# 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 the location data of users 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 obtain an effective assignment. Without considering the obfuscation of locations, directly applying existing methods may 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 analysis on the obfuscation locations, we propose an *Expected distance* metric to measure the distances among tasks/workers. Then we design a novel $\lambda$-Opting algorithmic framework and a game theoretic $k$-Switch framework. The frameworks use the aforementioned *Expected Distance* metric to generate an initial matching between workers and tasks, and then improves the matching by switching the tasks among workers. Extensive experimental results validate the effectiveness and efficiency of $k$-Switch.

**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


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

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. To support such collaborations, spatial crowdsourcing platforms have emerged to distribute spatial tasks among participants. Acting as workers, the participants move physically to specified locations to accomplish the tasks [11]. 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 select ad-hoc tasks such as taking photos, staying in line for food/good purchasing, and etc. A typical spatial crowdsourcing platform (e.g., gMission [8, 4] and MediaQ [12]) 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 application 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 leaking their private information. 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 [10]. 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. Theoretically, 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 [5]. 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 [18]. If we worry about the privacy of the sensor readings collected by workers, then anonymity techniques could be applied so that the sensing result from each worker is mixed with others, and subsequently neither the central server nor the end user could identify the original source [15, 19].

To prevent privacy leakage like the Facebook incident, we adopt the privacy-protection setting in which the centralized server could be un-trusted. In the traditional spatial crowdsourcing applications,

users upload their true locations, which are used by the centralized server to find a worker-task assignment. In our privacy-preserving framework, task requesters (customers) send only the obfuscated locations to the server. The location obfuscation process is done by adding random noise drawn from Laplace distribution, which ensures geographical differential privacy for users [3]. We propose a framework to properly match workers with tasks, based on the obfuscated locations of the tasks and the exact locations of workers, to minimize the total travel distance of workers. Though Hien To et. al. [16] have taken the privacy issue into concern, they adopt the online setting where tasks arrive one by one. Thus, their solution may not yield optimal result on our batch-based setting, which has been more widely studied [13].

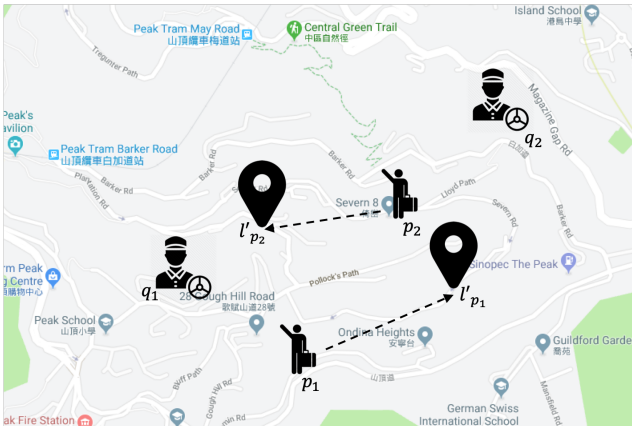



**Figure 1: An illustrative example**

Figure 1 is an example of our problem. Like the traditional spatial task assignment problem [13], we have a number of workers and tasks diversely distributed in the space. However, the locations of tasks are obfuscated. The reported location of each customer may be different from the true one. Note that in our problem setting, the worker locations are truly reported, as we assume that workers are employed by the application platform (e.g., the taxi drivers follow instructions from the taxi company). We will explain the details in Section 6 about how to adapt our proposed framework to the cases when the worker locations are also obfuscated.

By observing only the obfuscated task locations, 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, while the actual distance between them is large. The challenge is how to match workers with tasks, based on the obfuscated task locations, 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 adversarial. We propose the *Expected distance* metric, which is based on the probabilistic analysis on the relationship between the true distance and the observed distance between tasks and workers. Then we use the *Expected distance* based Successive Shortest Path Algorithm (SSPA) matching to obtain a baseline matching (a preliminary matching) of tasks and workers.

To further improve the matching quality, measured by the total travel distance of the workers to the task locations, we proposed $\lambda$-Opting framework. It first quantifies the likelihood of wrong assignment of task and workers in the baseline matching. Then based on the likelihood, the framework divides all the matched pairs into small groups of 2 worker-task matched pairs. Within the group, communication is enabled, so tasks could report their actual distance with the workers. If swapping of workers achieve lower total travel distance, such action is taken within the group. We iterate the process for $\lambda$ times to achieve optimization.

To further build on the $\lambda$-Opting framework, we propose a game theoretic $k$-Switch framework. Instead of dividing all the preliminary matched pairs from the baseline solution into group size of 2, now we try to divide them into group size of $k$ ($k \geq 3$). During the dividing process, we try to maximize the probability of assignment mistakes made in the preliminary matching. We show that finding the best grouping in the dividing process is NP-complete, and could not approximated with any finite factor unless *P=NP*. Thus, we carefully design a game theoretic mechanism in which matched pairs could move to maximize its own utility. After equilibrium is reached, which means when $k$-Switch finds the optimal grouping, the groups could use our proposed $k$-Talk protocol for internal communication, tasks swapping, and total travel distance optimization.

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 probabilistic analysis on the problem and show that the problem could not be solved exactly. We then derive an *Expected Distance* metric, which helps obtain a baseline matching with satisfactory quality in expectation in Section 4 .
- We propose a novel algorithmic $\lambda$-Opting framework in Section 5, which selects the best grouping of matched pairs of size 2 in an iterative manner, and allow workers to change tasks within the group.
- We further extend $\lambda$-Opting framework, and design a game theoretic $k$-Switch framework. It divides the matched pairs into group size of $k$ ($k \geq 3$), and utilizes a privacy-preserving communication protocol for internal tasks swapping and optimization. The framework is introduced in Section 6.
- Finally, we have conducted extensive experiments to validate the efficiency and effectiveness of the proposed 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 settings for spatial tasks assignment problem. Then we briefly introduce the essential principles of *geo-indistinguishability*, namely geographical differential privacy techniques. The background described in this section prepares us to formally define the privacy-preserving spatial crowdsourcing tasks assignment problem.

### 2.1 Spatial task assignment

Traditional spatial task assignment consider a group of tasks (e.g. taxi riders, fast-food ordering customers) as a set $P$. Each task $p_i \in P$ corresponds to a spatial location $l_{p_i}$, represented by a tuple $l_{p_i} = (x, y)$ where $x, y \in R$. Each task (or service requestor) requires a worker (e.g. taxi drivers, fast-food courier) to serve his/her request. Each worker $q_j \in Q$ also corresponds to a spatial location $l_{q_j}$, represented by $l_{q_j} = (x, y)$ where $x, y \in R$. A worker 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 travel cost of the matching. The travel cost is the summation of the distance between all worker-task pairs.

Let $\gamma$ 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 algorithm 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.

Existing works normally define the following cost function. For each possible matching $M$,

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

$\Psi(M)$ simply sums up the distance of the worker-task pairs of the assignment. Sometimes we call this minimization setting as the MaxMin matching problem, because on one hand, it produces the maximum cardinality of the matching, and on the other it minimizes the cost of all assignments. Linking back to the real-life application, in the ride-hailing example, the best assignment ensures serving the maximum number of customers, while minimizing the total pick-up travel distance among all taxi cabs.

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 of customer nodes on one side and workers nodes on the other [13]. There are a variety of existing algorithms for the MCF problem. Hungarian algorithm [14, 17], the cost scaling algorithm [9], 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 [13]. Let $V, E$ denote the nodes and edges of the graph, 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 by Dwork [6]. It is a notion of privacy protection such that when a single item in the original database $D$ is modified, the altered database $D'$ is still required to produce aggregate query result 'similar' to that of the original database $D$. By 'similar', it means that the difference of aggregate query results returned by $D$ and $D'$ is bounded by $e^{\epsilon}$.

As location-based service (LBS) becomes prevalent, how to protect the location information of individual users has drawn great attention from the research community. *k-anonymity* is a 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 methods using expectation of distance error and cloaking, are effective only under a certain assumption on the prior knowledge of an attacker. If a privacy adversary attacks in a way different from the assumed prior knowledge, these methods may fail to provide strong privacy protection.

A notion of *geo-Indistinguishability* is proposed by extending the differential privacy notion to the spatial domain [3]. It is proven that it provides strong privacy protection given any prior knowledge assumption. The notion is formally defined in Definition 3.1 in [3], which is listed as follows.

**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 $p_i$ |

**Definition 1.** (Geo-indistinguishability) For all true locations $x, x'$, A mechanism $M$ satisfies $\epsilon$-*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 the distributions associated with $x$ and $x'$.

A probabilistic mechanism, or an obfuscation scheme, essentially changes the spatial location of interest $x$ to another point in the planar area, with a probability distribution. The above definition of $\epsilon$-*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 in turn upper bounded by $\epsilon r$.

One particular mechanism satisfying the above property is drawing random noise from the planar Laplace distribution [3]. 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

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

where $\frac{\epsilon^2}{2\pi}$ is the normalization factor. We use the above noisy location density function to generate obfuscated locations from the true 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 taking another look at the illustrative example shown in Figure 1. At this time, there are two tasks $p_1$ and $p_2$ waiting to be served (for example, waiting for the taxi drivers to pick them up). There are two workers $q_1$ and $q_2$ available to the tasks. The figure demonstrates the perspective from the application server. The observed locations for $p_1$ and $p_2$ are $l'_{p_1}$ and $l'_{p_2}$. The application server needs to determine how to assign workers $q_1$ and $q_2$ to customers to minimize their travel distance.

As the figure shows, the true locations are denoted as $p_1$ and $p_2$. These locations are not observable by the application server, because the *geo-indistinguishability* Laplacian noise has been added to these locations to protect the customer privacy. For $p_1$, a random point is drawn from the planar Laplace distribution, with the probability density function of $\frac{\epsilon^2}{2\pi}e^{-\epsilon d(l_{p_1},x)}$, and becomes 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 *true locations* of the tasks, not the observed ones. Regarding the illustrative example above, if we assign the tasks to their closest workers based on the observed distance, then we may assign task $p_2$ to worker $q_1$, and task $p_1$ to worker $q_2$. However by looking at the picture, we know that the optimal solution should assign $p_1$ to $q_1$, and $p_2$ to $q_2$, because such matching has a lower total travel distance for the workers.

Now we define the Privacy-preserving Spatial Crowdsourcing Tasks Assignment problem (PSCTA 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 $\epsilon$ 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 travel distance of the workers,

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

Remark: as for the ultimate objective, the true location of the customer $l_p$ is used for measurement. Note that the true location is not available in the given input.

## 3.1 Worst-case

**Theorem 3.1.** *(Worst case of PSCTA problem) In the worst case of PSCTA problem, the ratio of the matching obtained by any given method with the optimal solution is arbitrarily bad.*

*Proof.* 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 know more about the original location of the task.

Since the perturbation could drag an original point to anywhere on a plane, an arbitrarily far away point from a worker could be potentially moved to be very close. We do not have any other prior knowledge, 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.

□

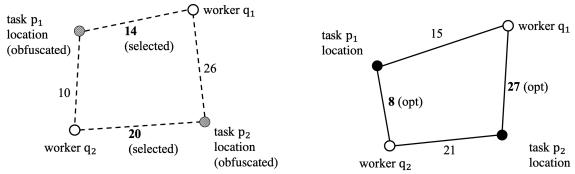
Nevertheless, based on the statistical analysis to be described in later sections, we could get an expected distance between the worker and the task. Based on the expected distance, we could produce a baseline matching. From the baseline matching, we develop other effective frameworks to obtain further improvement and produce good quality matching.

# 4. EXPECTED DISTANCES BASED MATCHING

We start with the state-of-the-art Successive Shortest Path Algorithm (SSPA) for the batch-based task assignment problem. As we will show in the following subsection, SSPA could not be applied to our new problem, which is formulated for privacy-preserving. So, we propose our *Expected Distance* based metric to modify the SSPA method, which considers the obfuscation of task location.

## 4.1 Existing solution

Because the obfuscated location of tasks, the server could only calculate the cost based on the observed location. Figure 2 shows a case when mismatching happens when SSPA is directly used based on the obfuscated location. The minimum cost that we could obtain by using the obfuscated location reported by the tasks is $14 + 20 = 34$, which is smaller than the other option $10 + 26 = 36$. However, based on the accurate locations of tasks shown in Figure 2, $q_1 - p_2, q_2 - p_1$ is the best choice.




(a) Use SSPA directly on the obfuscated locations (b) Minimal cost matching for the true locations

**Figure 2: SSPA fails on the privacy-preserving setting**

## 4.2 Statistical solution

With obfuscated locations of the tasks users, a statistical model is needed to improve the existing SSPA algorithm. We try to analyze the probability distribution of the actual distance between worker and tasks, given the observed distance of the reported location of tasks and workers, considering the possible obfuscation on the task location.

We did an approximation to show SSPA method could be improved. Then we introduce how to calculate the expected distance between tasks and workers, using both close-form and empirical analysis.

**Approximated BND.** Due to difficulty in analyzing planar Laplacian distribution directly, the Bi-variate Normal Distribution (BND) is used as an approximation. We first derive that the expected distance is different from the observed distance.

For better approximation, the mean and variance of BND is chosen to be the same with the planar Laplace distribution. The distribution could then be written as $BND(\mu, \Sigma)$, which is 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 $y_{l^*_{u_1}}$ refer to the two dimensions of the obfuscated location. $\sigma = \frac{\sqrt{2}r}{\epsilon}$ refers to the standard deviation of the planar Laplacian distribution for privacy-level $(\epsilon, r)$.

**Algorithm 1:** EDMA 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** $l_{u_i^*} \in L_u^*$ **do**
  **foreach** $l_{s_j}$ **do**
    $dis^* := \left| l_{u_i^*} - l_{s_j} \right|$
    $dis := D[dis^*]$
    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**
  $u_{P_i}, s_{P_i} = Pi$
  $sum_{cost} += \left| l_{u^{P_i}} - l_{s_{P_i}} \right|$
**return** $sum_{cost}$

Then we derive the distribution for the distance. We've only included the major steps. Please refer to the technical report [1] for the detailed derivation.

We could obtain the expected distance of real location as follows.

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

With the expected distance instead of the observed distance, we could obtain a more reasonable distance matrix for the the SSPA method.

To accelerate the calculation, we could store the expected distance in a pre-computed list. In this case, we focus on the integration with $\theta$,

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

With this, no matter which direction the task is facing the worker, the result wouldn't change. So we only need to calculate different expectation based on different distance between the two locations. A two-dimension calculation could be reduced to one dimension.

Since the expectation of $d^2$ is $\mu_{d^2} = d_{u^*,s}^2 + 2\sigma^2$, we have another interesting property deducted from the BND approximation,

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

For larger distance we can use the obfuscated distance instead of the expected distance. For smaller distance, our method tends to be more effective. At the same time, when the variance goes to 0, meaning the obfuscation level is minimal, the expected distance becomes the same with the observed distance. This also validates our probability analysis above.

Using the previous analysis, we derive an Expected Distance-based Matching Algorithm (EDMA). The major part for matching is based on SSPA and we mainly modify the preparation part and the examination part. Algorithm 1 presents EDMA.

In this algorithm, we iterate over each pair of user and server. The obfuscated distance is first calculated and used as a key to search in the pre-computed list storing the expected distance.

After the edges are added, it then calls SSPA to generate all the matched pairs. The true total travel distance sum over all the matched pairs is returned.

# 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 improve the matching between workers and tasks, which is measured by the total travel distance of the workers. There are three major steps in the λ-Opting Framework.

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

Step 2, 1-Opting step. The server notifies the selected small groups and allow them to improve the matching quality within the group, via internal communication. Internal communication allows the workers inside the group to access the true locations of the tasks, workers could swap tasks if it could obtain a better matching. The selected small groups then inform the server of the swapping results. This phase is called 1-Opting.

Step 3, λ-Opting. The server repeats the Measuring step and the 1-Opting step for $\lambda$ times, each time looking for groups with relatively high probability of sub-optimal assignment, and conduct internal communication for the groups.

We introduce the technical details of the λ-Opting Framework in the following subsections.

## 5.1 Preliminary matching

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 target is to divide all the workers into small groups of size 2, according to some statistical criterion to be described later. If the number of workers is an odd number, we may have a group containing only one worker. After dividing them into groups, communication inside the groups is allowed, so workers inside the small group could access to the true location of the tasks originally assigned to the other workers inside the group. Based on the true location, they could know the true travel distance for completing the tasks, and swap the tasks if it reduces the travel distance.

To implement the aforementioned framework, there are two main problems to be addressed. First, what is the 'statistical criterion' for properly grouping the workers? 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 a grouping. Intuitively, we hope to group workers such that swapping of their currently assigned tasks may lead to the best possible improvement of the total travel distance. Second, after grouping, how should we devise the inner-group communication to 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 is

the basis for the subsequent steps of our $\lambda$-Opting framework to optimize the worker-task matching quality.

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

### 5.2.1 Obfuscation score

We start by presenting an important observation on the matching results. 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, together with their assigned tasks obtained from the *Expected Distance* based SSPA algorithm. We analytically assign a score to this grouping to measure the possibility that the matching algorithm makes a mistake. The basic intuition is that the mistake-making probability would be high, if the two tasks are close to each other, while the positions of the workers are far away from 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* as:

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

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. The probability of wrong assignment is quantified by the *obfuscation-score*.

**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 hyperbola. We call the $\delta$ as the shift distance. 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 $\Delta_C - \Delta_D$.

The key idea here for the proof is to analyze when does the saved cost increase and decrease according to the properties of hyperbola. Please refer to the technical report for the complete proof. [1]

□

## 5.3 1-Opting

Based on the *o-score* defined in measuring step, we further 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 contains two different matched worker-task pairs, i.e., $g_i = (m_j, m_k)$, and $m_j.q \neq m_k.q$. Obviously we could calculate the obfuscation score *o-score* on the group $g_i$, i.e., *o-score*$(g_i)$.

Since the definition requires that the workers from the two matched pairs $(m_j, m_k)$ are different, we know the associated tasks are sure to be 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 basic intuition of the proof is that we could reduce the grouping problem to the maximum weight matching on a general graph. We construct an 1-on-1 mapping between a grouping and the matching obtained from a graph. We prove it by contradiction by showing that given any optimal grouping, there exists a corresponding maximum weight matching that could be obtained from the reduced graph, and vice versa. Please refer to the technical report for the detailed proof [1].

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.

□

Next we describe the 1-Opting step 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 communication to determine whether internal swapping could improve the matching quality.

**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  $response :=$ Internal communication 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** $M'$

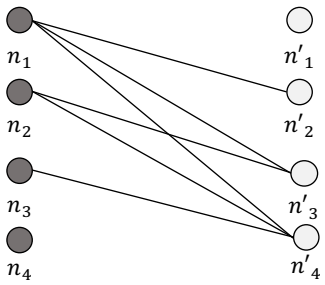



**Figure 3: Illustration of graph transformation**

The graph transformation in Algorithm 3 simply transforms a complete graph to an undirected bipartite graph. As shown in Figure 3, the basic idea is to copy $v$ nodes to $2v$ nodes, where the 1st node is copied to its copied node (denoted by $n'_i$). The original and the copied nodes are put in sets $L$ and $R$, respectively. 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.

**Algorithm 3:** Graph transformation

**Input:** Complete graph $G$
**Output:** Bipartite graph $G'$

1 Initialize an empty graph $G'$
2 Get the size of the vertex set, $v := |G.V|$
3 **foreach** *node* $n_i \in G$ **do**
4  Insert $n_i$ to $G$
5  Create a new node $n'_i$, insert it to $G$
6 **foreach** *node* $n_i \in G$ **do**
7  **foreach** *node* $n_j \in G$ **do**
8  $e :=$ edge between $n_i$ and $n_j$
9  $w := c(e)$
10  Create an edge between node $n_i$ and $n'_j$, with weight $w$
11 **return** $G'$

**Algorithm 4:** $\lambda$-Opting

**Input:** Parameters $\lambda, \theta$, 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 $\lambda$-Opting step

1 $i := 1$
2 $M_0 := M$
3 **for** $1 < i \leq \lambda$ **do**
4  $M_i :=$ Algorithm 2 on $M_{i-1}$
5  $Gain := \Psi(M_{i-1}) - \Psi(M_i)$
6  **if** $Gain \leq \theta$ **then**
7  Break
8  $i := i + 1$
9 **return** $M_i$

**Time complexity** 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 $O(n^{2.37})$.

## 5.4 $\lambda$-Opting framework

$\lambda$-Opting is an interative method, which runs for $\lambda$ rounds, each time improving the quality of the matching. The quality of the matching is measured by the total travel distance of the workers to reach their assigned tasks. It is described in Algorithm 4.

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

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

# 6. K-SWITCH FRAMEWORK

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

To further lower the total travel distance of the workers in the matching obtained, in this section, we manage to extend the group size from 2 to $k$ where $k \geq 3$. A larger group size allows more worker to exchange tasks with others, and thus leads to a better matching. We first formulate the $k$-Grouping problem and discuss its intractability (NP-hardness). Then we propose $k$-Switch, a game theoretic approach to re-assign the tasks within each group.

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, normally ranging from 3 to 6. Then we design a $k$-Talk internal communication protocol to improve the 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 $\lambda$-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 divide the matched pairs into small groups. But now we target groups of size $k$.

**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 contains $k$ different matched worker-task pairs, i.e., $g_i = (m_{i_1}, \cdots, m_{i_k})$, 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 two worker-task pairs in 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 $k$-groups. All the $k$-group should be exclusive, meaning they do not overlap, and the union of workers from each group cover the entire worker set $Q$.

**Definition 10.** (Score of a $k$-grouping) Given a $k$-grouping $G_k = \{g_i | g_i \in M \times M \cdots \times M\}$, we define its score as *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 $k$-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* $\leq_p$ $k$-grouping problem. Please refer to the technical report for the full proof [1].

Since *Balanced Graph Partition* has no polynomial time approximation algorithm with finite approximation factor unless P=NP [2], 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 high-quality groupings. The main idea is that sometimes we could improve the quality of the matching by moving a certain item (a preliminary matched pair) 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, because no further improvement could be made based on the current grouping.

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 a target $k$-group $g_t = \{m_{t_1}, \cdots, m_{t_k}\}$ and a matched pair $m_j$ which belongs to $g_t$, we have $\exists d, m_{t_d} = m_j$. Also, we are a given preliminary matched pair (also called an item), $m_i = (m_i.q, m_i.p, m_i.q \in Q, m_i.p \in P)$, and the source $k$-group $g_s$. 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 tuple $(m_i, g_s, g_t, m_j)$.

**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 *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 $g_t$ and putting this element $m_j$ back to the source group $g_s$. Thus we calculate the new *k-o-score* on both of the new groups after the item exchange, and calculate the score difference w.r.t. the original grouping.

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.

After all items make their moves, we jump to the next round. Each item now has a new chance to select other items and $k$-groups to exchange. After each round, the $k$-grouping could be different from last round. 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 5. At Line 2 and Line 4, $k$-Switch framework is calling its subroutine *Item exchange* defined in Algorithm 6 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 7. After the decentralized communication, the group notifies the server about the improved matching, and the server then updates the overall matching.

**Algorithm 5:** $k$-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

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

**Algorithm 6:** 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 $\delta$

$\delta := 0$
**foreach** $m_i \in M$ **do**
  $g_s :=$ original $k$-group $m_i$ belongs to in $G$
  *max_t* $:= i$
  *max_score* $:= 0$
  **foreach** $m_j \in M$ **do**
    **if** $m_i \neq m_j$ **then**
      $g_t :=$ $k$-group $m_j$ belongs to in $G$
      *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*$(g_t)$
      **if** *ex-score* $>$ *max_score* **then**
        *max_score* $=$ *ex-score*
        *max_t* $= j$
  $\delta$ += *max_score*
  $g'_s = m_{max\_t} \cup g_s \setminus m_i$
  $g'_t = m_i \cup g_s \setminus m_{max\_t}$
  $G' = \{g'_s, g'_t\} \cup G \setminus \{g_s, g_t\}$
  $G = G'$
**return** $G', \delta$

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. Thus one round of game mechanism runs in $O(n^2)$, where $n = |Q|$, the number of workers.

## 6.3 k-Talk protocol

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

The detailed protocol is shown in Algorithm 7.

In the first step, all workers communicate their locations with the tasks 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 w.r.t. the true distances. Since there are $O(k)$ 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), so the framework runs in $O(n^2)$.

## 6.4 Extension

We extend $k$-Switch framework to the cases when the worker locations are also obfuscated. First, in the the Expected distance based SSPA step, we run SSPA directly on the observed distance. Second, note at Line 5 of $k$-Talk in Algorithm 7, the workers notify all the tasks about their true location. Even though the locations are obfuscated to the server, they are sent to the tasks, so the tasks could calculate the actual distances from the workers. In this way, our framework could be well adapted to the cases when worker locations are also obfuscated.

# 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, $\lambda$-Opting and $k$-Switch. We also compared our methods with the SCGuard method proposed in [16].

## 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 $\lambda$-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 $\epsilon$, the flatter the distribution becomes. Being flatter means that larger value of $d(x_0, x)$ is more likely to appear, compared to the case where $\epsilon$ gets closer to 1.

In our experimental setup, if $\epsilon$ 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 $\epsilon$ 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.

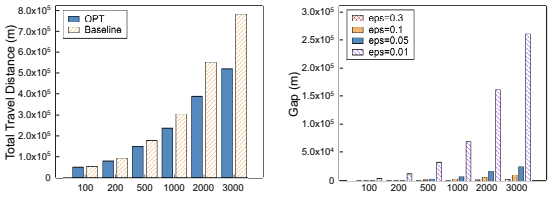
### *7.1.1 Data collection and generation*

**Real-world dataset** For the real-world data, we perform the experimental study on the taxi dataset from Didi Chuxing [20]. 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

**Algorithm 7:** $k$-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** $M_{g_i}$




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

**Figure 4: Optimal (OPT) versus Preliminary Matching (Baseline)**

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 $[0, 8, 000] \times [0, 8, 000]$.

## 7.2 Experimental Results

We perform evaluation based on the following evaluation metrics. The TTD and RT metrics are commonly used in previous studies [16, 13], while ITD and IGR are specifically designed for evaluating our proposed methods against the baseline.

- 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.

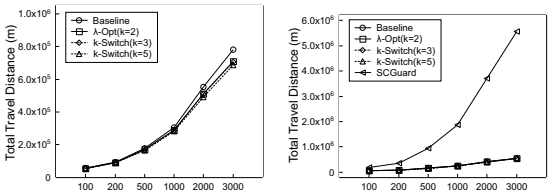
We call the preliminary matching the baseline method, and denote the task assignment (matching) obtained $M_{baseline}$. The actual location information of the tasks are not available to the application server, however as a way to measure performance, we use the actual locations to obtain an optimal 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 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. Please refer to the appendix of the technical report [1] for the cases when the number of workers and tasks are not the same.

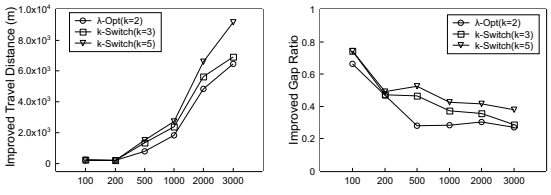
There exists a significant gap between the optimal solution and the baseline. As the input size gets larger, the gap is growing larger. The gap could constitute roughly 20% - 30% of the total travel distance (TTD) of the baseline solution.

We also look the impact of the value of $\epsilon$ on the gap. The value of $\epsilon$ 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 $\epsilon$, 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 $\epsilon$ is smaller, e.g., 0.01, there is a significant gap between the optimal solution and the baseline.




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

**Figure 5: Total travel distance (TTD) of various methods**




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

**Figure 6: Effectiveness measured by ITD and IGR**

### *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 [16]. 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. Our framework could be easily extend to the cases when both the worker and task obfuscate their locations. Also, we believe our problem formulation reflects the real-world scenario better, where the crowdsourcing platform has access to the true worker 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.

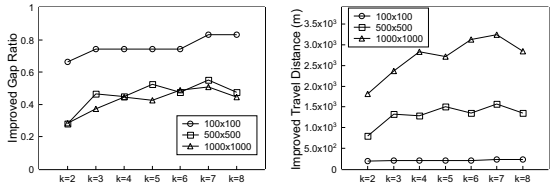
**Improved travel distance and improved gap ratio**

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 ($\epsilon$ 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




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

**Figure 7: Performance comparison on different $k$**

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 $\lambda$-Opting with $k$-Switch, the results show that $k$-Switch ($k = 3$) outperforms $\lambda$-Opting. Actually, $\lambda$-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 $\lambda$-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. We limit the range of $k$ here, to avoid privacy leakage among large groups. 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. The larger the group is, more workers are 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 additional gain on TTD and IGR becomes marginal. Considering a larger $k$ may lead to larger groups of leaking true task locations, it is most cost-effective to select $k = 5$.

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 $\epsilon$**

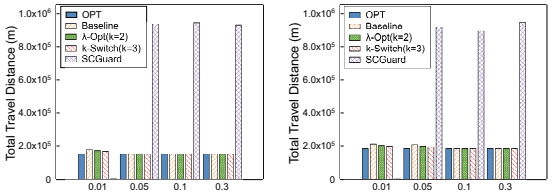
The value of $\epsilon$ plays an important role in the obfuscation process. The smaller the value is, the farther an obfuscated location is from its original location.

As shown in Figure 8, one could see that our methods dominate the performance of SCGuard, across different values of $\epsilon$. Another interesting note is that when $\epsilon = 0.3$, the baseline and the optimal solution are very close, so the gap is small. This is because $\epsilon = 0.3$ indicates the privacy restriction is low, the obfuscated locations are close to the true locations (only 1-5 meters away). 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**

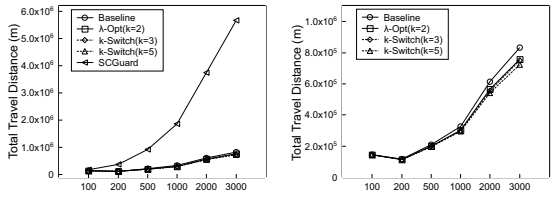
The results obtained by running experiments on synthetic dataset align with the results on the real-world data.

However, there are some interesting points worth noting. The improved gap ratio (IGR) tend to be quite stable across different



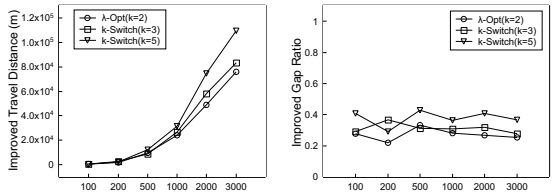

(a) Real-world data (b) Synthetic data

**Figure 8: Impact of $\epsilon$ for different methods (500 vs. 500)**




(a) TTD (b) TTD, excluding SCGuard

**Figure 9: TTD of various methods on synthetic data**




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

**Figure 10: ITD and IGR on synthetic data**

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.

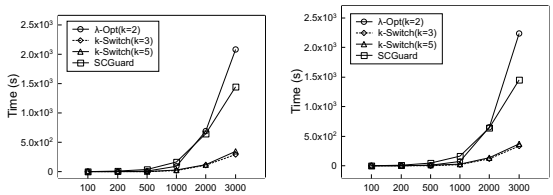
Quite surprisingly, $\lambda$-Opting is significantly slower than the other $k$-Switch methods. Though $\lambda$-Opting is a polynomial algorithm and has its running time bounded by $O(n^{2.37})$, where $n$ is the number of workers, as the input size gets larger, it becomes slower. According to the experimental results, $k$-Switch is around 3 times faster than $\lambda$-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**

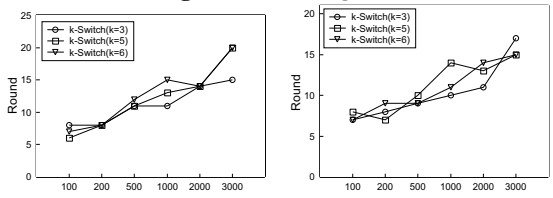
As a validation of the efficiency of $k$-Switch method, we record the number of rounds it takes to reach the equilibrium. Recall that, in each round of $k$-Switch, each item could move to other $k$-groups. The equilibrium is reached if no item is moving during the current round.

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.




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

**Figure 11: Running time**




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

**Figure 12: Convergence of $k$-Switch**

### 7.2.4 *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 (the OPT) and the baseline, our methods could fill up to 80% of the gap, and on average covering more than 50% of the gap. In terms of total travel distance, our methods achieve order-of-magnitude improvement over the online method SCGuard.

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

## 8. CONCLUSION AND FUTURE WORKS

In this work, we address a challenging problem – the privacy-preserving spatial crowdsourcing tasks assignment problem. The protection mechanism adds a random Laplacian noise to the reported location of tasks. The server could only access the obfuscated locations of the tasks.

We use the *Expected distance* based SSPA matching to obtain a baseline matching (a preliminary matching) of tasks and workers. Moreover, we proposed $\lambda$-Opting framework and $k$-Switch framework. Both of them utilize the quantitative analysis on the likelihood of wrong assignment of task and workers in the baseline matching. $\lambda$-Opting uses a polynomial matching method to find best groups of size 2, while $k$-Switch uses a game theoretic approach to find best groups of size $k$ ($k \geq 3$).

The experimental study shows that 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 cover up to 80% of the gap. It achieves order-of-magnitude improvement over existing online method SCGuard. Our methods are also very efficient on a broad range of input sizes. In particular, $k$-Switch is faster than $\lambda$-Opting, and around 3 times faster than SCGuard.

For the future work, different application scenarios in the spatial crowdsourcing platform could be considered. For example, the problem may become more challenging when we allow ride-sharing in tasks assignment. In addition, the differential privacy technique could be applied to different stages of the spatial crowdsourcing platform, which could lead to different research problems.

# 9. APPENDIX

## 9.1 Additional proofs

### 9.1.1 Section 4

**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 $\theta$ are independent because $\theta$ doesnt appear in this function. In Fig.13 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 $\theta_f$ are the distance and the direction angle between the location of server and the fake location of user respectively, while $r$ and $\theta$ for the possible and perturbed location of user.

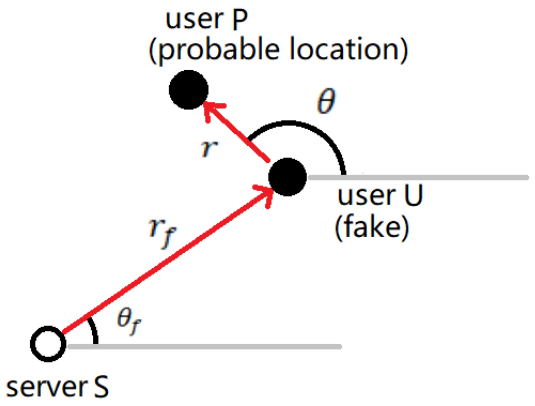



**Figure 13: The location of server $S$ and the perturbed location of user $U$ in polar coordinate. A possible true location of user $P$ is shown in figure as well with arbitrarily selected parameter $r$ and $\theta$.**

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:

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

We verify the integration by Matlab and the part of:

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

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 $\theta$, we can find:

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

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:

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

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.

### *9.1.2 Section 5*

**Theorem 9.1.** *(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 $\delta$ 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 $d_\Delta$ 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 $\pi_T$. 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 $\pi_T$ 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:

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

The normal vector intersect with $\pi_{T'}$ at $T'$, Then the minimum distance for $T'$ getting to $\pi_T$ 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:

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

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.14 and we only need the result in first first quadrant.

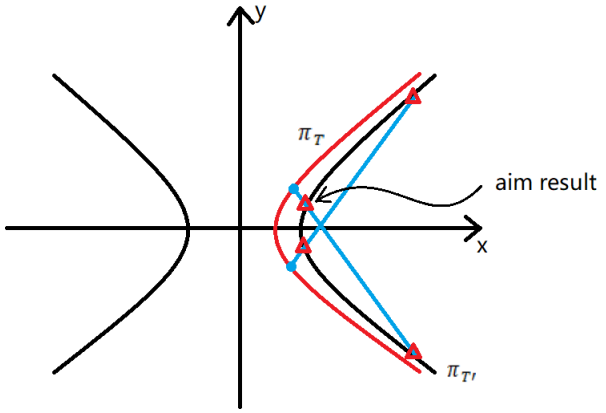



**Figure 14: Four results location of $T'$ in red triangles. The distance relationship does not change under different signs of $y$, so we pick only first quadrant to analysis and the nearest approach to realize shift distance $d_\Delta$ is point out black red arraw.**

Then we have the change rate while letting $d_\Delta \to 0$, which is a function of $x_0, \Delta_{T'}, d_w$. The $d_\Delta$ 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 use some variable substitutions to make the equation more clear:

$$\begin{cases} a = (\dfrac{\Delta_{T'}}{2})^2 \\ b = \dfrac{d_w^2 - \Delta_{T'}^2}{4} \\ c = 2d_\Delta\Delta_{T'} + d_\Delta^2 \end{cases} \Rightarrow \begin{cases} \dfrac{x_{T'}^2}{a} - \dfrac{y_{T'}^2}{b} = 1 \\ \dfrac{x_0^2}{a+c} - \dfrac{y_0^2}{b-c} = 1 \\ \dfrac{y_0(x_{T'}-x_0)}{b-c} + \dfrac{x_0(y_{T'}-y_0)}{a+c} = 0 \\ d_{TT'} = \sqrt{(x_{T'}-x_0)^2+(y_{T'}-y_0)^2} \end{cases}$$

Here one can notice that $c$ is the only variable that interacts with $d_\Delta$, which even is a surjection from the domain $R^+$ to $R^+$ as well as the proporty $c|_{d_\Delta=0} = 0$

Finally we can derive the $d_{TT'}$:

$$d_{TT'} = \sqrt{\frac{den_1 \cdot den_2 \cdot (b-c)}{num}}$$

where:

$$den_1 = x_0^2(a+b) - (a+c)^2$$

$$den_2 = (ac - key_1 + key_2)^2:$$

$$\begin{cases} key_1 = (a+b)x_0^2 - a^2 \\ key_2 = \sqrt{\dfrac{ab}{(a+c)(b-c)}} \cdot \sqrt{key_1^2 - c(key_3) + O(c^2)} \\ key_3 = 3a(key_1) \end{cases}$$

$$num = (a^3 + 2a^2c - a^2x_0^2 + ac^2 - acx_0^2 + b^2x_0^2 - bcx_0^2)^2$$

We can derive that the $D_C$ is the first-order small quantities of $c$. So the aim transfers to calculate

$$\lim_{d_\Delta \to 0} \frac{d_{TT'}}{d_\Delta} = \lim_{d_\Delta \to 0} \frac{c}{d_\Delta} \frac{d_{TT'}}{c} = 2\Delta_{T'} \cdot \lim_{c \to 0} \frac{d_{TT'}}{c}$$

Then we can derive that:

$$\begin{aligned} \lim_{c \to 0} \frac{d_{TT'}}{c} &= \sqrt{\lim_{c \to 0} \frac{den_1 \cdot \frac{den_2}{c^2} \cdot (b-c)}{num}} \\ &= \sqrt{\frac{\lim_{c \to 0} den_1 \cdot \lim_{c \to 0} \frac{den_2}{c^2} \cdot \lim_{c \to 0}(b-c)}{\lim_{c \to 0} num}} \end{aligned}$$

where:

$$\lim_{c \to 0} den_1 = x_0^2(a+b) - a^2$$

$$\lim_{c \to 0}(b-c) = b$$

$$\lim_{c \to 0}(num) = (a^3 - a^2x_0^2 + b^2x_0^2)^2$$

especially:

$$\frac{den_2}{c^2} = (\frac{ac - key_1 + key_2}{c})^2$$

$$\lim_{c \to 0} \frac{den_2}{c^2} = (a + \lim_{c \to 0} \frac{key_2 - key_1}{c})^2$$

$$\begin{aligned} &\lim_{c \to 0} \frac{key_2 - key_1}{c} \\ &= \lim_{c \to 0} \frac{\sqrt{\frac{ab}{(a+c)(b-c)}} \cdot \sqrt{key_1^2 - c \cdot key_3 + O(c^2)} - key_1}{c} \end{aligned}$$

This is a standard $\frac{0}{0}$ limitation and can be solved using L'Hospital rule.

$$\begin{aligned} \lim_{c \to 0} \frac{key_2 - key_1}{c} &= \lim_{c \to 0} \frac{\frac{\partial(key_2 - key_1)}{\partial c}}{\frac{\partial c}{\partial c}} \\ &= \frac{-\frac{key_1(b-a)}{2ab} - \frac{key_3}{2key_1}}{1} \\ &= -\frac{key_1(b-a)}{2ab} - 1.5a \end{aligned}$$

Then we can find:

$$\begin{aligned} \lim_{c \to} \frac{den_2}{c^2} &= \left(\frac{a}{2} + \frac{key_1(b-a)}{2ab}\right)^2 \\ &= \left(\frac{b^2x_0^2 - a^2x_0^2 + a^3}{2ab}\right)^2 = \frac{\lim_{c \to 0} num}{4a^2b^2} \end{aligned}$$

And now we can get that:

$$\begin{aligned}\lim_{c\to 0}\frac{d_{TT'}}{c} =&\sqrt{\frac{\lim_{c\to 0} den_1 \cdot \lim_{c\to 0}\frac{den_2}{c^2}\cdot \lim_{c\to 0}(b-c)}{\lim_{c\to 0} num}}\\ =&\sqrt{\frac{(x_0^2(a+b)-a^2)\cdot \frac{\lim_{c\to 0} num}{4a^2b^2}\cdot b}{\lim_{c\to 0} num}}\\ =&\frac{\sqrt{x_0^2(a+b)-a^2}}{2a\sqrt{b}}\end{aligned}$$

Now we transfer all the values of the equation to our input:

$$\begin{cases}\lim\limits_{c\to 0} x_0^2 = \lim\limits_{c\to 0}(a+c+\dfrac{(a+c)y_0^2}{b-c}) = a+\dfrac{ay_0^2}{b}\\ a=\left(\dfrac{\Delta_{T'}}{2}\right)^2\\ b=\dfrac{d_w^2-\Delta_{T'}^2}{4}\end{cases}$$

We choose $y_0$ instead of $x_0$ because each hyperbola has different domain for $x_0$ but full domain for $y_0$. The transform result is:

$$\lim_{d_\Delta\to 0}\frac{d_{TT'}}{d_\Delta} = \frac{2\sqrt{(d_w^2-\Delta_{T'}^2)^2+4d_w^2y_0^2}}{d_w^2-\Delta_{T'}^2}$$

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

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 $d_w^2, \Delta_{T'}^2, t_0^2$:

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

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

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

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:

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

And calculate the shift cost for each fake location of $T'$ and change amount of shift distance $d_\Delta$:

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

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:

$SAVE = (|AD|+|BC|) - (|AC|+|BD|)$

$\pi_1 = C(x_C, y_C) \in R^2$

$\pi_2 = \{D | D \in |AC|+|BD| > |AD|+|BC|\}$

$(|AC|+|BD|) - (|AD|+|BC|) = |\Delta_C - \Delta_D|$

$\leq d_{\Delta_C} + d_{\Delta_D} - ((|AD'|+|BC'|) - (|AC'|+|BD'|)) \quad = SAVE'$

$P(C,C')$ 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:

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

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

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

and define:

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

Then for the worker set $Q = q_1, q_2, ..., q_n$ and matched customer $P = p_{q_1}, pq_2, ..., pq_n$, a ranking description for 2-group is defined as:

$$\begin{aligned}Rank\,\{list(group_k = (q_i,q_j)|\\ i,j\in\{1,t\}, i\neq j|\\ m,n\in\left\{1,\frac{t(t-1)}{2}\right\}, m>n,\\ score_E(group_m) > score_E(group_n))\ \}\end{aligned}$$

where $score_E(group_m)$ is the score for a mismatching pair $m$ under the rule $E$. So we have:

$$\begin{aligned}&score_{E(\Delta)} \approx score_{E'(\Delta)}\\ &\approx max\left(SAVE'\cdot P_m(C,C')\cdot P_m(D,D')\right)\end{aligned}$$

We can get the maximum by calculate the derivation. The maximum is:

$$max\left(SAVE' \cdot P_m(C, C') \cdot P_m(D, D')\right)$$

$$= \frac{\epsilon^3}{8\pi^2\alpha} e^{-1-2\epsilon\alpha|\Delta_{C'} - \Delta_{D'}|}$$

where:

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

Using this value, we can calculate a much more explicit score for each pair of matching. □

**Theorem 9.2.** *(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 [7]. $V$ 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.

□

### 9.1.3 Section 6

**Theorem 9.3.** *(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 9.2. 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 $w(e) = o\text{-}score(g)$. 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 $score(G_k) = \sum_i k\text{-}o\text{-}score(g_i)$, it sums up all the *k-o-score* of all the $k$-groups. And for each $k$-group, $k\text{-}o\text{-}score(g_i) = \sum_{1 \leq s,t \leq k} o\text{-}score((m_{i_s}, m_{i_t}))$, 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 $o\text{-}score((n_i, n_j) = w(e)$. 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 $score(G) = \sum_i k\text{-}o\text{-}score(g_i)$. 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 [2], the $k$-grouping problem has the same property. □

**Algorithm 8:** Internal communication

**Input:** Two matched pairs $(q_1, p_1), (q_2, p_2)$
**Output:** True or False

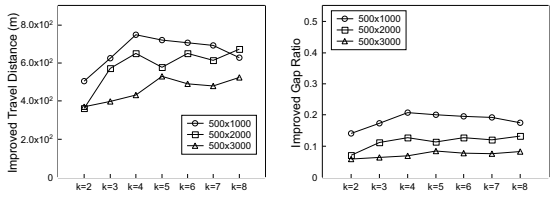
```
Workers q1, q2 notify their accurate locations to the tasks p1, p2
Task p1 calculate true distance cost_{p1,q1} = dist(l_{p1}, l_{q1}), cost_{p1,q2} = dist(l_{p1}, l_{q2}) using his/her own accurate location
Task p2 calculate true distance cost_{p2,q1} = dist(l_{p2}, l_{q1}), cost_{p2,q2} = dist(l_{p2}, l_{q2}) using his/her own accurate location
Task p1 notifies cost_{p1,q1} and cost_{p1,q2} to worker q1
Task p2 notifies cost_{p,2q1} and cost_{p2,q2} to worker q2
Two workers exchange all the cost
if cost_{p1,q1} + cost_{p2,q2} > cost_{p1,q2} + cost_{p2,q1} then
    Response := True
else
    Response := False
return Response
```

## 9.2 Additional experimental results

### 9.2.1 Imbalanced workers and tasks

To be concise, in previous sections we've only included 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 15, 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 $\lambda$-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.




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

**Figure 15: Effectiveness on imbalanced data input**

# 10. REFERENCES


[1] [online] Technical Report. https://cspcheng.github.io/pdf/kSwitchDP.pdf

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

[3] 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)
[4] 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)
[5] Christin, D.: Privacy in mobile participatory sensing: Current trends and future challenges. The Journal of Systems and Software **116**(57-68) (2016)
[6] Dwork, C.: Differential privacy. International Colloquium on Automata, Languages, and Programming, LNCS **4052**(1-12) (2006)
[7] Edmonds, J.: Paths, trees, and flowers. Canadian Journal of Mathematics **17**(449-467) (1965)
[8] gMission: `http://gimission.github.io`
[9] Goldberg, A.V., Kennedy, R.: An efficient cost scaling algorithm for the assignment problem. Mathematical Programming **71**(153-177) (1995)
[10] 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`
[11] 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)
[12] 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)
[13] 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)
[14] Munkres, J.: Algorithms for the assignment and transportation problems. Journal of the Society of Industrial and Applied Mathematics **5**(32-38) (1957)
[15] 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)
[16] 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)
[17] Torosluk, I.H., Ucoluk, G.: Incremental assignment problem. Information Sciences **177**(1523-1529) (2007)
[18] 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)
[19] 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)
[20] 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)