A k-Switch Framework for Privacy-preserving Spatial Crowdsourcing Tasks Assignment

Maocheng Li [†], Jiachuan Wang [†], Libin Zheng [†], Han Wu [*], Peng Cheng [*], Lei Chen [†], Xuemin Lin [#]
[†]*The Hong Kong University of Science and Technology, Hong Kong, China*
{csmichael, jwangey, lzhengab, leichen}@cse.ust.hk
[*]*East China Normal University, Shanghai, China*
han.wu@stu.ecnu.edu.cn, pcheng@sei.ecnu.edu.cn
[#]*The University of New South Wales, Australia*
lxue@cse.unsw.edu.au

ABSTRACT

As an emerging location-based mobile service, spatial crowdsourcing has gained great popularity over the recent years. However, its acquirement of users' location data also widely incurs privacy concerns in the society. In response to such concerns, we study the privacy-preserving spatial crowdsourcing in this paper, where customers (e.g., taxi riders) employ differential privacy techniques to obfuscate their locations when they request service. Consequently the application server no longer has access to their real locations. This prevents the single point of failure that an attack to the central application server results in unpredictable personal privacy information leakage. However, obfuscated locations bring challenges to the task assignment procedure, the key problem in spatial crowdsourcing. Existing task assignment methods take in actual and accurate locations of workers and tasks, to realize an effective assignment. Without considering the obfuscation of locations, directly applying existing methods could match a worker to a customer who is in fact far away. To address the challenge, we formulate the privacy-preserving spatial task assignment problem. By conducting probabilistic analyses on the obfuscation locations, we propose an *Expected Distance* metric to measure the distances among tasks/workers. Then we design a novel k-Switch algorithmic framework. The framework uses the aforementioned *Expected Distance* metric to generate an initial matching between workers and tasks, and then improves the matching by switching the tasks between workers. Extensive experimental results validate the effectiveness and efficiency of our technique.

PVLDB Reference Format:
Maocheng Li, Jiachuan Wang, Libin Zheng, Han Wu, Peng Cheng, Lei Chen, Xuemin Lin. A k-Switch Framework for Privacy-preserving Spatial Crowdsourcing Tasks Assignment. *PVLDB*, x(xxx): xxxx-yyyy, 2020.
DOI: https://doi.org/10.14778/xxxxxxx.xxxxxxx

1. INTRODUCTION

The mass adoption of GPS-equipped smart phones and high-speed wireless network enables individuals to collaborate, participate, consume and produce valuable information about the environment and themselves. Spatial crowdsourcing platforms distribute spatial tasks among participants, and they move physically to a specified locations to accomplish the tasks [10]. Examples of spatial tasks include taking a photo at the Eiffel Tower or picking up a passenger at Wall Street. The participants (also called workers) range from taxi drivers, car owners who offer ride-sharing service, to the general public who selects ad-hoc tasks to perform on a spatial crowdsourcing platform. A typical spatial crowdsourcing platform (e.g., gMission [7, 3] and MediaQ [11]) usually collects the location information of workers and tasks, then matches the suitable worker to the corresponding task. The matching (or task assignment) is often accomplished by executing a centralized assignment strategy on the spatial crowdsourcing server.

Spatial crowdsourcing brings convenience while facing serious privacy issues — the centralized server stores the private information of all its users, and thus becomes a potential single point of failure for privacy information leakage. When a malicious hacker attacks the central server, all users could suffer from private information being stolen. For example, Facebook faced intense scrutiny over whether it is handling users' private information responsibly, after the personal information of nearly 50 million users was exposed due to an attack [9]. Overall, privacy issues have raised significant concerns in recent years among various communities, ranging from policy makers, regulators, researchers, social organizations, technology start-ups, and multi-billion conglomerates.

Aiming at protecting individual's private information, various techniques and frameworks have been proposed. Every potential attack possesses a threat model, specifying characteristics like which entities are trustworthy, which entities may collude together to comprise users' privacy, which users are the targets of the privacy attack, and which stage of crowdsourcing the attack may happen [4]. If we do not trust the workers but the central server, then the central server could use cloaking technique to group nearby k customers together, so no worker could infer exactly where the customer is [17]. If we worry about the privacy of the sensor readings collected by workers, then anonymity technique could be applied such that the sensing result from each worker is mixed with others, and neither the central server nor the end user could identify the original source [14, 18]. If the centralized crowdsourcing server could be untrustworthy, then users should use obfuscation technique such as geographical differential privacy [2] to protect their own location. If only the perturbed location (not the original and accurate locations) of users are observable to the server, then new assignment techniques are needed to optimize the worker/task matching quality. [15] considers the online setting, in which task arrives one by

Proceedings of the VLDB Endowment, Vol. x, No. xxx
ISSN 2150-8097.
DOI: https://doi.org/10.14778/xxxxxxx.xxxxxxx

one.

Motivated by the controversial privacy leakage from Facebook, we formulate our privacy setting in which the centralized server could not be trusted. In the traditional spatial crowdsourcing applications, users upload their accurate location to the centralized application server, and the server determines the optimal assignment (i.e., worker-task pairs). In our new privacy-preserving framework, task requesters (customers) only upload their obfuscated locations to the server. We obfuscate customers' locations by adding random noise drawn from Laplace distribution, which ensures geographical differential privacy for users [2]. Our framework matches the suitable workers to tasks, based on the obfuscated locations of the tasks and accurate locations of workers, to minimizing the total moving distance of the workers. [15] fails to consider the global optimum, because it considers tasks one by one, and after the task assignment is determined, it is irrevocable.

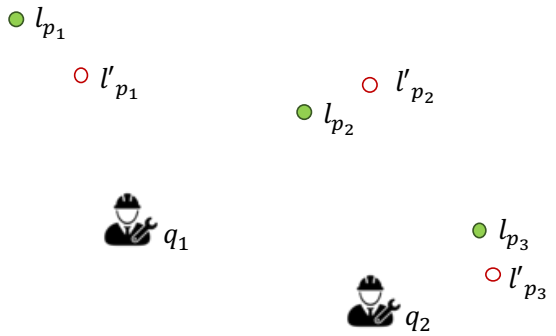


Figure 1: An Illustrative Example

Figure **??** is an example of our problem. Like traditional spatial task assignment problem [12], we have a number of workers and tasks diversely distributed geographically. However, the location of tasks is obfuscated. The reported location (the red empty circles in the figure) of each customer may be different from the actual location (the green filled circles in the figure). Note that in our problem setting, the location of the workers is accurately reported, as we assume that workers are employed by the application administrator (e.g., the taxi drivers follow policies from the taxi company). It simplifies the probabilistic analysis and we will discuss in more details about how to adapt our framework to the case when the locations reported by workers are also obfuscated in Section 6.

By observing only the obfuscated locations of tasks, the inferred distance between tasks and workers are now highly likely to be different from their actual distance. A task could be obfuscated to be very close to a worker, however the actual distance between them could be large. The challenge is how could we find the good matches between workers and tasks, based only on the obfuscated location (not necessarily accurate) of tasks to minimize the total moving distance of workers?

Based on the differential privacy definition and the property of the Laplacian distribution, we first derive that this problem could not be solved exactly, because the obfuscated locations could be arbitrarily worse. Nevertheless, we derive an metric. Using the new metric, our framework obtains matching result which is satisfactory in expectation.

Based on the newly designed *Expected Distance* metric, we propose a novel k-switch mechanism, by selecting the best k-sized subgroups of the original matchings. Within each subgroup, we design a coordinating protocol such that the actual locations of tasks are communicated and tasks are re-assigned to achieve optimal matching. We prove that selecting k-sized subset pairs out of the original matching pairs is NP-hard, and thus we devise efficient approximation methods with performance guarantees.

To summarize, we make the following contributions in the paper:

- We formulate the problem of privacy-preserving spatial task assignment in Section 3. We conduct non-trivial probabilistic analysis on the problem and show that the problem could not be solved exactly. We derive an *Expected Distance* metric, which helps obtain good matching result in expectation in Section 4 .
- We propose a novel k-Switch framework in Section **??**. It selects the best k-sized subgroups of a baseline matching results, and optimize the matching within each subgroup by letting workers and customers communicate their actual locations. The subgroup finding procedure is an NP-hard problem, and we devise efficient approximate algorithm with performance guarantee.
- We carefully design our communication protocol within each subgroup such that privacy leakage is minimized while ensuring the optimal assignment in Section 6.
- Finally, we have conducted extensive experiments to validate the efficiency and effectiveness of algorithms. The results are shown in Section 7.

In addition, we cover the preliminaries and necessary background in Section 2 and conclude the paper in Section 8.

2. BACKGROUND AND RELATED WORK

First we introduce the traditional setting and problem formulation for Spatial tasks assignment problem. Then we briefly cover the essential principles of Geo-indistinguishability, namely geographical differential privacy techniques. The background information here prepares us to formally define our privacy-preserving Spatial Crowdsourcing Tasks Assignment problem.

2.1 Spatial Task Assignment

Traditional spatial task assignment consider a group of customers (e.g. taxi riders, fast-food ordering customers) as a set P. Each customer $p_i \in P$ corresponds to a spatial location l_{p_i} in the Euclidean space, represented by a tuple $l_{p_i} = (x, y)$ where $x, y \in R$. Each customer (or service requestor) requires a service provider (e.g. taxi drivers, fast-food courier) to serve his/her request. Each service provider $q_j \in Q$ also corresponds to a spatial point l_{q_j} in the Euclidean space, represented by $l_{q_j} = (x.y)$ where $x, y \in R$. Each service provider also has a limit on the number of customers it could serve, namely a positive integer capacity $q.k \in N^*$.

The goal is to find a subset $M \subseteq Q \times P$, a matching that could simultaneously satisfy the capacity constraints, maximize the cardinality of the matching $|M|$, while minimizing the costs of the matching (the summation of the distance between all worker-task pairs).

For the maximum cardinality of the matching criterion, conventionally we set γ to be the maximum possible size of the matching. It is obvious that either all the tasks could be matched to a worker, so $\gamma = |P|$, or all workers got their hands full, meaning $\gamma = \sum_{q \in Q} q.k$. As a result, we set $\gamma = min\{|P|, \sum_{q \in Q} q.k\}$. Any algorithms aiming at finding the maximum cardinality of the matching should return a M with $|M| = \gamma$.

For the capacity constraint, each worker could not serve more customers than its capacity.

As for the minimizing cost part, conventionally we define the following cost function, for each possible matching M, we have

$$\Psi(M) = \sum_{(q,p)\in M} dist(l_q, l_p)$$

$\Psi(M)$ simply sums up the distance between the worker-task pair for each of the assignment. Sometimes we call this minimization setting as the MaxMin matching problem, because on one hand, it already produces the maximum cardinality of the matching, and on the other it minimizes the cost of all assignments. So linking back to the real-life application, in the ride-hailing example, the best assignment ensures that the maximum customers served, while minimizing the total pick-up travel distance among all taxis.

It is shown that this problem, also called as the Capacity Constrained Assignment (CCA) problem, could be reduced to the Minimum Cost Flow (MCF) problem on a bipartite graph consisting customer nodes on one side and workers nodes on the other [12]. There are a variety of existing algorithms for the MCF problem. Hungarian algorithm [13, 16], the cost scaling algorithm [8], and the Successive Shortest Path Algorithm (SSPA) are in-memory algorithms, while special techniques of incrementally adding edges to the network reduce unnecessary hard-disk access, which could be very slow and costly [12]. The state-of-the-art SSPA runs in $O(\gamma \cdot (|E| + |V| \cdot log|V|))$.

2.2 Geo-Indistinguishability

Differential privacy is first introduced in the area of statistical databases in the seminal work by Dwork [5]. It is a notion of privacy protection such that when a single item in the original database D is modified, aiming at protecting that item's privacy, the altered database D' is still required to produce aggregate query result 'similar' to the query result returned by the original database D. By 'similar' we mean that the difference of aggregate query results returned by D and D' is bounded by e^{ϵ}.

As location-based service (LBS) becomes prevalent, how to protect the location information of individual users attracts serious attention from the research community. *k-anonymity* is method that combines nearby k locations together, in order to 'hide' user's specific location behind a larger region. However *k-anonymity*, as well as other as methods including using expectation of distance error and cloaking, are effective only under a certain assumption on the prior knowledge of an attacker possesses. If a privacy adversary chooses to attack under a different prior knowledge assumption, these methods may not provide strong privacy protection.

A notion of Geo-Indistinguishability is proposed by extending the differential privacy notion to the spatial domain [2]. It is proven that it provides strong privacy protection given any prior knowledge assumption. The notion is defined formally as follows.

Definition 1. (Geo-indistinguishability) For all true locations x, x', A mechanism M satisfies ϵ-*Geo-Indistinguishability* if and only if:

$$d_\rho(M(x), M(x')) \leq \epsilon d(x, x')$$

where $d(x, x')$ is the Euclidean distance between x and x' while $d_\rho(M(x), M(x'))$ is the multiplicative distance between two distribution associated with x and x'.

A probabilistic mechanism, or an obfuscation scheme, essentially changes the spatial point of interest x to another point in the planar area, with a probability distribution. The above definition of ϵ-*Geo-Indistinguishability* ensures that for any point x' located within r distance from our point of interest x, the probability distributions of all possible obfuscated points for x' and x differ (measured by the multiplicative distance between two probability distributions) by at most $\epsilon d(x, x')$, which is upper bounded by ϵr.

Table 1: Notation.

Symbol	Description
P	a set of n customers
Q	a set of m workers
l_{p_i}	the location of a customer $p_i \in P, l_{p_i} = (x, y)$
l_{q_j}	the location of a worker $q_j \in Q, l_{q_j} = (x, y)$
$q_j.c$	the capacity constraint of a worker. q_j could do at most $q_j.c$ tasks
L_P	the set of all the locations of customers $L_P = \{l_{p_1}, l_{p_2}, \cdots, l_{p_n}\}$
L_Q	the set of all the locations of customers $L_Q = \{l_{q_1}, l_{q_2}, \cdots, l_{q_m}\}$
l'_{p_i}	the obfuscated location of a customer $p_i \in P$, $l'_{p_i} = (x', y')$
L'_P	the set of all the obfuscated locations of customers $L'_P = \{l'_{p_1}, l'_{p_2}, \cdots, l'_{p_n}\}$
$dist(l_1, l_2)$	the distance function $dist(l_1, l_2) = \sqrt{(l_1.x - l_2.x)^2 + (l_1.y - l_2.y)^2}$
$dist(l_{p_i}, l_{q_j})$	the distance cost between a customer p_i and a worker q_j
$L_{P,Q}$	the set of all the locations of both customers P and workers Q
$dist(l'_{p_i}, l_{q_j})$	the observed distance cost between a worker q_j and a privacy-preserved customer p_i

One particular mechanism satisfying the above property is drawing random noise from planar Laplace distribution [2]. Given a parameter $\epsilon \in \mathbb{R}^+$, the actual location $x_0 \in \mathbb{R}^2$, the probability density function of the obfuscated location (or noisy location), on another location $x \in \mathbb{R}^2$ is:

$$D_\epsilon(x_0)(x') = \frac{\epsilon^2}{2\pi} e^{-\epsilon d(x_0, x)} \quad (1)$$

$\frac{\epsilon^2}{2\pi}$ is the normalization factor. We use the above noisy location density function to generate obfuscated locations from actual locations.

3. PROBLEM DEFINITION

In this section we formulate the privacy-preserving spatial crowdsourcing task assignment problem formally. We summarize the notation used in this paper in Table 1.

We start the problem definition by presenting an illustrative example in Figure **??**.

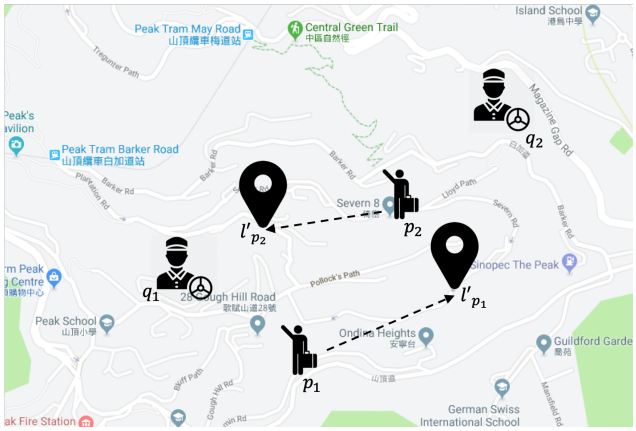


Figure 2: An example

At this time window, there are two customers p_1 and p_2 waiting to be served (waiting for the taxi drivers to pick them up). There are two workers q_1 and q_2 available to be assigned to taxi orders. The figure demonstrates the perspective from the application server. The observed location for p_1 and p_2 is l'_{p_1} and l'_{p_2}. Now based

on the observed location of the customers, the application server needs to seek for a matching algorithm to determine how to assign workers q_1 and q_2 to customers to minimize their travel distance from their current locations to the locations of the tasks.

As the figure shows, there exists the actual locations (or true locations) of customers p_1 and p_2. Their actual locations are denoted as l_{p_1} and l_{p_2}. These locations are not observable by the application server, because *Geo-Indistinguishability* Laplacian noise has been added to these locations. For l_{p_1}, a random point is drawn from the planar Laplace distribution, with the pdf of $\frac{\epsilon^2}{2\pi}e^{-\epsilon d(l_{p_1},x)}$, and becomes an observable location l'_{p_1} to the server. Same privacy-preserving technique was applied to l_{p_2}, giving us another observable location l'_{p_2}.

Note that the optimization goal is to minimize the total distance between workers and the *actual locations* of the tasks, not the observed ones. Using the illustrative example above, if we assign the tasks to their closest workers based on the observed distance, then we get a matching $M' = \{(p_2, q_1), (p_1, q_2)\}$. Assigning task p_2 to worker p_1 is because based on the observed distance, $d(l'_{p_2}, l_{q_1}) = 3 < 4 = d(l'_{p_2}, l_{q_2})$, meaning it 'looks' like q_1 is closer to p_2. Similar analysis assigns p_1 to q_2. When we look at the utility (or quality) of the matching, we look at the actual locations, because obviously the workers need to drive to the actual locations to pick up the customers. The quality of the matching in this case is $\Psi(M) = \sum_{(q,p)\in M} dist(q.l, p.l) = d(l_{p_2}, l_{q_1}) + d(l_{p_1}, l_{q_2}) = 3 + 4 = 7$. Note that this is not the optimal matching, because looking at the actual locations, assigning p_1 to q_1 and p_2 to q_2 actually lead to the optimal result $\widetilde{M}$, with the cost $\Psi(\widetilde{M}) = d(l_{p_1}, l_{q_1}) + d(l_{p_2}, l_{q_2}) = 2 + 3 = 5$.

Now we define the Privacy-preserving Spatial Crowd-sourcing Tasks Assignment problem.

We have the following inputs.:

- The set of customers P and their obfuscated locations L'_P
- The set of workers Q and their location L_Q
- The distance function $dist$ which measure the Euclidean distance between two locations
- the parameter ϵ used in *Geo-Indistinguishability*, which specifies the level of privacy protection. The parameter used in the obfuscation probability density function $\frac{\epsilon^2}{2\pi}e^{-\epsilon \cdot dist(x_0,x)}$

We expect an output of matching $M \subseteq Q \times P$, with the maximum cardinality $\gamma = min\{|P|, \sum_{q\in Q} q.k\}$, and the minimum cost,

$$\Psi(M) = \sum_{(q,p)\in M} dist(l_q, l_p)$$

In the ultimate objective, the actual location of the customer l_p is used for measurement. However this location is not available in the input given.

3.1 Worst-case

The hardness of the problem lies in the fact that the obfuscation function could perturb an original location to any possible location on the plane. And once the obfuscation is done, the input we are having is the set of fixed 'fake' locations of users. The random noise drawn from the Laplace distribution is done and set, so we do not have the chance to observe repeated experiment results, to gather more information about the original location of the task.

Since the perturbation could drag an original point to anywhere on a plane and produce a possible input, an arbitrarily far away point from a worker could be potentially moved to be very close. We do not have any other prior information, other than the observed distance. So statistically we would infer the worker should take the closest task. However as we purposely set the original location of the task to be arbitrarily far away from the task, the resulted matching could lead to an arbitrarily worst cost.

Because of the above analysis, we start presenting our statistical analysis framework in the following section, and propose a k-switch algorithm, which compromises small amount of privacy protection, but greatly enhances the matching quality.

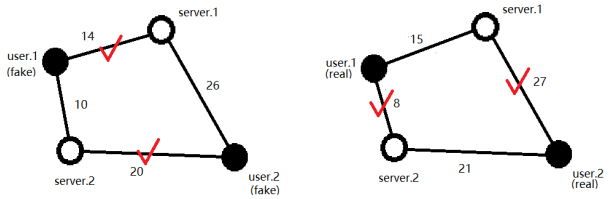
4. MINIMUM COST ASSIGNMENT WITH EXPECTED DISTANCES

In this section we will discuss the existing algorithms for off-line pairing and discuss the deficiency of it when applies to the privacy framework. Then we will derive a statistic model for the encrypted location model. Based on these calculation, some modifications are added for the original algorithms from different approach to achieve a reasonable assignment.

4.1 Common Solution for Off-line Matching

Successive Shortest Path Algorithm: this algorithm (short as **SSPA**) finds the global best solution for off-line matching problem. Users and servers are treated as two type of nodes connected with source node and sink node respectively. If an accessible cost exists for a user and a server to match, an edge will be added between them with the cost. To generate k pairs of users and servers, the algorithm runs k rounds of *Dijkstra algorithm* to seek for a road from source node to sink node followed by reshape of edges and cost for the chosen shortest path. Because the assignment is off-line and the matching algorithm only runs in the company, the time and space costs are acceptable enough.

But the performance of SSPA may not meet the expectation with privacy protection. The only information provided by the users are their fake locations. Company can only calculate the cost which in proportion to distance refer on the perturbed addresses. Obviously the costs vary when the true locations of their users land on different places. Figure 2 shows a situation where mismatching occurs when directly use the **SSPA**. The minimum cost using the information provided by users is $14 + 20 = 34$ while another choice costs $10 + 26 = 36$. But the real locations of users shown in Fig.**??** generate different cost for each edge and the $s_1 - u_2, s_2 - u_1$ is the best choice.



(a) Directly use SSPA on perturbed locations

(b) Actual minimum cost matching for real locations

Figure 3: A mismatching result from directly using SSPA based on the fake locations from users. The result of mismatching (a) and the real minimum cost matching (b) are both shown.

4.2 Statistical Analysis for Distance Distribution

With fake locations of the users, the statistic model is needed to improve the matching algorithm. From the previous part, we know that the locations were encrypted using Laplace distributions, which generates the mis-matching. Therefore, one can analysis the probability distribution of real locations and further analysis the distances. With more reasonable distances using for SSPA, the results of matching for real cost can be optimized.

Here we first introduce an approximation analysis to prove the distance for SSPA should be modified. The we generate the method for calculating the expectation of the distance using precise way and empirical way.

Approximated BND: This approximated is chosen to give a distribution analysis for online matching. Due to the difficulty in analysis for planar Laplace distribution, the bivariate normal distribution (BND) is used with a pretty good similarity. Here we will also derive a conclusion that the expectation of distance is different from the original one based on the encryption method.

For better approximation, the mean and variance of BND is chosen to be the same with the planar Laplace distribution. Then the distribution can be written as $BND(\mu, \Sigma)$, which featured by $\mu = \left[x_{l^*_{u_1}}, y_{l^*_{u_1}}\right]$ and $\Sigma = \begin{bmatrix} \sigma^2 & 0 \\ 0 & \sigma^2 \end{bmatrix}$. $x_{l^*_{u_1}}$ and $y_{l^*_{u_1}}$ refers to the two dimensions of a fake location. $\sigma = \frac{\sqrt{2}r}{\epsilon}$ refers to the standard deviation of planar Laplace distribution for privacy level $(\epsilon, r) - Geo - I$.

Then we can derive the distribution for the distance. What we focus on is $d(u_i, s_j) = \left|l_{u_i} - l_{s_j}\right|$. For now, we can only calculate the l_{u_i} using the distribution of $l^*_{u_i}$, approximated by $BND(l^*_{u_i}, \frac{2r^2}{\epsilon^2})$. So based on the Eula distance $d = \sqrt{d_x^2 + d_y^2}$, which means the distribution of two dimensions for calculation follows $BND(x_{l^*_{u_i}} - x_{l_{s_j}}, \frac{2r^2}{\epsilon^2})$ and $BND(y_{l^*_{u_i}} - y_{l_{s_j}}, \frac{2r^2}{\epsilon^2})$. We can use theory for bivariate random variable to get the distribution of $d^2 = d_x^2 + d_y^2$. The most important information for the quadratic from of distance can be calculated by moment-generating function including its mean and variance.

Generally, for variables $\mathbf{X} \sim N(\mu, \Sigma), \Sigma > 0$, the object quadratic form Q based on the quadratic matrix A have relationship $Q = \mathbf{X}'A\mathbf{X}$. In this question we have:

$$Q = X_1^2 + X_2^2 + 0 * X_1 X_2$$

$$A = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$$

And the moment generating function (mgf) can be simplified as:

$$M_Q(t) = \left|I - 2t\Sigma^{\frac{1}{2}} A \Sigma^{\frac{1}{2}}\right|^{\frac{1}{2}} \cdot e^{t\mu'\Sigma^{-\frac{1}{2}}(\Sigma^{\frac{1}{2}} A \Sigma^{\frac{1}{2}})(I - 2t\Sigma^{\frac{1}{2}} A \Sigma^{\frac{1}{2}})^{-1}\Sigma^{-\frac{1}{2}}\mu}$$

By replace it with our variables, we can derive:

$$M_Q(t) = \prod_{i=1}^{2} (1 - 2\sigma_i^2 t)^{\frac{1}{2}} \cdot e^{t\sum_{i=1}^2 \frac{\mu_i^2}{1 - 2\sigma_i^2 t}}$$

where $\mu = \left[x_{l^*_{u_i}} - x_{l_{s_j}}, y_{l^*_{u_i}} - y_{l_{s_j}}\right]'$ and $\boldsymbol{\sigma^2} = \left[\sigma^2, \sigma^2\right]$.

With the mgf of d^2, we can calculate the mean and variance of d^2:

$$\mu_{d^2} = M'_Q(0)$$

$$\sigma^2_{d^2} = M''_Q(0) - (M'_Q(0))^2$$

The MQ' and MQ'' refer to the first and second derivative of the mgf respectively.

Under such a condition, we can prove that:

$$M_Q(t) = e^{\frac{d^2_{u^*,s} t}{1 - 2\sigma^2 t}} \cdot \frac{1}{1 - 2\sigma^2 t}$$

where $d^2_{u^*,s} = (x_{l^*_{u_i}} - x_{l_{s_j}})^2 + (y_{l^*_{u_i}} - y_{l_{s_j}})^2$.Here we can calculate the results:

$$\mu_{d^2} = d^2_{u^*,s} + 2\sigma^2$$

$$\sigma^2_{d^2} = 4\sigma^2(\sigma^2 + d^2_{u^*,s})$$

Obviously, the expectation of distance is actually larger than the sighted distance of server and fake user location information $d^2_{u^*,s}$. The distance goes larger with higher variation.

Precise expectation calculated based on planar Laplace distribution is thus needed. Here we reshow the distribution:

$$D(l^*_u)(l_u) = \frac{\epsilon^2}{2\pi} e^{-\epsilon d(l^*_u, l_u)}$$

The l^*_u and l_u are the fake and real locations of user respectively. For more easily derivation, we transfer it in to polar Laplacian:

$$D(r, \theta) = \frac{\epsilon^2}{2\pi} r e^{-\epsilon r}$$

As we can observe, the two variable of polar Laplacian distribution r and θ are independent because θ doesnt appear in this function. In Fig.**??** we show a pair of user U and server S and an arbitrary selected point P as the probable real location of the user. r_f and θ_f are the distance and the direction angle between the location of server and the fake location of user respectively, while r and θ for the possible and perturbed location of user.

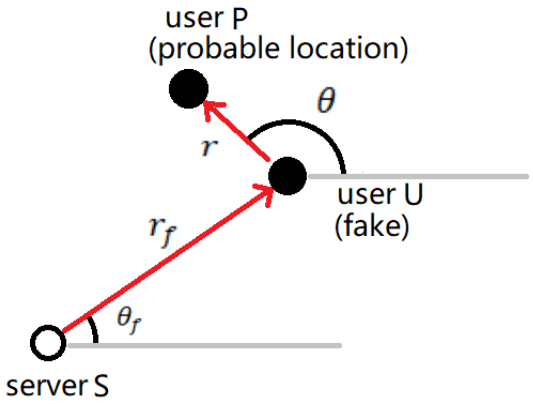


Figure 4: The location of server S and the perturbed location of user U in polar coordinate. A possible true location of user P is shown in figure as well with arbitrarily selected parameter r and θ.

In this situation, the probability of the chosen real point is:

$$D(r, \theta) = \frac{\epsilon^2}{2\pi} r e^{-\epsilon r}$$

And the real distance is expected to be:

$$d(S, P) = \overrightarrow{SP}_r = (\overrightarrow{SU} + \overrightarrow{UP})_r = \sqrt{r_f^2 + r^2 + 2r_f r cos(\theta_f - \theta)}$$

where $\overrightarrow{AB}$ refer to a vector from point A to B and $\overrightarrow{SP}_r$ means the value of r for vector $\overrightarrow{SP}$.

In the whole plane, we can finally calculate the expectation of distance for real location:

$$E(d) = \int_0^{\infty} \int_0^{2\pi} \frac{\epsilon^2}{2\pi} r e^{-\epsilon r} \cdot \sqrt{r_f^2 + r^2 + 2 r_f r cos(\theta_f - \theta)} d\theta dr$$
$$= \int_0^{\infty} \frac{\epsilon^2}{2\pi} r e^{-\epsilon r} \int_0^{2\pi} \sqrt{r_f^2 + r^2 + 2 r_f r cos(\theta_f - \theta)} d\theta dr$$

We verify the integration by Matlab and the part of:

$$\int_0^{2\pi} \sqrt{r_f^2 + r^2 + 2 r_f r cos(\theta_f - \theta)} d\theta$$

cannot be derived as a natural equation.

Fortunately, both spicy and Matlab provide us with numerical integration that can be used for this calculation. With the expectation of distance instead of directly calculated one, we can have more reasonable better distance matrix to feed the SSPA algorithm.

To make it calculate faster for the company, we use a precomputed list to save all the expectation of distance for SSPA with enough precision. In this case, refocus on the integration with θ, we can find:

$$\int_0^{2\pi} \sqrt{r_f^2 + r^2 + 2 r_f r cos(\theta_f - \theta)} d\theta$$
$$\xrightarrow{\theta = \alpha + \theta_f - \theta'_f} \int_{\theta'_f - \theta_f}^{2\pi + \theta'_f - \theta_f} \sqrt{r_f^2 + r^2 + 2 r_f r cos(\theta'_f - \alpha)} d\alpha$$
$$= \int_{\theta'_f - \theta_f}^{2\pi} \sqrt{r_f^2 + r^2 + 2 r_f r cos(\theta'_f - \alpha)} d\alpha$$
$$+ \int_{2\pi}^{2\pi + \theta'_f - \theta_f} \sqrt{r_f^2 + r^2 + 2 r_f r cos(\theta'_f - \alpha)} d\alpha$$
$$= \int_0^{2\pi} \sqrt{r_f^2 + r^2 + 2 r_f r cos(\theta'_f - \theta)} d\theta$$

With this conclusion, the result doesn't change no matter which direction the user is facing to its server. We only need to calculate different expectation based on different distance between the location of server and the fake location of user. A two-dimension list thus transfer to one dimension.

Another interesting characteristic can be referred from the approximation BND. While the expectation of d^2 is $\mu_{d^2} = d_{u^*,s}^2 + 2\sigma^2$, we can find that:

$$0 \leq \lim_{d_{u^*,s} \to \infty} (E(d) - d_{u^*,s})$$
$$\leq \lim_{d_{u^*,s} \to \infty} (\sqrt{E(d^2)} - d_{u^*,s}) = 0$$
$$\Rightarrow \lim_{d_{u^*,s} \to \infty} E(d) = d_{u^*,s}$$

This is also proved in later numerical integration. So for larger distance we can use the fake distance instead of the expectation without lose precision. And we calculate the list in a logistic space with dense points selected when $d_{u^*,s} \to 0$ and sparse calculations when its value goes bigger. At the same time, we can observe that if the variation is close to 0, which means the locations are not encrypted at all, the expectation is the same with the observed distance.

4.3 Statistical Solution

Using the probability result, we derive an Optimized Cost Offline Matching Algorithm (OCOMA) based on probability. The major part for matching is based on SSPA and we mainly modify the preparation part and examine part. Algorithm **??** presents the pseudo-code of OCPMA.

Algorithm 1: OCOMA Algorithm

Input: Set L_u, Set L_u^*, Set L_s, Edge set $E = \emptyset$, Expected distance dictionary D
Output: The total cost sum_{cost} for all the users

```
foreach l_{u_i^*} ∈ L_u^* do
    foreach l_{s_j} do
        dis^* := |l_{u_i^*} − l_{s_j}|
        dis := D[dis^*]
        add edge e(l_{u_i^*}, l_{s_j}) to E with cost dis
run algorithm SSPA to get the pairs in dictionary P
sum_cost = 0
for i := 1 to k do
    u_{P_i}, s_{P_i} = Pi
    sum_cost += |l_{u^{P_i}} − l_{s_{P_i}}|
return sum_cost
```

In this algorithm we firstly run a double loop to calculate cost for each pair of user and server. The fake distance is firstly calculated and used as a key to search in the precomputed list.

After the edges are added, the algorithm starts an SSPA to generate all pairs. Finally, the pairs back to the real locations and calculate the true cost to evaluate the algorithm.

5. λ-OPTING FRAMEWORK

In Section 4, we've proposed using our Expected Distance metric for SSPA algorithm to obtain a preliminary matching between workers and tasks.

In this section, we introduce a novel λ-Opting Framework to further optimize the matching quality between workers and tasks. The main challenge in the privacy-preserving setting is that the centralized application server does not have access to the accurate locations of the tasks. The only accessible information is the obfuscated locations. Based on only the obfuscated locations, a worker could be wrongly assigned a farther away task rather than a closer task, because the farther away task may 'seem' to be closer to the worker.

There are three major steps in the λ-Opting Framework.

Step 1, Measuring step. We utilize statistical analysis to measure the probability of 'mismatch' of small group of workers and tasks, based only on the observed obfuscated locations of the tasks.

Step 2, 1-Opting. The server notifies the selected small groups and allow them to achieve optimization of the matching quality within the group, via internal communication only within the group. Since internal communication allows the workers inside the group to access the actual locations of the tasks, workers could swap tasks if the swapping could lead to a better matching inside the group. The selected small groups notify the server about the internal swapping of the group.This phase is called 1-Opting.

Step 3, λ-Opting. The server repeats Measuring step and 1-Opting step for λ times, each time looking for other groups with relatively high probability of mismatches, and complete λ iterations to accomplish the λ-Opting Framework.

We introduce the technical details of each step of the λ-Opting Framework in this section.

5.1 Preliminary Matching

To make it clearer to present the techniques, we start by formulating the problem input.

From the Expected Distance metric based SSPA algorithm proposed in Section 4, we could obtain a preliminary matching between workers and tasks. A preliminary matching $M \subseteq Q \times P$ contains the worker-task pairs. $M = \{m_i | m_i = (m_i.q, m_i.p), i \in [0, |M|], m_i.q \in Q, m_i.p \in P\}$ where Q is the set of workers, and P is the set of tasks.

We have $|M|$ preliminary matched pairs. Each worker in the matching, $m_i.q$, is associated with an assigned task $m_i.p$. Our high-level goal is to divide all the workers into small groups of size 2, according to some statistical criterion to be described later. After dividing them into groups, communication inside the groups is allowed, so workers inside the small group could access to the accurate location of the tasks originally assigned to the other workers inside the group. Based on the accurate location information, they could measure the travel distance, and if swapping of tasks inside the group could yield smaller travel distance, the workers would swap tasks.

There are two main questions to be answered in this framework. First, the 'statistical criterion' of grouping certain workers together. Our group size is set to be 2, and every worker may be grouped with any other worker from the worker set Q. We will define an Obfuscation-score in the Measuring Step, which quantifies the quality of grouping. Intuitively, we hope to group workers such that swapping of their currently assigned tasks may lead to best possible improvement of travel distance. Second, after grouping, how should we enable within-group communication, and iteratively improve the matching quality?

We will illustrate the details to answer the above questions in the following steps.

5.2 Measuring Step

Based solely on the observed locations (reported obfuscated locations) of the tasks, the preliminary matching between workers and tasks could be sub-optimal. In Measuring step, our goal is to find the pairs of workers with higher probability of being assigned the tasks with sub-optimal travel distance. Finding such pairs could enable the techniques in later steps of our λ-Opting Framework to optimize the worker-task matching quality (travel distance) in iterative manners.

To measure the probability of sub-optimal matching, we define a score first - Obfuscation score.

5.2.1 Obfuscation score

We start by presenting an important observation on the matching result. We analyzed different cases when we make wrong assignments, in which we assign the worker to a task which seems closer, and we should have assigned the worker to another task in the optimal matching. We look at the grouping of size 2. Within each group, 2 workers are grouped together, with their assigned tasks obtained from the Expected Distance based SSPA algorithm. We try to analytically assign a score to this grouping to measure the possibility that the matching algorithm makes mistakes. The basic intuition is that the mistake making probability is higher, if the two tasks are closer to each other, while the positions of the workers are farther away to each other. We have the following definition.

Definition 2. (Obfuscation-score) From the preliminary matching results, given a grouping of size 2 , $g_i = \{(m_i, m_j) | m_i = (m_i.q, m_i.p), m_j = (m_j.q, m_j.p)\}$, $g \in M \times M$ we define the obfuscation-score, or *o-score* to be:

$$\textit{o-score}(g_i) = \frac{dist(l_{m_i.q}, l_{m_j.q})}{dist(l'_{m_i.p}, l'_{m_j.p})}$$

Note that this *o-score* gets larger when the distance between the observed locations of the tasks gets smaller. *o-score* also gets larger if the distance between the two workers is larger.

We analytically prove that the probability of wrongly assigning the task with the actual location farther to the worker, inversely correlates with the distance between the two tasks, while positively correlates with the distance between the two workers. So the probability of wrong assignment is quantified by the *obfuscation-score*. The sketch of the proof is shown below.

Theorem 5.1. *(Effectiveness of obfuscation-score) The obfuscation score defined in Definition 2 positively correlates with the probability of wrongly assigning a worse task (with longer distance) to a worker.*

Proof. For any workers A and B, we build a coordinate system with the midpoint of AB to be the origin, the direction of $\overrightarrow{AB}$ is the direction of x axis. Define $|AB| = d_w$, then we have all the points C which satisfy $|AC| - |BC| = \Delta$ on one hyperbolic curve. We call the δ as $shiftdistance$. So the curve for points having the same shift distance can be written as:

$$\pi_c : \frac{x^2}{(\frac{\Delta}{2})^2} - \frac{y^2}{(\frac{d_w^2 - \Delta^2}{4})} = 1$$

where:

$$\begin{cases} x > 0,\ \Delta > 0 \\ x < 0,\ \Delta < 0 \end{cases}$$

For any observed locations of customers C' and D' paired with A and B, we can calculate their shift distances $\Delta_{C'}, \Delta_{D'}$. Such an assignment refer to that:

$$\begin{aligned} &|AC'| + |BD'| < |AD'| + |BC'| \\ \Rightarrow &|AC'| - |BC'| < |AD'| - |BD'| \\ \Rightarrow &\Delta_{C'} < \Delta_{D'} \end{aligned}$$

With a disclosure of their real locations C, D to each other, a better assignment could occur when $\Delta_C > \Delta_D$ so the pairs are exchanged. The saved cost is $\Delta_C - \Delta_D$.

There are two critical parts to make the information disclosure more valuable:

1.The probability of observed point $C'(D')$ to be actually located at $C(D)$ is high; 2.The saved cost $\Delta_C - \Delta_D$ to be as high as possible.

To achieve the first aim, we can start from the possiblity distribution of encryption. The possibility is decreased as distance between fake and real locations goes farthur. For any workers A, B and a fake customer location T' at any place, we want the shift distance to be as big as possible within limited distance from its real location T.

Here we can define the hyperbola contains T':

$$\pi_{T'} : \frac{x^2}{(\frac{\Delta_{T'}}{2})^2} - \frac{y^2}{(\frac{d_w^2 - \Delta_{T'}^2}{4})} = 1$$

According to the symmetry we can only consider the case when $x > \frac{\Delta_{T'}}{2}$ and $y > 0$. The other cases is just inverse the value of x and y.

Then all the points T that have a shift distance d_Δ bigger than T' is located in the curve:

$$\pi_T : \frac{x^2}{(\frac{\Delta_{T'} + d_\Delta}{2})^2} - \frac{y^2}{(\frac{d_w^2 - (\Delta_{T'} + d_\Delta)^2}{4})} = 1$$

Now we can calculate the mimnum cost to shift from $\pi_{T'}$ to π_T. This result is consists of the properties of $\pi_{T'}$ and the points T'.

We randomly select a point $T(x_0, y_0)$, and we can discribe π_T as:

$$F_{\pi_T} = \frac{x^2}{(\frac{\Delta_{T'}+d_\Delta}{2})^2} - \frac{y^2}{(\frac{d_w^2-(\Delta_{T'}+d_\Delta)^2}{4})} - 1 = 0$$

So the normal function is:

$$\left.\frac{\partial F_{\pi_T}}{\partial y}\right|_{x=x_0,y=y_0}(x-x_0) - \left.\frac{\partial F_{\pi_T}}{\partial x}\right|_{x=x_0,y=y_0}(y-y_0) = 0$$
$$\Rightarrow \frac{y_0(x-x_0)}{(\frac{d_w^2-(\Delta_{T'}+d_\Delta)^2}{4})} + \frac{x_0(y-y_0)}{(\frac{\Delta_{T'}+d_\Delta}{2})^2} = 0$$

The normal vector intersect with $\pi_{T'}$ at T', Then the minimum distance for T' getting to π_T is $|TT'|$. Let $d_\Delta \to 0$, we can get the change rate of shift distance at $T(T')$.

The constraint conditions are shown below:

$$\begin{cases} \pi_{T'}: \frac{x^2}{(\frac{\Delta_{T'}}{2})^2} - \frac{y^2}{(\frac{d_w^2-\Delta_{T'}^2}{4})} = 1 \\ \pi_T: \frac{x^2}{(\frac{\Delta_{T'}+d_\Delta}{2})^2} - \frac{y^2}{(\frac{d_w^2-(\Delta_{T'}+d_\Delta)^2}{4})} = 1 \\ \frac{y_0(x-x_0)}{(\frac{d_w^2-(\Delta_{T'}+d_\Delta)^2}{4})} + \frac{x_0(y-y_0)}{(\frac{\Delta_{T'}+d_\Delta}{2})^2} = 0 \\ d_{TT'} = \sqrt{(x_{T'}-x_0)^2+(y_{T'}-y_0)^2} \end{cases}$$

where we have four free variables $x_{T'}, y_{T'}, y_0, d_{TT'}$ and four inputs $x_0, \Delta_{T'}, d_w, d_\Delta$. Four constraint functions can solve all these variables. We get four pair of results shown in Fig.**??** and we only need the result in the first quadrant.

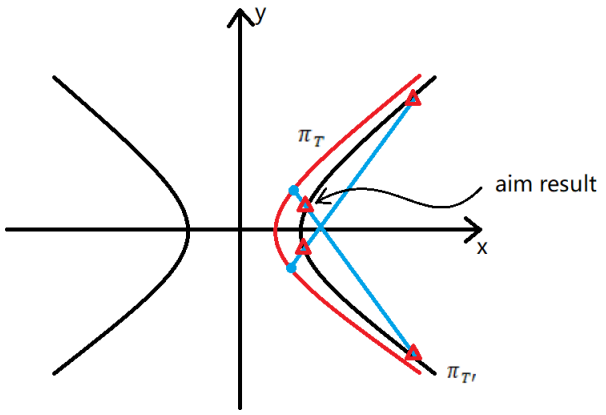


Figure 5: Four results location of T' in red triangles. The distance relationship does not change under different signs of y, so we pick only first quadrant to analysis and the nearest approach to realize shift distance d_Δ is point out black red arraw.

Then we have the change rate while letting $d_\Delta \to 0$, which is a function of $x_0, \Delta_{T'}, d_w$. The d_Δ is an infinitesimal with few probability to travel a long distance after encryption. Then we have $x_0 \approx x_{T'}$, enables the result to be applied to any observed fake location T'.

We can define the change rate of shift distance as:

$$D_C = \lim_{d_\Delta \to 0} d_{TT'} = D_C(x_0, \Delta_{T'}, d_w)$$

We can prove that

$$D_C = \frac{2d_\Delta\sqrt{(d_w^2-\Delta_{T'}^2)^2+4d_w^2y_0^2}}{d_w^2-\Delta_{T'}^2}$$

details are displayed in appendix.

Now we can analyze the impact of all inputs to the change rate of shift distance. As we are aiming to get the correlation, we can directly analyze $d_w^2, \Delta_{T'}^2, t_0^2$:

$$\frac{\partial D_C}{\partial y_0^2} = \frac{4d_w^2 d_\Delta}{(d_w^2-\Delta_{T'}^2)\sqrt{(d_w^2-\Delta_{T'}^2)^2+4d_w^2y_0^2}} > 0$$

$$\frac{\partial D_C}{\partial \Delta_{T'}^2} = \frac{8d_w^2y_0^2d_\Delta}{(d_w^2-\Delta_{T'}^2)^2\sqrt{(d_w^2-\Delta_{T'}^2)^2+4d_w^2y_0^2}} > 0$$

$$\frac{\partial D_C}{\partial d_w^2} = -\frac{4y_0^2(d_w^2+\Delta_{T'}^2)d_\Delta}{(d_w^2-\Delta_{T'}^2)^2\sqrt{(d_w^2-\Delta_{T'}^2)^2+4d_w^2y_0^2}} < 0$$

The change rate is positive correlate with y_0 and $\Delta_{T'}$ but negative correlate with d_w. With this conclusion we can get a simple score system as 2.

To give more explicit score, we can calculate score for any pair of workers A, B and their matched customers $T' = \{C', D'\}$. The location of workers and a customer are set as:

$$A(x_A, y_A), B(x_B, y_B), T'(x_{T'}, y_{T'})$$

Then we can get the needed inputs:

$$\begin{cases} d_w = \sqrt{(x_A-x_B)^2+(y_A-y_B)^2} \\ \Delta_{T'} = \sqrt{(x_A-x_{T'})^2+(y_A-y_{T'})^2} \\ \quad - \sqrt{(x_B-x_{T'})^2+(y_B-y_{T'})^2} \\ y_0 = \frac{|(y_A-y_B)(x_B-x_{T'})-(x_A-x_B)(y_B-y_{T'})|}{d_w} \end{cases}$$

And calculate the shift cost for each fake location of T' and change amount of shift distance d_Δ:

$$d_{TT'} = 2d_\Delta\frac{\sqrt{(d_w^2-\Delta_{T'}^2)^2+4d_w^2y_0^2}}{d_w^2-\Delta_{T'}^2}$$

Now for the second point, we need the saved cost $\Delta_C - \Delta_D$ to be as high as possible. The better condition occurs only when the real location has a different match result with the fake location, which means $(\Delta_C - \Delta_D)(\Delta_{C'} - \Delta_{D'}) < 0$, because of the low possibility of distance variation between real and fake locations, the $d_{\Delta_C} = |\Delta_C - \Delta_{C'}|$ and $d_{\Delta_D} = |\Delta_D - \Delta_{D'}|$ will be small. The constraint condition could be:

$$0 > (\Delta_C - \Delta_D)(\Delta_{C'} - \Delta_{D'})$$
$$\geq (|\Delta_{C'} - \Delta_{D'}| - d_{\Delta_C} - d_{\Delta_D})|\Delta_{C'} - \Delta_{D'}|$$

so we can derive that:

$$|\Delta_C - \Delta_D| \leq d_{\Delta_C} + d_{\Delta_D} - |\Delta_{C'} - \Delta_{D'}|$$

In order to get the biggest saved cost, the $|\Delta_{C'} - \Delta_{D'}|$ should be as less as possible.

Now we can analyze the expected save cost using probability model. For a pair of matching $A-C, B-D$, the value is:

$$E(\Delta) = \int_{\pi_1}\int_{\pi_2} SAVE \cdot P(C,C') \cdot P(D,D') d\pi_1 \pi_2$$

where:

$$\begin{aligned} &SAVE = (|AD| + |BC|) - (|AC| + |BD|) \\ &\pi_1 = C(x_C, y_C) \in R^2 \\ &\pi_2 = \{D | D \in |AC| + |BD| > |AD| + |BC|\} \\ &(|AC| + |BD|) - (|AD| + |BC|) = |\Delta_C - \Delta_D| \\ &\leq d_{\Delta_C} + d_{\Delta_D} - SAVE' \end{aligned}$$

$P(C, C')$ is the possibility of observed location C' to real locate at C:

$$P(C,C') = \frac{\epsilon^2}{2\pi} e^{\epsilon d(C,C')}$$

Replacing the distance with the approximate minimum distance cost for a shift distance $d(C, C')$, we have:

$$P_m(C,C') = \frac{\epsilon^2}{2\pi} e^{-2\epsilon d_{\Delta_{C'}} \frac{\sqrt{(d_w^2 - \Delta_{C'}^2)^2 + 4 d_w^2 y_0^2}}{d_w^2 - \Delta_{C'}^2}}$$

According to our reasoning, we can also get the shortest distance to realize an increasing of shift distance at d_{Δ_C}. Now we have:

$$E(\Delta) \approx \int_{\pi_1}\int_{\pi_2} SAVE' \cdot P(C,C') \cdot P(D,D') d\pi_1 d\pi_2$$

which can be further used to approximate a obfuscation score:

$$\begin{aligned} &max\left(SAVE' \cdot P_m(C,C') \cdot P_m(D,D')\right) \\ &= \frac{\epsilon^3}{8\pi^2 \alpha} e^{-1 - 2\epsilon\alpha |\Delta_{C'} - \Delta_{D'}|} \end{aligned}$$

where:

$$\begin{aligned} \alpha = min\,(\; & d_{\Delta_{C'}} \frac{\sqrt{(d_w^2 - \Delta_{C'}^2)^2 + 4 d_w^2 y_0^2}}{d_w^2 - \Delta_{C'}^2}, \\ & d_{\Delta_{D'}} \frac{\sqrt{(d_w^2 - \Delta_{D'}^2)^2 + 4 d_w^2 y_0^2}}{d_w^2 - \Delta_{D'}^2}\;) \end{aligned}$$

Details are listed in appendix. Using this value, we can calculate a much more explicit score for each pair of matching. □

5.3 1-Opting

In Measuring step, we defined *o-score*. In this section, we design the 1-Opting step to divide the pre-matched pairs into groups of size 2, and perform optimization internally.

Let's first formulate the grouping problem.

Definition 3. (Group) Given a preliminary matching $M = \{m_i | m_i = (m_i.q, m_i.p), m_i.q \in Q, m_i.p \in P\}$, we define a group $g_i \in M \times M$. This group has size 2, and contains two different matched worker-task pairs. $g_i = (m_j, m_k)$, and $m_j.q \neq m_k.q$. Obviously we could calculate the obfuscation score *o-score* on the group g_i, which is *o-score*(g_i).

Since the definition requires that the worker from the two matched pairs (m_j, m_k) are different, we know the associated tasks are different.

Definition 4. (Grouping) Given a preliminary matching $M = \{m_i | m_i = (m_i.q, m_i.p), m_i.q \in Q, m_i.p \in P\}$, a grouping is a set of groups, $G = \{g_i | g_i \in M \times M\}$. All the groups should be exclusive, meaning they don't overlap. The union of all the workers from each group cover the entire worker set Q.

Definition 5. (Score of a grouping) Given a grouping $G = \{g_i | g_i \in M \times M\}$, we define the quality score of a particular grouping to be *score*$(G) = \sum_i$ *o-score*(g_i).

Definition 6. (Grouping problem) Given a preliminary matching $M = \{m_i | m_i = (m_i.q, m_i.p), m_i.q \in Q, m_i.p \in P\}$, the grouping problem asks for the highest scoring grouping $G_{opt} = \{g_i | g_i \in M \times M\}$, such that for any other grouping G', *score*$(G_{opt}) \geq$ *score*(G').

Theorem 5.2. *(Hardness of grouping problem) The grouping problem could be solved in polynomial time.*

Proof. The grouping problem is equivalent to finding the maximum weight matching on a general graph.

We construct the graph as follows. The vertex (node) set of the graph V correspond to each matching m_i from the preliminary matching M. Between every vertex pair in the graph, we form an edge. It is a complete graph.

We define the weight of the edges as the following. Because each node in the graph corresponds to a matched pair, any edge is between two matched pairs. Let's say the edge is between node m_i and node m_j. Naturally these two matched pairs could form a group as defined in Definition 3. Let us call this group $g = (m_i, m_j)$. For this group, we have the obfuscation score *o-score*(g). We define the weight of the edge between node m_i and m_j to be *o-score*(g).

Now we show that if we could obtain the maximum weight matching on the graph constructed, we obtain the optimal solution to the grouping problem defined in Definition 6.

Let's recall the problem definition of maximum weight matching on a general graph. A matching in a graph is a set of edges without common vertices. The maximum weight matching is a matching that has the maximum sum of the weight on the edges of the matching.

We now show that given a maximum weight matching of the general graph, we could obtain the optimal solution to the grouping problem. First, say the optimal matching we obtain is $M_{opt} = \{e_i | e_i = (m_j, m_k)\}$. The matching is a set of edges. We map all the edges in the matching to a group in the grouping, $G = \{g_i | g_i = (m_j, m_k), \forall e \in M_{opt}, e = (m_j, m_k)\}$. Because of such construction, each group is having an 1-on-1 mapping to each edge in the optimal matching. Because it is a matching, edges in the matching do not have any common vertices. This means each vertex is only covered by at most one edge in the matching. Because of the 1-on-1 mapping between the matching and the grouping, it means each node m_i is included in at most 1 group. This ensures the exclusiveness property of the grouping.

Now we show the grouping is also the optimal grouping. We prove it by contradiction. Let's assume there exists a different optimal grouping $G_{opt} \neq G$. They differ by at least one group, say $g' = (m_j, m_k)$. There are three cases.

Case I: both of the nodes are not in any other groups in G. In that case, because G has a list of groups whose nodes are different (not shared among groups), the edge between m_j and m_k could be added to the optimal matching, which contradicts to the fact that the matching returned is the maximum weight matching of the graph.

Case II: if one node, let's say m_j is in one of the groups $g_i = (m_j, m_p)$ in G, but m_k is not in any groups. In this case, if we replace the edge $e = (m_j, m_p)$ by edge $e' = (m_j, m_k)$, by the 1-on-1 mapping of the matching and the grouping, we know we c

ould have obtained a larger weight matching in the optimal matching. This contradicts to the fact that the matching returned is the optimal one.

Case III: both of the nodes m_j and m_k are in some of the groups in G, but they are not in the same group. So the case now is that in the optimal grouping, we have a group $g' = (m_j, m_k)$. In the grouping obtained by the matching, we have two groups, $g_1 = (m_j, m_p)$, $g_2 = (m_k, m_q)$. Basically that means in the grouping G, m_j and m_k are grouped with some other nodes, m_p and m_q. Because g' is in the optimal grouping, we know the obfuscation score on the two groups, *o-score(g')*+*o-score*$((m_p, m_q)) \geq$ *o-score*$((m_j, m_p)$ + *o-score*$((m_k, m_q))$. Because otherwise this grouping is not the optimal grouping. If we have this, then we know we could construct a matching by including the edge $e = (m_j, m_k)$ and $e = (m_p, m_q)$, and this new matching should have been a matching with larger weight, because the weight on the edges are defined to be the obfuscation score of the two nodes. This contradicts with the fact that the matching is a maximum matching.

For the maximum weight matching problem on a general graph, It is shown that it is among the 'hardest' problem that could be solved in polynomial time. It has $O(|V|^3)$ running time algorithm [6]. V here is the set of vertices of the graph constructed, which is the number of preliminary matched pairs, also roughly equal to the number of workers.

Note that finding the maximum weight matching on a bipartite graph could be done by the matrix multiplication algorithm, with a time complexity of $O(|V|^{2.37})$.

For the general graph we construct, because it is a complete graph, we transform the graph to a bipartite graph, which could be solved by the matrix multiplication algorithm. This concludes the proof, that the grouping problem could be solved in polynomial time.

□

1-Opting step is described in Algorithm 2. Note that at line 12, server runs the matrix multiplication algorithm to get a matching from a bipartite graph, which is transformed from a complete graph. At line 17, the server notifies the two matched pairs $(q_1, p_1), (q_2, p_2)$ and let them do decentralized communication to determine whether internal swapping could lead to optimization.

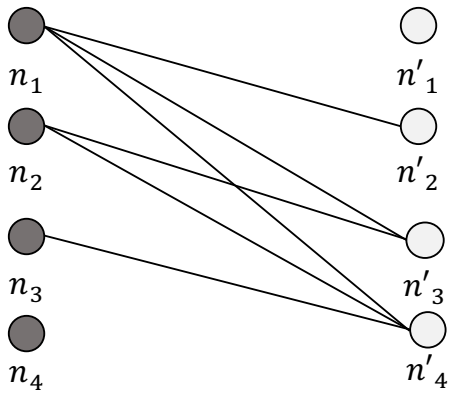


Figure 6: Illustration of graph transformation

The graph transformation in Algorithm 3 simply transforms a complete graph to a bipartite graph. The basic idea is to copy v nodes to $2v$ nodes, where the 1st node is copied to its copied node (denoted by n'_i). All the original node are in set L. All copied nodes are in set R. Then we connect every node in set L with every other node in set R, except for the node copy of itself. Also, we don't add redundant edges, meaning we only add edges between n_2 to node n'_3 and n'_4, but not to n'_1, because edge (n_2, n'_1) is the same as (n_1, n'_2). See Figure 3 for an example of a graph containing 4 nodes. We omit the rigorous proof, however it is straightforward to observe that a matching obtained on the transformed bipartite graph corresponds to a matching on the original complete graph.

The time complexity of Algorithm 2 is $O(n^{2.37})$, where n is the number of tasks. The analysis is as follows. Because the preliminary matched pairs is upper bounded by the number of tasks, so the matched pairs has at most n pairs. By the way we construct the graph, the number of nodes in the graph constructed is also n. The matrix multiplication algorithm at Line 12 runs in $O(|V|^{2.37})$ for the bipartite graph we built. So overall, the algorithm runs in $O(n^{2.37})$.

The time complexity of Algorithm 4 is $O(1)$, however it does require remote communication. We assume that communication delay is negligible.

Algorithm 2: 1-Opting step

Input: Preliminary matched pairs $M = \{m_i | m_i = (m_i.q, m_i.p), m_i.q \in Q, m_i.p \in P\}$ Q is the set of workers, P is the set of tasks.
Output: Improved matched pairs M', after 1-Opting step

1 Initialize an empty graph G
2 **foreach** $m_i \in M$ **do**
3 Add node m_i to G
4 **foreach** $m_j \in M$ **do**
5 Add node m_j to G
6 Get location of the 2 workers $l_{m_i.q}$, $l_{m_j.q}$
7 Get the obfuscated location of the 2 tasks $l'_{m_i.p}$, $l'_{m_j.p}$
8 *o-score* $:= \frac{dist(l_{m_i.q}, l_{m_j.q})}{dist(l'_{m_i.p}, l'_{m_j.p})}$
9 Add an edge$e$ between m_i and m_j
10 Set the weight of the edge $c(e) :=$ *o-score*
11 $G' :=$ Algorithm 3 on G
12 Run the matrix multiplication algorithm to obtain a matching M_G for the bipartite graph G'
13 **foreach** $e_i \in M_G$ **do**
14 Get two matched pairs(nodes) m_1 and m_2 from edge e_i
15 Get both workers $q_1 := m_1.q$, $q_2 := m_2.q$
16 Get both tasks $p_1 := m_1.p$, $p_2 := m_2.p$
17 Notify q_1, q_2, p_1, p_2 about the grouping
18 $response :=$ Algorithm 4 on q_1, q_2, p_1, p_2
19 **if** $response == True$ **then**
20 $m'_1 := (q_1, p_2)$
21 $m'_2 := (q_2, p_1)$
22 Insert m'_1, m'_2 to M'
23 **else**
24 Insert m_1, m_2 to M'
25 **return** M'

5.4 λ-Opting Framework

λ-Opting is an interative method. It runs for λ rounds, each time improving the quality of the matching.

λ is a hyper-parameter tune-able by the application server. In previous section, we know that the time complexity for running 1-Opting is $O(n^{2.37})$. So running it for λ rounds, the time needed is $O(\lambda \cdot n^{2.37})$, taking λ as an input. To control the running time, the application server could strike for a balance between the quality of

Algorithm 3: Graph transformation

Input: Complete graph G
Output: Bipartite graph G'
Initialize an empty graph G'
Get the size of the vertex set, $v := |G.V|$
foreach *node* $n_i \in G$ **do**
  Insert n_i to G
  Create a new node n'_i, insert it to G
foreach *node* $n_i \in G$ **do**
  foreach *node* $n_j \in G$ **do**
    $e :=$ edge between n_i and n_j
    $w := c(e)$
    Create an edge between node n_i and n'_j, with weight w
return G'

Algorithm 4: Internal communication

Input: Two matched pairs $(q_1, p_1), (q_2, p_2)$
Output: True or False
Workers q_1, q_2 notify their accurate locations to the tasks p_1, p_2
Task p_1 calculate true distance $cost_{p_1,q_1} = dist(l_{p_1}, l_{q_1})$, $cost_{p_1,q_2} = dist(l_{p_1}, l_{q_2})$ using his/her own accurate location
Task p_2 calculate true distance $cost_{p_2,q_1} = dist(l_{p_2}, l_{q_1})$, $cost_{p_2,q_2} = dist(l_{p_2}, l_{q_2})$ using his/her own accurate location
Task p_1 notifies $cost_{p_1,q_1}$ and $cost_{p_1,q_2}$ to worker q_1
Task p_2 notifies $cost_{p,2q_1}$ and $cost_{p_2,q_2}$ to worker q_2
Two workers exchange all the cost
if $cost_{p_1,q_1} + cost_{p_2,q_2} > cost_{p_1,q_2} + cost_{p_2,q_1}$ **then**
  $Response := True$
else
  $Response := False$
return $Response$

the matching and the running time, via the parameter λ.

We also introduce another hyper-parameter θ, as the threshold of quality gain. If the gain obtained from one step of 1-Opting is smaller than θ, the λ-Opting method stops. This is also for quicker convergence. The default value of θ could be set to 0, so the method keeps running for λ rounds, or until no quality gain could be achieved, whichever is earlier.

λ-Opting method is described in Algorithm 5.

6. K-SWITCH FRAMEWORK

In previous section, from a sub-optimal preliminary matched pairs, we group matched pairs into small group of size 2, and allow internal communication within the group to obtain better worker-task arrangement for lower travel cost.

It is natural to extend the idea to think about, could we generalize the idea from size 2 to size k?

In this section, we first formulate the k-Grouping problem and discuss its intractability (NP-hardness). Then we propose k-Switch, a game theoretic approach to address the problem.

Under our k-Switch framework, we first group the preliminary matched pairs into small group of size k, by a game theoretic approach. k is a hyper-parameter adjustable by the application server,

Algorithm 5: λ-Opting

Input: Parameters λ, θ, preliminary matched pairs $M = \{m_i | m_i = (m_i.q, m_i.p), m_i.q \in Q, m_i.p \in P\}$ Q is the set of workers, P is the set of tasks.
Output: Improved matched pairs M', after λ-Opting step
$i := 1$
$M_0 := M$
for $1 < i \leq \lambda$ **do**
  $M_i :=$ Algorithm 2 on M_{i-1}
  $Gain := \Psi(M_{i-1}) - \Psi(M_i)$
  if $Gain \leq \theta$ **then**
    Break
  $i := i + 1$
return M_i

normally ranging from 3 to 6. Then we utilize a k-Talk internal communication protocol designed by us, for better worker-task assignment within the group. After the decentralized communication and tasks swapping, we obtain an improved matching.

6.1 k-Grouping problem

Same as the λ-Opting framework, we start with a preliminary matched pairs of workers and tasks, $M = \{m_i | m_i = (m_i.q, m_i.p), m_i.q \in Q, m_i.p \in P\}$. Similarly, we hope to divide the matched pairs into small groups. Different from the fixed size of 2 in λ-Opting, now we hope to have a adjustable group size of k.

Let's formulate the k-Grouping problem.

Definition 7. (k-group) Given a preliminary matching $M = \{m_i | m_i = (m_i.q, m_i.p), m_i.q \in Q, m_i.p \in P\}$, we define a k-group $g_i \in M \times M \cdots \times M$. The k-group has size k, and contains k different matched worker-task pairs. $g_i = (m_{i_1}, \cdots, m_{i_k})$, and $\forall s, t, 1 \leq s < t \leq k, m_{i_s}.q \neq m_{i_t}.q$.

Definition 8. (*o-score* of a k-group) For a k-group, we define the obfuscation score *k-o-score* on the group g_i,
k-o-score$(g_i) = \sum_{1 \leq s,t \leq k}$ *o-score*$((m_{i_s}, m_{i_t}))$.

Recall that *o-score* is defined on a group of size 2. Here the *k-o-score* enumerates all the combinations of a 2-size subset of the k-size group, and sum up the *o-score* of all such subsets.

Definition 9. (k-grouping) Given a preliminary matching $M = \{m_i | m_i = (m_i.q, m_i.p), m_i.q \in Q, m_i.p \in P\}$, a k-grouping G_k is a set of k-group, as defined in Definition 7. All the k-Group should be exclusive, meaning they don't overlap, and the union of all the workers from each group cover the entire worker set Q.

Definition 10. (Score of a k-grouping) Given a k-grouping $G_k = \{g_i | g_i \in M \times M \cdots \times M\}$ defined in Definition 9, we define the quality score of a particular k-grouping to be *score*$(G_k) = \sum_i$ *k-o-score*(g_i).

Intuitively, a k-grouping divides the preliminary matched pairs ($|M|$ pairs) into group of size k (k-group). Each possible grouping has an associated score, which is the sum of all the *k-o-score* calculated from each of the k-group.

Definition 11. (k-grouping problem) Given a preliminary matching $M = \{m_i | m_i = (m_i.q, m_i.p), m_i.q \in Q, m_i.p \in P\}$, the grouping problem asks for the highest scoring k-grouping $G_k = \{g_i | g_i \in M \times M \cdots \times M\}$, such that for any other k-grouping G', *score*$(G_k) \geq$ *score*(G').

Theorem 6.1. *(Hardness of k-grouping problem) The k-grouping problem has no polynomial time approximation algorithm with finite approximation factor unless P=NP.*

Proof. We show a polynomial reduction of the Balanced Graph Partition problem to the k-grouping problem.

Balanced Graph Partition: The balanced graph partition problem could be stated in the following way. Given a graph $G = (V, E)$, weights on the edge $w(e) \in R$ for each $e \in E$. For an integer $p \geq 2$, a p partition is p disjoint subsets with equal size, $V = V_1 \cup \cdots \cup V_p$, $|V_1| = \cdots |V_p| = \frac{|V|}{p}$.

The decision problem is that given a positive integer W, could we have a p partition and if $E' \subset E$ is the set of edges that have two endpoints in the two different sets V_i, $E' = \{(v_i, v_j) \in E | v_i \in V_i, v_j \in V_j, i \neq j\}$, such that the sum of the weights on all such edges is smaller or equal to the given integer W, $\sum_{e \in E'} w(e) \leq W$?

We now show the reduction. There is a yes instance for the balanced graph partition problem if and only if there is a yes instance for the decision version of the k-grouping problem.

Let's define the decision version problem of the k-grouping problem. Given a positive integer J, do we have a k-grouping of the preliminary matched pairs $G_k = \{g_i | g_i \in M \times M \cdots \times M\}$, such that $score(G_k) \geq J$?

Note that the decision problem for the balanced graph partition corresponds to a minimization problem, while the decision problem for the k-grouping problem corresponds to a maximization problem.

For the k-grouping problem, we construct a graph $G' = (V', E')$ similar to the proof in Theorem 5.2. The vertex set V' corresponds to each matching m_i from the preliminary matching M. Between every vertex pair in the graph, we form an edge $e = (m_i, m_j)$. This edge corresponds to a size-2 group, so we could calculate the obfuscation score *o-score* on this group $g = (m_i, m_j)$. We define the weight to be $w(e) =$ *o-score*(g). We define T as the total sum of all the edges in the constructed graph $T = \sum_{e \in E'} w(e)$.

Now we show the reduction. If direction, if there is a yes instance to the k-grouping problem, we could find a yes instance to the p-partition problem. Say for a given integer J, we could obtain a k-grouping $G_k = \{g_i\}$, such that $score(G_k) \geq J$. Such grouping corresponds to a partition. For each k-group, it contains k nodes. We have $p = \lfloor \frac{|V'|}{k} \rfloor$ groups in total. $score(G_k) \geq J$, we look closely at what edges are included. Since $score(G_k) = \sum_i$ *k-o-score*(g_i), it sums up all the *k-o-score* of all the k-groups. And for each k-group, *k-o-score*$(g_i) = \sum_{1 \leq s,t \leq k}$ *o-score*$((m_{i_s}, m_{i_t}))$, summing up the edges weight between the nodes within a particular k-group. Let's take $W' = T - score(G_k)$. This corresponds to the total summation of all the edges, minus all the edges that are within a particular k-group. The weight sum W corresponds to all the edges which cross different k-groups. And because $score(G_k) \geq J$, we know the $W' \leq T - J = W$. So we find a p partition, with each partition of k nodes, and the sum of all the edges which have endpoints in different partition is smaller or equal to a given integer W.

Only-if direction: if there is a yes instance to the p-partition problem, then we could obtain a yes instance for the k-grouping problem. Given the graph and the p-partition, we know we have $V = V_1 \cup \cdots \cup V_p$, $|V_1| = \cdots |V_p| = \frac{|V|}{p}$. And for a given positive integer W, for the edges $E' \subset E$ that have two endpoints in the two different sets V_i, $E' = \{(v_i, v_j) \in E | v_i \in V_i, v_j \in V_j, i \neq j\}$, the sum of the weights on all such edges is smaller or equal to the given integer W, $\sum_{e \in E'} w(e) \leq W$. First we transform the graph to be a complete graph by adding edges between nodes that don't edges, and setting the edge weight to be 0. Since we know we have a yes instance, adding such 0 weight edges to the graph would not increase the cut edges (edges cross different partitions), and the transformed instance would still be a yes instance.

We then transform each V_i to a corresponding k-group g_i, where we set $k = \frac{|V|}{p}$. Each g_i contains all the nodes in V_i. Since we've transformed the graph to be complete graph, each pair of ndoes in g_i have edges between then, and for any edge $e = (n_i, n_j)$ we define *o-score*$((n_i, n_j) = w(e)$. So obviously for this constructed k-size g_i, we have the *k-o-score* defined, by summing up the weight of all the edges within the group. Finally, now we have a grouping $G = \{g_i\}$, where each g_i is transformed from the partition V_i, and has size k. We define the score for G to be $score(G) = \sum_i$ *k-o-score*(g_i). Because we know all the cross-partition edges sum, W' is less or equal to W, so all the in-partition edges sum, which is $score(G) = T - W' \geq T - W$, T is the summation of weight edges of all the edges in the graph. If we define $J = T - W$, we've obtained a yes instance to the grouping problem, $score(G) \geq J$.

This concludes the proof that *Balanced Graph Partition* $\leq_p$ k-grouping problem.

Since *Balanced Graph Partition* has no polynomial time approximation algorithm with finite approximation factor unless P=NP [1], the k-grouping problem has the same property. □

6.2 k-Switch, a game theoretic approach

To address the intractable k-grouping problem, we design k-Switch, a game theoretic approach for finding good quality groupings.

The main idea is that sometimes we could improve the quality of the matching by moving a certain item (preliminary matched pairs) from one k-group to another. If we treat each item as a single player, then each player tries to maximize its own utility by selecting the best target k-group to move to. If we give all players fair opportunities to compete/move, then after a certain period of chaos, the entire system moves towards equilibrium, which gives matching of acceptable quality in expected sense.

Given the preliminary matched pairs, we start with a random k-grouping G_0. Each group contains k items (matched pairs).

Given a k-grouping, an item m_i, g_s the source k-group that it currently belongs to, a target k-group g_t, and an item m_j in g_t, we define an exchange score *ex-score*(m_i, g_s, g_t, m_j).

Definition 12. (Item exchange) Given preliminary matched pair (also called an item),
$m_i = (m_i.q, m_i.p, m_i.q \in Q, m_i.p \in P)$, the source k-group g_s, a target k-group $g_t = \{m_{t_1}, \cdots, m_{t_k}\}$. For a given m_j which belongs to g_t, we have $\exists d, m_{t_d} = m_j$. We define the *item exchange* as follows. An *item exchange* is when we move item m_i from a source k-group g_s to a target k-group g_t, by replacing an item m_j in g_t. An *item exchange* could be denoted as a 4-element set (m_i, g_s, g_t, m_j).

Definition 13. (Item exchange score) Given an *item exchange* denoted by m_i, g_s, g_t, m_j. The item exchange score is defined as follows. *ex-score*$(m_i, g_s, g_t, m_j) =$ *k-o-score*$(\{m_j\} \cup g_s \setminus \{m_i\}) +$ *k-o-score*$(\{m_i\} \cup g_t \setminus \{m_j\}) -$ *k-o-score*$(g_s) -$ *k-o-score*(g_t).

The item exchange score is basically measuring how much gain we could have if we insert an element m_i into a target k-group g_t, by removing another element m_j from g_t. We put this element m_j back to the source group g_s. So we calculate the new *k-o-score* on both of the new groups after the item exchange, and calculate the difference of the new sum with the previous sum.

Algorithm 6: k-Switch

Input: Preliminary matched pairs $M = \{m_i | m_i = (m_i.q, m_i.p), m_i.q \in Q, m_i.p \in P\}$ Q is the set of workers, P is the set of tasks.
Output: Improved matched pairs M', after k-Switch

1 $G_0 :=$ Random k-grouping from M
2 $G', \delta :=$ Run *Item Exchange* in Algorithm 7 on G_0, M
3 **while** $\delta \neq 0$ **do**
4 $\quad G', \delta :=$ Run *Item Exchange* in Algorithm 7 on G', M
5 **foreach** $g_i \in G'$ **do**
6 $\quad M_{g_i} :=$ Run k-*Talk protocol* from Algorithm 8 on worker-task pairs from g_i
7 $\quad$ **foreach** $m' \in M_{g_i}$ **do**
8 $\quad\quad m_0 :=$ Original matched pair from M
9 $\quad\quad M' = \{m'\} \cup M \setminus \{m_0\}$
10 **return** M'

We design the game mechanism as follows. In each round, each item m_i has a chance to select the best target k-group to move to, and at the same time it picks the element m_j that it hopes to replace. In selecting the best item m_j to exchange groups, the subject m_i tries to maximize its utility for this round. The utility is measured by the *item exchange score* defined in Definition 13.

Then we continue to the next round. Each item has a chance to select other items and k-groups to do exchange. After each round, the k-grouping could be different from last round. If there is any difference, we continue to execute the game mechanism, until no exchange happens in a round. By then, equilibrium is reached.

The k-Switch is described in Algorithm 6.

At Line 2 and Line 4, k-Switch framework is calling its subroutine *Item exchange* defined in Algorithm 7 to perform one round of game mechanism, in which all the items could find the best destination group to move to, according to its measured utility.

At Line 6, the server notifies the worker-task pairs from a given k-group, letting them to do internal communication according to the designed k-Talk protocol in Algorithm 8. After the decentralized communication, the group notifies the server about the improved matching between them, and the server uses the new pairs to update the overall matching.

For each round of game mechanism, each item has a chance to find the best destination. It needs to run through all the other pairs to compare the utility score. So overall one round of game mechanism runs in $O(n^2)$, where $n = |Q|$, the number of workers.

It is shown in our experimental study that k-Switch typically terminates in less than 10 rounds, so the overall time complexity of the algorithm is $O(n^2)$.

6.3 k-Talk protocol

The decentralized communication protocol is designed for inner-group communication and better arrangement of worker-task pairs. Because the accurate location is communicated in a carefully controlled way, we could optimize the internal matching.

The detailed protocol is shown in Algorithm 8.

The first step, all workers communicate their location with the task simultaneously, and the task calculates the distance from itself to all the workers. Since there are k workers for each task to process, this step takes $O(k)$ time. Then all the tasks send their calculated distance to the workers. The workers select a random representative and share the distance matrix. Finally it runs a matching algorithm on the true distance matrix. Since there $O(k)$ nodes, we know the state-of-the-art matrix multiplication algorithm runs in $O(k^{2.37})$. The overall time complexity is $O(k + k^{2.37}) = O(k^{2.37})$.

Under the main k-Switch framework, the k-Talk protocol will only be called after the equilibrium of k-grouping is reached. Each group runs the protocol in parallel, so the overall time complexity of k-Switch is $O(n^2 + k^{2.37})$. In real-world application, k is normally a very small number (less than 10), the framework runs in $O(n^2)$.

Algorithm 7: Item Exchange

Input: All matched pairs M, a k-grouping $G = \{g_i | g_i \in M \times \cdots \times M, |g_i| = k\}$
Output: Improved k-grouping G', after 1 round of game. The utility gain δ

1 $\delta := 0$
2 **foreach** $m_i \in M$ **do**
3 $\quad g_s :=$ original k-group m_i belongs to in G
4 $\quad$ *max_t* $:= i$
5 $\quad$ *max_score* $:= 0$
6 $\quad$ **foreach** $m_j \in M$ **do**
7 $\quad\quad$ **if** $m_i \neq m_j$ **then**
8 $\quad\quad\quad g_t :=$ k-group m_j belongs to in G
9 $\quad\quad\quad$ *ex-score* $:=$ *ex-score*$(m_i, g_s, g_t, m_j) =$ *k-o-score*$(\{m_j\} \cup g_s \setminus \{m_i\})$+*k-o-score*$(\{m_i\} \cup g_t \setminus \{m_j\})$ − *k-o-score*(g_s) − *k-o-score*(g_t)
10 $\quad\quad\quad$ **if** *ex-score* $>$ *max_score* **then**
11 $\quad\quad\quad\quad$ *max_score* $=$ *ex-score*
12 $\quad\quad\quad\quad$ *max_t* $= j$
13 $\quad$ δ += *max_score*
14 $\quad g'_s = m_{max\_t} \cup g_s \setminus m_i$
15 $\quad g'_t = m_i \cup g_s \setminus m_{max\_t}$
16 $\quad G' = \{g'_s, g'_t\} \cup G \setminus \{g_s, g_t\}$
17 $\quad G = G'$
18 **return** G', δ

7. EXPERIMENTAL STUDY

We conducted extensive experiments on both real-world dataset and synthetic dataset to validate the effectiveness and efficiency of the two methods we propose, λ-Opting and k-Switch. We also compared our methods with the SCGuard method proposed in [15].

We introduce the experimental setup, including data collection/generation process, the general information of the dataset, and other experiment environment in Section 7.1. Then we present our experimental results and discussion in Section 7.2.

7.1 Experimental setup

The main parameters we select for the experiments are as follows. Number of workers $n = |Q| \in \{100, 200, \mathbf{500}, 1,000, 2,000, 3,000\}$.
Number of tasks $m = |P| \in \{100, 200, \mathbf{500}, 1,000, 2,000, 3,000\}$.
For λ-Opting, we test an early-stopping rounds $ES \in \{5, \mathbf{10}, 15, 20\}$.
For k-Switch, we test various values of $k \in \{3, 4, \mathbf{5}, 6, 7, 8\}$. The number in bold are the default parameter we select when test against other variables.

As for the differential privacy-preserving mechanism on task locations, geo-indistinguishability, we test various $\epsilon \in \{0.01, \mathbf{0.05}, 0.1, 0.3\}$. According to the obfuscation function $D_\epsilon(x_0)(x') = \frac{\epsilon^2}{2\pi} e^{-\epsilon d(x_0, x)}$, the smaller is the parameter ϵ, the flatter the distribution becomes.

Algorithm 8: k-Talk protocol

Input: A k-group $g_i = \{m_{i_1}, \cdots, m_{i_k}\}$
Output: The set of modified matched pairs M_{g_i} for the k-group g_i, $M_{g_i} = \{m'_{i_1}, \cdots, m'_{i_k}\}$

Let $q_1, \cdots, q_k$ be the workers from all the matched pairs $q_j = m_{i_j}.q$
Let $p_1, \cdots, p_k$ be the tasks from all the matched pairs $p_j = m_{i_j}.p$
foreach q_m **do**
  foreach p_n **do**
    Worker q_m notifies its location to task p_n
    Task p_n calculates the true distance using its accurate location $cost_{p_n,q_m} = dist(l_{p_n}, l_{q_m})$
    Task p_n sends $cost_{p_n,q_m}$ back to worker q_m
All workers q_m select a random representative worker q_r
All workers q_m send its knowledge of all the true distance $cost_{p_n,q_m}$ to q_r
The representative q_r runs an exact matching algorithm to obtain a new matching M_{g_i}
return M_{g_i}

Being flatter means that larger value of $d(x_0, x)$ is more likely to appear, compared to the case where ϵ gets closer to 1.

In our experimental setup, if ϵ is close to = 1, most of the noise points generated are within distance of 1-2 meters away from their original point. When $\epsilon = 0.01$, distance ranging from 10 meters to 50 meters are more likely to appear. We select the range to test ϵ very carefully based on the data distribution, which validates the effectiveness of our method when applied to the real-world data.

All experiments were performed on an Intel(R) Xeon(R) CPU E3-1220 v6 @ 3.00GHz machine, running CentOS Linux release 7.6.1810 (Core). The methods were implemented in Java.

We discuss how we collect the real-world data and how we generate the synthetic dataset. We conducted experiments on both datasets to validate the robustness of our method.

7.1.1 Data collection and generation

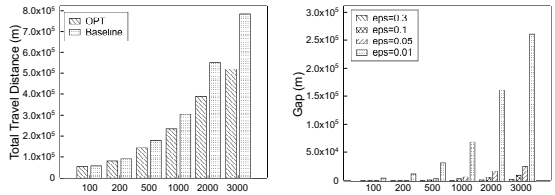
Real-world dataset. For the real-world data, we perform the experimental study on the taxi dataset from Didi Chuxing [19]. The dataset contains detailed trajectory samples from the taxi cabs running in Xi'an city, Shanxi province of China. On each day, the dataset contains around 30 million samples, with each data sample recording the GIS coordinates (latitude and longitude) of the taxi, the taxi ID, the passenger ID, and the timestamp.

Since there is no explicit drop-off location (the end of a trip) and the start location (the task location), we adopt the following approach. Sorted by the timestamp of the data samples, if one data sample shows that a certain taxi X is carrying passenger A, and the next data sample shows that this taxi X is carrying another passenger B, then we consider this sequence of actions to be dropping off a customer and picking up a new customer. Since the sampling rate of the dataset is around few seconds for each taxi, we consider it to be a good sampling rate, and it is reasonable to adopt our way of inferring drop-off and pick-up locations.

For the drop-off location, we use it as the worker location in our problem. For the pick-up location, we use it as the task location. Scanning through the dataset using this method, we could collect location information of workers and tasks. Given a time period (e.g., 30 minutes), we could produce a sample set of workers and tasks, and their corresponding locations, and we use this sample set as the input of our experimental study.

The sampled worker and task locations are all in GIS coordinates (latitude and longitude), and we convert it back to Cartesian coordinates in meters (in X and Y) using a common technique – Equi-rectangular projection, and shifting the lower-left boundary point back to (0,0). All of our points are within the range of $[0, 8, 003.5] \times [0, 8, 258.4]$.

Synthetic dataset. Based on the range of the real-world data, the synthetic data generation follows the same range. So we randomly sample points from the range $[0, 8, 000] \times [0, 8, 000]$.



(a) OPT vs. Baseline ($\epsilon = 0.01$) (b) Impact of ϵ on different input

Figure 7: Optimal (OPT) versus Preliminary Matching (Baseline)

7.2 Experimental Results

We perform evaluation based on the following evaluation metrics.

- Total travel distance (TTD). The quality of the assignment between worker and tasks is measured by the total travel distance for each matched task/worker pair. Refer to Section 2 for $\Psi(M)$.
- Improved travel distance (ITD). From the preliminary matching we obtain by running the matching algorithm on the observed obfuscated location, we could always obtain a TTD for a preliminary matching (Baseline). From the baseline, we measure the improved distance (ITD). ITD=$\Psi(M_{baseline}) - \Psi(M')$, where $M_{baseline}$ is the preliminary matching (baseline), and M' is the improved matching after running one of our proposed methods.
- Improved gap ratio (IGR) There exists an optimal task assignment if all the location information are accurate. The difference between the optimal matching and the baseline shows how much room of improvement in total we have, $Gap = \Psi(M_{baseline}) - \Psi(M_{OPT})$. Improved gap ratio (IGR)= $\frac{ITD}{Gap}$.
- Running time (RT). The running time of various methods. This is to evaluate the efficiency of the proposed algorithms.

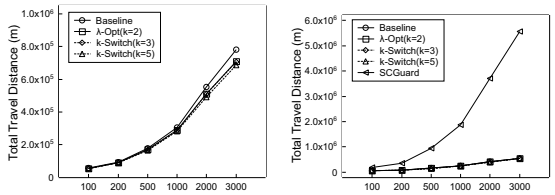
7.2.1 Baseline and the gap

A preliminary matching (we refer as baseline method throughout this section) could always be obtained by directly performing task assignment on the obfuscated locations of tasks. The methods we introduce in the paper first use Expected Distance as described in Section 4 to obtain a preliminary matching, and then use different ways described in Section 5 and 6 first to achieve optimization.

We call the preliminary matching the baseline method, and denote the task assignment (matching) obtained $M_{baseline}$. The actual location information of the tasks are not available to the application server, however as a way to measure performance, we use the actual locations to obtain an optimal task assignment. We refer to the optimal matching as M_{OPT}.

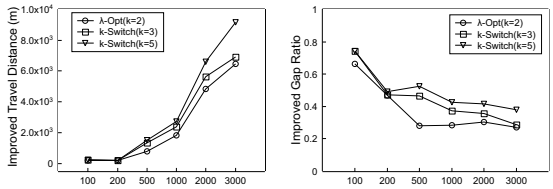
The best method could at most improve as much as the difference between the optimal solution and the preliminary matching. We denote the gap between the two as $gap = \Psi(M_{baseline}) - \Psi(M_{OPT})$

Figure 4 shows the gap on different input size. 100 refers to the input size of 100 workers versus 100 tasks, and 500 refers to the input size of 500 workers versus 500 tasks. In most part of the



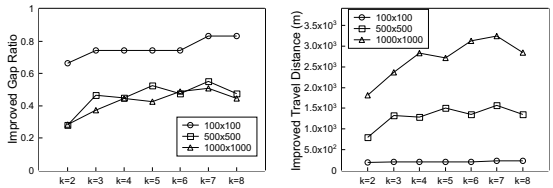
(a) TTD ($\epsilon = 0.01$) (b) TTD ($\epsilon = 0.05$)

Figure 8: Total travel distance (TTD) of various methods



(a) Improved travel distance (ITD) (b) Improved gap ratio (IGR)

Figure 9: Effectiveness measured by ITD and IGR



(a) IGR on different k (b) ITD on different k

Figure 10: Performance comparison on different k

experimental study, we select the worker-task size to be 100 vs. 100, 200 vs. 200, 500 vs. 500, 1,000 vs. 1,000, 2,000 vs. 2,000, and 3,000 vs. 3,000 to make the results concise. When the number of workers and tasks are not the same, our methods are effective as well.

There exists a significant gap between the optimal solution and the baseline. As the input size gets larger, the gap is growing larger. The gap could constitute roughly 20% - 30% of the total travel distance (TTD) of the baseline solution.

We also look the impact of the value of ϵ on the gap. The value of ϵ plays an important role in the obfuscation process. The smaller the value is, the farther an obfuscated location is from its original location. From Figure 4(b) we could see that, the larger is the value of ϵ, the less is the degree of obfuscation. And thus, the baseline method could achieve a matching which is very similar to the optimal matching. On the other hand, when the value of the ϵ is smaller, e.g., 0.01, there is a significant gap between the optimal solution and the baseline.

7.2.2 Effectiveness

Total travel distance

We first look at the total travel distance (TTD). As shown in Figure 5(a), all of our proposed methods obtain better matching then the baseline method. k-Switch ($k = 5$) achieved the smallest (best) TTD among all the methods.

When the size of the input gets larger, our method perform better, obtaining significant (close to 20%) improvement over the baseline method.

We tend to compare the methods we propose with the online method SCGuard [15]. For $\epsilon = 0.05$, the TTD of the matching obtained by SCGuard and our methods are shown in Figure 5(b). Our methods are order-of-magnitude better than SCGuard. Because of the TTD value of SCGuard is relatively large, different curves for different methods are overlapping with each other in the figure.

On one hand, SCGuard is an online method, which does not consider global optimum. On the other hand, our problem formulation assumes the location of the worker to be accurate, while the online method considers the cases when both the worker and the task obfuscate their locations. We believe our problem formulation reflects the real-world application better, because the leading crowdsourcing platform in U.S. and China have their proprietary vehicles, and it is location of workers are always accessible. Furthermore, our framework could be easily extend to the cases when both the worker and task obfuscate their locations.

When the obfuscation level is high, $\epsilon = 0.01$, SCGuard fails to return meaningful results. While as shown in Figure 5(a), our methods return more favorable results. It further validates the robustness of our methods against higher level privacy requirement.

Improved travel distance and improved gap ratio

As shown previously in Figure 4, the gap between the optimal matching and the baseline method could vary. For example, if the privacy requirement is low (ϵ is close to 1), then the gap between baseline and the optimal solution could be small. While when the privacy requirement is strict, the gap could be huge.

If the gap itself is small, looking at the absolute value of TTD may not give the full picture of which method is more effective than the others. Thus we use the improved travel distance (ITD) to measure the relative distance improvement over the baseline method.

As shown in Figure 6(a), all of our methods show significant improvement across all input sizes. When the input size gets larger, it makes sense that the improved distance gets larger, as ITD sums over all the matched pairs.

It is interesting to see that k-Switch ($k = 5$) performs the best across all input sizes. k is the size of the small group in which internal communication is enabled, it is sensible that the larger the group is, the more effective is the method.

When compare λ-Opting with k-Switch, the results show that k-Switch ($k = 3$) outperforms λ-Opting. Actually, λ-Opting could be considered a generalized version of k-Switch when $k = 2$, but the internal grouping methods are completely different.

In addition, we use improved gap ratio (IGR) to capture the percentage of gap that the method fills. The relative range of 0 to 1 gives us a better picture about the effectiveness of the methods. As shown in Figure 6(b), the best method, k-Switch ($k = 5$) could fill around 60% of the gap, k-Switch ($k = 3$) could fill around 50% of the gap, and λ-Opting could fill around 30% of the gap.

Different values of k

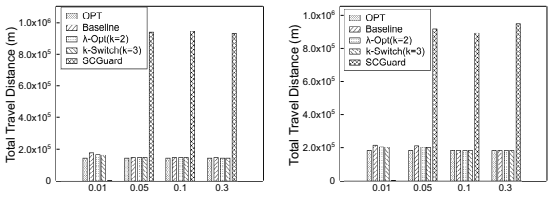
We then look at k-Switch in more details. We test different values of k and test the performance of the method. As shown in Figure 7(a), the general trend is that the larger the value k is, the better performance we could obtain. It makes sense, as the parameter k controls the group size in which internal communication is enabled. So it makes that the larger the group is, more workers could be involved to exchange assigned tasks, and obtain a better matching, which is closer to the optimal task assignment.

It is also shown in the figure that the value of 5 could be the best value to select for k, as the performance tends to level off beyond $k = 5$. So beyond this point, even if we could increase the value of k, the cost/benefit trade-off could be marginal.

We could see that for the improved travel distance (ITD) metric shown in Figure 7(b), for different value of k, the distance value itself tend to be stable. This is due to the value scale of the distance. Since the travel distance itself is large, from a visualization point of view, the effectiveness of different values of k is not as clear as the one shown in Figure 7(a).

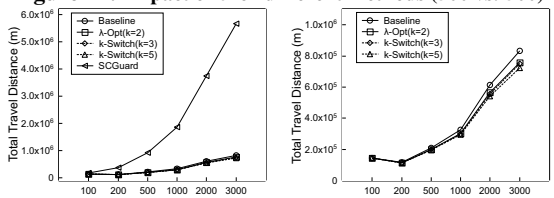
Different values of ϵ

The value of ϵ plays an important role in the obfuscation process.



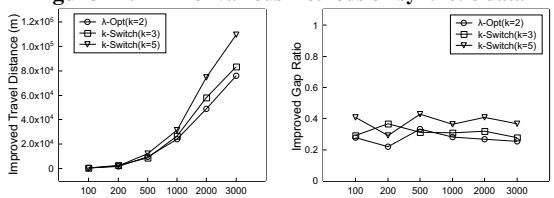
(a) Real-world data (b) Synthetic data

Figure 11: Impact of ϵ for different methods (500 vs. 500)



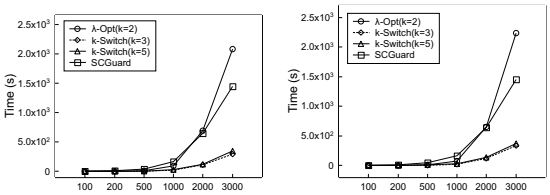
(a) TTD (b) TTD, excluding SCGuard

Figure 12: TTD of various methods on synthetic data



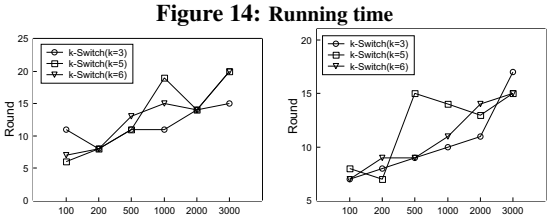
(a) ITD on synthetic data (b) IGR on synthetic data

Figure 13: ITD and IGR on synthetic data



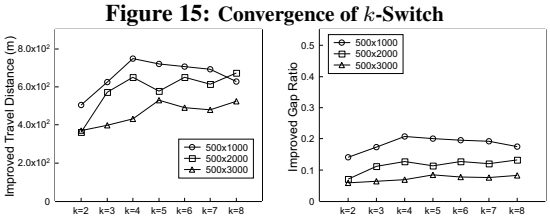
(a) Real-world data (b) Synthetic data

Figure 14: Running time



(a) Real-world data (b) Synthetic data

Figure 15: Convergence of k-Switch



(a) Improved total distance (b) Improved gap ratio

Figure 16: Effectiveness on imbalanced data input

The smaller the value is, the farther an obfuscated location is from its original location.

As shown in Figure 8, one could see that our methods dominate the performance of SCGuard, across different values of ϵ. Another interesting note is that when $\epsilon = 0.3$, the baseline and the optimal solution are very close, so the gap is small. Our methods are more effective when the privacy requirement is stricter ($\epsilon = 0.01$). The same pattern could be observed on the synthetic data, as shown in Figure 8(b).

Synthetic data

The results obtained by running experiments on synthetic dataset align with the results on the real-world data.

However, there are some interesting points worth noting. The improved gap ratio (IGR) tend to be quite stable across different data size. As compared to the real-world data shown in Figure 6(b), the larger the size of the input is, the lower the IGR is. It makes sense, as the input size correlates with the difficulty of the problem. While our methods are considerably much more robust across different input size, as shown in Figure 10(b).

Our methods still dominate the performance of SCGuard on the synthetic dataset, as shown in Figure 9(a). k-Switch ($k = 5$) performs the best.

7.2.3 Efficiency

We record the running time for all the methods for both real-world dataset and synthetic dataset. The results are shown in Figure 11.

Quite surprisingly, λ-Opting is significantly slower than the other k-Switch methods. Though λ-Opting is a polynomial algorithm and has its running time bounded by $O(n^{2.37})$, where n is the number of workers, as the input size gets larger, it becomes slower. k-Switch does not have a guarantee on the round it takes to reach equilibrium, but according to the experimental results, it is around 3 times faster than λ-Opting and SCGuard.

When the value of k gets larger, it is expected that the method takes longer running time. As shown in the figure, $k = 5$ is a bit slower than $k = 3$, but not by too much. Compared to the magnitude of the running time of SCGuard, k-Switch is very efficient.

Convergence of k-Switch

As a validation of the efficiency of k-Switch method, we record the number of rounds it takes to reach the equilibrium. Recall that, in each round of k-Switch, each item could move to other k-groups. The equilibrium is reached if no item is moving during the current round.

The result is shown in Figure 12. Normally it takes around 15-20 rounds for the method to converge. In general, the larger the input size is, the more rounds it takes to reach the equilibrium. But even for large size of 3,000 versus 3,000, it takes less than 20 rounds to reach the equilibrium.

7.2.4 Imbalanced workers and tasks

For the purpose of being concise, we only include experimental results on input dataset that has the same number of workers and tasks. Nevertheless, our method generalizes well on the imbalanced cases, where the number of workers and tasks are not the same.

As shown in Figure 13, we test our methods on various input combinations, e.g., 500 tasks versus 1,000 workers, 500 tasks versus 3,000 workers. Here, we refer to the λ-Opting method as $k = 2$. As we could see, all of our methods could cover around 20% of the gap between the optimal solution and the baseline. When the imbalance between the workers and tasks gets larger, the less is the gap. Because of this, in general, our method works better on the input of 500 versus 1,000, as compared to the input of 500 versus 3,000.

7.2.5 Results summary

We conducted extensive experiments on both real-world dataset and synthetic dataset.

From effectiveness point of view, our methods achieve significant travel distance improvement over the baseline. If we consider the gap between the best possible solution and the baseline, our methods could fill around 40% of the gap. In terms of total travel distance, our methods achieve order-of-magnitude improvement over previous proposed online method SCGuard.

While achieving the best effectiveness, k-Switch maintains efficiency on different input sizes. It is faster than λ-Opting, and around 3 times faster than SCGuard.

8. CONCLUSION

In this work, we formulate the traditional spatial crowdsourcing tasks assignment problem in a privacy-preserving setting. Traditional task assignment problem takes a set of reported locations of tasks and workers, and produces the best quality matching. The matching minimizes the total travel distance of workers to their assigned tasks. Our newly formulated privacy-preserving setting uses the most commonly adopted technique – *Geo-indistinguishability*, a scheme that could protect user location in a differential private way, to obfuscate tasks' location. The obfuscation scheme adds a random Laplace noise to reported location of tasks, which perturbs the actual location to a 'fake' but nearby location. The overall obfuscation brings challenge to the traditional task assignment problem.

We did the worst-case analysis on this new problem. We found that the there is no exact optimal algorithm for the new problem, because the obfuscation method is random and once the location could be obfuscated arbitrarily far away from its actual location, which makes the matching result arbitrarily bad. Despite the challenge and the difficulty of the problem, we device a Expected distance metric, which improves the observed distance between worker and task by probabilistic analysis. Then we develop a k-Switch framework to further improve the matching produced by the Expected distance metric. The initial matching are sub-optimal because we could only observe the reported (not actual) locations of tasks. The framework allows small subsets (usually size from 2 to 4) of workers to communicate with each other and exchange actual locations of tasks. If the interchange of tasks between workers lead to better assignment (lower travel distance), k-Switch allows workers to do such optimization.

The experimental study shows that k-Switch could cover up to 42% of the performance gap between baseline solution (matching produced by just observing the obfuscated locations) and the underlying optimal matching. The method is stable and efficient across various input sizes.

The future direction beyond this work would be considering more application scenarios in the spatial crowdsourcing platform. For example, the problem may become more challenging when we allow ride-sharing in tasks assignment. The privacy-preserving techniques could be applied in various stages of the applications, which lead to many interesting potential areas of research.

9. REFERENCES

[1] Andreev, K., Racke, H.: Balanced graph partitioning. Theory of Computing Systems **39**(6) (2006)

[2] Andres, M.E., Bordenabe, N.E., Cjhatzikokolakis, K., Palamidessi, C.: Geo-indistinguishability: differential privacy for location-based systems. In: Proceedings of the 2013 ACM SIGSAC conference on Computer and communications security, pp. 901–914 (2013)

[3] Chen, Z., Fu, R., Zhao, Z., Liu, Z., Xia, L., Chen, L., Cheng, P., Cao, C.C., Tong, Y.: gmission: A general spatial crowdsourcing platform. Proceedings of the VLDB Endowment **7**(13) (2014)

[4] Christin, D.: Privacy in mobile participatory sensing: Current trends and future challenges. The Journal of Systems and Software **116**(57-68) (2016)

[5] Dwork, C.: Differential privacy. International Colloquium on Automata, Languages, and Programming, LNCS **4052**(1-12) (2006)

[6] Edmonds, J.: Paths, trees, and flowers. Canadian Journal of Mathematics **17**(449-467) (1965)

[7] gMission: `http://gimission.github.io`

[8] Goldberg, A.V., Kennedy, R.: An efficient cost scaling algorithm for the assignment problem. Mathematical Programming **71**(153-177) (1995)

[9] Isaac, M., Frenkel, S.: Facebook security breach exposes accounts of 50 million users. `https://www.nytimes.com/2018/09/28/technology/facebook-hack-data-breach.html`

[10] Kazemi, L., Shahabi, C.: Geocrowd: enabling query answering with spatial crowdsourcing. In: Proceedings of the 20th International Conference on Advances in Geographic Infomation Systems (SIGSPATIAL), pp. 189–198 (2012)

[11] Kim, S.H., Lu, Y., Constantinou, G., Shahabi, C., Wang, G., Zimmermann, R.: Mediaq: mobile multimedia management system. In: Proceedings of the 5th ACM Multimedia Systems Conference, pp. 224–235. ACM (2014)

[12] Leong, H.U., Yiu, M.L., Mouratidis, K., Namoulis, N.: Capacity constrained assignment in spatial databases. In: Proceedings of the 2008 ACM SIGMOD International Conference on Management of Data, pp. 15–28 (2008)

[13] Munkres, J.: Algorithms for the assignment and transportation problems. Journal of the Society of Industrial and Applied Mathematics **5**(32-38) (1957)

[14] Shin, M., Cornelius, C., Peebles, D., Kapadia, A., Kotz, D., Triandopoulos, N.: Anonysense: a system for anonymous opportunistic sensing. Journal of Pervasive Mobile Computing **7**(16-30) (2010)

[15] To, H., Shahabi, C., Xiong, L.: Privacy-preserving online task assignment in spatial crowdsourcing with untrusted server. In: Proceedings of the 34th IEEE International Conference on Data Engineering (ICDE), pp. 833–844 (2018)

[16] Torosluk, I.H., Ucoluk, G.: Incremental assignment problem. Information Sciences **177**(1523-1529) (2007)

[17] Vu, K., Zheng, R., Gao, J.: Efficient algorithms for k-anonymous location privacy in participatory sensing. In: Proceedings of the 31st IEEE International Conference on Computer Communications (INFOCOM), pp. 2399–2407 (2012)

[18] Wang, C.J., Ku, W.S.: Anonymous sensory data collection approach for mobile participatory sensing. In: Proceedings of the 28th IEEE Conference on Data Engineering Workshops (ICDEW), pp. 220–227 (2012)

[19] Xu, Z., Li, Z., Guan, Q., Zhang, D., Li, Q., Nan, J., Liu, C., Bian, W., Ye, J.: Large-scale order dispatch in on-demand ride-hailing platforms: A learning and planning approach. In: KDD '18 Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pp. 905–913 (2018)