number of constraints created by each obfuscated location in Equ. (5) is at most $K(K-1)$. Therefore, in the worst case, the total number of Geo-Ind constraints for all the obfuscated locations $v_1, ..., v_K$ is $O(K) \times O(K(K-1)) = O(K^3)$.

To improve time efficiency without sacrificing the optimality of the GTS solution, we utilize the transitivity property of Geo-Ind constraints in graphs to reduce the number of Geo-Ind constraints.

It is important to highlight that, the transitivity property of Geo-Ind, as defined in this paper, differs from the one presented in [16]. In the context of [16], Geo-Ind was imposed on every pair of adjacent locations (nodes) in the mobility graph. Conversely, in the current paper, Geo-Ind is exclusively enforced between each pair of locations v_i and v_j that share the same peer location v_k (according to Equ. (5)), i.e., v_i and v_j have to be in the same peer location set $\mathcal{P}_k$. Consequently, the transitivity property expounded in [16] cannot be applied to our present scenario.

As a solution, we first define *neighboring peers* in Definition 4.1 and prove that, to enforce Geo-Ind for all the pairs in $\mathcal{P}_k$, it is sufficient to enforce Geo-Ind only for each pair of neighboring peers in $\mathcal{P}_k$. We then propose a time-efficient algorithm to create the Geo-Ind constraints for all the neighboring peers in each peer location set $\mathcal{P}_k$, of which the detailed pseudo code is shown in Algorithm 1 and Algorithm 2.

Definition 4.1. *(*Neighboring peers*) Given an obfuscated location v_k's peer location set $\mathcal{P}_k$, a pair of locations (v_i, v_j) is called* ***neighboring peers*** *if no other location $v_l \in \mathcal{P}_k$ is in the shortest path between v_i and v_j (in both directions). We use $\mathcal{N}_{i,k}$ to denote the set of v_i's neighboring peers in $\mathcal{P}_k$.*

Theorem 4.1. *(*Transitivity of Geo-Ind in the peer location set*) To enforce Geo-Ind for each pair of locations in $\mathcal{P}_k$, it is sufficient to enforce Geo-Ind only for each pair of neighboring peers in $\mathcal{P}_k$.*

Proof. We pick up any pair of locations in $\mathcal{P}_k$. Without loss of generality, we denote the two locations by (v_1, v_M) and denote their shortest path in $\mathcal{P}_k$ by $\mathcal{S}_{(v_1,v_M)} = ((v_1, v_2), ..., (v_{M-1}, v_M))$, as Fig. 6 shows. We then prove that (v_1, v_M) satisfies Geo-Ind if all the neighboring peers in $\mathcal{P}_k$ satisfy Geo-Ind.

We use $v_{m_1}, ..., v_{m_n}$ ($1 < m_1 < ... < m_n < M$) to denote the locations in $\mathcal{P}_k$ along the shortest path $\mathcal{S}_{(v_1,v_M)}$, and let $m_0 = 1$ and $m_{n+1} = M$. Since $v_{m_1}, ..., v_{m_n}$ are in the shortest path from v_1 to v_M sequentially, $c_{1,M} = \sum_{l=0}^{n} c_{m_l,m_{l+1}}$. Because each neighboring peer $(v_{m_l}, v_{m_{l+1}})$ ($l = 1, ..., M-1$) satisfies Geo-Ind, for each obfuscated location v_k,

$$z_{1,k} - e^{\epsilon c_{1,M}} z_{M,k} = z_{1,k} - e^{\epsilon \sum_{l=0}^{n} c_{m_l,m_{l+1}}} z_{M,k} \quad (14)$$

$$= \sum_{l=0}^{n} \underbrace{\left(z_{m_l,k} - e^{\epsilon c_{m_l,m_{l+1}}} z_{m_{l+1},k}\right)}_{\le 0 \text{ since } (v_{m_l}, v_{m_{l+1}}) \text{ satisfy Geo-Ind}} e^{\epsilon \sum_{h=1}^{l} c_{m_h,m_{h+1}}} \le 0,$$

indicating that (v_1, v_M) satisfy Geo-Ind. The proof is completed. □

Note that Theorem 4.1 presented in this paper is a generalized version of the transitivity property established in [16].

Algorithm 1: The Geo-Ind constraint formulation for v_k.

Input : $\mathcal{P}_k$
Output: The set of Geo-Ind constraints for v_k
1 **for** *each* $v_i \in \mathcal{V}$ **do**
2 Build the shortest path tree $\mathcal{T}_i$ rooted at v_i using the Dijkstra algorithm;
3 $\mathcal{N}_{i,k} \leftarrow \phi$;
4 $\mathcal{N}_{i,k} \leftarrow$ DFS$(v_i, \mathcal{T}_i, \mathcal{P}_k, \mathcal{N}_{i,k})$;
5 **for** *each* $v_j \in \mathcal{N}_{i,k}$ **do**
6 Add the Geo-Ind constraint for (v_i, v_j) to the LP formulation;
7 **return**;

Algorithm 2: DFS$(v_j, \mathcal{T}_i, \mathcal{P}_k, \mathcal{N}_{i,k})$.

Input : $v_j, \mathcal{T}_i, \mathcal{P}_k, \mathcal{N}_{i,k}$
Output: $\mathcal{N}_{i,k}$
1 **if** $v_j \in \mathcal{P}_k$ **then**
2 Add v_j to $\mathcal{N}_{i,k}$; // Step ①
3 **return** $\mathcal{N}_{i,k}$; // Step ②
4 **else**
5 **if** v_j *is a leaf node of* $\mathcal{T}_i$ **then**
6 **return** $\mathcal{N}_{i,k}$;
7 **for** *each child* v_l *of* v_j *in* $\mathcal{T}_i$ **do**
8 DFS$(v_l, \mathcal{T}_i, \mathcal{P}_k, \mathcal{N}_{i,k})$;

Neighboring peers' searching algorithm. Theorem 4.1 states that enforcing Geo-Ind only for neighboring peers in $\mathcal{P}_k$ is sufficient to enforce it for all pairs of locations in $\mathcal{P}_k$. To implement this, Algorithm 1 formulates the Geo-Ind constraints for v_k using a shortest path tree approach. Specifically, using the Dijkstra algorithm [27] (line 2), the algorithm constructs the shortest path tree $\mathcal{T}_i$ rooted at v_i for each $v_i \in \mathcal{V}$. Then, the algorithm traverses v_i's neighboring peers $\mathcal{N}_{i,k}$ in $\mathcal{P}_k$ using a *depth-first-search (DFS)* approach [27] (line 3–4) and formulates the Geo-Ind constraints for v_i and its neighboring peers (line 5–6). Algorithm 2 provides the pseudo-code for DFS. As shown in Fig. 7, when a neighboring peer of v_i (step ①) is reached, such as v_j, the algorithm backtracks to v_j's parent (step ②) without exploring v_j's descendants. This is because to reach a descendant of v_j, such as v_l, from v_i, the path must pass through v_j, indicating that v_l cannot be v_i's neighboring peer according to *Definition 4.1*.

Complexity of GTS after the Geo-Ind constraint reduction. After applying the Geo-Ind constraint reduction, the number of Geo-Ind constraints for each obfuscated location v_k is approximately equal to the number of edges in the worker mobility graph $\mathcal{G}$. Note that $\mathcal{G}$ retrieved from real-world maps closely resembles a planar graph, such as the city road map shown in Fig. 9 and the campus road map shown in Fig. 19(b). $\mathcal{G}$ can be also a constant number

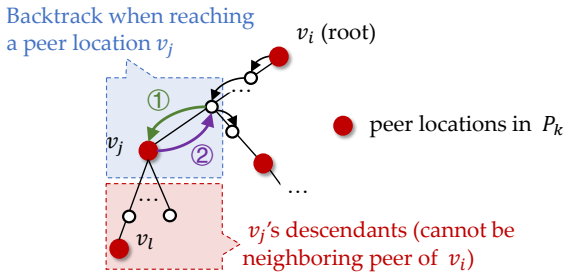


Figure 7: DFS of v_i's neighboring peers in v_i's SPT.

of planar graphs, such as the building map with 2 floors shown in Fig. 19(a), where each floor corresponds to a planar graph. Since the number of edges in a planar graph is $O(K)$, the number of Geo-Ind constraints for each obfuscated location v_k is up to $O(K)$. Consequently, the total number of Geo-Ind constraints for all the obfuscated locations $v_1, ..., v_K$ is $O(K^2)$. As Theorem 4.1 suggests, the reduced constraints are sufficient to maintain the optimality of the original GTS constraints.

Despite reducing the number of Geo-Ind constraints from $O(K^3)$ to $O(K^2)$, the Geo-Ind constraint reduction technique in GTS still results in $O(K^2)$ decision variables and $O(K^2)$ Geo-Ind constraints. Thus, the size of the constraint matrix in GTS remains at $O(K^2) \times O(K^2)$. Our experimental results in Section 5 reveal that the computation time required to solve the LP problem with such a large size using classic algorithms (e.g., the simplex method or the interior point algorithm) is prohibitively high (e.g., when $K \geq 100$).

In the next section, by leveraging the structural characteristics of the GTS constraints, we will design optimization decomposition techniques to tackle the problem. By doing so, we can effectively reduce both the number of decision variables and Geo-Ind constraints to $O(K)$, providing a much more manageable size for the problem.

4.3 The Column Generation (CG) Algorithm

Recall that each decision vector $\mathbf{z}_k = [z_{1,k}, ..., z_{K,k}]^\top$ $(k = 1, ..., K)$ denotes the probabilities of selecting v_k as the obfuscated location given the real locations $v_1, ..., v_K$. If the obfuscation matrix $\mathbf{Z}$ is reshaped to a vector $\mathbf{z} = [\mathbf{z}_1^\top, ..., \mathbf{z}_K^\top]^\top$, then the constraint matrix of $\mathbf{z}$ consists of two parts, as depicted in Fig. 8(a): (1) *Joint constraints*, i.e., the probability unit measure (Equ. (3)), link all $\mathbf{z}_1, ..., \mathbf{z}_K$ together; (2) *Disjoint constraints*, including the constraints of peer locations (Equ. (2)) and Geo-Ind (Equ. (5)), are decomposed to a set of matrix blocks. Each block k contains the constraints of $\mathbf{z}_k$.

As Fig. 8(b) shows, the feasible region of each $\mathbf{z}_k$ defined by each block matrix k is a polyhedron Λ_k. Replacing each $\mathbf{z}_k \in \Lambda_k$ by a convex combination of Λ_k's extreme points, we can obtain the *Dantzig Wolfe (DW) formulation* of GTS, of which the decision variables are the weights $\boldsymbol{\lambda}$ assigned to the extreme points of the polyhedrons $\Lambda_1, ..., \Lambda_K$ (the detailed DW formulation can be found in Section 8.1 in the Supplementary file). Although the number of the decision variables in the DW formulation is exponential with respect to K (as each polyhedron might have an exponential number of extreme points), a majority of its extreme points are not visited during the simplex method search. In this regard, we

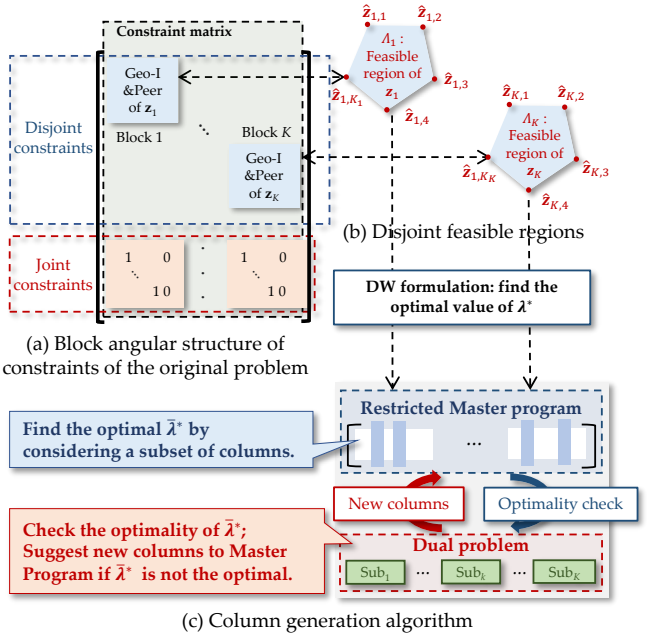


Figure 8: Structure of the CG algorithm.

only need to search a subset of extreme points (columns) to find the optimal solution using the CG algorithm [25].

More precisely, as Fig. 8(c) shows, the CG algorithm starts with a *restricted master program (RMP)* by considering only a subset of columns of the original DW constraint matrix, and then use its *dual problem (DRMP)* to test the optimality of RMP's solution $\bar{\lambda}^*$. DRMP can be further decomposed into a set of *subproblems*; if $\bar{\lambda}^*$ hasn't reached the optimal of the original DW formulation, the subproblems can identify new columns to add to the master program to improve the solution. This process is repeated until $\bar{\lambda}^*$ converges to the optimal. A more detailed description of CG can be found in Appendix.

Complexity of CG. The number of decision variables and the number of constraints in both RMP and DRMP are $O(K)$, which can be efficiently solved using the simplex algorithm. The only remaining question is how many iterations are needed to converge $\bar{\lambda}^*$ to the optimal. This aspect will be further discussed in our experiment in Fig. 13 in Section 5. In the experiment, to check how close $\bar{\lambda}^*$ can achieve the optimal, we compare $\bar{\lambda}^*$ with a lower bound of the DW's optimal provided by Theorem 8.1 (Details can be found in Section 8.2 in the Supplementary file).

5 PERFORMANCE EVALUATION

In this section, we assess the performance of our fine-grained geo-obfuscation algorithm, labeled as "FineGeo" for brevity.

In Section 5.1, we first conduct a large-scale trace-driven simulation using the map and the vehicle trajectory dataset of Shenzhen city. In Section 5.2, we carry out two real-world experiments in a building and a campus using the SC geo-obfuscation prototype we developed.

Benchmarks. We compare FineGeo against the following three benchmarks, all of which use ϵ-Geo-Ind as the privacy criterion:

(i) *Grid-based geo-obfuscation (labeled as "Grid")* [12]. "Grid" discretizes the location field into a grid map, rendering the locations of workers indistinguishable within individual grid cells. Similar to FineGeo, Grid's objective is to minimize the expected quality loss by employing an LP framework, while abstaining from any constraint reduction or optimization decomposition methods. Consequently, Grid has to construct the obfuscation matrix based on a less finely-grained location set. In particular, our initial investigation reveals that when the number of locations exceeds 50, it is hard for LP to calculate the obfuscation matrix efficiently. As such, we let "Grid" discretize the target regions into 50 grid cells, denoted as "Grid-50". For the sake of comparison, we also consider the scenario where "Grid" discretizes the target regions into 40 grid cells, referred to as "Grid-40".

(ii) *Laplacian noise (labeled as "Laplace")* [10], where the obfuscated location of each real location v_i follows a polar Laplace distribution $z_{i,k} \propto e^{-\epsilon c_{i,k}}$ ($v_k \in \mathcal{V}$). The Laplacian noise naturally satisfies ϵ-Geo-Ind [10] without using LP, and its time complexity is significantly lower than LP-based methods. As such, Laplace can be developed on a fine-grained location set like FineGeo. However, in Laplace, users' mobility is considered only on a 2-dimensional plane, without considering any mobility restrictions they may have. Additionally, it does not optimize the distribution of obfuscation locations to minimize quality loss.

(iii) *Vehicle-based geo-obfuscation (labeled as "VGO")* [8], which is our prior work aiming to protect the location privacy of vehicles in SC. VGO takes into account the network-constrained mobility features of the vehicles and employs LP to minimize quality loss. However, when selecting obfuscated locations, it does not impose any constraints to mitigate the resulting quality loss for SC.

Metrics. We test the following metrics of the different geo-obfuscation methods:

(1) *Computation time* to execute algorithms. The experiments are performed on a server with two Intel Xeon Silver 4309Y CPUs, each with 8 cores. The server runs Rocky Linux 8.6. For FineGeo, we measure the computation time of peer location searching (introduced in Section 4.1) and the CG algorithm (introduced in Section 4.3), including the restricted master program (RMP), its dual problem DRMP, and the subproblems.

(2) *Quality loss*, measured by the average estimation error of traveling distance ($\Delta(\mathbf{Z})$ defined by Equ. (6)). We didn't select "traveling time" as the metric as it is highly impacted by factors other than algorithms, e.g., the moving speed of workers.

(3) *Expected inference error (EIE)*, which describes the expected distortion from the estimated location $\hat{v}$ by the attacker (using Beyesian inference attack [14]) to the actual location v_i. Higher EIE implies a higher privacy level achieved by geo-obfuscation.

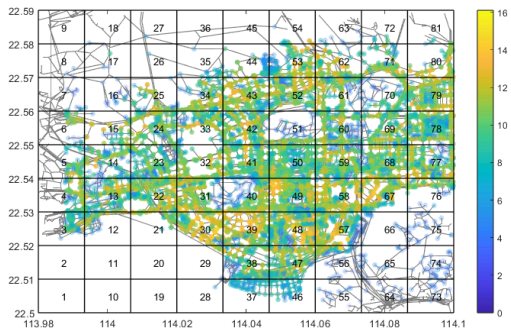
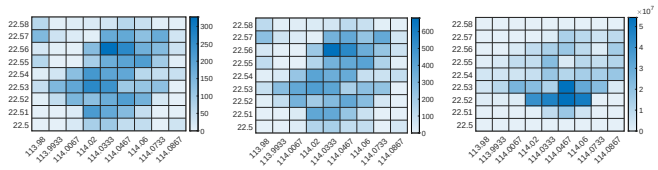


Figure 9: The road map of Shenzhen (including the heap map of the GPS record density).



(a) Node density (b) Edge density (c) GPS record density

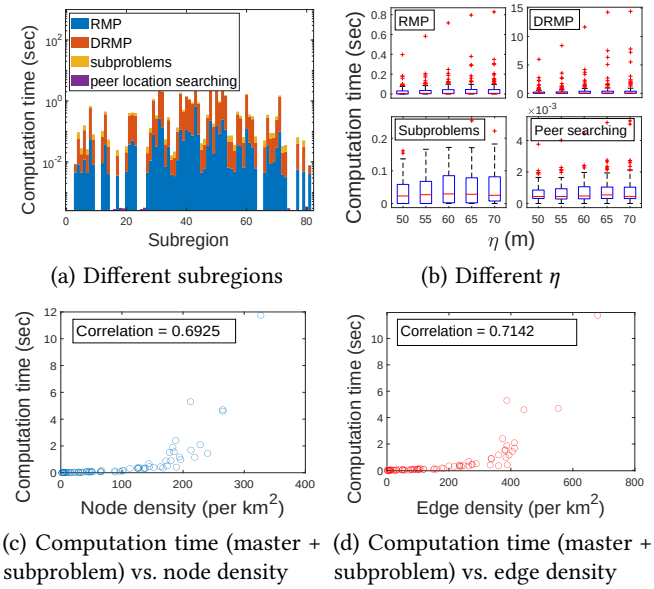
Figure 10: Dataset statistics.

5.1 Trace-Driven Simulation

5.1.1 Dataset. As Fig. 9 shows, we select the *Futian district in Shenzhen* city, China, as the target region of SC. The graph model of the district is extracted by OpenStreetMap [28], which provides fine-grained location (node) and road (edge) information of the city, i.e., the average distance between adjacent locations (nodes) is 184.9m. To crop the road map data, we used a bounding box with a south-west corner coordinate of *(latitude = 22.50, longitude = 113.98)* and a north-east corner coordinate of *(latitude = 22.59, longitude = 114.10)*, as per the municipal information of Shenzhen. We partition the whole target region into 81 subregions indexed by $\{1, 2, ..., 81\}$. Fig. 10(a)(b) show the heat map of the node density (the number of discrete locations (nodes) per square kilometer) and the edge density (the number of edges per square kilometer) of the mobility graph across the 81 subregions.

The vehicle trajectory dataset used for the simulation contains the timestamps, GPS positions, and velocities of around 27,996 vehicles in Shenzhen [29], including 15,610 taxicabs and 12,386 customized transit service vehicles in Dada Car corporation. We use taxicabs and transit service vehicles as a proxy of crowdsourcing workers by assuming that workers' locations follow the same probability distribution with taxicabs and transit service vehicles in the road network. Fig. 10(c) shows the heat map of the vehicles' GPS records across the 81 subregions.

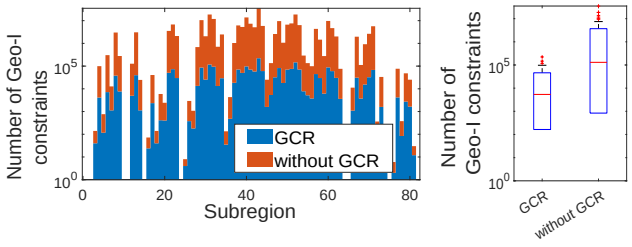
5.1.2 Time efficiency of FineGeo. We first evaluate FineGeo's computation time, which includes the four components: peer location searching, RMP, DRMP, and subproblems. This evaluation covers the 81 subregions; the results are presented in Fig. 11(a). The figure shows that the average computation time of peer location searching, RMP, DRMP, and subproblems are 0.0008 seconds, 0.0477 seconds, 0.6244 seconds, and 0.0488 seconds, respectively, and the total computation time to create an obfuscation matrix is 0.7218 seconds.



(a) Different subregions (b) Different η

(c) Computation time (master + subproblem) vs. node density (d) Computation time (master + subproblem) vs. edge density

Figure 11: Computation time of FineGeo.

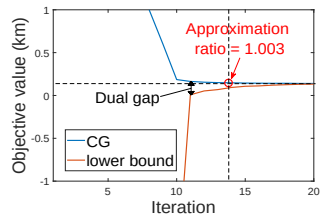


(a) Different subregions (b) Box plot

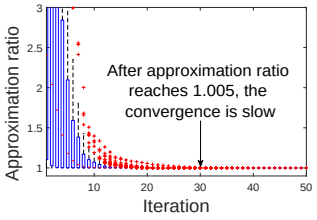
Figure 12: Number of Geo-Ind constraints with and without GCR.

Fig. 11(b) displays the computation time of FineGeo's four components given different values of η. Fig. 20(c) (Fig. 20(d), respectively) displays the correlation between the total computation time of FineGeo and the node density (edge density, respectively). Our findings indicate that the total computation time of FineGeo is positively correlated with η, node density, and edge density. For instance, the correlation between the total computation time and the node density (edge density, respectively) is 0.6925 (0.7142, respectively). This is because a higher value of η, node density, or edge density causes a larger peer location set for each obfuscated location. This, in turn, requires more pairs of real locations to satisfy the Geo-Ind constraints and leads to higher computation time.

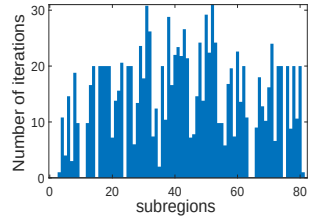
To demonstrate the time efficiency improved by FineGeo, we use the MATLAB LP toolbox `linprog` [30] to solve GTS directly. Specifically, we employ the algorithms `dual-simplex` and `interior-point`, both of which terminate without reaching the optimal solution due to the large size of GTS.



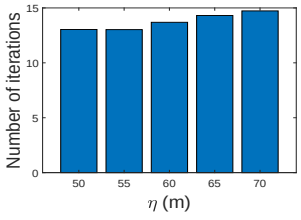
(a) CG convergence in one trace



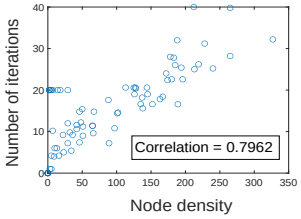
(b) Convergence statistics



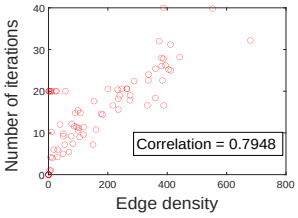
(c) Number of iterations in different subregions



(d) Number of iterations given different η



(e) Number of iterations vs. node density



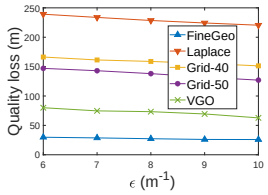
(f) Number of iterations vs. edge density

Figure 13: Convergence of CG.

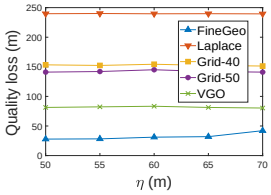
5.1.3 Time efficiency improved by constraint reduction and decomposition. As introduced in Section 4.2 and Section 4.3, FineGeo improves time efficiency in solving GTS via the two steps: *Geo-Ind constraint reduction (label as "GCR")* and the *column generation* algorithm (label as "*CG*"). This part tests how these two steps improve time efficiency.

Fig. 12(a)(b) presents a comparison between the numbers of Geo-Ind constraints in the LP formulation (Equ. (7)-(8)) with and without GCR. From the figures, we find that GCR reduces the number of Geo-Ind constraints of LP by 99.04%. Note that, on average, the number of neighboring peers for each obfuscated location in the mobility graphs is 0.8819 times higher than the number of nodes in the graphs, indicating that there are $O(K)$ neighboring peers (each neighboring peer corresponds to two Geo-Ind constraints) for each obfuscated location. Hence, the simulation result demonstrates that GCR reduces the number of Geo-Ind constraints in LP from cubic to approximately quadratic with respect to the number of locations, which is consistent with the complexity analysis in Section 4.2.

We proceed by evaluating the time efficiency of CG. Recall that, in each iteration of CG, RMP, DRMP, and all the subproblems have only $O(K)$ decision variables and $O(K)$ linear constraints, which can be solved quickly using simplex methods. Therefore, the remaining question is how many iterations are needed for CG to converge to a near-optimal solution. Fig. 13(a)(b) presents an example of the convergence of CG, where we compare the quality



(a) Quality loss vs. ϵ



(b) Quality loss vs. η

Figure 14: QL of the different algorithms.

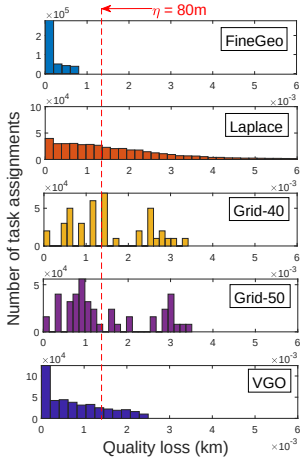


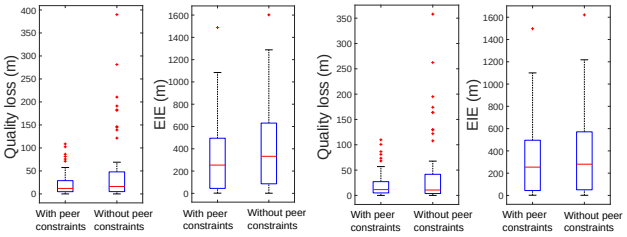
Figure 15: QL distribution of the different algorithms.

loss achieved by CG in one trace ($\epsilon = 10\text{m}^{-1}$ and $\eta = 80$m) with the lower bound derived by Equ. (27) over iterations. Notably, the dual gap between CG's quality loss and the lower bound contains the minimum quality loss. Fig. 13(a) reveals that CG can attain a near-optimal solution at the 14th iteration, with the *approximation ratio* (i.e., the ratio of the quality loss attained by CG and the quality loss's lower bound) reaching 1.003.

Fig. 13(b) shows the boxplot of the CG convergence for 81 different target regions in Shenzhen. This plot reveals a long tail in the convergence of the CG algorithm, as evidenced by its slow decrease in the approximation ratio after the ratio reaches 1.005. To ensure time efficiency, in the following simulation, we set an acceptable approximation threshold of $\xi = 1.005$, prompting the algorithm to terminate once the ratio of quality loss and its lower bound reaches ξ. Notably, setting $\xi = 1.005$ results in an average of 13.758 iterations required for CG termination. Fig. 13(c) shows the number of iterations to terminate CG across 81 subregions. Remarkably, the average number of iterations to terminate CG of all the subregions is 13.758 when $\eta = 50$m.

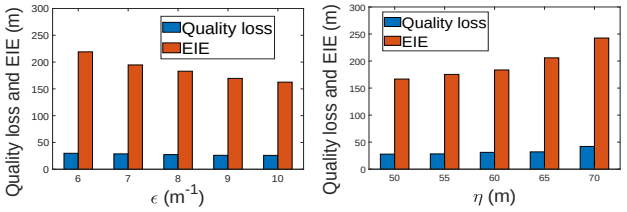
We then test how the value of η impacts the convergence of CG. Fig. 13(d) shows the average number of iterations in the 81 subregions increase with the increase of η (from 50m to 70m). Fig. 13(e) (Fig. 13(f), respectively) illustrates the correlation between the number of iterations required to terminate CG and the node density (edge density, respectively). The figures show a positive correlation between the number of iterations and both node density and edge density, with correlation coefficients of 0.7962 and 0.7948, respectively.

5.1.4 Comparison of quality loss with the benchmarks. We created 5,000 tasks and 20,000 workers that are uniformly deployed across 81 subregions. Fig. 14(a)(b) compare the average quality loss of the five algorithms in the 81 subregions, with various ϵ and η, respectively. Both figures demonstrate that FineGeo achieves significantly lower quality loss than Grid-40, Grid-50, Laplace, and VGO. The quality loss of FineGeo is in the range of [20m, 40m], and on average, it is 87.98%, 82.63%, 79.95%, and 61.76% lower than that of Laplace,



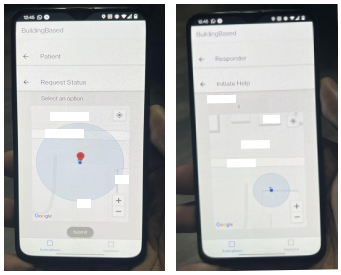
(a) $\epsilon = 6\text{m}^{-1}$ (b) $\epsilon = 10\text{m}^{-1}$

Figure 16: Privacy loss caused by the peer location constraint.



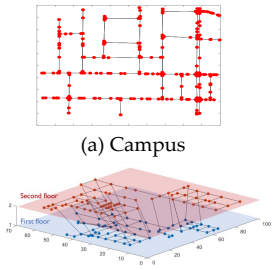
(a) Different ϵ (b) Different η

Figure 17: QL and EIE of FineGeo.



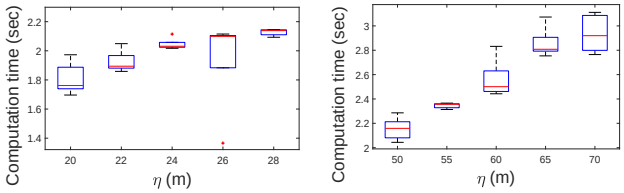
(a) Patient (b) Responder

Figure 18: User interface of the prototype.



(a) Campus (a) Building

Figure 19: Real-world mobility graph.



(a) Building (b) Campus

Figure 20: Computation time with different η.

Grid-40, Grid-50, and VGO, respectively. For example, assuming 30 kilometers/hour driving speed, the quality loss of FineGeo results in only up to 4 seconds estimation error of traveling time for vehicles. Such estimation error of traveling time should be acceptable for time-sensitive SC applications like CPR assignments.

Notably, Laplace has the highest quality loss since it assumes workers can move freely without considering their mobility restrictions. The quality loss of Grid-40 and Grid-50 is higher than FineGeo as both Grid-40 and Grid-50 are developed based on coarse-grained location sets, which cannot accurately estimate the traveling costs in SC. Not surprisingly, the quality loss of Grid-40 is higher than that of Grid-50, since lower granularity location representation causes higher quality loss. Finally, FineGeo outperforms VGO since, besides minimizing the quality loss, FineGeo additionally sets the peer location constraint to enforce the estimation error of all the traveling costs to be bounded by η.

Fig. 14(a) shows that all five algorithms have a lower quality loss when ϵ is higher. According to the definition of ϵ-Geo-Ind (Equ. (4)), a higher privacy budget ϵ enforces less restriction on the obfuscated location probability, allowing the algorithms to select the obfuscated location near to the real location, ultimately leading to a lower quality loss. Fig. 14(b) shows that the quality loss of FineGeo increases with the increase of η. It is because a higher η allows a larger obfuscation range, which, on average, introduces a higher estimation error of traveling distance.

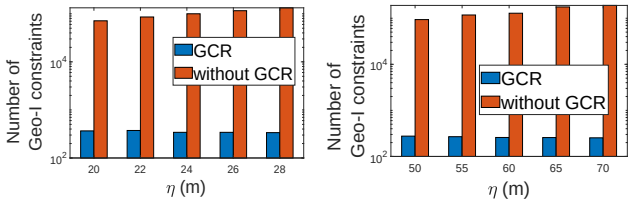
By setting $\eta = 80\text{m}$ and $\epsilon = 10\text{m}^{-1}$, we depict the quality loss distribution of the five algorithms in Fig. 15. From the figure, we find that only FineGeo has its quality no higher than the threshold η, because FineGeo enforces the obfuscation range to the peer location set of which the quality loss is upper bounded by η. In contrast, the other four algorithms don't have such a constraint.

5.1.5 Pivacy loss caused by peer location constraint. As discussed in Fig. 14 and Fig. 15, FineGeo can achieve the lowest quality loss since it restricts the selection of obfuscated locations to the peer location set. The reduced obfuscation range, however, might sacrifice the achieved privacy levels. To further evaluate how much privacy and quality loss are reduced by the peer location constraints, in Fig. 16(a)(b), we compare the *expected inference error (EIE)* and the quality loss of FineGeo in the 81 subregions with and without peer location constraints when $\epsilon = 6\text{m}^{-1}$ and $\epsilon = 10\text{m}^{-1}$, respectively. The two figures show that, when applying the peer location constraints to the obfuscated location selection, the EIE of FineGeo is reduced by 14.01% and 8.13% when $\epsilon = 6\text{m}^{-1}$ and $\epsilon = 10\text{m}^{-1}$, and quality loss is reduced by 49.77% and 43.83%, respectively. This indicates that incorporating peer location constraints significantly improves the accuracy of travel cost estimation while maintaining a high level of privacy. As illustrated in Fig. 4 in Section 3.1.2, peer locations exhibit similar travel costs to the exact location of the worker, which reduces the quality loss significantly. On the other hand, peer locations remain sufficiently distant from the worker's actual position, leading to a sufficiently high EIE by attackers.

Additionally, we conducted a comparison of FineGeo's performance with varying values of η and ϵ, as shown in Fig. 17(a) and Fig. (b) respectively. The two figures reveal that FineGeo achieves an average EIE of 190.3 meters, which is 5.36 times higher than its quality loss. This substantial location inference error makes it challenging for potential attackers to accurately track the precise location of the SC worker.

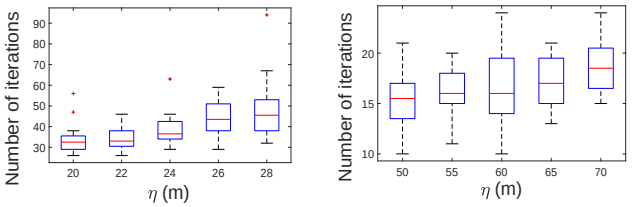
5.2 Real-World Experiment

In this part, we carried out a pilot study to test the performance of FineGeo in real-world scenarios within the UNT main campus (4,9 square kilometers of land area) and the building "Discovery Park" (0.055 square kilometers of land area) using the prototype we



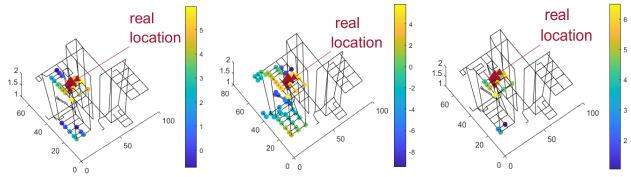
(a) Building (b) Campus

Figure 21: Number of Geo-Ind constraints reduced by GCR.



(a) Building (b) Campus

Figure 22: Convergence of CG.



(a) $\epsilon = 35\text{m}^{-1}, \eta$ =28m (b) $\epsilon = 15\text{m}^{-1}, \eta$ =28m (c) $\epsilon = 15\text{m}^{-1}, \eta$ =20m

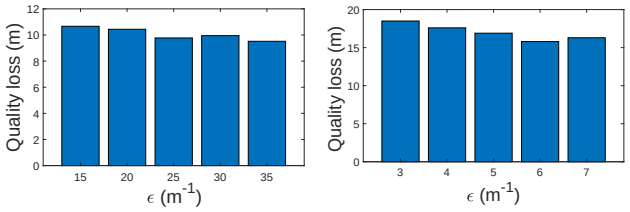
Figure 23: Obfuscation distribution given different ϵ and η. *In the figures, color value = $\log(1000 z_{i,k})$

developed. The prototype includes the main functions of SC like geo-obfuscation and SC task assignment.

More specifically, on the user side, we developed an Android smartphone app based on the Google map API. Fig. 18(a) and (b) show the user interfaces of the Android APP. As shown in Fig. 18(a), a requester (or patient) can upload his/her task with the task location specified. Workers (or participants) who opted in can receive the nearby tasks and report their obfuscated location to the server, as shown in Fig. 18(b). After receiving the workers' reported location, the server sends a list of task requests to the worker, along with the traveling cost estimated based on the worker's obfuscated location. By selecting a task request, a route will be displayed on the map to navigate this worker to the selected task location.

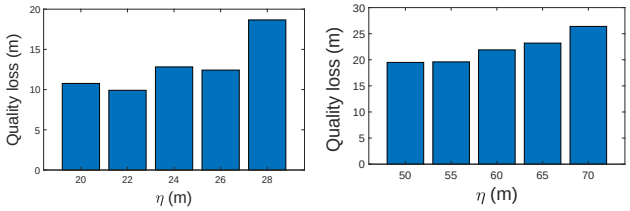
Fig. 19(a) displays the mobility graph of the campus extracted from OpenStreetMap [28]. The average distance between adjacent locations (nodes) is 28.2 meters. Fig. 19(b) shows the mobility graph of the building, which was extracted from the department's two-floor maps. In this graph, the intersections of the passageways and staircases are considered "nodes", and the arcs connecting the junction locations of the passageways are considered "edges". The average distance between nodes in this graph is 7.71 meters.

5.2.1 Time efficiency of FineGeo. Fig. 20(a)(b) shows the computation time (including peer location searching, RMP, DRMP, and



(a) Building (Exp1) (b) Campus (Exp2)

Figure 24: QL of FineGeo with different ϵ.



(a) Building (Exp3) (b) Campus (Exp4)

Figure 25: QL loss of FineGeo with different η.

subproblems) of FineGeo for the two target regions with different η values. Our results show that the average computation time for the building and the campuses is 2.04 seconds and 2.53 seconds, respectively. As expected, the computation time of FineGeo in both figures increases with the increase of η, as higher η introduces more Geo-Ind constraints in LP. This observation is consistent with the simulation results in Fig. 11(b).

Fig. 21(a)(b) compare the numbers of Geo-Ind constraints in the LP formulation (Equ. (7)-(8)) with and without GCR in the two target regions. The figures show that, on average, GCR reduces the number of constraints by 99.71% and 99.85% for the building and the campus, respectively.

Fig. 22(a)(b) show the number of iterations needed to terminate CG for the building and the campus, respectively, with the different η values. Like the simulation, we set the acceptable approximation ratio $\xi = 1.005$. The figures show that it takes 16.8 iterations (39.9 iterations, respectively) to terminate the CG algorithm when the target region is the building (the campus, respectively). Moreover, in both figures, CG converges more slowly when η is higher, which is consistent with the simulation results in Fig. 11(b).

5.2.2 Quality loss of FineGeo. We conducted four experiments to evaluate the quality loss of FineGeo for the two target regions.
Exp1 (building): η=20m and ϵ is increased from 15m^{-1} to 35m^{-1}.
Exp2 (campus): η=50m and ϵ is increased from 3m^{-1} to 7m^{-1}.
Exp3 (building): $\epsilon = 15\text{m}^{-1}$ and η is increased η from 20m to 28m.
Exp4 (campus): $\epsilon = 3\text{m}^{-1}$ and η is increased from 50m to 70m.
For each experiment, we collected 1,000 location reports from the participants.

Fig. 24(a)(b) and Fig. 25(a)(b) show the results of Exp 1–Exp4. The figures indicate that the quality loss of FineGeo increases with an increase in η and decreases with an increase in ϵ. These findings are consistent with the simulation results shown in Fig. 14(a)(b). The average quality loss of FineGeo in the building and the campus

is 11.49 meters and 19.57 meters, respectively. This results in 3.48 seconds and 5.93 seconds estimation errors of traveling time for pedestrians, assuming a running speed of 3.3 meters/second [31], which are acceptable for applications like CPR assignment .

The heat maps in Fig. 23(a)(b)(c) illustrate how the obfuscated location distribution of FineGeo is impacted by ϵ and η for the target region building. The figures show that when ϵ is higher (by comparing Fig. 23(a)(b)), or η is lower (by comparing Fig. 23(b)(c)), the obfuscated location has a higher probability of being close to the real location, ultimately resulting in a lower quality loss.

6 RELATED WORKS

The study of location privacy can date back to almost two decades ago when Gruteser and Grunwald [32] introduced the notion of *location k-anonymity*. This notion was then extended to *l-diversity*, meaning that a user's location cannot be distinguished from other nearby $l-1$ locations [14]. However, *l*-diversity oversimplifies the threat model by assuming that all the dummy locations are equally probable to be the real location from the attacker's perspective, making it susceptible to different inference attacks [8, 10, 14]. In recent years, Andr'es *et al.* [10] proposed a formal privacy concept, known as *Geo-Ind*, which is based on the statistical notion of *differential privacy (DP)*. This work has spurred the development of several new geo-obfuscation strategies [9, 10, 12, 14]. For instance, Andrés *et al.* [10] not only introduced the Geo-Ind notion but also developed a geo-obfuscation technique that adds noise drawn from a polar Laplacian distribution to the actual location to achieve Geo-Ind.

On the flip side, geo-obfuscation inevitably leads to errors in users' reported locations and loss of quality in LBS. To address this issue, researchers have been exploring the trade-off between privacy and quality of service. For instance, Bordenabe *et al.* [18] proposed an optimization framework for geo-obfuscation to minimize the quality loss for each user while adhering to the Geo-Ind restrictions. Chatzikokolakis *et al.* [21] defined privacy mass over the points of interest on the plane and set the privacy budget ϵ of Geo-Ind for a location based on the local features of each area. Wang *et al.* [12] considered the quality loss incurred by all users as a whole and proposed a location privacy-preserving task assignment algorithm to minimize the total traveling cost.

Most of the existing geo-obfuscation works adopt an LP framework, which has relatively high time complexity if no constraint reduction/decomposition is applied. To enhance computational efficiency, as illustrated in Fig. 26, those works have to discretize the location field of geo-obfuscation with low granularity [12, 14, 19–22]. For instance, recent works such as [12, 14] discretize the workers' location field into a grid map, where locations are indistinguishable in each grid (e.g., the grid cell size g ranges from 766m×766m [14] to 1.0 km×1.0 km [12]). In such cases, the estimation error of traveling time (or *cost* for simplicity) caused by discretization alone can be as high as $\sqrt{2}g$, which is unacceptable in time-sensitive SC applications such as CPR worker assignment. Delaying the initiation of CPR by even a minute can decrease the probability of survival by 7-10% [23]. Although some works like [8, 25, 33, 34] have designed geo-obfuscation at a higher granularity, they still have to restrict the target region to a small area, such as 0.055km^2 [8], which is insufficient for SC applications.

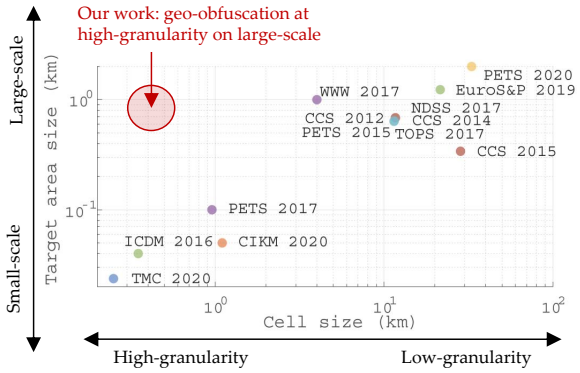


Figure 26: Comparison of location precision and target region scale of the existing works, including:
TMC 2020 [8], PETS 2020 [35], CIKM 2020 [25], EuroS&P 2019 [22], WWW 2017 [12], TOPS 2017 [36], PETS 2017 [33], CCS 2012 [9], NDSS 2017 [14], ICDM 2016 [37], CCS 2015 [20], CCS 2014 [11].

The works most relevant to this work are our recent work [8, 25], in which we aimed to protect vehicles' location privacy in the road network. Both [8, 25] consider workers' mobility constraints and discretize vehicle locations with relatively high granularity. However, these approaches cannot be directly applied to scenarios such as buildings, where the location field requires even finer discretization and additional mobility restrictions. Moreover, while the works in [8, 25] aim to maximize the privacy criterion *expected inference errors (EIE)* and minimize quality loss, but with no guarantee for the quality of single task assignment, making them unsuitable for time-sensitive SC.

7 CONCLUSIONS

In this paper, we have developed a new geo-obfuscation approach to protect workers' location privacy in time-sensitive SC. We enforce the workers' obfuscated location to be within his/her "peer location set" such that the traveling cost estimation errors of each task are bounded by a threshold. We then formulated a new obfuscation generation problem, called GTS, and applied the Geo-Ind constraint reduction and the DW optimization decomposition to solve GTS in a time-efficient manner. The experimental results from simulation and real-world tests have demonstrated the effectiveness of our approach.

We envision several promising directions to continue this research. Firstly, by considering the mobility features of workers in different environments, we will think like an adversary, and develop new threat models that can possibly use those mobility features to infer workers' exact locations. The countermeasures of the new attacks also need to be developed. Secondly, in our current framework, we still consider homogeneous workers, where a single graph is sufficient to describe workers' mobility. In reality, the workers might be heterogeneous, e.g., a mixture of pedestrians and vehicles. Then, how to model the mobility of heterogeneous workers using multiple graphs in geo-obfuscation is another problem to address.

REFERENCES

[1] L. Kazemi and C. Shahabi. Geocrowd: Enabling query answering with spatial crowdsourcing. In *Proc. of ACM SIGSPATIAL*, pages 189–198, 2012.

[2] Leyla Kazemi, Cyrus Shahabi, and Lei Chen. Geotrucrowd: Trustworthy query answering with spatial crowdsourcing. In *Proc. of ACM International Conference on Advances in Geographic Information Systems (SIGSPATIAL)*, pages 314–323, 2013.

[3] Yongxin Tong, Lei Chen, and Cyrus Shahabi. Spatial crowdsourcing: Challenges, techniques, and applications. *VLDB Endow.*, 10(12):1988–1991, August 2017.

[4] Uber. https://www.uber.com/, 2022. Accessed: 2022-07-27.

[5] Gigwalk. https://www.gigwalk.com/, 2022. Accessed: 2022-07-27.

[6] H. To, G. Ghinita, L. Fan, and C. Shahabi. Differentially private location protection for worker datasets in spatial crowdsourcing. *IEEE TMC*, pages 934–949, 2017.

[7] Pulsepoint. https://www.pulsepoint.org/, 2022. Accessed: 2022-07-21.

[8] C. Qiu, A. C. Squicciarini, C. Pang, N. Wang, and B. Wu. Location privacy protection in vehicle-based spatial crowdsourcing via geo-indistinguishability. *IEEE Transactions on Mobile Computing*, pages 1–1, 2020.

[9] R. Shokri, G. Theodorakopoulos, C. Troncoso, J. Hubaux, and J. L. Boudec. Protecting location privacy: Optimal strategy against localization attacks. In *Proc. of ACM CCS*, pages 617–627, 2012.

[10] M. Andrés et al. Geo-indistinguishability: Differential privacy for location-based systems. In *Proc. of ACM CCS*, pages 901–914, 2013.

[11] K. Fawaz and K. G. Shin. Location privacy protection for smartphone users. In *Proc. of ACM CCS*, pages 239–250. ACM, 2014.

[12] 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 *Proc. of WWW*, 2017.

[13] Hengzhi Wang, En Wang, Yongjian Yang, Jie Wu, and Falko Dressler. Privacy-preserving online task assignment in spatial crowdsourcing: A graph-based approach. In *IEEE INFOCOM 2022 - IEEE Conference on Computer Communications*, pages 570–579, 2022.

[14] L. Yu, L. Liu, and C. Pu. Dynamic differential location privacy with personalized error bounds. In *Proc. of ACM NDSS*, 2017.

[15] G. Ghinita et al. Private queries in location based services: Anonymizers are not necessary. In *Proc. of ACM SIGMOD*, 2008.

[16] C. Qiu and A. C. Squicciarini. Location privacy protection in vehicle-based spatial crowdsourcing via geo-indistinguishability. In *Proc. of IEEE ICDCS*, 2019.

[17] H. To, C. Shahabi, and L. Xiong. Privacy-preserving online task assignment in spatial crowdsourcing with untrusted server. In *Proc. of IEEE ICDE*, 2018.

[18] N. Bordenabe, K. Chatzikokolakis, and C. Palamidessi. Optimal geo-indistinguishable mechanisms for location privacy. In *Proc. of ACM CCS*, 2014.

[19] R. Shokri. Privacy games: Optimal user-centric data obfuscation. *Proceedings on Privacy Enhancing Technologies*, 2015(2):299 – 315, 2015.

[20] Yonghui Xiao and Li Xiong. Protecting locations with differential privacy under temporal correlations. In *Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security*. ACM, oct 2015.

[21] K. Chatzikokolakis, C. Palamidessi, and M. Stronati. Constructing elastic distinguishability metrics for location privacy. *PoPETs*, 2015:156–170, 2015.

[22] Simon Oya, Carmela Troncoso, and Fernando Pérez-González. Rethinking location privacy for unknown mobility behaviors. In *2019 IEEE European Symposium on Security and Privacy (EuroS&P)*, pages 416–431, 2019.

[23] Huang LH, Ho YN, Tsai MT, Wu WT, and Cheng FJ. Response Time Threshold for Predicting Outcomes of Patients with Out-of-Hospital Cardiac Arrest. *Emerg Med Int.*, 02 2021.

[24] F. S. Hillier. *Linear and Nonlinear Programming*. Stanford University, 2008.

[25] C. Qiu, A. C. Squicciarini, Z. Li, C. Pang, and L. Yan. Time-efficient geo-obfuscation to protect worker location privacy over road networks in spatial crowdsourcing. In *Proc. of ACM CIKM*, 2020.

[26] Chenxi Qiu, Li Yan, Anna Squicciarini, Juanjuan Zhao, Chengzhong Xu, and Primal Pappachan. Trafficadaptor: An adaptive obfuscation strategy for vehicle location privacy against traffic flow aware attacks. In *Proceedings of the 30th International Conference on Advances in Geographic Information Systems*, SIGSPATIAL '22, New York, NY, USA, 2022. Association for Computing Machinery.

[27] Harsh Bhasin. *Algorithms: Design and Analysis*. Oxford Univ Press, 2015.

[28] Openstreetmap. https://www.openstreetmap.org/, 2022. Accessed: 2022-07-27.

[29] Li Yan and et al. Catcharger: Deploying in-motion wireless chargers in a metropolitan road network via categorization and clustering of vehicle traffic. *IEEE Internet of Things Journal*, pages 1–1, 2021.

[30] MATLAB. https://www.mathworks.com/products/matlab.html, 2019. Accessed: 2019-07-22.

[31] J.B. Morin, P. Samozino, K. Zameziati, and A. Belli. Effects of altered stride frequency and contact time on leg-spring behavior in human running. *Journal of Biomechanics*, 40(15):3341–3348, 2007.

[32] M. Gruteser and D. Grunwald. Anonymous usage of location-based services through spatial and temporal cloaking. In *Proc. of ACM MobiSys*, 2003.

[33] Raed Al-Dhubhani and Jonathan M. Cazalas. An adaptive geo-indistinguishability mechanism for continuous lbs queries. *Wirel. Netw.*, 24(8):3221–3239, nov 2018.

[34] Leye Wang, Daqing Zhang, Dingqi Yang, Brian Lim, and Xiaojuan Ma. Differential location privacy for sparse mobile crowdsensing. pages 1257–1262, 12 2016.

[35] Ricardo Mendes, Mariana Cunha, and Joao Vilela. Impact of frequency of location reports on the privacy level of geo-indistinguishability. *Proceedings on Privacy Enhancing Technologies*, 2020:379–396, 04 2020.

[36] Reza Shokri, George Theodorakopoulos, and Carmela Troncoso. Privacy games along location traces: A game-theoretic framework for optimizing location privacy. *ACM Trans. Priv. Secur.*, 19(4), dec 2016.

[37] Leye Wang, Daqing Zhang, Dingqi Yang, Brian Y. Lim, and Xiaojuan Ma. Differential location privacy for sparse mobile crowdsensing. In *2016 IEEE 16th International Conference on Data Mining (ICDM)*, pages 1257–1262, 2016.

8 APPENDIX

8.1 The Column Generation Algorithm

8.1.1 Dantzig-Wolfe (DW) formulation. As Fig. 8(b) shows, the feasible region of each $\mathbf{z}_k$ is constrained by Equ. (2) (5) is a polyhedron Λ_k, which has L_k extreme points $\mathcal{Z}_k = \{\hat{\mathbf{z}}_{k,1}, ..., \hat{\mathbf{z}}_{k,L_k}\}$. Any feasible decision vector $\mathbf{z}_k \in \Lambda_k$ can be represented as a convex combination of Λ_k's extreme points [24]

$$\mathbf{z}_k = \sum_{\hat{\mathbf{z}}_{k,r} \in \mathcal{Z}_k} \lambda_{k,r} \hat{\mathbf{z}}_{k,r}, \tag{15}$$

where $\sum_{\hat{\mathbf{z}}_{k,r} \in \mathcal{Z}_k} \lambda_{k,r} = 1$ and $\lambda_{k,r} \geq 0$.

Replacing each $\mathbf{z}_k$ by $\sum_{\hat{\mathbf{z}}_{k,r} \in \mathcal{Z}_k} \lambda_{k,r} \hat{\mathbf{z}}_{k,r}$, we can rewrite GTS defined in Equ. (7)–(8) as the following *DW formulation*:

$$\min \quad \sum_{k=1}^{K} \sum_{\hat{\mathbf{z}}_{k,r} \in \mathcal{Z}_k} \mathbf{c}_k^\top \hat{\mathbf{z}}_{k,r} \lambda_{k,r} \tag{16}$$

$$\text{s.t.} \quad \sum_{k=1}^{K} \sum_{\hat{\mathbf{z}}_{k,r} \in \mathcal{Z}_k} \lambda_{k,r} \hat{\mathbf{z}}_{k,r} = \mathbf{1}, \tag{17}$$

$$\sum_{\hat{\mathbf{z}}_{k,r} \in \mathcal{Z}_k} \lambda_{k,r} = 1, \forall k \text{ and } \lambda_{k,r} \geq 0, \forall k, r \tag{18}$$

The decision variables in the aforementioned DW formulation are $\lambda_{k,r}$ ($k = 1, ..., K$ and $r = 1, ..., L_k$). Each $\lambda_{k,r}$ corresponds to an extreme point $\hat{\mathbf{z}}_{k,r}$ of the polyhedron Λ_k.

Although the DW formulation has only $O(K)$ constraints, the number of decision variables is $\sum_{k=1}^{K} L_k$, i.e., the total number of extreme points in all the polyhedrons $\Lambda_1, ..., \Lambda_K$. Note that each L_k can be exponential with respect to K, thereby making the DW formulation computationally demanding if we employ classic LP algorithms to solve it directly. Nonetheless, a majority of the extreme points in DW are not visited during the simplex method search. In this regard, the column generation algorithm provides an efficient solution [25].

8.1.2 The column generation (CG) algorithm. The basic idea of CG is to start by formulating a *restricted master program (RMP)* by only considering a small portion of decision variables in the original DW formulation in Equ. (16)–(18) and iteratively add decision variables to RMP to improve the objective function Equ. (16). Fig. 8(c) shows the framework of the column generation algorithm:

1) Restricted master program - *Find a feasible solution of the DW formulation.*

We formulate the *restricted master program* RMP($\overline{\mathcal{Z}}_1, ..., \overline{\mathcal{Z}}_K$) in Equ. (19) by picking up only a constant number of *columns* (each column corresponds to a decision variable) $\overline{\mathcal{Z}}_k$ ($\overline{\mathcal{Z}}_k \subset \mathcal{Z}_k$) in each polyhedron Λ_k:

$$\overline{\boldsymbol{\lambda}}^* = \left\{ \begin{array}{ll} \min & \sum_{k=1}^{K} \sum_{\hat{\mathbf{z}}_{k,r} \in \overline{\mathcal{Z}}_k} \mathbf{c}_k^\top \hat{\mathbf{z}}_{k,r} \lambda_{k,r} \\ \text{s.t.} & \sum_{k=1}^{K} \sum_{\hat{\mathbf{z}}_{k,r} \in \overline{\mathcal{Z}}_k} \hat{\mathbf{z}}_{k,r} \lambda_{k,r} = \mathbf{1} \\ & \sum_{\hat{\mathbf{z}}_{k,r} \in \overline{\mathcal{Z}}_k} \lambda_{k,r} = 1, \forall k, \lambda_{k,r} \geq 0, \forall k, r \end{array} \right\}. \tag{19}$$

where $\overline{\boldsymbol{\lambda}}^*$ denotes the optimal solution of RMP($\overline{\mathcal{Z}}_1, ..., \overline{\mathcal{Z}}_K$).

Since both the number of constraints and the number of decision variables in RMP($\overline{\mathcal{Z}}_1, ..., \overline{\mathcal{Z}}_K$) are $O(K)$, RMP($\overline{\mathcal{Z}}_1, ..., \overline{\mathcal{Z}}_K$) can be solved efficiently using the simplex method. However, the optimal solution $\overline{\boldsymbol{\lambda}}^*$ of RMP($\overline{\mathcal{Z}}_1, ..., \overline{\mathcal{Z}}_K$) might not achieve the optimal extreme point of the original DW since RMP($\overline{\mathcal{Z}}_1, ..., \overline{\mathcal{Z}}_K$) only includes a subset of DW's decision variables. Therefore, we need to check the optimality of $\overline{\boldsymbol{\lambda}}^*$.

2) Optimality test - *Check the optimality of $\overline{\boldsymbol{\lambda}}^*$; if optimality hasn't been achieved, add new columns (extreme points) to the restricted master program to improve the solution.*

More specifically, we check the optimality of $\overline{\boldsymbol{\lambda}}^*$ by formulating the dual problem DRMP($\overline{\mathcal{Z}}_1, ..., \overline{\mathcal{Z}}_K$) of RMP($\overline{\mathcal{Z}}_1, ..., \overline{\mathcal{Z}}_K$), defined as:

$$(\overline{\boldsymbol{\pi}}^*, \overline{\boldsymbol{\mu}}^*) = \left\{ \begin{array}{ll} \min & \sum_k \pi_k + \sum_l \mu_l \\ \text{s.t.} & \sum_k \hat{\mathbf{z}}_{k,r} \pi_k + \mu_l \leq \mathbf{c}_k^\top \hat{\mathbf{z}}_{k,r}, \\ & \forall \hat{\mathbf{z}}_{k,r} \in \overline{\mathcal{Z}}_k,\ l = 1, ..., K. \end{array} \right\} \tag{20}$$

where $(\overline{\boldsymbol{\pi}}^*, \overline{\boldsymbol{\mu}}^*)$ ($\overline{\boldsymbol{\pi}}^* = [\overline{\pi}_1^*, ..., \overline{\pi}_K^*]$ and $\overline{\boldsymbol{\mu}}^* = [\overline{\mu}_1^*, ..., \overline{\mu}_K^*]$) denote the optimal solution of DRMP($\overline{\mathcal{Z}}_1, ..., \overline{\mathcal{Z}}_K$).

According to the *optimality test criteria* [25], $\overline{\boldsymbol{\lambda}}^*$ achieves the optimal of the original DW formulation if and only if $(\overline{\boldsymbol{\pi}}^*, \overline{\boldsymbol{\mu}}^*)$ satisfies

$$\max_{\mathbf{z}_k \in \Lambda_k} \left\{ \sum_k z_{k,l} \overline{\pi}_k^* + \overline{\mu}_l^* - \mathbf{c}_k^\top \mathbf{z}_k \right\} \leq 0 \ (l = 1, ..., K). \tag{21}$$

The optimality test in Equ. (21) can be further decomposed into a set of independent subproblems sub_k ($k = 1, ..., K$), where each sub_k is to check whether

$$\sum_k z_{k,l}^* \overline{\pi}_k^* + \overline{\mu}_l^* - \mathbf{c}_k^\top \mathbf{z}_k^* \leq 0, \tag{22}$$

where $\overline{\mathbf{z}}_k^*$ is the optimal solution of the following LP problem:

$$\max \sum_k z_{k,l} \pi_k + \mu_l - \mathbf{c}_k \mathbf{z}_k \quad \text{s.t. } \mathbf{z}_k \in \mathcal{Z}_k. \tag{23}$$

In other words, the optimality of DW can be achieved only if, for each sub_k ($k = 1, ..., K$),

$$\sum_k z_{k,l}^* \overline{\pi}_k^* + \overline{\mu}_l^* - \mathbf{c}_k^\top \mathbf{z}_k^* \leq 0. \tag{24}$$

Each sub_k is an LP problem with only $O(K)$ decision variables and $O(K)$ linear constraints and is independent of other subproblems. Hence, sub_1, ..., sub_K can be solved in parallel. If exists $\mathbf{z}_k^*$ with

$$\sum_k \hat{z}_{k,l}^* \overline{\pi}_k^* + \overline{\mu}_l^* - \mathbf{c}_k^\top \mathbf{z}_k^* > 0, \tag{25}$$

the optimality criterion is not achieved, and sub_k sends $\mathbf{z}_k^*$ to the restricted master program as a new column to add. In the next iteration, the updated master program finds its solution and tests the optimality again. This process is repeated until the optimal solution is found.

8.1.3 Time complexity analysis of CG. In each iteration, the restricted master program RMP($\overline{\mathcal{Z}}_1, ..., \overline{\mathcal{Z}}_K$), its dual problem DRMP($\overline{\mathcal{Z}}_1, ..., \overline{\mathcal{Z}}_K$), and the subproblems sub_k ($k = 1, ..., K$) each have $O(K)$ decision variables and $O(K)$ linear constraints. Moreover, given the selected columns in RMP($\overline{\mathcal{Z}}_1, ..., \overline{\mathcal{Z}}_K$), the optimality test can be conducted directly without solving RMP($\overline{\mathcal{Z}}_1, ..., \overline{\mathcal{Z}}_K$) in each iteration. The above features of CG enable an efficient solution using LP in each iteration. Then, the remaining question is how many iterations CG needs to converge the optimal solution.

To test the convergence of CG, we derive a *lower (dual) bound* of the GTS optimal in Theorem 8.1 to measure how close our solution is to the optimal over iterations. Here, we use the superscript $^{(n)}$ to denote the values set/derived in each iteration n.

8.2 The Lower Bound of DW's Optimal

Theorem 8.1. *For each* sub_k *in each iteration n, we let*

$$\delta_{l,k}^{(n)} = \sum_k z_{k,l}^* \overline{\pi}_k^{*(n)} + \overline{\mu}_l^{*(n)} - \mathbf{c}_k^\top \mathbf{z}_k^* \tag{26}$$

and $\delta_l^{(n)} = \max_k \delta_{l,k}^{(n)}$. *Then,*

$$\xi^{(n)} = \sum_k \overline{\pi}_k^{*(n)} + \sum_l \left(\overline{\mu}_l^{*(n)} - \delta_l^{(n)}\right) \tag{27}$$

is a lower bound of the DW's optimal.

Proof. Based on Equ. (26), we obtain that, in each iteration n,

$$\max_{\mathbf{z}_k \in \Lambda_k} \left\{\sum_k z_{k,l} \overline{\pi}_k^{*(n)} + \left(\overline{\mu}_l^{*(n)} - \delta_l^{(n)}\right) - \mathbf{c}_k^\top \mathbf{z}_k\right\} \leq 0, \tag{28}$$

implying $\left\{\overline{\pi}_1^{*(n)}, ..., \overline{\pi}_K^{*(n)}, \left(\overline{\mu}_1^{*(n)} - \delta_1^{(n)}\right), ..., \left(\overline{\mu}_K^{*(n)} - \delta_K^{(n)}\right)\right\}$ construct a feasible solution to the dual problem. Therefore, the corresponding objective value in the dual problem $\sum_k \overline{\pi}_k^{*(n)} + \sum_l \left(\overline{\mu}_l^{*(n)} - \delta_l^{(n)}\right)$ offers a lower bound of the GTS optimal (according to *weak duality* [24]). □