Influence-aware Task Assignment in Spatial Crowdsourcing (Technical Report)

Xuanhao Chen¹, Yan Zhao², Kai Zheng¹*, Bin Yang², Christian S. Jensen²
¹University of Electronic Science and Technology of China, China
²Department of Computer Science, Aalborg University, Denmark
xhc@std.uestc.edu.cn, yanz@cs.aau.dk, zhengkai@uestc.edu.cn, {byang, csj}@cs.aau.dk

Abstract—With the widespread diffusion of smartphones, Spatial Crowdsourcing (SC), which aims to assign spatial tasks to mobile workers, has drawn increasing attention in both academia and industry. One of the major issues is how to best assign tasks to workers. Given a worker and a task, the worker will choose to accept the task based on her affinity towards the task, and the worker can propagate the information of the task to attract more workers to perform it. These factors can be measured as worker-task influence. Since workers’ affinities towards tasks are different and task issuers may ask workers who performed tasks to propagate the information of tasks to attract more workers to perform them, it is important to analyze worker-task influence when making assignments. We propose and solve a novel influence-aware task assignment problem in SC, where tasks are assigned to workers in a manner that achieves high worker-task influence. In particular, we aim to maximize the number of assigned tasks and worker-task influence. To solve the problem, we first determine workers’ affinities towards tasks by identifying workers’ historical task-performing patterns. Next, a Historical Acceptance approach is developed to measure workers’ willingness of performing a task, i.e., the probability of workers visiting the location of the task when they are informed. Next, we propose a Random reverse reachable-based Propagation Optimization algorithm that exploits reverse reachable sets to calculate the probability of workers being informed about tasks in a social network. Based on worker-task influence derived from the above three factors, we propose three influence-aware task assignment algorithms that aim to maximize the number of assigned tasks and worker-task influence. Extensive experiments on two real-world datasets offer detailed insight into the effectiveness of our solutions.

Index Terms—worker-task influence, task assignment, spatial crowdsourcing

I. INTRODUCTION

With the near-ubiquitous diffusion of smartphones and similar devices, a new kind of crowdsourcing has emerged, namely Spatial Crowdsourcing (SC), where smartphone users serve as workers that perform tasks at specific physical locations. In SC, examples of spatial tasks include reporting local hot spots, taking photos or videos of a POI, and monitoring traffic conditions [1].

SC has received substantial attention in the last years [2]–[10]. Studies exist that aim to maximize the total number of completed tasks [11], the diversity score of assignments [12], the number of completed tasks for a worker with an optimal schedule [13], etc. These studies generally focus on the spatio-temporal information of workers and tasks during task assignment, while they do not consider worker-task influence, i.e., how to ensure that assigned tasks satisfy workers’ affinities towards tasks and are well-known among workers who are likely to visit the locations of tasks. Visiting the location of a task is equivalent to accepting the task. In real-world scenarios, different workers prefer different kinds of tasks. Moreover, when completing tasks, workers can propagate information on available tasks to their friends through social networks. Workers who are informed can choose to perform tasks based on their historical task-performing patterns. It is important to analyze such phenomena when assigning tasks. For example, the owner of a new restaurant may want to publish a leaflet distribution task to promote the restaurant as widely as possible. Some free meal coupons and VIP cards are offered to workers who accept the task and help to propagate the news about the restaurant. If we only consider spatio-temporal information, an available worker who close to the restaurant at the current time will be assigned the task, but the worker may not be able to promote the restaurant widely. Thus, the case will not be a successful promotion. In addition, the real-time locations of workers are temporary, which ignores the worker’s historical task-performing patterns. Moreover, by analyzing the social networks that workers are in, we can obtain valuable insights about interactions among workers, which can be further utilized to improve the quality of spatial task assignments.

Recent studies have explored the effects of social impact in task assignment, where social network features are used to extract preference of worker groups [14], [15]. However, these studies do not consider the interactions among workers, which include information propagation patterns and social network structures. Different workers have different abilities to propagate information [16] and different probabilities to visit the locations of tasks [17]. This indicates that different workers contribute differently to worker-task influence. Moreover, it is important to infer task execution behaviors based on historical task-performing records of workers. Several approaches use past task-performing patterns to deduce worker preferences for tasks [1], [18], but they do not analyze the willingness of
workers to perform tasks, i.e., the probabilities that workers will visit the task locations. If a worker previously performed tasks near a new task, the worker is more likely to visit the location of the new task [19]. Lastly, we are not aware of any existing task assignment techniques that combine social networks and historical task-performing patterns to determine worker-task influence, which is a key factor for improving the quality of task assignment in SC.

To address these challenges, we propose the Influence-aware Task Assignment (ITA) problem, where the objective is to assign tasks to suitable workers so as to maximize both the total number of assigned tasks and worker-task influence, which consists of workers’ affinities (namely worker-task affinity) towards tasks, the probability (namely worker willingness) of workers visiting the locations of tasks and the probability (namely worker propagation) of workers being informed about tasks in social networks. Larger worker-task influence means that workers’ affinities towards tasks are larger, and the number of people who are willing to visit the locations of tasks after informed is larger. An example of the ITA problem is illustrated in Figure 1. Workers, \( w_1, w_2, \) and \( w_5 \) performed tasks, \( s_1, s_2, \) and \( s_3 \), at time \( t_1 \), respectively. At time \( t_2 \), workers \( w_4 \) and \( w_5 \) are online, and tasks \( s_4 \) and \( s_5 \) become available. These are tasks published by new restaurants that ask workers to take photos and then advertise the restaurants on social media. The requirement of tasks is to increase the number of people who are willing to visit the location of the restaurant after knowing about it, i.e., enlarging worker-task influence. The circle around each worker denotes the reachable region of the worker at the current time. Because of the budgets of the restaurant, only one worker is required to perform each task, while the worker who are assigned the task should enlarge worker-task influence. A simple greedy approach is to assign tasks to the nearest worker, which gives the task assignment \( \{(s_4, w_3), (s_5, w_5)\} \), where the value of worker-task influence is \(1.67 + 0.85 = 2.52\). However, adopting an influence-aware task assignment approach, we can achieve a higher worker-task influence with assignment \( \{(s_4, w_4), (s_5, w_5)\} \), where the value of worker-task influence is \(4.25 + 0.85 = 5.1\).

Worker-task influence can be computed by worker-task affinity (i.e., workers’ affinities towards tasks), worker willingness (i.e., the probability of workers visiting the locations of tasks) and worker propagation (i.e. the probability of workers being informed about tasks in social networks). However, there exists a challenge of how to combine worker-task influence with existing objectives such as maximizing the number of assigned tasks. In other words, influence-aware assignment should optimize for worker-task influence without sacrificing other objectives. To achieve this, we propose a Data-driven Influence-aware Task Assignment (DITA) framework, consisting of two primary components. First, worker-task influence is calculated, which not only considers online interactions among workers, but also captures workers’ historical task-performing patterns and real-time task assignments mode. Second, we design three algorithms to maximize the overall task assignments by giving higher priorities to workers who generate higher worker-task influence at every time instance.

The paper’s contributions can be summarized as follows:

i) We formalize and study an Influence-aware Task Assignment (ITA) problem in the context of SC. To the best of our knowledge, this is the first study in SC that considers worker-task influence and assigns tasks based on the influence.

ii) We calculate worker-task influence by taking into account worker-task affinity, worker willingness and worker propagation.

iii) We design three alternative algorithms to solve the ITA problem, including basic Influence-aware Assignment, Entropy-based Influence-aware Assignment, and Distance-based Influence-aware Assignment.

iv) We conduct extensive experiments on two real-world datasets to offer insight into the effectiveness of the proposed methods.

II. PROBLEM STATEMENT

We present necessary preliminaries, define the problem addressed, and give an overview of our solution framework. Table 1 lists notation used throughout the paper.

A. Preliminary Concepts

Definition 1 (Spatial Task) A spatial task, denoted by \( s = (l, p, \varphi, C) \), has a location \( s.l \), a publication time \( s.p \), a valid time \( \varphi \) (meaning that it will expire at \( s.p + s.\varphi \)), and multiple category labels \( s.C \).

Definition 2 (Worker) A worker, denoted by \( w = (l,r) \), consists of a location \( w.l \) and a reachable distance \( w.r \).
reachable range of worker \( w \) is a circle with center \( w.l \) and radius \( w.r \), within which \( w \) can accept assignments.

A spatial task \( s \) can be completed only if a worker arrives at its location before the expiration deadline \( s.p+s.p \). With single-task assignment mode, the SC server assigns each task to one worker at a time.

**Definition 3 (Worker-Task Influence)** Given a worker \( w \) and a task \( s \). Worker-task influence (calculated in Section III), denoted as \( if(w,s) \), consists of \( w \)'s affinity towards \( s \), the probability of other workers visiting the location of \( s \) after informed by \( w \), and the probability of other workers who are informed by \( w \) through social networks.

**Definition 4 (Spatial Task Assignment)** Given a set of tasks \( S \) and a set of workers \( W \), a spatial task assignment, denoted by \( A \), consists of a set of worker-task pairs of the form \((s,w)\), where task \( s \) is assigned to worker \( w \) satisfying the spatio-temporal constraints, and where each worker or task can be assigned at most once.

We use \(|A|\) to denote the total number of assigned tasks in task assignment \( A \). The problem investigated is stated as follows:

**ITA Problem Statement.** Given a set of workers and a set of tasks at the current time in an SC platform, our problem is to find a task assignment \( A_{opt} \) that achieves the following goals:

1) primary optimization goal: maximize the total number of assigned tasks (i.e., \( \forall A_i \in A, (|A_i| \leq |A_{opt}|) \)), where \( A \) denotes all possible assignments; and

2) secondary optimization goal: maximize worker-task influence of assignments.

**Lemma 1** The ITA problem is NP-hard.

**Proof** We can prove the lemma through a reduction from the 0-1 knapsack problem, which is described as follows: Given a set \( U \) with \( n \) items, in which each item \( u_i \) is labelled with a weight \( l_i \) and a value \( h_i \), the 0-1 knapsack problem is to find a subset \( U^* \) of \( U \) that maximizes \( \sum_{u_i \in U^*} h_i \), subject to \( \sum_{u_i \in U^*} l_i \leq L \), where \( L \) is the maximum weight capacity.

Consider the following instance of the ITA problem. Given a task set \( S \) with \( n \) tasks, each task \( s_i \in S \) is associated with a worker (corresponding to the weight \( l_i = 1 \) of the 0-1 knapsack problem). Here, the number of workers is sufficiently large. The value \( h_i \) of each task \( s_i \) that is a function related to task completion and worker-task influence, is at least as hard as the \( h_i \) (that is a constant) in the 0-1 knapsack problem, so that this difference does not make our problem easier. In addition, we have \( L \) workers. Therefore, the ITA problem is to identify a task subset \( S^* \) of \( S \) that maximizes \( \sum_{s_i \in S} h_i \) subject to \( \sum_{s_i \in S} l_i \leq L \).

If the ITA problem instance can be solved in polynomial time, a 0-1 knapsack problem can be solved by being transformed to the corresponding ITA problem instance and then it can be solved in polynomial time. This contradicts the fact that the 0-1 knapsack problem is NP-hard [20], and so there cannot be an efficient solution (i.e., in polynomial time) to the ITA problem instance that is then NP-hard. Since the ITA problem instance is NP-hard, the ITA problem is also NP-hard.

**B. Framework Overview**

We propose a framework, Data-driven Influence-aware Task Assignment (DITA), to solve the ITA problem. The framework has two components: worker-task influence modeling and task assignment, as shown in Figure 2.

The first component aims to calculate worker-task influence. Specifically, we employ Latent Dirichlet Allocation (LDA) to measure workers' affinities (i.e., worker-task affinity) towards tasks, where we treat the categories of tasks that workers have already completed as documents to train the LDA model, and then the categories of tasks and workers at the current time are input into the trained LDA model to compute the worker-task affinity. For worker willingness calculation, we propose a Historical Acceptance (HA) algorithm to measure the probability of a worker visiting the location of a task based on the task-performing history of the worker and the real-time locations of the worker and task. For worker propagation, we first exploit an Independent Cascade (IC) model to simulate information propagation process of tasks in a given social network, and then we propose a Random reverse reachable-based Propagation Optimization (RPO) algorithm to calculate worker propagation based on IC and social networks.

In the task assignment component, considering the spatio-temporal constraints (i.e., the reachable regions of workers and expiration times of tasks) of workers and tasks, we optimize the task assignment based on worker-task influence at each time instance and propose a basic Influence-aware Assignment (IA) method. Taking worker-task influence and location entropy into account, we propose an Entropy-based Influence-aware Assignment (EIA) method. Moreover, a Distance-based Influence-aware Assignment (DIA) method which considers worker-task influence and workers’ travel costs is developed.
III. WORKER-TASK INFLUENCE CALCULATION

We proceed to detail how to calculate worker-task influence for a worker and a task. We cover worker-task affinity, worker willingness, worker propagation and worker-task influence.

A. Worker-Task Affinity Calculation

In SC, different workers exhibit different affinities (i.e., preferences) for the same categories of tasks, leading to different task-performing behaviors. For example, a worker may like to report a hot spot, while another worker may prefer to monitor traffic conditions. Since task categories contain semantic information (e.g., restaurant) and the Latent Dirichlet Allocation (LDA) model [21] performs well at modeling semantic affinity (i.e., semantic matching) between text documents by learning topics, we employ it to quantify worker-task affinity.

In LDA, a document is regarded as a set of words generated by several topics, where each topic is described by terms following a probability distribution. The modeling process can be formalized as follows:

$$P(v_i | d) = \sum_{j=1}^{[Top]} P(v_i | t_j) P(t_j | d)$$

Here $P(v_i | d)$ is the probability of term $v_i$ for a document $d$ and $[Top]$ is the number of topics. Next, $P(v_i | t_j)$ is the probability of picking a term from $t_j$ in document $d$. LDA estimates the topic-term distribution, $P(v_i | t_j)$, and the document-topic distribution, $P(t_j | d)$, using Dirichlet priors. It iterates multiple times over each term in $d$ until the parameters in LDA converge. This way, we get the topic distribution of each document. Each topic is a probability distribution over a set of words. In the LDA model, words that are related semantically have high probability of belonging to the same topic.

In order to adapt LDA to model worker-task affinity, we treat each task category as a word, and we treat the categories of tasks in the historical task-performing records of worker $w_i$ as a document, denoted by $dc_{w_i}$. The documents across all workers on an SC platform form a set of documents that is used to train the LDA model, cf. Figure 3. Based on the documents, LDA can learn topics. Each topic is represented by a probability distribution over categories. For worker $w_i$ and task $s_i$ at the current time, we can use the trained LDA model to calculate the topic distribution, where the topic distribution of $w_i$ is calculated from the historical task-performing records that reflect the preferred category distribution of $w_i$, and the topic distribution of $s_i$ is calculated based on its categories. Then the learned topics and the document $dc_{s_i}$ formed by the categories of the location of $s_i$ are used to estimate the worker-task affinity, $P_{aff}$, as follows:

$$P_{aff}(w_i, s_i) = \sum_{t \in Top} P(w_i | t) \cdot P(s_i | t),$$

where $t$ denotes a topic and $Top$ is a set of learned topics. Further, $P(w_i | t)$ and $P(s_i | t)$ quantify how well topic $t$ matches the topic distribution of $w_i$'s historical task-performing records and the topic distribution of task $s_i$, respectively. A larger $P_{aff}(w_i, s_i)$ value indicates that $w_i$ is more likely to perform $s_i$, since the preferred category distribution of $w_i$ and that of the task $s_i$ are correlated better.

B. Worker Willingness Calculation

In general, different workers exhibit different willingness to visit the location of a task. Previous studies only consider real-time locations of workers and tasks [22], [23] when assigning tasks. However, measuring a worker’s willingness to visit the location of a task according to the distance between the worker’s real-time location and the task location represents an incomplete picture. The real-time location is temporary, and this ignores the worker’s historical task-performing patterns.

To tackle this issue, we propose a Historical Acceptance (HA) approach to measure the willingness of a worker $w$ to visit the locations of particular tasks based on the worker’s historical task-performing records (denoted as $S_w$) and the real-time locations of workers and tasks, where $S_w = \{(s_1, t_1, t_1), (s_2, t_2, t_2), \ldots, (s_n, t_n, t_n)\}$ and each triplet $(s_i, t_{s_i}^a, t_{s_i}^c)$ consists of task $s_i$, a task arrival time $t_{s_i}^a$, and a task completion time $t_{s_i}^c$. The worker willingness is measured as the probability that a worker moves from the locations of the tasks they have performed to the location of the current task.
In particular, HA computes worker willingness in terms of stationary distribution modeling of workers’ historical mobility and movement probability density calculation.

1) Stationary Distribution Modeling of Workers’ Historical Mobility: The stationary distribution of a worker’s historical mobility captures the probability that a worker w stays at the location of a performed task \( s_i \), denoted as \( P_w(w, s_i) \). This probability can be computed using the Random Walk with Restart (RWR) method, which is an efficient approach to simulating the movement of objects [24]. In order to adapt the RWR method to compute the stationary distribution of a worker’s historical mobility, we exploit workers’ historical task-performing records \( S_w \) (ordered by check-in time) to construct an \( n \times n \) weight matrix for worker w \( (w \in W) \), where \( n \) is the number of tasks performed by w. The weight of item \( w_{ij} \) in the \( i \)-th row and \( j \)-th column is set to \( 1/\sum_j m_{ij} \), where \( m_{ij} = 1 \) if w performed tasks at the \( j \)-th location; otherwise, \( m_{ij} = 0 \).

2) Movement Probability Density Calculation: The movement probability density of worker w is the probability density of moving from the location of task \( s_i \) to the location of the next task, \( s_{i+1} \), denoted as \( f_w(d(s_i, s_{i+1})) \), where \( f_w \) is a probability density function and \( d(s_i, s_{i+1}) \) is the distance between the location of \( s_i \) and the location of \( s_{i+1} \). Previous studies [25], [26] show that the movements of workers are self-similar. Since the random variable described by the Pareto distribution obeys the self-similarity property [27], we choose the Pareto distribution to measure the movement probability density of worker w, denoted as \( f_w(x; \pi; \omega) = \frac{\pi \omega}{\pi - 1} x^{\pi-1} \), where \( x \) is the distance between locations of tasks. \( \pi \) is a shape parameter that can be calculated using maximum likelihood estimation, and \( \omega \) is the minimum value of \( x \). Since real-time tasks are not known in advance, we use \( S_w \) (ordered by check-in time) to compute \( f_w \). Note that different task-performing orders will lead to different \( f_w \). As a worker may perform several tasks at the same location, the minimum value of \( d(s_i, s_{i+1}) \) is 0, where \( s_i \) and \( s_{i+1} \) are tasks in \( S_w \). We set \( x_i \) of \( f_w \) to \( d(s_i, s_{i+1}) + 1 \) to avoid \( x_i \) being 0. In this case, \( \omega = 1 \). Based on \( x_i \), we employ maximum likelihood estimation to estimate \( \pi \) of \( f_w \). Further, \( \pi \) can be calculated using the following equation:

\[
\frac{d}{d\pi} \left[ \frac{|S_w|}{\pi} \prod_i \left( \frac{\pi}{x_i^{\pi}} \right)^{1/\pi} \right] = 0
\]

where \(|S_w|\) is the number of performed tasks of w and \( x_i \geq 1 \).

Accordingly, \( \pi \) is given by Equation 1.

\[
\pi = \frac{|S_w|}{\sum_i x_i \ln x_i}, \text{ where } \sum_i |S_w| \ln x_i \neq 0
\]

Since a worker may stay at different locations of performed tasks, we first need to compute the probability of worker w staying at the location of a performed task \( s_i \), and then we combine the probability with the probability that w moves from the location of \( s_i \) to the location of the current task \( s \) to compute worker willingness. Based on Sections III-B1 and III-B2, the willingness of w to visit the location of s can be calculated as follows:

\[
P_{\omega}(w, s) = \sum_{s_i} P_w(w, s_i) \cdot \int_{d(s_i, s)}^{\infty} f_w(x)dx
\]

\[
= \sum_{s_i} P_w(w, s_i) \cdot (d(s_i, s) + 1)^{-\pi}
\]

C. Worker Propagation Calculation

When knowing information of a task, a worker has the potential to propagate the task’s information to other workers independently through social media. We propose worker propagation to measure the probability of workers being known tasks.

There are two main challenges when computing worker propagation. First, since the information propagation in a social network is complex, it is important to simulate the propagation process reasonably. Second, based on the propagation process, the computation of worker propagation should be completed in limited time since we aim to assign tasks to workers online. To address these challenges, we propose an approximation method, called Random reverse reachable-based Propagation Optimization (RPO), for calculating the worker propagation.

1) Random Reverse Reachable Set Generation: We first detail how to generate Random Reverse Reachable (RRR) sets for workers, which will be used in the RPO method. The definition of an RRR set follows.

Definition 5 (Random Reverse Reachable Set) Given a social network \( G = (W, E) \), constructing the reverse graph \( G' \) of \( G \) and selecting a worker \( w_i \) uniformly at random from \( G' \), a subgraph \( g_i \) is a directed graph sampled from \( G' \) under a given propagation model. A random reverse reachable (RRR) set for \( w_i \), is a set of workers in \( g_i \) that can reach \( w \).

To generate an RRR set for each worker, it is important to select a suitable propagation model. Independent Cascade (IC) [17], [28]–[33] is a commonly-used propagation model, where users inform their neighbors independently. In our ITA problem, a worker knowing a task has the potential to propagate information about the task to the neighbors independently, which can be well modeled by IC. Therefore, we use the IC model to simulate the information propagation process of tasks and sample subgraphs from \( G' \) to generate RRR sets.

The IC model is an iterative model. At the beginning of IC model, a worker \( w \) who knows task s is selected to inform the neighbors independently. In each iteration, if a worker \( w_i \) has more than one neighbor knowing the task information in the current iteration, the worker will be informed by these neighbors independently. The probability, \( P_k(w_i) \), of worker \( w_i \) being informed by the neighbors in the \( k \)-th iteration is calculated as follows:

\[
P_k(w_i) = 1 - \prod_{w_j \in NE_{k-1}(w_i)} (1 - P_j(w_j, w_i)),
\]
where \( NE_{k-1}(w_i) \) is the neighbors of \( w_i \) who know a given task in the \((k-1)\)-th iteration, and \( P_j(w_j, w_i) \) is an in-degree-based probability that neighbor \( w_j \) of \( w_i \) informs \( w_i \), which is a ratio between 1 and \( w_i \)'s in-degree.

Workers who are informed have only one chance to inform their neighbors. In the \( k \)-th iteration of the IC model, a worker who is not informed is added to a subgraph \( g_s \) with the probability \( P_k \), and the edges connecting the worker and the neighbors who are informed in the \((k-1)\)-th iteration are added to \( g_s \) with the probability \( P_s \). When no new workers are informed, the propagation process terminates, and \( g_s \) is constructed. Accordingly, the RRR set of worker \( w_s \) is a set of workers that can reach \( w_s \) along a finite number of edges in the directed graph \( g_s \).

2) Random Reverse Reachable-based Propagation Optimization Method: Next, we present the Random reverse reachable-based Propagation Optimization (RPO) method.

Based on Definition 5, if a worker \( w_s \) appears in an RRR set of another worker \( w_i \), the propagation process from \( w_s \) should have a certain probability to inform \( w_i \). Specifically, we can get following lemma.

**Lemma 2** Given two workers \( w_s \) and \( w_i \), the probability that \( w_i \) is informed by \( w_s \) under a propagation process equals the probability that \( w_s \) belongs to an RRR set of \( w_i \) [30].

Given a set of \( N \) RRR sets, \( \mathcal{R} = \{R_1, R_2, \ldots, R_N\} \), and a worker \( w_s \) in \( G \), let \( \mathcal{R}_i \) be a set of RRR sets that are generated by worker \( w_i \) in \( G \). Based on Lemma 2 and the linearity of expectation, we can calculate the informed probability \( P_{pro}(w_s, w_i) \) of \( w_i \) being informed by \( w_s \), as follows:

\[
P_{pro}(w_s, w_i) = \frac{|W|}{N} \sum_{j=1}^{N} P_j \sum_{j=1}^{R_j} v_j,
\]

where \( v_j = 0 \) if \( \{w_j\} \cap R_j = \emptyset \); otherwise, \( v_j = 1 \). \( R_j \) is the \( j \)-th set of \( \mathcal{R}_i \), \( |W| \) is the number of workers in \( G \), \( |R_j| \) is the size of \( R_j \), and \( E \left[ \sum_{j=1}^{R_j} v_j \right] \) is the expected number of RRR sets generated by \( w_i \) that cover \( w_s \). To ensure that the estimation of \( P_{pro}(w_s, w_i) \) is accurate, it is essential that \( N \) is sufficiently large to ensure that \( \sum_{j=1}^{R_j} v_j \) do not deviate significantly from its expectation. The analysis of how to choose a setting for \( N \) is covered in Section III-E.

The whole process of the RPO method is covered in Algorithm 1. The main computational challenge is the huge search space when enumerating all possible RRR sets of each worker, which increases exponentially with respect to the number of workers. Therefore, it is important to obtain a limited number of RRR sets that make it possible to guarantee an approximation ratio of the probability that workers are informed. To achieve this, we propose two lower bounds on the number of RRR sets (an iteration-based lower bound \( N_R(k) \) and a threshold-based lower bound \( N_R'(\gamma) \)). Specifically, given a worker \( w_s \) who knows task \( s \) and a social network \( G = (W, E) \) as input, Algorithm 1 iteratively generates \( N_R(k) \) RRR sets (stored in \( \mathcal{R} \)) based on \( G \) (lines 4–6), where \( N_R(k) \) is the iteration-based lower bound on the number of RRR sets. Then for each worker \( w_i \in W \), the algorithm computes the number \( N_p(w_i) \) of workers which \( w_i \) can propagate the task information to based on the current \( \mathcal{R} \) and then finds the maximal \( N_p(w_i) \), denoted as \( N_p^{opt} = \max_{w_i \in W} N_p(w_i) \) (lines 7–8). If \( N_p^{opt} \) is larger than a threshold \( \gamma \), a threshold-based lower bound \( N_R'(\gamma) \) on the number of RRR sets is computed based on the threshold \( \gamma \) and \( N_p^{opt} \) (lines 9–11); otherwise, \( \mathcal{R} \) is set to \( \emptyset \) (lines 12–13). Next, the algorithm continues to generate \( \mathcal{R} \) (lines 14–17). Getting a set \( \mathcal{R} \) of suitable RRR sets, we can compute \( P_{pro}(w_s, w_i) (w_i \in W) \) (according to Equation 3) and output the worker propagation for a worker, i.e., \( WP_{w_s} = \{P_{pro}(w_s, w_1), \ldots, P_{pro}(w_s, w_{|W|})\} \) (lines 18–21). The computation of \( k, N_R(k), N_p(w_i), N_p^{opt}, N_R'(\gamma), \gamma \) and the approximation ratio are discussed in Section III-E.

**Algorithm 1: RPO**

Input: a worker \( w_s \) who knows task \( s \); a social network \( G = (W, E) \)

Output: worker propagation \( WP_{w_s} \) of \( w_s \)

1. \( \mathcal{R} \leftarrow \emptyset; \)
2. \( k \leftarrow |W|/2; \)
3. repeat
4. Compute \( N_R(k); \)
5. // \( N_R(k) \) is the iteration-based lower bound of the number of RRR sets.
6. Generate \( N_R(k) \) RRR sets based on \( G \) (according to Section III-C1);
7. Insert these RRR sets into \( \mathcal{R}; \)
8. Compute \( N_p(w_i) (w_i \in W) \) based on \( \mathcal{R}; \)
9. // \( N_p(w_i) \) denotes the number of workers which \( w_i \) can propagate the task information to.
10. Find the maximal \( N_p(w_i) \), denoted as \( N_p^{opt}; \)
11. if \( N_p^{opt} \geq \gamma \) then
12. Compute \( N_R'(\gamma) \) based on \( N_p^{opt}; \)
13. // \( N_R'(\gamma) \) is the threshold-based lower bound of the number of RRR sets.
14. break;
15. else
16. \( \mathcal{R} \leftarrow \emptyset; \)
17. \( \mathcal{R} \leftarrow \emptyset; \)
18. Compute \( \{P_{pro}(w_s, w_1), \ldots, P_{pro}(w_s, w_{|W|})\}; \)
20. for each \( w_i \in W \) do
21. Compute \( P_{pro}(w_s, w_i) \) based on \( \mathcal{R} \) (according to Equation 3);
22. \( WP_{w_s} \leftarrow \{P_{pro}(w_s, w_1), \ldots, P_{pro}(w_s, w_{|W|})\}; \)
23. Return \( WP_{w_s} \)

The complexity of RPO is dominated by the generation of RRR sets, which takes \( O(|E| + |R| \cdot |M| \cdot |E|/|W|) \) time, where \( |E| \) is the number of edges in \( G \), \( |R| \) is the size of \( \mathcal{R} \), \( |M| \) is the number of workers who can be informed by the greedy informed worker (see Definition 8), and \( |W| \) is the number of workers in \( G \).

**D. Worker-Task Influence Calculation**

We combine worker-task affinity \( P_{aff} \), worker willingness \( P_{wil} \), and worker propagation \( WP_{w_s} \) to calculate the...
worker-task influence, if \((w_s, s)\), of worker \(w_s\) and task \(s\), as follows:
\[
if(w_s, s) = P_{aff}(w_s, s) \cdot \sum_{w_i \in W \setminus \{w_s\}} P_{wd}(w_i, s) \cdot P_{pro}(w_s, w_i),
\]
where \(w_s\) is a worker who knows the information of \(s\), \(W\) denotes the worker set in the social network, and \(P_{pro}(w_s, w_i)\) is the \(i\)-th value in \(WP_{w_s}\).

**E. Feasibility Analysis**

In order to guarantee an approximation ratio of computing worker propagation based on Random Reverse Reachable (RRR) sets, we present a feasibility analysis of computing a suitable number \(N\) of RRR sets. First, we introduce the notions of informed range and martingale and then use these to propose lemmas that facilitate the computation of the number of RRR sets. Then corresponding proofs are provided to guarantee a high approximation ratio between the estimated worker propagation and the worker propagation computed based on RRR sets. The notions of informed range and martingale [34] are defined as follows:

**Definition 6 (Informed Range)** Given a worker \(w_s\) and an RRR set \(R = \{R_1, R_2, \ldots, R_N\}\), the informed range \(\sigma(w_s)\) of \(w_s\) is the estimated fraction of workers that are informed by \(w_s\).

\[
\sigma(w_s) = \sum_{i=1}^{\lvert W \rvert} P_{pro}(w_s, w_i) = \frac{\lvert W \rvert}{N} \cdot \mathbb{E} \left[ \sum_{s=1}^{N} v_j \right],
\]
where \(v_j = 0\) if \(\{w_s\} \cap R_j = \emptyset\); otherwise, \(v_j = 1\).

**Definition 7 (Martingale)** A sequence of random variables \(x_1, x_2, \ldots, x_i\) is a martingale if and only if \(\mathbb{E}[|x_i|] < +\infty\) and \(\mathbb{E}[x_i | x_1, x_2, \ldots, x_{i-1}] = x_{i-1}\) for any \(i\).

An important property of martingales [34] is shown as follows:

**Lemma 3** Given a martingale \(x_1, x_2, \ldots\) such that \(|x_1| \leq l_1\) and \(|x_j - x_{j-1}| \leq l_1\) for \(j \in [2, l]\), and \(\text{Var}[x_1] + \sum_{j=2}^{l} \text{Var}[x_j | x_1, x_2, \ldots, x_{j-1}] \leq l_2\), for any \(\epsilon > 0\),

\[
Pr [x_i - \mathbb{E}[x_i] \geq \epsilon] \leq \exp \left( -\frac{\epsilon^2}{2l_1 \epsilon + 2l_2} \right),
\]
where \(\text{Var}[\cdot]\) is the variance of a random variable.

Given a worker \(w_s\), since each worker \(w_i\) is selected uniformly at random to generate \(R_i\) and the generation of \(R_i\) is independent of \(R_1, R_2, \ldots, R_{i-1}\), we have \(\mathbb{E}[v_{i} | v_1, v_2, \ldots, v_{i-1}] = \mathbb{E}[v_i] = \sigma(w_s)/\lvert W \rvert\). Let \(\alpha = \sigma(w_s)/\lvert W \rvert\) and \(x_i = \sum_{j=1}^{i} (v_j - \alpha)\). It is clear that \(\mathbb{E}[x_i] = 0\) and \(\mathbb{E}[x_i | x_1, x_2, \ldots, x_{i-1}] = x_{i-1}\), which indicates that \(x_1, x_2, \ldots, x_N\) is a martingale. Moreover, based on \(x_i = \sum_{j=2}^{i} (v_j - \alpha)\), it is clear that \(|x_1| \leq 1\) and \(|x_i - x_{i-1}| \leq 1\) for any \(i \in [2, N]\). Combining this with the independence of \(R_i\), we have:

\[
\text{Var}[x_1] + \sum_{i=2}^{N} \text{Var}[x_i | x_1, x_2, \ldots, x_{i-1}] = N\alpha(1 - \alpha) \leq N\alpha,
\]

Based on Lemma 3 and Equation 4, the following corollary is derived.

**Corollary 1** For any \(\epsilon > 0\),
\[
Pr \left[ \sum_{j}^{N} v_j - N \cdot \alpha \geq \epsilon \cdot N \cdot \alpha \right] \leq \exp \left( -\frac{\epsilon^2}{2 + \frac{2}{3} \epsilon} \cdot N \cdot \alpha \right)
\]

We obtain the corollary by applying Lemma 3 to \(-x_1, -x_2, \ldots, -x_N\).

**Corollary 2** For any \(\epsilon > 0\),
\[
Pr \left[ \sum_{j}^{N} v_j - N \cdot \alpha \leq -\epsilon \cdot N \cdot \alpha \right] \leq \exp \left( -\frac{\epsilon^2}{2} \cdot N \cdot \alpha \right)
\]

Given a set \(R = \{R_1, R_2, \ldots, R_N\}\) of RRR sets, let \(f_R(w_s)\) be the fraction of RRR sets in \(R\) that cover \(w_s\). It is clear that the number \(N_p(w_s)\) of workers who can be informed by \(w_s\) is \(|W| \cdot f_R(w_s)\). Based on Corollary 2, we have following lemma.

**Lemma 4** Let \(\lambda \in (0, 1), \epsilon > 0, and \)
\[
N' = \frac{2|W| \cdot \ln(1/\lambda)}{\sigma(w_s) \cdot \epsilon^2}
\]

If \(N' \leq N\), \(N_p(w_s) \geq (1 - \epsilon) \cdot \sigma(w_s)\) holds with at least probability \(1 - \lambda\).

Lemma 4 shows that when \(N'\) is sizable, the calculation of worker propagation based on Equation 5 guarantees a \((1 - \epsilon)\)-approximate solution. However, the value of \(\sigma\) is different for different workers, which makes it hard to derive a suitable value of \(N'\). Moreover, \(N'\) should be as small as possible to reduce the computation time. To address this issue, we derive a lower bound on \(N'\). Let \(w_s^*\) be a worker with maximum informed range, i.e., \(\sigma(w_s^*) \geq \sigma(w_s)\), for any \(w_s\) in \(G\). We can rewrite Lemma 4 as follows:

**Lemma 5** Let \(\lambda \in (0, 1), \epsilon > 0, and \)
\[
N'_R(\gamma) = \frac{2|W| \cdot \ln(1/\lambda)}{\sigma(w_s^*) \cdot \epsilon^2}
\]

If \(N'_R(\gamma) \leq N\), \(N_p(w_s^*) \geq (1 - \epsilon) \cdot \sigma(w_s^*)\) holds with at least probability \(1 - \lambda\).

The \(\gamma\) in Lemma 5 is a threshold, to be computed in Lemma 6. Since we aim to guarantee an approximation that is as high as possible, the setting of \(\lambda\) should be as low as possible (e.g., \(\lambda = 1/|W|^\alpha\), where \(\alpha \geq 1\)).

However, in real cases, \(\sigma(w_s^*)\) is unknown in advance. To address this problem, we derive a lower bound on \(\sigma(w_s^*)\) with the help of a so-called greedy informed worker that can be calculated in advance. A greedy informed worker is defined as follows:

**Definition 8 (Greedy Informed Worker)** A greedy informed worker \(w_s^0\) is a worker generated by a greedy approach that maximizes \(f_R: w_s^0 = \arg \max_{w_s \in W} f_R(w_s)\).

In order to use \(w_s^0\) to derive the lower bound on \(\sigma(w_s^*)\), we construct a test \(T(\cdot)\) related to \(w_s^0\) on a set of values (denoted
as \( K = \{k_1, k_2, \ldots \} \) and run the test on \( K \). If \( k_i > \sigma(w^*_i) \) then \( T(k_i) = \text{false} \) with a high probability, and \( k_i \) can be considered as a lower bound of \( \sigma(w^*_i) \). Let \( N^\text{opt}_p = |W| \cdot f_R(w^*_p) \). Based on Corollary 1, we can construct \( T() \) based on following lemma:

**Lemma 6** Given \( \epsilon^* > 0 \), let \( k_i \in K \), \( \gamma = (1 + \epsilon^*) \cdot k_i \), \( \lambda^* \in (0,1) \), and

\[
N \geq N_R(k_i) = \frac{(2 + \frac{2}{3} \epsilon^*) \cdot (\ln(|W|) + 1/\lambda^*) \cdot |W|}{\epsilon^* \cdot 2 \cdot k_i}
\]

If \( \sigma(w^*_i) < k_i \), we get \( N^\text{opt}_p < \gamma \) with at least probability \( 1 - \lambda^* \).

Based on Lemma 6, it is easy to see that if \( N^\text{opt}_p \geq \gamma \), then \( \sigma(w^*_i) \geq k_i \) holds at least with probability \( 1 - \lambda^* \), and \( \sigma(w^*_i) \) can be set to \( N^\text{opt}_p \cdot k_i / \gamma \). We can set \( K = \{\lfloor |W|/2\rfloor, \lfloor |W|/4\rfloor, |W|/8, \ldots, 2\} \) and then run the test \( T(k_i) : N^\text{opt}_p \geq \gamma \) on \( O(\log_2 |W|) \) values of \( K \) to compute the lower of \( \sigma(w^*_i) \). Moreover, since we need to guarantee the approximation with high probability (e.g., \( 1 - 1/|W|^\rho \)), the setting of \( \lambda^* \) can be \( \frac{1}{|W|^\rho \log_2 |W|} \).

Based on Lemmas 5 and 6, \( N \) should be set to \( \max\{N'_R(\gamma), N_R(k_i)\} \) to guarantee the approximation. However, it is difficult to choose suitable settings for \( \epsilon \) and \( \epsilon^* \) that minimize \( \max\{N'_R(\gamma), N_R(k_i)\} \). To address this problem, \( \max\{N'_R(\gamma), N_R(k_i)\} \) can be approximated with a simple function of \( \epsilon \) and \( \epsilon^* \), and then \( \epsilon^* = \sqrt{2} \epsilon \) is derived as the minimizer of the function.

The proofs of above lemmas are shown as follows:

**Proof of Lemma 4.** Given any worker \( w_s \), let \( \alpha = \mathbb{E}[f_R(w_s)] \), by Lemma 2, \( \alpha = \mathbb{E}[f_R(w_s)] = \sigma(w_s)/|W| \).

Based on Corollary 2, we have

\[
\Pr[N_p(w_s) \leq (1 - \epsilon) \cdot \sigma(w_s)] = \Pr[f_R(w_s) - \alpha \leq -\epsilon \cdot \alpha] = \Pr[N \cdot f_R(w_s) - N \cdot \alpha \leq -\epsilon \cdot N \cdot \alpha] = \Pr[\sum_{j=1}^{N} v_j - N \cdot \alpha \leq -\epsilon \cdot N \cdot \alpha] \leq \exp(-\epsilon^2 \cdot N \cdot \alpha/2) \leq \exp(-\epsilon^2 \cdot N^* \cdot \alpha/2) = \lambda
\]

The proof of Lemma 5 is similar to that of Lemma 4 when setting \( \alpha = \mathbb{E}[f_R(w^*_s)] \).

**Proof of Lemma 6.** Given any worker \( w_s \), let \( \alpha = \mathbb{E}[f_R(w_s)] \). The following equation can be derived.

\[
\alpha = \mathbb{E}[f_R(w_s)] \leq \mathbb{E}[f_R(w^*_s)] = \sigma(w^*_s)/|W| < k_i/|W|
\]

Let \( \beta = \frac{(1+\epsilon^*) \cdot k_i}{|W|^\rho} - 1 \). By Equation III-E, we can get \( \beta > \frac{\epsilon^* \cdot k_i}{|W|^\rho} > \epsilon^* \).

Based on Corollary 1, we have

\[
\Pr[N_p(w_s) \geq \gamma] = \Pr[f_R(w_s) - \alpha \geq \left(\frac{\gamma}{|W|^\rho \cdot \alpha} - 1\right) \cdot \alpha] = \Pr[N \cdot f_R(w_s) - N \cdot \alpha \geq \left(\frac{\gamma}{|W|^\rho \cdot \alpha} - 1\right) \cdot N \cdot \alpha] \leq \exp\left(-\frac{\beta^2 \cdot N \cdot \alpha}{2 + \frac{2}{3} \beta}\right) \leq \exp\left(-\frac{1}{2} / (\epsilon^* \cdot 2 + 3/2) \cdot \ln(|W|/\lambda^*) \cdot \epsilon^*\right) = \lambda^*/|W|
\]

According to the union bound and \( N_p(w_s) \leq N^\text{opt}_p \), we have \( N^\text{opt}_p \leq \gamma \) with at least \( 1 - \lambda^* \) probability.

**IV. INFLUENCE-AWARE TASK ASSIGNMENT**

We propose three algorithms, including basic, Entropy-based, and Distance-based Influence-aware Assignment, abbreviated IA, EIA, and DIA, respectively, that solve the ITA problem.

**A. Influence-aware Assignment**

Taking worker-task influence as the priority of task assignment, we propose a basic Influence-aware Assignment (IA) algorithm to solve the ITA problem by transforming it to a Minimum Cost Maximum Flow (MCMF) [11] problem.

To adapt MCMF to the ITA problem, we first construct a task assignment graph based on the available workers and tasks. Specifically, given a set of workers \( W = \{w_1, w_2, \ldots\} \), and a set of tasks \( S = \{s_1, s_2, \ldots\} \) at time \( t \), we construct a graph \( G = (N, E) \), where \( N \) and \( E \) denote sets of nodes and edges, respectively. Let \( |N| = |W| + |S| + 2 \) and \( |E| = |W| + |S| + m \), where \( m \) is the number of available assignments for all workers. Since tasks expire at their deadlines and workers only accept tasks in their reachable range, the available assignments for worker \( w \), denoted as \( w.A \), should satisfy the following conditions:

i) task \( s \) is located in the reachable circular range of worker \( w \), i.e., \( d(w, l, s.l) \leq w.r \).

ii) worker \( w \) has enough time to reach the location of \( s \) before it expires, i.e., \( t + d(w, l, s.l) \leq s.p + s.t \).

We use \( d(w, l, s.l) \) to denote the Euclidean distance between \( w.l \) and \( s.l \), and use \( t(w, l, s.l) \) to denote the travel time from \( w.l \) to \( s.l \). For the sake of simplicity, we assume all the workers share the same travel speed, meaning that the travel time and distance are equivalent. However, the proposed algorithms can also address the cases where workers are moving at different speeds. Let \( |w.A| \) be the number of available assignments for worker \( w \), and thus \( m \) can be derived by summing \( |w.A| \) for all workers: \( m = \sum_{w \in W} |w.A| \).

In graph \( G \), nodes \( n_i \) and \( n_{|W|+j} \) correspond to a worker \( w_i \) and a task \( s_j \), respectively. Moreover, we add two new nodes (denoted as \( n_0 \) and \( n_{|W|+|S|+1} \)) as the source (\( N_s \)) and destination (\( N_d \)), respectively. An example graph \( G \) for
four workers and four tasks at the same time is illustrated in Figure 4. The graph is generated by following steps:

i) \( N_s \) connects all worker nodes, and the capacities of the corresponding edges are set to 1, i.e., \( c = 1 \), since each worker can perform only one task at a time. The costs of these edges are set to 0.

ii) Each task node connects with \( N_d \), and the capacities of the corresponding edges are set to 1, indicating that each task can be assigned to at most 1 worker. The costs of these edges are set to 0.

iii) If the assignment \((s_j, w_i)\) is available, i.e., \((s_j, w_i) \in W.A\), we add an edge from worker node \( n_i \) to task node \( n_{W|j|} \). The capacities of the corresponding edges are set to 1, and the cost (denoted as \( w(n_i, n_{W|j|}) \)) is the ratio between 1 and the worker-task influence, \( \text{if} (w_i, s_j) \), of \( w_i \) and \( s_j \), i.e.,

\[
 w(n_i, n_{W|j|}) = \frac{1}{\text{if}(w_i, s_j) + 1}. 
\]

Then the task assignment problem is converted into an MCMF problem in the directed graph \( G \) from \( N_s \) to \( N_d \), which is to achieve the maximum flow (i.e., maximizing the task assignments) while minimize the cost (i.e., maximizing worker-task influence). The Ford-Fulkerson algorithm [35] is employed to compute the maximum flow of the graph, and then linear programming is used to minimize the cost of the flow [11].

### B. Entropy-based Influence-aware Assignment (EIA)

In SC, each task has a location. If many workers are close to a task, i.e., the relative proportion of workers close to the task is high, the task is more likely to be completed. Considering that location entropy [11], [36] is an efficient metric to measure the relative proportion of workers in the location of a specific task. Lower location entropy indicates that the location of task \( s \), and let \( \text{Num}_w \) denote the total number of visits of all workers to the location of task \( s \). Then the location entropy \( s.e \) of task \( s \), is computed as follows:

\[
 s.e = - \sum_{w \in W_s} P_s(w) \cdot \ln P_s(w),
\]

where \( W_s \) is a set of workers that have performed task \( s \) historically, and \( P_s(w) = \text{Num}_w / \text{Num}_s \).

Considering worker-task influence and location entropy, EIA adapts IA by setting the cost \( w(n_i, n_{W|j|}) \) of each edge that connects \( w_i \) and \( s_j \) to \((s.e + 1)/(\text{if}(w_i, s_j) + 1)\), where \( \text{if}(w_i, s_j) \) is the worker-task influence of worker \( w_i \) and task \( s_j \).

### C. Distance-based Influence-aware Assignment (DIA)

The IA algorithm fails to consider travel costs between the locations of workers and tasks. Workers are more likely to perform nearby tasks [1], [19], and travel cost is a critical factor when workers choose which tasks to perform. We compute the travel cost between a worker \( w_i \) and a task \( s_j \), denoted as \( d(w_i, l, s_j, l) \), using Euclidean distance. Workers who are closer to tasks will be given higher priority to perform them. To achieve this, we propose a Distance-based Influence-aware Assignment (DIA) algorithm that uses travel costs to discount worker-task influence. Specifically, DIA modifies IA by setting the cost \( w(n_i, n_{W|j|}) \) of each edge that connects \( w_i \) and \( s_j \) to \( 1/(F(w_i, l, s_j, l) \cdot \text{if}(w_i, s_j) + 1) \), where \( F(w_i, l, s_j, l) = 1 - \min(1, d(w_i, l, s_j, l)/w_i.r) \).

### V. Experimental Evaluation

#### A. Experimental Setup

Due to the lack of benchmarks for spatial crowdsourcing task assignment algorithms, two check-in datasets consisting of social networks of workers, and workers’ check-ins from Brightkite (BK) [37] and FourSquare (FS) [38] are used to simulate a spatial crowdsourcing scenario. This is common practice when evaluating SC platforms [3], [13], [39], [40]. Since BK does not contain category information of venues, we exact categories of the venues with the aid of the FourSquare API\(^1\). BK has 58,228 users, 214,078 social connections, and 4,491,143 check-ins from April 2008 to October 2010. FS has 11,326 users, 47,164 social connections, and 1,385,223 check-ins from January to December 2011.

We assume that all users are workers since users who check in at different spots are good candidates to perform nearby spatial tasks, and we assume that their locations are those of the most recent check-ins. Moreover, we set the time granularity to one day, during which the available tasks and workers are entered into our framework. We also assume that users who check in at a time instance are available workers for that time instance, and we assume that a worker is online until the worker is assigned a task. For each check-in venue, we use its location and the earliest check-in time of the day as the location and publication time of a task. Further, the categories of check-in locations are regarded as task categories. We set the number of topics used to extract worker-task affinity to 50, i.e., \( \text{Top} = 50 \). The informed probability of each social network edge, \( e \), is set to \( 1/|d_e| \) [29], [31], [41], i.e., \( P_e = 1/|d_e| \), where \( |d_e| \) denotes the number of edges with the same end point with \( e \). The parameters \( \epsilon \) and \( o \) in the Random reverse reachable-based Propagation Optimization approach are set to 0.1 and 1, respectively. Travel costs are calculated

\(^1\)https://developer.foursquare.com/docs/
using Euclidean distance, and the speeds of workers is set to 5 km/h. The default values of all parameters used in the experiments are shown in Table II. In task assignment experiments, we run the algorithms over 4 days of a month on BK and FS, and we report average results. All experiments are run on a Linux (Ubuntu 16.04) machine with Intel(R) Xeon(R) E5-2650 v4 2.20GHz processor and 256G memory.

### TABLE II

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Default value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Number of tasks</td>
<td>S</td>
</tr>
<tr>
<td>Number of workers</td>
<td>W</td>
</tr>
<tr>
<td>Valid time of tasks</td>
<td>ϕ</td>
</tr>
<tr>
<td>Workers’ reachable radius</td>
<td>r</td>
</tr>
</tbody>
</table>

**B. Experimental Results**

1) **Influence Modeling Performance:** We first evaluate the performance of worker-task affinity, worker willingness, and worker propagation and their impact on worker-task influence. We consider the IA algorithm and three variants of it to study the contribution to worker-task influence of the three aspects. The methods are as follows:

i) **IA:** Our basic Influence-aware Assignment algorithm, which considers worker-task influence and aims to maximize total task assignment and worker-task influence.

ii) **IA-WP:** A variant of IA that considers worker willingness and worker propagation.

iii) **IA-AP:** A variant of IA that considers worker-task affinity and worker propagation.

iv) **IA-AW:** A variant of IA that considers worker-task affinity and worker willingness.

Since we aim to maximize the influence of tasks, we propose Average Influence, $AI$, to evaluate the performance of each algorithm, which is calculated as follows:

$$AI = \frac{\sum_{(s,w) \in A} \text{if}(w, s)}{|A|},$$

where if$(w, s)$ is the worker-task influence of worker $w$ and task $s$, and $|A|$ is the number of assignments.

**Effect of $S$:** We first study the effect of $|S|$ on $AI$. As illustrated in Figure 5, IA achieves the largest $AI$, followed by IA-AP, for any $|S|$. The reason is that IA considers worker-task affinity, worker willingness and worker propagation, while none of the variants consider all aspects. IA-AP performs better than IA-WP and IA-AW. This may be due to the fact that the average probability of workers visiting locations of tasks is small, which means that the weight of worker willingness on computing task influence is smaller than that of worker-task affinity and worker propagation. Another observation is that $AI$ of IA is highest when $|S| = 500$. The reason is that the number of workers who can generate larger worker-task influence is small and that most of them are selected when $|S| = 500$.

**Effect of $W$:** Next we study the effect of $|W|$. From Figure 6, we can see that the $AI$ of IA-WP is lowest among these algorithms in most cases, which means the weight of worker-task affinity plays a more important role on modeling worker-task influence than worker willingness and worker propagation.

2) **Performance of Influence-aware Task Assignment:** Next, we evaluate the different task assignment algorithms.

(a) Average Influence on BK

(b) Average Influence on FS

Fig. 5. Effect of $|S|$

(a) Average Influence on BK

(b) Average Influence on FS

Fig. 6. Effect of $|W|$

(a) Average Influence on BK

(b) Average Influence on FS

Fig. 7. Effect of $ϕ$

(a) Average Influence on BK

(b) Average Influence on FS

Fig. 8. Effect of $r$
i) MTA: The Maximum Task Assignment algorithm [11] that maximizes the number of assigned tasks by computing the maximum flow of the task assignment graph.

ii) IA: Our basic Influence-aware Assignment algorithm.

iii) EIA: Our Entropy-based IA algorithm.

iv) DIA: Our Distance-based IA algorithm.

v) MI: The Maximum Influence algorithm that is divided into two steps: 1) Select multiple workers for each task based on the spatio-temporal constraints; 2) assign a task to each worker to maximize the total worker-task influence.

When a worker knows a task, the worker will propagate the information of the task to other workers in the social network. More workers knowing the information of the task leads to larger worker-task influence. Thus we introduce a metric, Average Propagation, $AP$, to evaluate the performance of the task assignment algorithms.

$$AP = \frac{\sum_{(s_i, w_i) \in A} \sum_{w_j \in W \setminus \{w_i\}} P_{pro}(w_i, w_j)}{|A|},$$

where $W$ is the set of all workers and $P_{pro}(w_i, w_j)$ is the probability that worker $w_j$ knows task $s_i$ from worker $w_i$.

Four additional metrics are also used to compare the algorithms: 1) CPU time: the CPU time costs for computing a task assignment during a time instance; 2) the total number of assigned tasks; 3) $AI$; and 4) travel cost: the average travel costs for workers performing tasks.

**Effect of $|S|$**: We first study the effect of the number of tasks. We generate five datasets containing 500 to 2,500 tasks by random selection from the original dataset. As shown in Figures 9(a) and 10(a), the CPU costs of all methods exhibit a similar increasing trend when $|S|$ grows. The reason is that a larger $|S|$ means that the task assignment graph becomes larger, which results in higher CPU time to compute task assignments. We can see that the CPU time is highest for EIA, followed by IA, DIA, MI and MTA. However, the number of tasks assigned by EIA is larger than those of the others (see Figures 9(b) and 10(b)), which demonstrates the superiority of the location entropy strategy. MI has the smallest number of assigned tasks while it has the largest Average Influence, $AI$, (see Figures 9(c) and 10(c)). This is due to the fact that MI aims to maximize the total worker-task influence and ignores to maximize the total number of assigned tasks, which increases the value of $AI$. The $AI$ of IA is larger than that of DIA, EIA, and MTA. The reason is that EIA and DIA adopt the location entropy and travel cost strategies, respectively, which reduces the effect of worker-task influence. DIA takes into account the travel cost of workers with the result that the worker willingness (see Equation 2) of DIA is larger than that of EIA. Thus, the $AI$ of DIA is larger than that of EIA for all values of $|S|$. As expected, the Average Propagation $AP$ of MI, IA, EIA, and DIA is larger than that of MTA (see Figures 9(d) and 10(d)). The reason is that worker propagation is considered in MI, IA, EIA, and DIA, while being ignored in MTA. Since workers who can generate larger worker propagation have priority to perform tasks, we see that with the increase of $|S|$, workers with smaller worker propagation have more chances to perform tasks. Moreover, DIA yields the smallest average travel costs, as shown in Figures 9(e) and 10(e). This is due to the fact that DIA takes into account the travel cost. Workers who are closer to tasks will be given higher priority to perform them. The average travel costs of all algorithms decrease with the increase of $|S|$, since the probability of assigned tasks located near workers increases.

**Effect of $|W|$**: Next, we study the effect of $|W|$ by varying it from 400 to 2,000. Figures 11(a) and 12(a) show that the CPU time increases when $|W|$ grows. The reason is that more workers tend to have more available task assignments, which leads to more edges in the task assignment graph. Since more workers can take part in task assignment, more tasks can be assigned, so the number of assigned tasks grows with the increase in the number of workers (see Figures 11(b) and 12(b)). As shown in Figures 11(c) and 12(c), the $AI$ of MI,
IA, EIA, and DIA are larger than that of MTA. Figures 11(d) and 12(d) show that the \( AP \) of all methods changes randomly. The reason may be that workers are selected at random from the original datasets, which means that workers who can generate larger \( AP \) have probability of being selected for \( |W| \). Moreover, the average travel cost of DIA is the smallest, and that of MTA is the highest (see Figures 11(e) and 12(e)). The reason is that DIA takes workers’ travel costs into account, while MTA disregards any location information.

**Effect of \( \varphi \):** As expected, the CPU costs of all methods increase when \( \varphi \) grows (see Figures 13(a) and 14(a)). This occurs because workers can reach more tasks when \( \varphi \) grows, which means that the number of available task assignments increases, i.e., more edges exist in the task assignment graph. As shown in Figures 13(b) and 14(b), the number of assigned tasks of all methods grows with growing \( \varphi \). The reason is that the task assignment graph becomes larger with larger \( \varphi \), which means that the probability of workers being assigned a task increases. Figures 13(c), 13(d), 14(c), and 14(d) show that the \( AI \) and \( AP \) of MI, IA, EIA, and DIA are larger than for MTA. The average travel costs of MTA are larger than those of other algorithms (see Figures 13(e) and 14(e)). Moreover, the average travel costs of all methods increase when \( \varphi \) grows (see Figures 13(e) and 14(e)). The reason is that with the increase of \( \varphi \), the probability of workers performing tasks with larger travel costs increases, which means that some workers are assigned tasks with larger travel costs. The average travel costs of EIA are larger than those of IA, DIA and MI since tasks with lower location entropy have higher priority to be assigned when applying EIA, which indicates workers travel longer to reach tasks.

**Effect of \( r \):** We proceed to consider the effect of \( r \) by varying it from 5 to 25 km. Figures 15(a), 15(b), 16(a) and 16(b) show that the CPU time and the number of assigned tasks of
AI it takes travel costs into account when assigning tasks. The but also achieved larger numbers of assigned tasks than the effect of worker-task influence. EIA is more time-consuming, illustrated in Figures 15(c), 15(d), 16(c), and 16(d), the tasks that far from them, which leads to more assignments. As being assigned, increasing the probability that workers accept the tasks with fewer workers nearby have higher priority of tasks that are far from them is small. When applying EIA, from workers increases, and the probability of workers accept tasks between EIA and the other approaches increases. The It can also be seen that the gap in the number of assigned each worker has higher probability to be assigned a task. available in each worker’s reachable range, which means that problems in SC is task assignment. Kazemi et al. [11] consider range of studies [5], [8], [9], [42]–[49]. One of the core AP and Influence (c) Average Influence (d) Average Propagation and Average Propagation (e) Travel Cost (f) Travel Cost workers can choose nearby spatial tasks without the need to coordinate with the SC-server. In SAT mode, the server assigns tasks to workers with the aim of maximizing the number of assigned tasks [1], [3], [50], [51] or maximizing the number of performed tasks for a worker with optimal schedule [13]. Zeng et al. [52] study a latency-oriented task completion problem that addresses the trade-off between quality and latency for task assignment. Cheng et al. [53] focus on cooperation-aware spatial crowdsourcing, where more than one worker is required to complete a task. In contrast to these studies, we study a novel task assignment problem based on worker-task influence.

Next, quality assurance is a core challenge in spatial task assignment. Workers tend to complete tasks with good quality if a quality strategy exists. Zhao et al. [51] study preference-aware task assignment, which considers temporal preferences of workers. Zhao et al. [54] propose a preference-aware task assignment for on-demand taxi dispatching that aims to maximize the expected total profits. However, these studies simply infer workers’ preferences from historical task-performing records, and they ignore workers’ social impact.

Some recent studies try to improve task assignment based on social networks. Li et al. [14] focus on group task assignment, which employs social features to learn social impact-based preferences of different worker groups. Wang et al. [55] propose two algorithms, Basic-Selector and Fast-Selector, to select a subset of workers to maximize the temporal-spatial coverage. However, these studies ignore the interactions among all workers in social networks and workers’ long-term task performing patterns.

VI. RELATED WORK

Spatial Crowdsourcing (SC) has been the subject of a range of studies [5], [8], [9], [42]–[49]. One of the core problems in SC is task assignment. Kazemi et al. [11] consider two task publication modes, namely Worker Selected Tasks (WST) and Server Assigned Tasks (SAT). In WST mode, workers can choose nearby spatial tasks without the need to coordinate with the SC-server. In SAT mode, the server assigns tasks to workers with the aim of maximizing the number of assigned tasks [1], [3], [50], [51] or maximizing the number of performed tasks for a worker with optimal schedule [13]. Zeng et al. [52] study a latency-oriented task completion problem that addresses the trade-off between quality and latency for task assignment. Cheng et al. [53] focus on cooperation-aware spatial crowdsourcing, where more than one worker is required to complete a task. In contrast to these studies, we study a novel task assignment problem based on worker-task influence.

Next, quality assurance is a core challenge in spatial task assignment. Workers tend to complete tasks with good quality if a quality strategy exists. Zhao et al. [51] study preference-aware task assignment, which considers temporal preferences of workers. Zhao et al. [54] propose a preference-aware task assignment for on-demand taxi dispatching that aims to maximize the expected total profits. However, these studies simply infer workers’ preferences from historical task-performing records, and they ignore workers’ social impact.

Some recent studies try to improve task assignment based on social networks. Li et al. [14] focus on group task assignment, which employs social features to learn social impact-based preferences of different worker groups. Wang et al. [55] propose two algorithms, Basic-Selector and Fast-Selector, to select a subset of workers to maximize the temporal-spatial coverage. However, these studies ignore the interactions among all workers in social networks and workers’ long-term task performing patterns.

VII. CONCLUSION

In this paper, we take an important step towards effective task assignment in spatial crowdsourcing that takes into account worker-task influence. Unlike most existing studies that only consider real-time worker and task locations, we further consider social networks to capture the interactions


