World Wide Web (2020) 23:2131–2151
https://doi.org/10.1007/s11280-019-00728-3

User experience-driven secure task assignment in spatial crowdsourcing

Wei Peng[1] · An Liu[1] · Zhixu Li[1] · Guanfeng Liu[2] · Qing Li[3]

Received: 20 February 2019 / Revised: 15 July 2019 / Accepted: 9 September 2019 /
Published online: 28 February 2020

Abstract
With the ubiquity of mobile devices and wireless networks, Spatial Crowdsourcing (SC) has earned considerable importance and attention as a new strategy of problem-solving. Tasks in SC have location constraints and workers need to move to certain locations to perform them. Current studies mainly focus on maximizing the benefits of the SC platform. However, user average waiting time, which is an important indicator of user experience, has been overlooked. To enhance user experience, the SC platform needs to collect lots of data from both workers and users. During this process, the private information may be compromised if the platform is not trustworthy. In this paper, we first define user experience-driven secure task assignment problem and propose two privacy-preserving online task assignment strategies to minimize the average waiting time. We securely construct an encrypted bipartite graph to protect private data. Based on this encrypted graph, we propose a secure Kuhn-Munkres algorithm to realize task assignment without privacy disclosure. Theoretical analysis shows the security of our approach and experimental results demonstrates its efficiency and effectiveness.

Keywords Spatial crowdsourcing · Task assignment · User experience · Privacy-preserving

1 Introduction

With the rapid development of mobile device and wireless networks, Spatial Crowdsourcing (SC) is a new extension of traditional crowdsourcing, where crowd workers are assigned by

This article belongs to the Topical Collection: *Special Issue on Web Information Management and Applications*
Guest Editors: Yi Cai and Jianliang Xu

✉ An Liu
anliu@suda.edu.cn

[1] Institute of Artificial Intelligence, School of Computer Science and Technology, Soochow University, Suzhou, China

[2] Department of Computing, Macquarie University, Sydney, Australia

[3] Department of Computing, The Hong Kong Polytechnic University, Hong Kong, China

the crowdsourcing platform to perform spatial tasks. Spatial Crowdsourcing provides on-demand ability and quality services and spends less monetary cost than hiring specialized technical personnel. Now, it has been successfully applied in various industries, for example, to online taxi-calling service (e.g., DiDi and Uber), handyman service (e.g., TaskRabbit) and food delivery service (e.g., Eleme and Meituan). In spacial crowdsourcing services, workers are ready for task assignment in SC platform and requesters usually raise their requests to the SC platform, then the platform assigns these tasks to workers. The assigned workers always need to travel to the location of corresponding task physically. The basic problem in SC is how to choose suitable workers to perform right tasks. To ensure practicability, it is significant to study this problem in the online scenario (online task assignment problem) where workers and tasks appear on the SC platform dynamically.

By observing the release modes of SC task assignment proposed in existing studies, we found that many works focus on optimizing the benefits of SC platform during the process of task assignment. For example, [29] and [36] aim to minimize the total cost and then the benefit of platform is maximized. However, *user experience* is equal significant and can not be ignored in practice for the SC platform. The foundation of SC platform is the number of users. There always are several similar platforms which provide the same service, and users prefer to choose the platform which provides service with higher user experience. As a result, it is vital for the SC platform to focus on improving the user experience. Besides, many previous works chose the travel distance instead of the travel time between workers and tasks. This could save some computation, but it makes experimental results inaccurate and unpractical. The following example demonstrates the importance of user experience and the necessary of taking the velocity attribute into consideration:

We choose online taxi-calling service as an example, users (passengers) raise their requests to the SC platform and hope that they only need to wait a short time so that the taxi can reach their locations and take them to their destinations as soon as possible. As is shown in Figure 1, we assume that a taxi-calling platform has 3 tasks ($s_1 - s_3$) and 5 workers ($u_1 - u_5$) in 2D space, the ID, distance and location of them and the velocity of workers are marked in this figure. The arriving time of all tasks and workers are shown in Table 1. We assume that worker u_3 and worker u_5 get stuck in traffic and their speeds are slower than

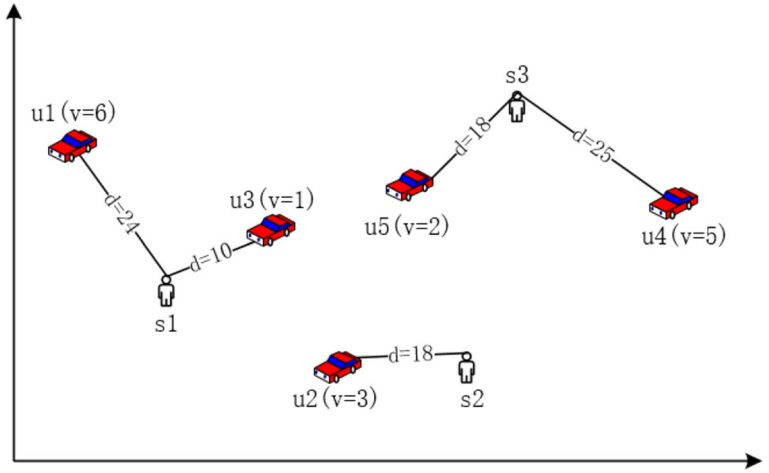


Figure 1 An example of 3 passengers and 5 taxis

Table 1 Arriving Time of workers and tasks

Time	0	1	2	3	4	5	6	7
$Task/worker$	u_1	s_1	u_2	u_3	s_2	u_4	s_3	u_5

others such as the speed of u_3 is 1. What's more, a task can only be assigned to one worker and each worker can perform only one task. Then we compare the average waiting time (the evaluation indicator of user experience) of the following two different strategies. In the first strategy, we set the optimization goal as minimizing the average waiting time of all tasks, and the solution will be that s_1, s_2, s_3 are matched to u_1, u_2, u_4. And the average waiting time is $(24/6 + 18/3 + 25/5)/3 = 5$ (Here we ignore the time of waiting for assignment and we will consider it in the body of this paper). In the second strategy, we set optimization goal as minimizing the average traveling distance, the solution will be that s_1, s_2, s_3 are matched to u_3, u_2, u_5. The average waiting time is $(10/1 + 18/3 + 18/2)/3 = 8.3$. During this process, users need to wait for the taxi and the waiting time should be as short as possible because long waiting time may urge users to use competitive platforms and then the benefits will be significantly reduced. In the second strategy, user (passenger) s_1 and s_3 wait too long and achieve a bad user experience so that they may choose another taxi-calling platform instead of using this platform next time. Obviously, the SC platform cannot assign tasks based on the distance of workers and tasks rather than the traveling time of workers because the speed of workers (traffic condition) could have a great influence on user's waiting time and user experience.

In order to make optimal task assignment, it is necessary for the SC platform to know the locations and speeds of workers and tasks. However, the locations and speeds are privacy for workers and tasks because they can lead to workers being tracked or tasks being destroyed, etc. Task assignment may become inefficient when workers and tasks are hesitant to share their locations or others due to privacy concerns. For each worker, his/her location and speed are both private and should be protected. For tasks (users), the location is private and should be protected. We aim to encrypt the locations and speeds based on Paillier cryptosystem and then calculate the traveling time based on the encrypted data without compromising workers and tasks privacy. In order to calculating traveling time by locations and speed, we need to use division operation. How to perform division efficiently and accurately on encrypted data is still an open problem. In [16], the author proposed a protocol which can divide data securely based on ElGamal cryptosystem. However, this protocol cannot be applied to large SC system for it's key length should be set large enough to avoid computation overflow and the large key size will lead to prohibitive computation cost in encrypted data. To overcome this weakness, we transform the secure division problem into a secure least common multiple (LCM) problem so that we can calculate the traveling time of workers securely and preserve the privacy of workers and tasks.

We summarize our main contributions as follows:

- We formally define user experience-driven secure task assignment problem in Section 2, which tries to minimize average waiting time and protect private information during task assignment simultaneously.
- We propose two methods to construct encrypted bipartite graph to protect private information of workers and users in Section 3.1.
- We propose in Section 3.2 a secure Kuhn-Munkres algorithm which takes an encrypted bipartite graph as input, and outputs a plan of task assignment. Security and complexity of this approach is discussed in Section 4.

– We conduct extensive experiments on synthetic data and show the effectiveness and efficiency of our approach in Section 5.
– Compared with our previous work [13], the problem considered in this paper is more complex and reasonable in practice. To handle this complicated problem, we propose methods of constructing encrypted bipartite graph and making KM algorithm workable on encrypted data.

The rest of the paper is organized as follows. We present the preliminary in Section 2. In Section 3, we present two online privacy-preserving task assignment algorithms: SKM-EG and SKM-AG. Security and complexity analysis is presented in Section 4. Further, we report experimental results on synthetic dataset in Section 5 and review related work in Section 6. Finally, we conclude the paper in Section 7.

2 Preliminaries

2.1 System model

The core of a spatial crowdsourcing system is a platform connecting a set of workers with a set of tasks. Workers and tasks arrive the platform dynamically. Without loss of generality, we take a periodical task assignment model. Before explaining it, we give several basic definitions.

Definition 1 (Worker) A worker u is denoted by a triple $\langle l, v, x \rangle$, where l is u's location, v is u's speed, and x is the time when u appears on the platform.

Definition 2 (Task) A task s is denoted by a tuple $\langle l, x \rangle$, where l is its location and x is the time when s is posted on the platform.

Definition 3 (Task Assignment Instance Set) Given a set of workers U and a set of tasks S, a task assignment instance set, denoted by I, is a set of pairs in the form of $p_{ij} = (u_i, s_j)$ where (u_i, s_j) means worker $u_i \in U$ is assigned to task $s_j \in S$.

In the above definition, we assume that one worker is assigned to only one task, which is quite common in real spatial crowdsourcing applications such as DiDi (i.e., one taxi is assigned to one passenger). Once a worker u_i is assigned to a task s_j, both of them will be removed from the platform. We don't care how long it takes for u_i to complete s_j, but once s_j is completed and u_i becomes available, u_i will appear on the platform again. Based on this setting, we define the periodical task assignment model as follows:

Definition 4 (Periodical Task Assignment Model) The platform performs task assignment every τ time units. More specifically, at the beginning of one cycle, the platform generates a task assignment instance set I, given all available workers U and all available tasks S at that time. The workers and tasks involved in I will be removed from U and S, respectively. In the following τ time units, U and S may be updated due to the appearance of new workers and new tasks, or the logout of workers and the cancellation of tasks. After τ time units, the platform generates another task assignment instance set, and so on.

Remark: The periodical task assignment model can be regarded as a trade-off between offline task assignment and online task assignment. When τ is so small that every task is assigned to an available worker as soon as it is posted on the platform, this model is clearly a complete online task assignment process. On the other hand, if τ is large enough, there is only one round of offline task assignment. Therefore, the exact value of τ is important and should be adjusted dynamically in practice. Further, it is very likely that not all tasks/workers can be matched to some workers/tasks in one round of assignment. In this case, these tasks/workers will participate the next round of task assignment until they are assigned or offline.

As mentioned earlier, user experience plays a crucial role in spatial crowdsourcing applications. In practice, once a task is posted in the platform, it should be completed the sooner the better. Here, a task is thought to be completed once the assigned worker arrives at its location (e.g., a passenger has got on a taxi). Note that we do not consider the time that a worker needs to perform a task, as sometimes this depends on the task itself (e.g., the distance between the source and the destination in a taxi-calling task), which is beyond the scope of task assignment. Therefore, we consider using the interval between task posting and task completion to be a specific index of user experience. In particular, we have the following definitions.

Definition 5 (Travel Time) The travel time of worker u_i to task s_j, denoted by t_{ij}, is calculated as follows:

$$t_{ij} = d(l_i, l_j)/v_i \tag{1}$$

where d is the direct Euclidean distance between location l_i and location l_j.

Definition 6 (Average Waiting Time) Given cycle τ in the periodical task assignment model and an interval of $k\tau$ time units, the average waiting time during this interval is calculated as follows:

$$\omega = \left(\sum_{c=0}^{k-1} \sum_{p_{ij} \in I_c} (c\tau + t_{ij} - x_j) + \sum_{s} k\tau \right) / n \tag{2}$$

where n is the total number of tasks posted during the period of $k\tau$ time units. The left part $\sum_{c=0}^{k-1} \sum_{p_{ij} \in I_c} (c\tau + t_{ij} - x_j)$ is the sum of waiting time of tasks that have matched workers, noting that task s_j has waited $c\tau$ time units before getting assigned in the c-th round of assignment. The right part $\sum_s k\tau$ is the sum of waiting time of tasks that have not been getting assigned in the k rounds of assignment.

In the above definition, we do not consider overdue tasks for simplicity. In practice, it is sometimes not easy to know the exact deadline of a task, for example, a passenger can cancel a taxi-calling request anytime.

2.2 Problem definition

We focus on the user experience-driven secure task assignment problem in spatial crowdsourcing under the semi-honest model. In particular, our objective is to minimize the average waiting time of all tasks in a given period without disclosing private information of work-

ers and tasks to unauthorized parties. Based on the aforementioned definitions, we can formalize the User experience-driven Secure Task Assignment (USTA) problem as follows:

Definition 7 (USTA Problem) Given cycle τ in the periodical task assignment model and an interval of $k\tau$ time units, the USTA problem is finding k task assignment instance sets to minimize the average waiting time of all tasks defined in (2), while satisfying the security requirement defined in (3).

2.3 Adversary model

In this paper, we try to protect the following private data: the location and speed of workers, and the location of tasks. All these private data should not be disclosed to unauthorized parties during the procedure of task assignment. To accurately describe the ability of unauthorized parties, we adopt the well-known *semi-honest* model [9]. In this model, each party will stick to a pre-defined protocol, showing the honest aspect. On the other hand, each party will try to derive extra information from what it received in the execution of the protocol, showing the dishonest aspect. The security under the semi-honest model can be formally defined as follows:

Definition 8 (**Security under Semi-honest Model** [9]) Suppose that $\mathcal{F}(x_1, \cdots, x_n) = (\mathcal{F}_1, \cdots, \mathcal{F}_n)$ is a functionality computed by n parties jointly, where x_i and $\mathcal{F}_i$ are the input and output of the i-th party ($1 \leq i \leq n$). For $\mathcal{I} = \{i_1, \cdots, i_\kappa\} \subset \{1, \cdots, n\}$, we let $\mathcal{F}_\mathcal{I}$ denote the subsequence $\mathcal{F}_{i_1}, \cdots, \mathcal{F}_{i_\kappa}$. Consider a protocol for computing $\mathcal{F}$. The view of the i-th party during an execution of this protocol, denoted as $VIEW_i$, is (x_i, y, m_i) where y represents the outcome of the i-th party's internal coin tosses (i.e., a random integer) and m_i represents the messages that the party has received. In other words, $VIEW_i$ is all the data that the i-th party can observe during the execution of the protocol. Let $VIEW_\mathcal{I} \stackrel{\text{def}}{=} (\mathcal{I}, VIEW_{i_1}, \cdots, VIEW_{i_\kappa})$. Then, we say that the protocol securely computes $\mathcal{F}$ if there exists a polynomial-time algorithm, denoted as $\mathcal{A}$, such that for every $\mathcal{I}$ above

$$\mathcal{A}(\mathcal{I}, (x_{i_1}, \cdots, x_{i_\kappa}, \mathcal{F}_\mathcal{I})) \stackrel{C}{=} VIEW_\mathcal{I}, \tag{3}$$

where $\stackrel{C}{=}$ denotes computational indistinguishability.

To achieve privacy-preserving, we introduce a semi-honest crypto cloud provider (CCP) who holds secret keys and provides crypto services. We assume the CCP and the SC platform do not collude by observing that both SC platforms (e.g., DiDi and Uber) and CCPs (e.g., pCloud Crypto and Boxcryptor) are typically run by large companies. Clearly, it is unlikely for a CCP and an SC platform to collude as it will damage their reputation which in turn affects their revenues. Therefore, this assumption has been widely used recently, for example, in [1, 7, 17, 19, 20].

2.4 Cryptosystems

The privacy-preserving property of our protocol is built on several well-known cryptosystems: PRG [24] and Paillier [22]. The details of PRG and Paillier can be found in the given references and all of them are proved to be secure. Here we only emphasize some important properties of these cryptosystems.

Pseudo-random number generator is an algorithm which can be used to generate numbers by a hash function. For Paillier, their encryption are denoted as E_p. In the same way, their decryption are denoted as D_p. This cryptosystem has some important properties and we listed them here in below:

– **Homomorphic Properties of Paillier:** Given two messages m_1 and m_2, we encrypt them and then we have:

$$E_p(m_1)E_p(m_2) = E_p(m_1 + m_2). \tag{4}$$

Beyond that, given a message m, we encrypt it and then we have:

$$E_p(m)^k = E_p(km). \tag{5}$$

3 User-experience-driven secure task assignment

USTA is essentially a global optimization problem. Generally, the optimal solution cannot be found unless we have a global view of the problem. In practice, workers and tasks enter into the SC platform dynamically. Obviously, the SC platform cannot have a global view of the problem. At any time, it only has a local view of available tasks and workers with their reported locations. It is therefore impossible for the SC platform to perform the optimal task assignment. As a result, we adopt a local optimal task assignment strategy, that is, we consider the k rounds of task assignment in USTA to be independent and for each round our objective is minimizing the average waiting time of all tasks in that round. Note that, this strategy is not new and has already been used in [11]. Solutions found by this strategy are typically sub optimal. To improve the quality of solutions, some advanced task assignment strategies, for example, those considering future tasks and workers [4, 31] have been proposed recently. In this paper, however, we only consider this simple local optimal strategy as it helps us focus on privacy-preserving during task assignment which is the main contribution of our work.

Each round of task assignment in USTA can be reduced to the weighted bipartite matching problem as follows. In round k, suppose U_k and S_k are the set of workers and tasks that are available respectively. Let $G_k = (V_k, E_k)$ be a bipartite graph with V_k as the set of vertices and E_k as the set of edges. Each worker u_i in U_k maps to a vertex v_i in V_k and each task s_j in S_k maps to a vertex $v_{|U_K|+j}$ in V_k. If a worker $u_i \in U_k$ can be assigned to a task $s_j \in S_k$, an edge e_{ij} connecting the vertex v_i to the vertex $v_{|U_K|+j}$ is added to E_k. Besides, e_{ij} is associated with a weight w_{ij} which is the travel time of worker u_i to task s_j, that is, t_{ij} defined in (1). Clearly, G_K is a weighted bipartite graph, so finding a task assignment instance set to minimize the average waiting time of all tasks in round k is equivalent to solve the matching problem on G_k.

Following the above frame, the weighted bipartite matching problem can be solved by the classic Hungarian algorithm (also known as the Kuhn-Munkres (KM for short) algorithm [12, 21]). In our problem setting, however, the weights of the edges in G_k cannot be obtained directly due to privacy concerns. More specifically, the location of tasks, and the location and speed of workers, should not be disclosed to the unauthorized parties, such as the SC platform. The weights of the edges in G_k also should be encrypted because they are intermediate results of the calculation. Next, we will introduce two construction methods of encrypted G_k which aims to realize privacy-preserving, and named them EG for encrypted

exact graph construction and AG for encrypted approximate graph construction respectively. After that, we will introduce the secure KM algorithm (denoted as SKM) on the encrypted bipartite graph G_k which aims to solve the task assignment problem and get the task assignment instance set I_k in round k. In particular, we combined the following two construction methods EG and AG with the secure KM algorithm SKM and named them SKM-EG and SKM-AG.

3.1 Construction of encrypted bipartite graph

3.1.1 Encrypted exact bipartite graph (EG)

All available workers and tasks are hold by the SC platform, so it is easy for SC platform to obtain V_k, the set of vertices in G_k. If a worker u_i can be assigned to a task s_j, there is a corresponding edge e_{ij} in G_k and this edge's weight is t_{ij} which is the time u_i takes to travel to the location of s_j. To calculate t_{ij}, the SC platform needs to know d_{ij}, the distance between u_i and s_j, and v_i, the speed of u_i. Unfortunately, v_i is the private information of u_i. Furthermore, to calculate d_{ij}, the SC platform needs to know the locations of u_i and s_j, which are also private. To enable travel-time computation without disclosing private information, we adopt homomorphic encryption-based methodology. On one hand, we protect private data by encrypting them, so any party without secret key cannot learn anything from the encrypted data. On the other hand, it is feasible to calculate the distance and traveling time on ciphertext with the homomorphic property of encryption systems.

Secure distance computation is a common problem in the security domain. Here we follow our previous work [17] and [32] and use Paillier cryptosystem [22] to encrypt private data. Specifically, u_i and s_j do not report their locations to the SC platform directly. Instead, s_j uses the public key of CCP to encrypt its location and the encrypted location is forwarded to u_i via the SC platform. Using the homomorphic addition property, u_i can calculate his/her squared Euclidean distance from s_j as follows:

$$E(d_{ij}^2) = E((l_{ix})^2 + (l_{iy})^2)E((l_{jx})^2 + (l_{jy})^2)E(l_{jx})^{-2l_{ix}}E(l_{jy})^{-2l_{iy}} \tag{6}$$

In the above equation, $E((l_{jx})^2 + (l_{jy})^2)$, $E(l_{jx})$, and $E(l_{jy})$ constitute the encrypted location of s_j. In terms of computation cost, s_j needs to perform encryption three times. In contrast, u_i only needs to do encryption one time to encrypt $(l_{ix})^2 + (l_{iy})^2$.

With the help of the computation of workers, the SC platform can easily obtain $E(d_{ij}^2)$ for every possible task assignment pair (u_i, s_j). Finding square root over encrypted values is not supported by Paillier, so the SC platform needs to turn to CCP for help. To prevent CCP from knowing real values of distance, the SC platform needs to send the encrypted message to the CCP and then get return. When the SC platform send message to CCP, it must satisfying

$$a^* = \alpha(a) + \beta \tag{7}$$

where α and β are two numbers randomly selected from prime field $\mathcal{Z}_q$, and a is the encrypted message which is sent from SC to CCP. Specially, the SC platform encrypt the value $E(d_{ij}^2)$ again with the (7) and the secret key of Paillier as $E(\alpha(d_{ij}^2) + \beta)$. Then the real value d_{ij} can be obtained by subtracting the random noise from the decryption result. Based on this real distance and his/her speed, u_i can calculate t_{ij} and send $E(t_{ij})$ to the SC plat-

form. Once the SC platform receives the data from all workers, it has all elements of G_k and can run secure KM algorithm for task assignment. We will present secure KM algorithm in Section 3.2.

3.1.2 Encrypted approximate bipartite graph (AG)

In terms of security, the method of constructing encrypted bipartite graph presented in the last section is intricate. To further improve the security of bipartite graph construction, we present another method in this section. Instead of computing the exact value of t_{ij} over ciphertext, we consider an approximate value of it and construct an approximate bipartite graph G_k. This is motivated by the fact that the square root of an encrypted value is hard to evaluate, and consequently, the distance between two objects is usually approximated by its square in the security domain. In some cases, SC platform perform task assignment by approximation result may result in sub optimal results. Therefore, we should design a good approximation so that we can obtain satisfied task assignment on the approximate bipartite graph. With sacrificing a little bit of accuracy, we can still ensure that strong security can be achieved during the p.

Our task assignment approximation strategy is based on the following lemma:

Lemma 1 *Let $U = \{u_1, \cdots, u_n\}$ be a set of n workers, $S = \{s_1, \cdots, s_m\}$ be a set of m tasks, and $D = \{d_{11}, \cdots, d_{1m}, d_{21}, \cdots, d_{nm}\}$ be the distances between u_i and s_j, V_{lcm} be the Least Common Multiple of all workers' speed, and $v_i' = V_{lcm}/v_i$ where $1 \leq i \leq n$. For any two different workers $u_i, u_k \in U$ and two different tasks $s_j, s_l \in S$, $d_{ij}/v_i < d_{kl}/v_k$ holds if $d_{ij}v_i' < d_{kl}v_k'$.*

Proof $d_{ij}v_i' < d_{kl}v_k' \Longleftrightarrow d_{ij}v_i'/V_{lcm} < d_{kl}v_k'/V_{lcm} \Longleftrightarrow d_{ij}/v_i < d_{kl}/v_k$. □

The above lemma tells us the time inequation still holds when the scaled speed v_i' is used. This is important as our task assignment is based on travel time. Furthermore, division operation is not needed when comparing travel times. Instead, we only need to compute the product of two numbers, which can be supported by the Paillier cryptosystem. Based on lemma 1, we propose t_{ij}', an approximation for t_{ij}, as follows:

$$t_{ij}' = d_{ij}^2 v_i'^2 = d_{ij}^2 V_{lcm}^2 / v_i^2. \tag{8}$$

As discussed in the last section, every user can compute his/her squared Euclidean distance from a task over ciphertext directly. According to (8), the encrypted approximated travel time will be send to SC platform for graph construction and it can be calculated by user u_i as follows:

$$E(t_{ij}') = E(d_{ij}^2)^{v_i'^2} \tag{9}$$

The value of $v_i'^2$ is straightforward as long as u_i knows V_{lcm}, which also needs to be calculated securely. To compute V_{lcm} securely, we adopt an aggregation protocol denoted as AP [14] which can calculate the sum of multiple numbers in a privacy-preserving manner. It works as follows:

Algorithm 1 Calculating LCM.

Input: the maximal speed V_{max}, the speed v_i of every worker $u_i (1 \leq i \leq n)$
Output: the LCM of all speeds V_{lcm}

1: The SC platform and all workers perform the same exclusion algorithm on V_{max} to get a same list L of 2-tuples $< p, c_p >$ where p is a prime meeting $p \leq V_{max}$ and c_p is the maximal times of p meeting $p^{c_p} \leq V_{max}$.
2: Every worker u_i computes his own factorization F_i of v_i by Pollard's rho algorithm.
3: AP performs $\sum_{p \in P} p * (c_p + 1)$ key generations and assigns these secrets respectively
4: **for** each prime p in L **do**
5: **for** number $k (0 \leq k \leq c_p)$ **do** ¡¡¡¡
6: Every worker u_i generates his own flag data $f[k]$, encrypts it by
7: the assigned AP secrets and sends it to the SC platform.
8: $V_{lcm} = 1$
9: **for** each prime p in L **do**
10: **for** number $k (c_p \geq k \geq 0)$ **do** ¡¡¡¡
11: The SC platform decrypts the sum of all $f[k]$, denoted as H.
12: **if** $H > 0$ **then**
13: $V_{lcm} = V_{lcm} * p^H$
14: break ¡¡¡¡
15: **return** V_{lcm}

Key generation: Let X be a set of nc random numbers where n is the number of workers and c is a random number. Then, divide X into n random disjoint subsets X_i with c numbers and define $M = 2^{\lceil \log_2 n\Delta \rceil}$ where Δ is maximum value of workers' data. At last, send k_i to u_i and the sum k_0 to the SC platform where $k_i = \left(\sum_{v' \in V_i} v'\right) \bmod M$ and $k_0 = \left(\sum_{v' \in V} v'\right) \bmod M$.

Encryption E_a: For each worker u_i, he/she encrypts data m_i by computing:

$$c_i = (k_i + m_i) \bmod M \tag{10}$$

Decryption D_a: The SC platform can decrypt the sum by computing:

$$V\left(\sum_{i=1}^{n} m_i\right) = \left(\sum_{i=1}^{n} c_i - k_0\right) \bmod M \tag{11}$$

Based on a credible assumption that the maximal worker's speed is limited and known to all, we explain the Algorithm 1 as follows: In line 1 and 2, exclusion algorithm is performed to get the list L of 2-tuples $< p, c_p >$ whose complexity is $O(n \log(\log n))$. For example, our maximal speed is 10. Then 3 is one prime where $3 < 10$, and its maximal times is 2 for $3^2 \leq 10$. So the tuple $< 3, 2 >$ will be inserted into the list. Besides, every worker calculates the factorization F_i of his own speed v_i by Pollard's rho algorithm whose complexity is $O(n^{\frac{1}{4}})$. For example, the factorization F of a worker($v_i = 6$) is $F = 2 * 3$ for $6 = 2 * 3$. Based on the list L, the AP generates $\sum_{p \in L} p * (c_p + 1)$ different keys for same key may disclose workers' speed in line 3. In line 4 to 7, each worker u_i generates his flag data $f[k] (k \in [0, c_p])$ as follows:

$$f[k] = \begin{cases} 1, & AT[p] = k \\ 0, & otherwise \end{cases} \tag{12}$$

where $AT[p]$ is the appearance times of p in the corresponding F_i. Then, encrypts and sends flag data. In the above examples, when $p = 3$, this worker($v_i = 6$) generates these flag data $f[0] = 0$, $f[1] = 1$, $f[2] = 0$. In line 9 to 14, the LCM is computed by $V_{lcm} = \prod_{p \in L} p^H$. For example, the factorization of another worker($v_i = 9$) is $3 * 3$. If $p = 3$, this worker generates flag data $f[0] = 0$, $f[1] = 0$, $f[2] = 1$. So the maximal times of 3 is 2 for the decrypted sum of $f[2]$ meets the condition in line 12. Meanwhile, the maximal times of 2,5,7 are 1,0,0 respectively. So $V_{lcm} = 2^1 * 3^2 * 5^0 * 7^0 = 18$ will be returned.

3.2 Secure KM on encrypted bipartite graph (SKM)

3.2.1 KM algorithm

In this section, KM algorithm is adopted to solve the matching problem on bipartite graph. But before we present how to run secure KM algorithm on encrypted bipartite graph (SKM), we will give a brief introduction to KM firstly. We start from some basic definitions of KM algorithm.

Definition 9 (Feasible Vertex Labeling) Let $l(v)$ be the vertex labeling of vertex v and e_{ij} be the weight of edge between v_i and v_j. Given a weighted bipartite graph G, if we have $l(v_i) + l(v_j) \geq e_{ij}$ for each edge, the vertex labeling l is a feasible vertex labeling.

Definition 10 (Tight Edge) Given a weighted bipartite graph G and feasible vertex labeling l, if $l(v_i) + l(v_j) = e_{ij}$, edge e_{ij} is a tight edge.

Definition 11 (Subgraph of Tight Edges) Given a weighted bipartite graph $G = \{V, E\}$ and feasible vertex labeling l, let $E_l = \{e_{ij} \in E | l(v_i) + l(v_j) = e_{ij}\}$. Subgraph of tight edges is a graph G_l which consists of edge set E_l and corresponding vertex.

Definition 12 (Augmenting Path) An augmenting path has the following properties: 1) all edges in augmenting path are tight edges; 2) The number of edges included in augmenting path is odd, and the number of odd numbered edges are one more than the number of even numbered edges; 3) The vertices in augmenting path start from the vertex corresponding to the worker and end at the vertex corresponding to the task, and appear in this two kinds of vertices alternately; 4) There is no repeated vertices in augmenting path; 5) Both the starting and finishing vertices in augmenting path are not included in the selected matching pairs, while the other vertices belong to the selected matching pairs.

To find the perfect match where all the workers are assigned to one task with minimum weight in total, KM starts by initializing all the vertex with feasible vertex labeling l. Then it starts from vertex v_1 and iterates over all the vertexes to find the augmenting path and inserts all the founded worker-task match into the subgraph G_l accordingly. Everytime before iteration over vertexes, it also updates the vertex labeling l based on the following rules:

$$l'(v) = \begin{cases} l(v) + d, & v \in U \cap P' \\ l(v) - d, & v \in S \cap P' \\ l(v), & else \end{cases} \tag{13}$$

where v is a vertex in graph G_l and P' is the failing augmenting path corresponding to the minimum $d = min\{l_i + l_j - e_{ij}\}$. The iteration stops when all the workers are assigned with one task and the matching result is included in subgraph G_l.

3.2.2 Secure KM algorithm

It is noteworthy that there are three main operations in the KM algorithm: initialize feasible vertex labeling, determine if it is tight edges and modify the vertex labeling. Specifically, we need to initialize the vertex labeling by selecting the maximum edge weight associated with each vertex at the beginning of algorithm. Besides, we also need to determine whether the edge weight is equal to the sum of corresponding vertex labeling. Furthermore, we need to choose the $d = min\{l_i + l_j - e_{ij}\}$ and modify the vertex labeling by addition and subtraction operation. All these operations are straightforward in plaintext. However, here we only have the encrypted bipartite graph where all the weights are in ciphertext which makes the implementation very challenging, especially the maximum selection and compare operation.

To implement secure max, min and compare operations without compromising participator's privacy, we employ CCP's capability of decryption. A general idea can be that SC sends the encrypted message like a and b to CCP, CCP decrypts them and do the operation over the decrypted message. And eventually returns the result in ciphertext to SC. However, since the encrypted message can be sensitive information which should be kept privacy from CCP too, it is inappropriate to send them directly. Instead, we disguise all the message with random values as $x^* = x^\alpha * E_p(\beta)$ which can be easily achieved by homomorphic encryption. By sending the disguised value to CCP, the privacy of original data is protected. And also, since all the values are disguised with same random value, the relationship (max, min, compare) is still hold. For example, SC wants the min value of two encrypted messages a and b, instead of raw value, it sends disguised values $a^* = a^\alpha * E_p(\beta)$ and $b^* = b^\alpha * E_p(\beta)$ to CCP, where α and β are generated according to the (7). With decryption, CCP can easily select the min value (a^* for example) and return its ciphertext back to SC. Thus, SC can obtain the min value (a) by comparing the returned ciphertext locally.

With the defined secure max, min and compare operations, we can go forward and design the secure KM algorithm. The detail of secure KM is shown in Algorithm 2. Step 1 reads in the bipartite graph G_k as $e_{ij} = E_p(w_{ij})$ where $j \in m$, $i \in n$ and $m = n$. $E_p(w_{ij})$ is calculated as follows:

$$E(w_{ij}) = \begin{cases} E(-\infty), & u_i \in LW(k) \\ E(-\infty), & s_i \in LS(k) \\ E(-t'^2_{ij}) or E(-t_{ij}), & else. \end{cases} \tag{14}$$

where $LW(k)$, $LS(k)$ is the set of logic workers and tasks. Here, we generate logic workers or tasks to equal the number of worker and task for further applying of KM algorithm. $Xvisit$ and $Yvisit$ initialized as $False$ (step 3 and 7) are used for recording whether or not the vertex has been accessed during the process of finding augmenting path. Step 4 and 8 initialize the feasible vertex labeling of part X (worker vertex) as the ciphertext of maximum traveling time which is connected to the corresponding vertex and initialize the feasible vertex labeling of part Y (task vertex) as the ciphertext of 0. In step 5, we set the $match$ array as 0 for that the corresponding worker is not matched to the task. In step 9 to 30, we continuously expand the subgraph of tight edges by traversing the worker vertex. Specifically, in each round of expanding the subgraph of tight edges, we initialize the difference value of each task $less$ as the ciphertext of infinity in step 11. Then, we repeat the process of finding the augmenting path for worker i from step 12 to 30. During this process, we running the $SecAP$ algorithm (details in algorithm 3) to get the maximum perfect match for the current subgraph of tight edges in step 17. If we cannot find the maximum perfect match, we

will update the feasible vertex labeling (from step 19 to 30) and then repeat to find the augmenting path by $SecAP$ algorithm. For the feasible vertex labeling X, Y, the difference of each path *diff* and $less$ and the edge weights which are involved in algorithm 3 step 5 are ciphertext of Paillier, we calculate and modify them based on the homomorphic properties of Paillier. What's more, algorithm 3 $SecAP$ is a secure algorithm to find augmenting path (maximum perfect match) recursively.

Algorithm 2 KM in encrypted data.

Input: bipartite graph G_k with encrypted edge weights
Output: task assignment instance set I_p

1: The SC platform get the encrypted edge weight e_{ij} by the input bipartite graph G_k, $e_{ij} \leftarrow E_p(t_{ij}'^2)$ where $j \in m$ and $i \in n$. And $m = n$
2: **for** each worker i in n **do**
3: $Xvisit[i] \leftarrow False$
4: $X[i] \leftarrow Secmax(E_p(t_{i1}'^2), ..., E_p(t_{im}'^2))$
5: $match[i] \leftarrow 0$
6: **for** each task j in m **do**
7: $Yvisit[j] \leftarrow False$
8: $Y[j] \leftarrow E_p(0)$
9: **for** each worker i in n **do**
10: **for** each task j in m **do**
11: $less[j] \leftarrow E_p(\infty)$
12: **while** true **do**
13: **for** each worker i in n **do**
14: $Xvisit[i] \leftarrow False$
15: **for** each task j in m **do**
16: $Yvisit[j] \leftarrow False$
17: **if** SecAP(i) **then**
18: break;
19: $diff \leftarrow E(\infty)$
20: **for** each task j in m **do**
21: **if** $Yvisit[j] = True$ **then**
22: $diff \leftarrow Secmin(diff, less[j])$
23: **for** each worker i in n **do**
24: **if** $Xvisit[i] = True$ **then**
25: $X[i] \leftarrow X[i] * diff^{-1}$
26: **for** each task j in m **do**
27: **if** $Yvisit[j] = True$ **then**
28: $Y[j] \leftarrow Y[j] * diff$
29: **else**
30: $less[j] \leftarrow less[j] * diff^{-1}$
31: the SC platform get I_p from $match[i], i \in m$
32: **return** I_p

Algorithm 3 SecAP.

Input: Worker ID i
Output: True or False

```
Xvisit[i] ← True
for each task j in m do
    if Yvisit[j] = True then
        continue;
    gap ← X[i] * Y[j] * e_ij^-1
    if SecCompare(gap,0) then
        Yvisit[j] ← True
        if match[j] = 0 || DFS(match[j]) then
            match[j] ← i, return True
    else
        less[j] ← Secmin(less[j], gap)
return False
```

4 Security and complexity analysis

4.1 Security analysis

Theorem 1 *Our SKM-EG model is allowed to be privacy-preserving with* $K_0 = V_{lcm}$, $K_{-1} = \{V_{lcm}, \alpha(d_{ij}^2) + \beta, \alpha(w_{ij}) + \beta\}$ *and* $K_i = V_{lcm}(1 \leq i \leq n)$ *extra knowledge.*

Proof We firstly consider the privacy of SC platform. For SC, the observed view is $V_0 = \{E_p(s_j), E_p(t_{ij}), E_p(t'_{ij}), V_{lcm}\}$. We also assume there is a probabilistic polynomial-time simulator P_0 that generates $V'_0 = \{E_p(s_j), E_p(x_i), E_p(y_i), V_{lcm}\}$ where s_j is a serial number defined by SC platform randomly and $x_i(1 \leq i \leq m * n)$ and $y_i(1 \leq i \leq m * n)$ are random numbers uniformly distributed in $\mathbb{Z}_N$. As Paillier is secure, it is clear that view V_0 is indistinguishable from view V'_0, thus, the privacy of SC is preserved.

Next we analyze each worker u_i with $K_i = V_{lcm}(1 \leq i \leq n)$. The view for each worker u_i is $V_i = \{V_{lcm}, E_p((l_{jx})^2 + (l_{jy})^2), E_p(l_{jx}), E_p(l_{jy})\}$ and the view generated by simulator P_i is $V'_i = \{V_{lcm}, E_p(x_1), E_p(x_2), E_p(x_3)\}$ where $x_i(i = 2, 3)$ are random numbers follow Gaussian $\mathcal{N}(500, 400^2)$ and x_1 is the sum of the square of x_2 and $x3$. Based on the semantic security of Paillier, we can easily verify that $V_i \equiv V'_i(1 \leq i \leq n)$, thus, the privacy of each worker is also preserved.

Finally, we analyze the privacy of CCP u_{-1} with $K_{-1} = \{V_{lcm}, \alpha(d_{ij}^2) + \beta, \alpha(w_{ij}) + \beta\}$. The view observed by CCP is $V_{-1} = \{V_{lcm}, \alpha(d_{ij}^2) + \beta, \alpha(w_{ij}) + \beta\}$. We also assumed that there is a probabilistic polynomial-time simulator P_{-1} generating $V'_{-1} = \{V_{lcm}, \alpha(x_i) + \beta, \alpha(y_j) + \beta\}$ where each $x_i(1 \leq i \leq m * n)$ and and each $y_j(1 \leq j \leq m * n)$ are the squares of the distance which are computed from two random numbers following Gaussian $\mathcal{N}(500, 400^2)$. What's more, α and β are two numbers randomly selected from prime field $\mathcal{Z}_q$. As the randomness of α and β and also the security of (7), V_{-1} is indistinguishable from v'_{-1} which means the privacy of CCP holds.

Based on the above proofs, our SKM-EG model is secure with K disclosure where has neglected effects on individual privacy. □

Theorem 2 *Our SKM-AG model is allowed to be privacy-preserving with* $K_0 = V_{lcm}$, $K_{-1} = \{V_{lcm}, \alpha(w_{ij}) + \beta\}$ *and* $K_i = V_{lcm}(1 \leq i \leq n)$ *extra knowledge.*

Proof Here, we focus on the security proof of CCP as the security proof of the SC platform u_0 with $K_0 = V_{lcm}$ and every worker u_i with $K_i = V_{lcm}(1 \leq i \leq n)$ are similar to the proof of Theorem 1. For CCP u_{-1} with $K_{-1} = \{V_{lcm}, \alpha(w_{ij}) + \beta\}$, the view is $V_{-1} = \{V_{lcm}, \alpha(w_{ij}) + \beta\}$. We also assume there is a probabilistic polynomial-time simulator P_{-1} that generates $V'_{-1} = \{V_{lcm}, \alpha(x_i) + \beta\}$ where each $x_i(1 \leq i \leq m * n)$ is the square of the distance which is computed from two random numbers following Gaussian $\mathcal{N}(500, 400^2)$. Since α and β are two random numbers selected from prime field $\mathcal{Z}_q$, and the security of (7) holds, thus it's easy to prove that V_{-1} is indistinguishable from v'_{-1} and the security of CCP holds.

Based on the above proofs, our SKM-AG model is secure with K disclosure where has neglected effects on individual privacy. □

4.2 Complexity analysis

In our system, every worker computes and communicates in parallel. Ignoring some cheap operations, we will analyze the complexity of SKM-EG and SKM-AG from aspect of SC platform, CCP, and each worker. L_i $(i = p, a)$is the key size of Paillier or AP encryption strategy. Due to the size of ciphertext by Paillier is larger than plaintext and the ciphertext by AP, we exclude the latter two from communication cost E_p, E_a, are the encryption and decryption operation of Paillier and AP. We assume that we have n workers and m tasks at round p. At each round, our system needs to run this flow for a round. The complexity of SKM-EG and SKM-AG in cycle p are shown in Tables 2 and 3.

5 Experiment study

5.1 Related algorithms introduction

There are many existing works on spatial crowdsourcing task assignment, most of them adopt greedy selection strategies. However, the spatial crowdsourcing models and problems in these works are different from ours and the existing algorithms cannot be chosen for comparison directly. In order to evaluate the effectiveness of our privacy-preserving task assignment strategies SKM-EG and SKM-AG, we summarize the basic idea of these works

Table 2 Computation Cost of SKM-EG and SKM-AG

	Computation Cost		
	SC platform	CCP	Worker u_i
EG	$3mE_p$	mnD_p	$2E_p$
AG	$3mE_p$	D_a	E_p+E_a
SKM	$2E_p$	mnD_p+$5mmnD_p$	0

Table 3 Communication Cost of SKM-EG and SKM-AG

	Communication Cost		
	SC platform	CCP	Worker u_i
EG	$3mL_p$	mnL_p	$5mL_p$
AG	$3mL_p$+nL_p	$(n+1)L_a$	$4mL_p$+$2L_a$
SKM	mnL_p+nL_p +$8nmmL_p$	mnL_p+nL_p+$8nmmL_p$	0

and design three task assignment algorithms based on different heuristic strategies for comparison: G-MP, G-MT and G-MD. Like the SKM-EG and SKM-AG strategies, these three algorithms are performing task assignment based on the encrypted bipartite graph. Specifically, G-MP strategy is selecting the minimum travel time task-worker pair based on sorted travel times iteratively. G-MT strategy is selecting minimum travel time worker for each task when it is released on the SC platform. G-MD strategy is selecting minimum travel distance worker for each task.

5.2 Experiment settings

We conduct our experiments on synthetic dataset. We will introduce the data setting in below. We use a $1000 * 1000$ 2 dimensional space as working space. We generate workers and tasks that appear the SC platform dynamically. The location of them follows Gaussian $\mathcal{N}(500, 400^2)$ and the speed of workers follows Gaussian $\mathcal{N}(5, 2.5^2)$. The arriving time of all workers and tasks follow Poisson distribution. What's more, for comparing effectiveness, we also generate tasks and workers with their arriving time follows uniform distribution. The distance function dis is Euclidean distance function.

Two criteria are introduced to evaluate our proposed framework, namely computing time and average waiting time respectively. For computing time, we compare the construction ways of encrypted bipartite graph EG and AG and compare SKM-EG and SKM-AG strategies to those three greedy strategies which are mentioned in Section 5.1. For average waiting time, we compare the results which are produced by our SKM-EG, SKM-AG strategies and above three greedy strategies.

Tables 4 and 5 summarize the parameters in comparisons. In our simulation, we set the number of workers and tasks as 100,200,300,400,500 and the time interval τ are 2,5,10,20,50. We assume that the time window of our system is 10.

The algorithms are implemented in Python and the experiments are performed on a PC with i7-7700K CPU and 16G memory.

5.3 Experimental results

5.3.1 Efficiency

As shown in Figure 2a, the computing time of EG strategy is more than AG strategy. It is easy to explain that the EG strategy should send the encrypted edge weight to CCP for

Table 4 Evaluation Settings of Efficiency

Parameter	Value
Number of workers $\|U\|$ and tasks $\|S\|$	50, 60, 70, 80, 90, 100

Table 5 Evaluation settings of effectiveness

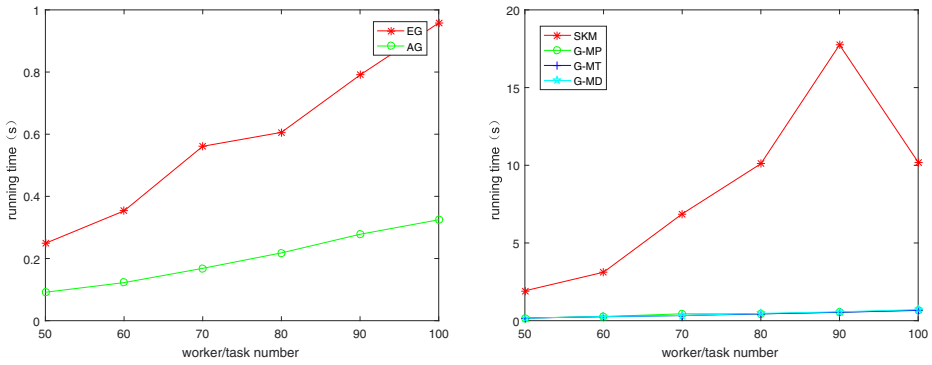
Parameter	Value
Number of workers $\|U\|$	100, 200, 300, 400, 500
Number of tasks $\|S\|$	100, 200, 300, 400, 500
Time interval τ	2, 5, 10, 20, 50

decryption and then CCP need to square it. What's more, the secondary infilling for communication also take some time. Besides, we could find that the running time of both EG and AG are short and we could apply it in practice.

Figure 2b shows the running time of Secure KM, G-MP, G-MT and G-MD. It is obvious that the running time of SKM is clearly longer than others. However, the running time is acceptable and practicability. We can observe that the running time of SKM when worker/task number is 100 is nearly equal to the worker/task number is 80. The reason is that it ran into a coincidence when the worker/task number is 100 and it does not have to do many rounds of calculations to get the assignment set. What's more, our work solve the problem that key size cannot be set too long which was meet in [16]. We can compute the real traveling time without consider the problem that too much workers may lead to the overflow of result. In other words, the most important meaning for our work is to break through the speed limitation of Liu et al.'s framework, and we put this method into real-world practice.

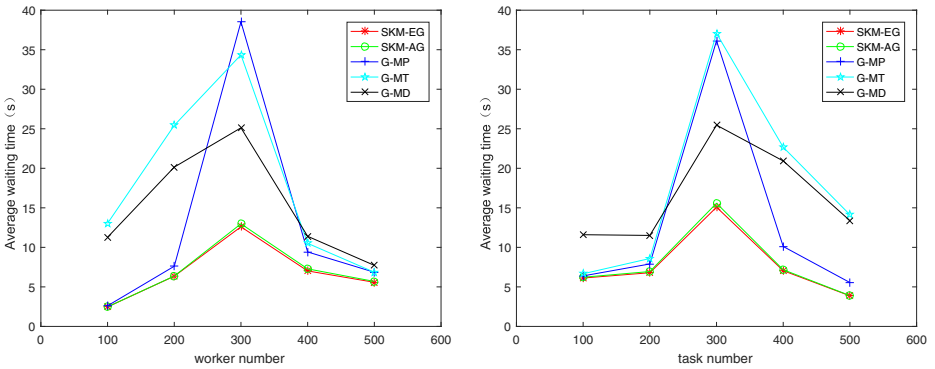
5.3.2 Effectiveness

Figure 3a and b show the performance of our strategies on varying the number of workers and tasks. It is obvious that our SKM-EG and SKM-AG strategies perform much better than others and SKM-AG is very close to the local optimal solution SKM-EG. Specifically, in Figure 3a, we set the number of tasks as 300 and set the number of workers as 100 to 500. We observe that the average waiting time when task number is 300 is higher than others. To explain this, it is necessary for algorithms to match all of the workers and tasks and all of them could be matched because the number of workers is equal to the number of tasks. As a result, some remote workers and tasks are matched and it may cause a long waiting time. What's more, when the worker number is less than 300, we can find that SKM-EG,



(a) Effect of the number of workers/tasks (b) Effect of the number of workers/tasks

Figure 2 Effect of Workers/Tasks Number

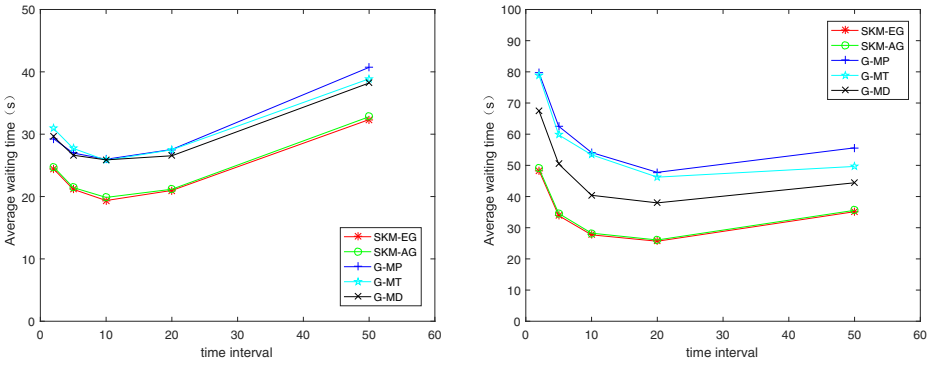


(a) Effect of the number of workers $|U|$ (b) Effect of the number of tasks $|S|$

Figure 3 Effect of Workers/Tasks Number

SKM-AG and G-MP strategies perform better than others. The reason is that G-MT and G-MD do task assignment for each task when task arrived while task number is more than worker number and this could miss better matching for the loss of perspective. When the worker number is more than 300, G-MD performs worst shows that it is better for us to choose travel time instead of travel distance. In Figure 3b, we set the number of workers as 300 and the number of tasks as 100 to 500. The characteristics of the result are similar to Figure 3a whether the number of workers and tasks is equal or not. Finally, our SKM-AG strategy is not sensitive to the quantitative relationship between tasks and workers and it always performs well.

Figure 4a and b show the performance of our strategies on varying the time interval τ. We generate workers and tasks while their arriving time follow Poisson distribution and uniform distribution. Firstly, we can intuitively observe that our SKM-AG strategy is very close to the local optimal solution SKM-EG and performs much better than others. Secondly, when time interval τ is small, the average waiting time of all tasks is longer than the adjacent value. The reason is that only a few number of workers are available when time interval τ is small and it is inevitable that some of worker-task pairs have a long travel time. Thirdly,



(a) Effect of the number of workers $|U|$ (b) Effect of the number of tasks $|S|$

Figure 4 Effect of time interval τ

when time interval τ is large, the average waiting time of all task is longer than the adjacent value too. It is caused by that there exists many tasks which appears early but waits a long time for matching.Finally, both of the Figure 4a and b show that it is better for us to set time interval τ as 10 to 20 instead of too large or too small. The distribution of both tasks' and workers' arriving time has no effect on our SKM-AG strategy and it has similar performance in Figure 4a and b.

6 Related work

In this section, we review representative work from three aspects, task assignment for spatial crowdsourcing, online matching problems and privacy protection in spacial crowdsourcing.

Spatial crowdsourcing is becoming prevalent in both research community (e.g., [2, 3, 28, 29]) and industry (e.g., Waze, DiDi). Task assignment is the core problem in spatial crowdsourcing [3–6, 11, 30, 31, 33–35, 38]. To our best knowledge, [11] firstly proposed task assignment problem in spatial crowdsourcing. Their objective is to maximize the total number of assigned tasks. Some follow up works also focused on the same objective such as [30]. In addition, some other works focused on how to do better task assignment and propose different constraints and goals for new application scenarios. [11] aims to find a maximum-cardinality matching with minimum total distance and [5] aims to maximize the reliability and diversity of finished tasks. In [38], the workers acceptance was maximized to improve the system throughput. In the real world scenario, both workers and tasks arrive platform dynamically. However, some of the earlier studies focused on static scenario. [10] first used an online model to describe the assignment process. Most of existing works consider the static scenario and the economic benefits for the crowdsourcing platform while we focus on the user experience and aim to find a maximum-cardinality matching with minimum total time between the matched worker-task pairs in the online scenario.

In [28], Tong et.al categorizes task assignment in spatial crowdsourcing into (static) offline and (dynamic) online scenarios. In online scenarios, there are two ways to deal with the workers and tasks which dynamically appear one by one in the physical space: (1) batch-based matching approach [3, 6, 11, 29]; (2) online matching approach (i. e. , workers/tasks are assigned as soon as they reach the platform [4, 31, 33, 35]). Our work utilizes batch-based matching approach in online scenarios and extends [3, 6, 11, 29] in the privacy protection aspect.

Privacy protection is an emerging issue in spatial crowdsourcing. It focuses on protecting the information (e.g., location, speed) of workers and tasks in dynamic scenarios. State-of-the-art techniques about privacy protection are as follows: (1)Cloaked Locations-based protection technique (i.e., the location of workers is transformed as a cloaked area in [23]); (2)Differential Privacy-based protection technique, (i.e., workers send their locations to a trusted third party and the third party sanitizes the location of workers according to differential privacy techniques, such as [8, 26, 27, 37]); (3)Encrypted Data-based protection(i.e., the exact distances between tasks and workers can be computed based on their encrypted locations in [15, 16, 18, 25]). Our work utilizes Encrypted Data-based protection technique to improve [16, 18, 25] in two aspects. First, [18, 25] is based on the encrypted data of workers' location and tasks' location to calculate the encrypted data of the distance, involving the add operation of encrypted data. However, our work needs us to calculate the encrypted data of the travel time , involving the ciphertext division operation. Second, [16] can only be small calculation, including homomorphic division operation on the ciphertext. While

our division operation is based on the least common multiple ways, which can process large-scale computing.

7 Conclusion

In this paper, we have studied a novel task assignment problem in spatial crowdsourcing, named user experience-driven secure task assignment problem, which finds k task assignment instance sets in $k\tau$ time units to minimize the average waiting time of all tasks while satisfying the security requirement. We have presented two methods to construct encrypted bipartite graph to protect private data of both workers and users. We have also enhanced conventional KM algorithm and made it workable on encrypted graph. Extensive experiments have been conducted to demonstrate the efficiency and effectiveness of our proposed approach.

Acknowledgements This paper is partially supported by Natural Science Foundation of China (Grant No. 61572336, No. 61572335, No. 61632016, No. 61772356, No. 61802344, No. 61602400, No. 61702227), and Natural Science Research Project of Jiangsu Higher Education Institution (No. 18KJA520010, No. 17KJA520003), and a Hong Kong Polytechnic University start-up fund (project no. 1.9B0V), and a Project Funded by the Priority Academic Program Development of Jiangsu Higher Education Institutions.

References

1. Araki, T., Furukawa, J., Lindell, Y., Nof, A., Ohara, K.: High-throughput semi-honest secure three-party computation with an honest majority. In: Proceedings of the 2016 ACM SIGSAC conference on computer and communications security, Vienna, Austria, October 24-28, 2016, pp. 805–817 (2016)
2. Chen, Y.-Y., Guo, D.-K., Zhou, T.-Q., Xu, M.: A survey on task and participant matching in mobile crowd sensing. JCST **33**(4), 768–791 (2018)
3. Cheng, P., Jian, X., Chen, L.: An experimental evaluation of task assignment in spatial crowdsourcing. VLDB **11**(11), 1428–1440 (2018)
4. Cheng, P., Lian, X., Chen, L., Shahabi, C.: Prediction-based task assignment on spatial crowdsourcing. In: ICDE, pp. 997–1008 (2017)
5. Cheng, P., Lian, X., Chen, Z., Fu, R., Chen, L., Han, J., Zhao, J.: Reliable diversity-based spatial crowdsourcing by moving workers. VLDB **8**(10), 1022–1033 (2015)
6. Deng, D., Shahabi, C., Zhu, L.: Task matching and scheduling for multiple workers in spatial crowdsourcing. In: SIGSPATIAL, no. 21 (2015)
7. Dong, C., Chen, L., Wen, Z.: When private set intersection meets big data: an efficient and scalable protocol. In: 2013 ACM SIGSAC conference on computer and communications security, CCS'13, Berlin, Germany, November 4-8, 2013, pp 789–800 (2013)
8. Fan, L., Xiong, L.: An adaptive approach to real-time aggregate monitoring with differential privacy. TKDE **26**(9), 2094–2106 (2014)
9. Goldreich, O.: Foundations of cryptography: volume 2, basic applications. Cambridge University Press, Cambridge (2009)
10. Hassan, U.U., Curry, E.: A multi-armed bandit approach to online spatial task assignment. In: 11rd IEEE international conference on ubiquitous intelligence and computing and autonomic and trusted computing and scalable computing and communications, U.C-ATC-ScalCom 2014, Bali, Indonesia, Dec 9-12, 2014, pp. 64 (2014)
11. Kazemi, L., Shahabi, C.: Geocrowd: Enabling query answering with spatial crowdsourcing. In: SIGSPATIAL, pp. 189–198 (2012)
12. Kuhn, H.W.: The hungarian method for the assignment problem. Nav. Res. Logist. Q. **2**(1-2), 83–97 (1955)
13. Li, J., Liu, A., Wang, W., Li, Z., Liu, G., Zhao, L., Zheng, K.: Towards privacy-preserving travel-time-first task assignment in spatial crowdsourcing. In: APWeb-WAIM, pp. 19–34 (2018)

14. Li, Q., Cao, G., La Porta, T.F.: Efficient and privacy-aware data aggregation in mobile sensing. TDSC **11**(2), 115–129 (2014)
15. Liu, A., Li, Z.-X., Liu, G.-F., Zheng, K., Zhang, M., Li, Q., Zhang, X.: Privacy-preserving task assignment in spatial crowdsourcing. J. Comput. Sci. Technol. **32**(5), 905–918 (2017). [Online]. Available: https://doi.org/10.1007/s11390-017-1772-5
16. Liu, A., Wang, W., Shang, S., Li, Q., Zhang, X.: Efficient task assignment in spatial crowdsourcing with worker and task privacy protection. GeoInformatica **22**, 335–362 (2018)
17. Liu, A., Zheng, K., Li, L., Liu, G., Zhao, L., Zhou, X.: Efficient secure similarity computation on encrypted trajectory data. In: ICDE, pp. 66–77 (2015)
18. Liu, B., Chen, L., Zhu, X., Zhang, Y., Zhang, C., Qiu, W.: Protecting location privacy in spatial crowdsourcing using encrypted data. In: EDBT (2017)
19. Liu, J., Yang, J., Xiong, L., Pei, J.: Secure skyline queries on cloud platform. In: 33rd IEEE International Conference on Data Engineering, ICDE 2017, San Diego, CA, USA, April 19-22, 2017, pp. 633–644 (2017)
20. Meng, X., Zhu, H., Kollios, G.: Top-k query processing on encrypted databases with strong security guarantees. In: 34th IEEE International Conference on Data Engineering, ICDE 2018, Paris, France, April 16-19, 2018, pp. 353–364 (2018)
21. Munkres, J.: Algorithms for the assignment and transportation problems. J. Soc. Ind. Appl. Math. **5**(1), 32–38 (1957)
22. Paillier, P., et al.: Public-key cryptosystems based on composite degree residuosity classes. In: Eurocrypt, vol. 99. Springer, pp. 223–238 (1999)
23. Pournajaf, L., Xiong, L., Sunderam, V., Goryczka, S.: Spatial task assignment for crowd sensing with cloaked locations. In: MDM (2014)
24. Reddaway, S.: Pseudo-random number generators, May 14 1974, uS Patent 3,811,038
25. Sun, Y., Liu, A., Li, Z., Liu, G., Zhao, L., Zheng, K.: Anonymity-based privacy-preserving task assignment in spatial crowdsourcing. In: WISE, pp. 263–277 (2017)
26. To, H., Ghinita, G., Fan, L., Shahabi, C.: Differentially private location protection for worker datasets in spatial crowdsourcing. TMC **16**(4), 934–949 (2017)
27. To, H., Shahabi, C., Ghinita, G.: A framework for protecting worker location privacy in spatial crowdsourcing. VLDB **7**(10), 919–930 (2014)
28. Tong, Y., Chen, L., Shahabi, C.: Spatial crowdsourcing: Challenges, techniques, and applications. VLDB **10**(12), 1988–1991 (2017)
29. Tong, Y., She, J., Ding, B., Chen, L., Wo, T., Xu, K.: Online minimum matching in real-time spatial data: Experiments and analysis. VLDB **9**(12), 1053–1064 (2016)
30. Tong, Y., She, J., Ding, B., Wang, L., Chen, L.: Online mobile micro-task allocation in spatial crowdsourcing. In: ICDE, pp. 49–60 (2016)
31. Tong, Y., Wang, L., Zhou, Z., Ding, B., Chen, L., Ye, J., Xu, K.: Flexible online task assignment in real-time spatial data. VLDB **10**(11), 1334–1345 (2017)
32. Xiao, M., Ma, K., Liu, A., Zhao, H., Li, Z., Zheng, K., Zhou, X.: Sra: Secure reverse auction for task assignment in spatial crowdsourcing. IEEE Trans. Knowl. Data Eng. **35**, 1–1 (2019)
33. Xiao, M., Wu, J., Huang, L., Cheng, R., Wang, Y.: Online task assignment for crowdsensing in predictable mobile social networks. TMC **16**(8), 2306–2320 (2017)
34. Xiao, M., Wu, J., Huang, L., Cheng, R., Wang, Y.: Online task assignment for crowdsensing in predictable mobile social networks. IEEE Trans. Mob. Comput. **16**(8), 2306–2320 (Aug 2017)
35. Zeng, Y., Tong, Y., Chen, L., Zhou, Z.: Latency-oriented task completion via in spatial crowdsourcing. In: ICDE, pp. 478–481 (2018)
36. Zeng, Y., Tong, Y., Chen, L., Zhou, Z.: Latency-oriented task completion via spatial crowdsourcing. In: ICDE, pp. 317–328 (2018)
37. Zhai, D., Sun, Y., Liu, A., Li, Z., Liu, G., Zhao, L., Zheng, K.: Towards secure and truthful task assignment in spatial crowdsourcing. World Wide Web **22**(5), 2017–2040 (2019). [Online]. Available: https://doi.org/10.1007/s11280-018-0638-2
38. Zheng, L., Chen, L.: Maximizing acceptance in rejection-aware spatial crowdsourcing. TKDE **29**(9), 1943–1956 (2017)