Influence-aware Task Assignment in Spatial Crowdsourcing

Xuanhao Chen[1], Yan Zhao[2], Kai Zheng[1*], Bin Yang[2], Christian S. Jensen[2]
[1]University of Electronic Science and Technology of China, China
[2]Department 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 patters. 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

[*]Corresponding author: Kai Zheng.

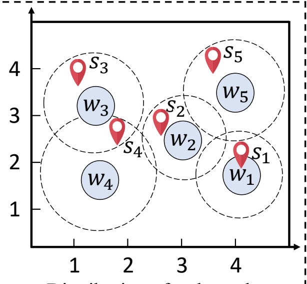
workers to perform tasks, i.e., the probabilities that workers will visit the task locations. If a worker previously performed tasks near a new task, the worker is more likely to visit the location of the new task [19]. Lastly, we are not aware of any existing task assignment techniques that combine 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_3 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

Worker	Time	Task
w_1	t_1	s_1
w_2	t_1	s_2
w_3	t_1	s_3

Historical task-performing records



Distribution of tasks and workers at time t_2

	w_1	w_2	w_3	w_4	w_5
s_4	1.42	3.56	1.67	4.25	5.23
s_5	2.28	6.17	0.32	0.18	0.85

Worker-task influence at t_2

Task assignment at time t_2

Worker	Time	Task
w_3	t_2	s_4
w_5	t_2	s_5

Greedy task assignment

Worker	Time	Task
w_4	t_2	s_4
w_5	t_2	s_5

Influence-aware task assignment

Fig. 1. Running Example

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 I 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 φ (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 $w.r$. The*

TABLE I
SUMMARY OF NOTATION

Symbol	Definition
s	Spatial task
$s.l$	Location of spatial task s
$s.p$	Publication time of spatial task s
$s.\varphi$	Valid time of spatial task s
$s.C$	Categories of spatial task s
S	A spatial task set
w	Worker
$w.l$	Location of worker w
$w.r$	Reachable distance of worker w
W	A worker set
$if(w,s)$	Worker-task influence of worker w and spatial task s
A	A task assignment
$\lvert A\rvert$	The total number of assigned tasks in task assignment A
A_{opt}	The optimal task assignment
$\mathbb{A}$	A task assignment set

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 $s.p + s.\varphi$. 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., $\forall\ A_i \in \mathbb{A}\ (|A_i| \le |A_{opt}|)$), where $\mathbb{A}$ 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 $\sum_{u_i \in U^*} h_i$ subjected to $\sum_{u_i \in U^*} l_i \le L$, 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 $\sum_{s_i \in S} h_i$ subjected to $\sum_{s_i \in S} l_i \le L$.

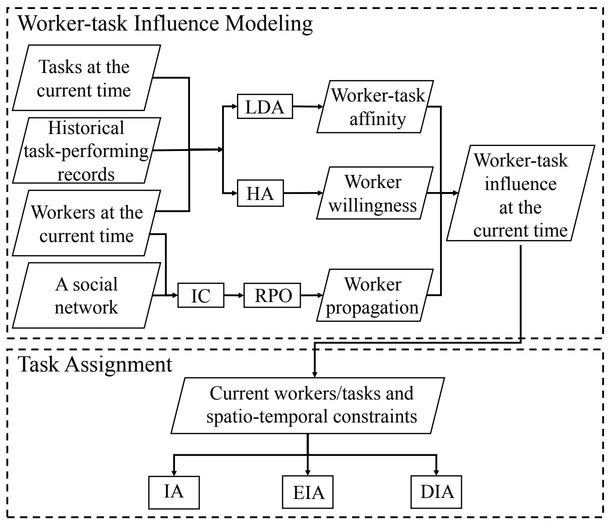
If the ITA problem instance can be solved in polynominal time, a 0-1 knapsack problem can be solved by being transformed to the corresponding ITA problem instance and then it can be solved in polynominal time. This contradicts the fact that the 0-1 knapsack problem is NP-hard [20], and so there cannot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



Worker-task Influence Modeling
Tasks at the current time
Historical task-performing records
Workers at the current time
A social network
LDA
HA
IC
RPO
Worker-task affinity
Worker willingness
Worker propagation
Worker-task influence at the current time
Task Assignment
Current workers/tasks and spatio-temporal constraints
IA
EIA
DIA

Fig. 2. DITA Framework

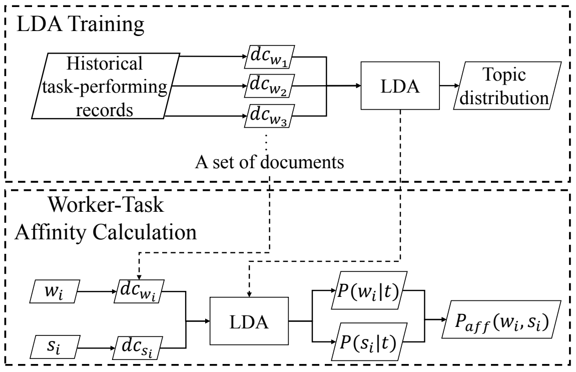


Fig. 3. Worker-task Affinity Calculation

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 v_i within topic t_j, and $P(t_j|d)$ 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:

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

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 patters.

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 $S_w = \{(s_1, t^a_{s_1}, t^l_{s_1}), (s_2, t^a_{s_2}, t^l_{s_2}), \ldots, (s_n, t^a_{s_n}, t^l_{s_n})\}$ and each triplet $(s_i, t^a_{s_i}, t^l_{s_i})$ consists of task s_i, a task arrival time $t^a_{s_i}$, and a task completion time $t^l_{s_i}$. 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 $1/\sum_j^n m_{ij}$, 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 $f_w(x; \pi; \omega) = \frac{\pi\omega^\pi}{x^{\pi+1}}$, where x is the distance between locations of tasks, π is a shape parameter that can be calculated using maximum likelihood estimation, and ω 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 π of f_w. Further, π can be calculated using the following equation:

$$\frac{d}{d\pi} \prod_i^{|S_w|-1} \frac{\pi}{x_i^{\pi+1}} = 0$$

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

Accordingly, π is given by Equation 1.

$$\pi = \frac{|S_w|-1}{\sum_i^{|S_w|-1} \ln x_i}, \text{ where } \sum_i^{|S_w|-1} \ln x_i \neq 0 \tag{1}$$

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:

$$\begin{aligned} P_{wil}(w, s) &= \sum_{s_i}^{S_w} P_w(w, s_i) \cdot \int_{d(s_i,s)}^{\infty} f_w(x)dx \\ &= \sum_{s_i}^{S_w} P_w(w, s_i) \cdot (d(s_i, s) + 1)^{-\pi} \end{aligned} \tag{2}$$

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_i.*

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_s 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_j. 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, $\mathbb{R} = \{R_1, R_2, \ldots, R_N\}$, and a worker w_s in G, let $\mathbb{R}_i$ ($\mathbb{R}_i \subseteq \mathbb{R}$) 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:

$$P_{pro}(w_s, w_i) = \frac{|W|}{N} \cdot \mathbb{E}\left[\sum_{j=1}^{|\mathbb{R}_i|} v_j\right], \tag{3}$$

where $v_j = 0$ if $\{w_s\} \cap R_j = \emptyset$; otherwise, $v_j = 1$. R_j is the j-th set of $\mathbb{R}_i$, $|W|$ is the number of workers in G, $|\mathbb{R}_i|$ is the size of $\mathbb{R}_i$, and $\mathbb{E}\left[\sum_{j}^{|\mathbb{R}_i|} v_j\right]$ 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 $\sum_{j}^{|\mathbb{R}_i|} v_j$ 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 $\mathbb{R}$) 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 $\mathbb{R}$ 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 γ, a threshold-based lower bound $N'_R(\gamma)$ on the number of RRR sets is computed based on the threshold γ and N_p^{opt} (lines 9–11); otherwise, $\mathbb{R}$ is set to $\emptyset$ (lines 12–13). Next, the algorithm continues to generate $(N'_R(\gamma) - |\mathbb{R}|)$ RRR sets when the size of the current $\mathbb{R}$ is too small, i.e., $|\mathbb{R}| < N'_R(\gamma)$ that is computed in the iteration (lines 16–17). Getting a set $\mathbb{R}$ 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., $WP_{w_s} \leftarrow (P_{pro}(w_s, w_1), \ldots, P_{pro}(w_s, w_{|W|}))$ (lines 18–21). The computation of k, $N_R(k)$, $N_p(w_i)$, N_p^{opt}, $N'_R(\gamma)$, γ 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  $\mathbb{R} \leftarrow \emptyset$;
2  $k \leftarrow |W|/2$;
3  repeat
4      Compute $N_R(k)$;
       // $N_R(k)$ is the iteration-based lower bound of the
          number of RRR sets.
5      Generate $N_R(k)$ RRR sets based on $G$ (according to
       Section III-C1);
6      Insert these RRR sets into $\mathbb{R}$;
7      Compute $N_p(w_i)$ ($w_i \in W$) based on $\mathbb{R}$;
       // $N_p(w_i)$ denotes the number of workers which $w_i$
          can propagate the task information to.
8      Find the maximal $N_p(w_i)$, denoted as $N_p^{opt}$;
9      if $N_p^{opt} \geq \gamma$ then
10         Compute $N'_R(\gamma)$ based on $N_p^{opt}$;
           // $N'_R(\gamma)$ is the threshold-based lower bound of
              the number of RRR sets.
11         break;
12     else
13         $\mathbb{R} \leftarrow \emptyset$;
14         $k \leftarrow k/2$;
15 until $k = 2$;
16 if $|\mathbb{R}| < N'_R(\gamma)$ then
17     Generate $(N'_R(\gamma) - |\mathbb{R}|)$ RRR sets and insert them into $\mathbb{R}$;
18 for each $w_i \in W \setminus \{w_s\}$ do
19     Compute $P_{pro}(w_s, w_i)$ based on $\mathbb{R}$ (according to Equation 3);
20 $WP_{w_s} \leftarrow (P_{pro}(w_s, w_1), \ldots, P_{pro}(w_s, w_{|W|}))$;
21 Return $WP_s$
```

The complexity of RPO is dominated by the generation of RRR sets, which takes $O(|E| + |\mathbb{R}| \cdot |M| \cdot |E|/|W|)$ time, where $|E|$ is the number of edges in G, $|\mathbb{R}|$ is the size of $\mathbb{R}$, $|M|$ 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, $\mathit{if}(w_s, s)$, of worker w_s and task s, as follows:

$$\mathit{if}(w_s, s) = P_{\mathit{aff}}(w_s, s) \cdot \sum_{w_i \in W \setminus \{w_s\}} P_{wil}(w_i, s) \cdot P_{pro}(w_s, w_i),$$

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 $\mathbb{R} = \{R_1, R_2, \dots, R_N\}$, the informed range $\sigma(w_s)$ of w_s is the estimated fraction of workers that are informed by w_s.*

$$\sigma(w_s) = \sum\nolimits_{i=1}^{|W|} P_{pro}(w_s, w_i) = \frac{|W|}{N} \cdot \mathbb{E}\left[\sum\nolimits_{j=1}^{N} v_j\right],$$

where $v_j = 0$ if $\{w_s\} \cap R_j = \emptyset$; otherwise, $v_j = 1$.

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

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

Lemma 3 *Given a martingale $x_1, x_2, \dots$ such that $|x_1| \le l_1$ and $|x_j - x_{j-1}| \le l_1$ for $j \in [2, i]$, and $Var[x_1] + \sum_{j=2}^{i} Var[x_j | x_1, x_2, \dots, x_{j-1}] \le l_2$, for any $\epsilon > 0$,*

$$Pr\left[x_i - \mathbb{E}[x_i] \ge \epsilon\right] \le \exp\left(-\frac{\epsilon^2}{\frac{2}{3}l_1\epsilon + 2l_2}\right),$$

where $Var[\cdot]$ 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 $R_1, R_2, \dots, R_{i-1}$, we have $\mathbb{E}[v_i | v_1, v_2, \dots, v_{i-1}] = \mathbb{E}[v_i] = \sigma(w_s)/|W|$. Let $\alpha = \sigma(w_s)/|W|$ and $x_i = \sum_{j=1}^{i}(v_j - \alpha)$. It is clear that $\mathbb{E}[x_i] = 0$ and $\mathbb{E}[x_i | x_1, x_2, \dots, x_{i-1}] = x_{i-1}$, which indicates that $x_1, x_2, \dots, x_N$ is a martingale. Moreover, based on $x_i = \sum_{j=1}^{i}(v_j - \alpha)$, it is clear that $|x_1| \le 1$ and $|x_i - x_{i-1}| \le 1$ for any $i \in [2, N]$. Combining this with the independence of R_i, we have:

$$Var[x_1] + \sum\nolimits_{i=2}^{N} Var[x_i | x_1, x_2, \dots, x_{i-1}] = N\alpha(1-\alpha) \le N\alpha \quad (4)$$

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

Corollary 1 *For any $\epsilon > 0$,*

$$Pr\left[\sum\nolimits_{j}^{N} v_j - N \cdot \alpha \ge \epsilon \cdot N \cdot \alpha\right] \le \exp\left(-\frac{\epsilon^2}{2 + \frac{2}{3}\epsilon} \cdot N \cdot \alpha\right)$$

We obtain the corollary by applying Lemma 3 to $-x_1, -x_2, \dots, -x_N$.

Corollary 2 *For any $\epsilon > 0$,*

$$Pr\left[\sum\nolimits_{j}^{N} v_j - N \cdot \alpha \le -\epsilon \cdot N \cdot \alpha\right] \le \exp\left(-\frac{\epsilon^2}{2} \cdot N \cdot \alpha\right)$$

Given a set $\mathbb{R} = \{R_1, R_2, \dots, R_N\}$ of RRR sets, let $f_R(w_s)$ be the fraction of RRR sets in $\mathbb{R}$ 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*

$$N' = \frac{2|W| \cdot \ln(1/\lambda)}{\sigma(w_s) \cdot \epsilon^2} \quad (5)$$

If $N' \le N$, $N_p(w_s) \ge (1 - \epsilon) \cdot \sigma(w_s)$ 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 σ is different for different workers, which makes it hard to derive a suitable value of N'. Moreover, N' should be as smaller as possible to reduce the computation time. To address this issue, we derive a lower bound on N'. Let w_s^τ be a worker with maximum informed range, i.e., $\sigma(w_s^\tau) \ge \sigma(w_s)$, for any w_s in G. We can rewrite Lemma 4 as follows:

Lemma 5 *Let $\lambda \in (0, 1)$, $\epsilon > 0$, and*

$$N'_R(\gamma) = \frac{2|W| \cdot \ln(1/\lambda)}{\sigma(w_s^\tau) \cdot \epsilon^2}$$

If $N'_R(\gamma) \le N$, $N_p(w_s^\tau) \ge (1 - \epsilon) \cdot \sigma(w_s^\tau)$ holds with at least probability $1 - \lambda$.

The γ 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 λ should be as low as possible (e.g., $\lambda = 1/|W|^o$, where $o \ge 1$).

However, in real cases, $\sigma(w_s^\tau)$ is unknown in advance. To address this problem, we derive a lower bound on $\sigma(w_s^\tau)$ 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^θ is a worker generated by a greedy approach that maximizes f_R: $w_s^\theta = \arg\max_{w_i \in W} f_R(w_i)$.*

In order to use w_s^θ to derive the lower bound on $\sigma(w_s^\tau)$, we construct a test $T(\cdot)$ related to w_s^θ on a set of values (denoted as $K = \{k_1, k_2, \ldots\}$) and run the test on K. If $k_i > \sigma(w_s^\tau)$ then $T(k_i) = \mathit{false}$ with a high probability, and k_i can be considered as a lower bound of $\sigma(w_i^\tau)$. Let $N_p^{opt} = |W| \cdot f_R(w_s^\theta)$. Based on Corollary 1, we can construct $T(\cdot)$ 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*

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

If $\sigma(w_s^\tau) < k_i$, we get $N_p^{opt} < \gamma$ with at least probability $1 - \lambda^*$.

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

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 ϵ and ϵ^* that minimize $\max\{N_R'(\gamma), N_R(k_i)\}$. To address that problem, $\max\{N_R'(\gamma), N_R(k_i)\}$ can be approximated with a simple function of ϵ and ϵ^*, and then $\epsilon^* = \sqrt{2}\epsilon$ is derived as the minimizer of the function.

The proofs of Lemmas 4, 5, and 6 are found in the technical report[1].

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., $d(w.l, s.l) \leq w.r$.

[1]http://zheng-kai.com/paper/icde_2022_chen_long.pdf

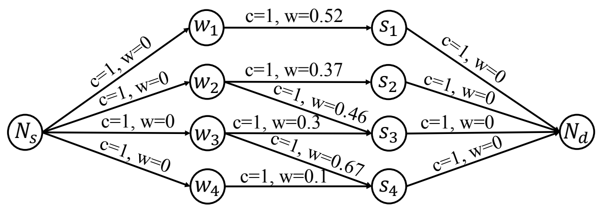


Fig. 4. Task Assignment Graph

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

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, $\mathit{if}(w_i, s_j)$, of w_i and s_j, i.e., $w(n_i, n_{|W|+j}) = \frac{1}{\mathit{if}(w_i, s_j)+1}$.

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 total number of workers in the location of a task as well as the relative proportion of their visits to that location, we use it to measure the relative proportion of workers in the location of a specific task. Lower location entropy indicates that the distribution of the visits to that task is restricted to only a few workers. To maximize the total number of task assignments, a task located in a region with smaller location entropy should be given higher priority when making assignments. Let Num_w denote the historical number of visits of worker w to the location of task s, and let Num_s 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:

$$s.e = -\sum\nolimits_{w \in W_s} P_s(w) \cdot \ln P_s(w),$$

where W_s is a set of workers that have performed task s historically, and $P_s(w) = Num_w / Num_s$.

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 $(s.e + 1)/(if(w_i, s_j) + 1)$, where $if(w_i, s_j)$ 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 $1/(F(w_i.l, s_j.l) \cdot if(w_i, s_j) + 1)$, 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 crowsourcing 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[2]. 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., $|Top| = 50$. The informed probability of each social network edge, e, is set to $1/id_e$ [29], [31], [41], i.e., $P_j = 1/id_e$, where id_e denotes the number of edges with the same end point with e. The parameters ϵ and o in the Random reverse reachable-based Propagation Optimization approach are set to 0.1 and 1, respectively. Travel costs are calculated 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 summarized 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
PARAMETER SETTINGS

Parameter	Default value
Number of tasks $\|S\|$	1500
Number of workers $\|W\|$	1200
Valid time of tasks φ	5 h
Workers' reachable radius r	25 km

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.

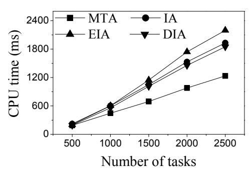
Since we aim to maximize the influence of tasks, we propose Average Influence, AI, to evaluate the performance of each algorithm, which is calculated as follows:

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

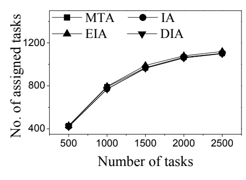
where $if(w, s)$ is the worker-task influence of worker w and task s, and $|A|$ is the number of assignments.

Due to the space limitation, we only show the effect of $|S|$. Additional results can be found in an extended technical report. As illustrated in Figure 5, IA achieves the largest AI, followed by IA-AP, for any $|S|$. The reason is that IA

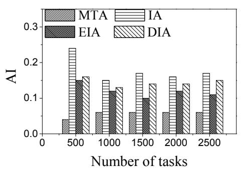
[2]https://developer.foursquare.com/docs/



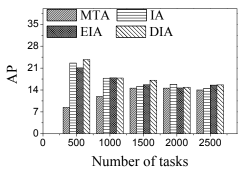
(a) CPU Time



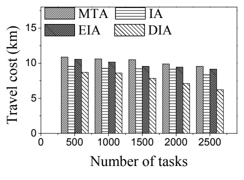
(b) Number of Assigned Tasks



(c) Average Influence

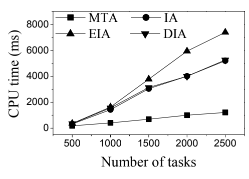


(d) Average Propagation

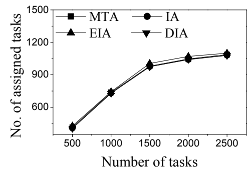


(e) Travel Cost

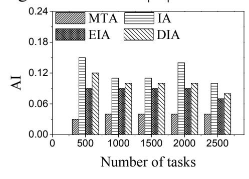
Fig. 6. Effect of $|S|$ on BK



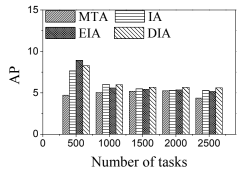
(a) CPU Time



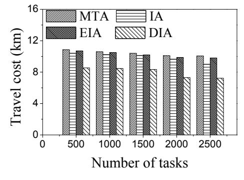
(b) Number of Assigned Tasks



(c) Average Influence



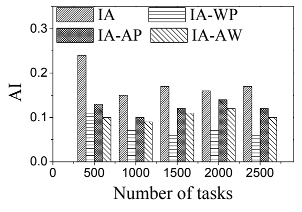
(d) Average Propagation



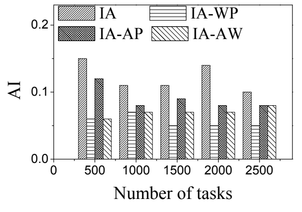
(e) Travel Cost

Fig. 7. Effect of $|S|$ on FS

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 $|S| = 500$.



(a) Average Influence on BK



(b) Average Influence on FS

Fig. 5. Effect of $|S|$

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

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.

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.

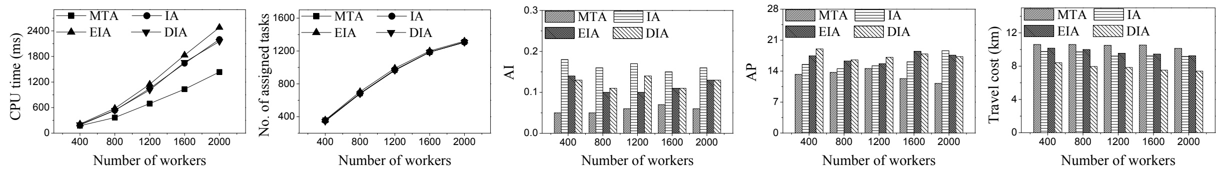
$$AP = \frac{\sum_{(s_i,w_i)\in A} \sum_{w_j \in W\setminus\{w_i\}} P_{pro}(w_i, w_j)}{|A|},$$

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.

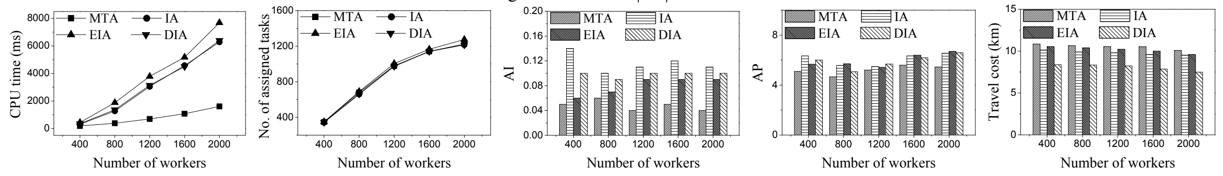
Effect of $|S|$. We first study the effect of the number of tasks. We generate five datasets containing 500 to 2,500 tasks by random selection from the original dataset. As shown in Figures 6(a) and 7(a), the CPU costs of all methods exhibit a similar increasing trend when $|S|$ grows. The reason is 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, and MTA. However, the number of tasks assigned by EIA is larger than those of the others (see Figures 6(b) and 7(b)), which demonstrates the superiority of the location entropy strategy. In Figures 6(c) and 7(c), IA has the largest Average Influence, AI, followed by DIA, EIA, and MTA. This is due to the fact 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 AP of IA, EIA, and DIA is larger than that of MTA (see Figures 6(d) and 7(d)). The reason is that worker propagation is considered in 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 6(e) and 7(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 8(a) and 9(a) show that the



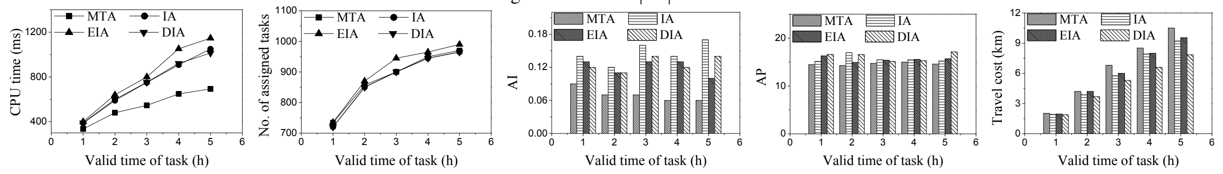
(a) CPU Time (b) Number of Assigned Tasks (c) Average Influence (d) Average Propagation (e) Travel Cost

Fig. 8. Effect of $|W|$ on BK



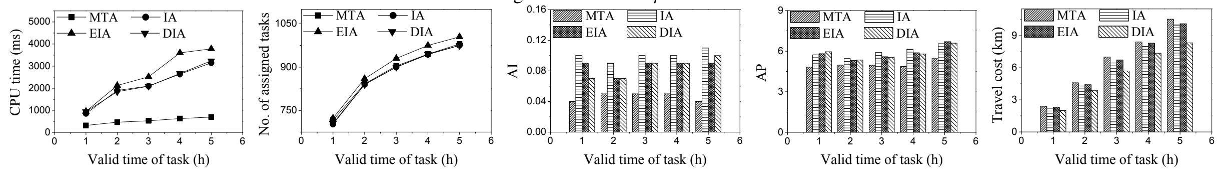
(a) CPU Time (b) Number of Assigned Tasks (c) Average Influence (d) Average Propagation (e) Travel Cost

Fig. 9. Effect of $|W|$ on FS



(a) CPU Time (b) Number of Assigned Tasks (c) Average Influence (d) Average Propagation (e) Travel Cost

Fig. 10. Effect of φ on BK



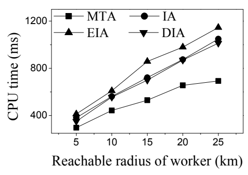
(a) CPU Time (b) Number of Assigned Tasks (c) Average Influence (d) Average Propagation (e) Travel Cost

Fig. 11. Effect of φ on FS

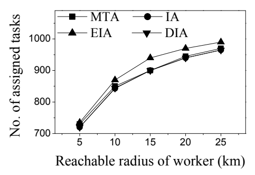
CPU time increases when $|W|$ grows. The reason is that more workers tend to have more available task assignments, which leads to more edges in the task assignment graph. Since more workers can take part in task assignment, more tasks can be assigned, so the number of assigned tasks grows with the increase in the number of workers (see Figures 8(b) and 9(b)). As shown in Figures 8(c) and 9(c), the AI of IA, EIA, and DIA are larger than that of MTA. Figures 8(d) and 9(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 8(e) and 9(e)). The reason is that DIA takes workers' travel costs into account, while MTA disregards any location information.

Effect of φ: As expected, the CPU costs of all methods increase when φ grows (see Figures 10(a) and 11(a)). This occurs because workers can reach more tasks when φ 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 10(b) and 11(b), the number of assigned tasks of all methods grows with growing φ. The reason is that the task assignment graph becomes larger with larger φ, which means that the probability of workers being assigned a task increases. Figures 10(c), 10(d), 11(c), and 11(d) show that the AI and AP of IA, EIA, and DIA are larger than for MTA. The average travel costs of MTA are larger than those of other algorithms (see Figures 10(e) and 11(e)). Moreover, the average travel costs of all methods increase when φ grows (see Figures 10(e) and 11(e)). The reason is that with the increase of φ, 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 and DIA since tasks with lower location entropy have higher priority to be assigned when applying EIA, which indicates workers travel longer to reach tasks.

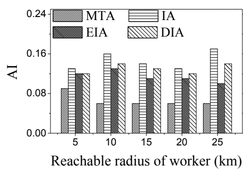
Effect of r: We proceed to consider the effect of r by varying it from 5 to 25 km. Figures 12(a), 12(b), 13(a) and 13(b) show that the CPU time and the number of assigned tasks of all methods exhibit a similar increasing trend when r grows. The reason is that with the increase of r, more tasks are available in each worker's reachable range, which means that each worker has higher probability to be assigned a task. It can also be seen that the gap in the number of assigned



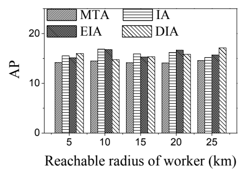
(a) CPU Time



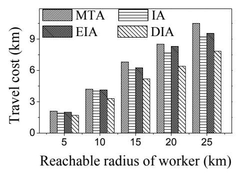
(b) Number of Assigned Tasks



(c) Average Influence

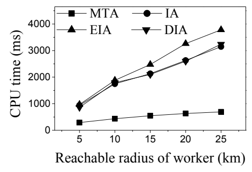


(d) Average Propagation

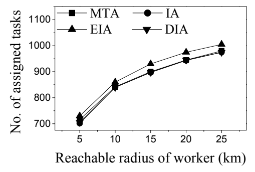


(e) Travel Cost

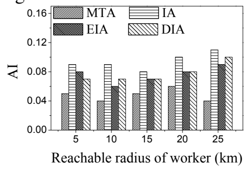
Fig. 12. Effect of r on BK



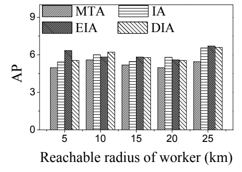
(a) CPU Time



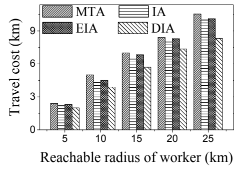
(b) Number of Assigned Tasks



(c) Average Influence



(d) Average Propagation



(e) Travel Cost

Fig. 13. Effect of r on FS

tasks between EIA and the other approaches increases. The reason is that when r grows, the number of tasks that are far from workers increases, and the probability of workers accept tasks that are far from them is small. When applying EIA, the tasks with fewer workers nearby have higher priority of being assigned, increasing the probability that workers accept tasks that far from them, which leads to more assignments. As illustrated in Figures 12(c), 12(d), 13(c), and 13(d), the AI and AP of MTA are lower than for the other approaches, which demonstrates the superiority of the influence-aware assignment strategy. Since more tasks are assigned and workers can reach tasks with larger travel costs when r grows, the average travel costs of all methods increase (cf. Figures 12(e) and 13(e)).

According to the above analysis, the time cost of MTA is the lowest, while the number of assigned tasks, Average Influence (AI), and Average Propagation (AP) of MTA are the smallest. IA has the largest AI value because other the algorithms adopt different strategies to improve the number of assigned tasks, which reduces the effect of worker-task influence. EIA is more time-consuming, but also achieved larger numbers of assigned tasks than the other algorithms. The travel cost of DIA is the smallest since it takes travel costs into account when assigning tasks.

VI. Related Work

Spatial Crowdsourcing (SC) has been the subject of a range of studies [5], [8], [9], [42]–[46]. 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], [47], [48] or maximizing the number of performed tasks for a worker with optimal schedule [13]. Zeng et al. [49] study a latency-oriented task completion problem that addresses the trade-off between quality and latency for task assignment. Cheng et al. [50] 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. [48] study preference-aware task assignment, which considers temporal preferences of workers. Zhao et al. [51] 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. [52] 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 among workers, and we employ historical task-performing records to extract long-term task performing patterns of workers. We propose three task assignment algorithms that maximize the number of assigned tasks and worker-task influence. To the best of our knowledge, this is the first study in spatial crowdsourcing that considers worker-task influence in task assignment. An extensive empirical study based on real-world data demonstrates that the proposed methods can significantly improve the effectiveness of task assignment.

Acknowledgment

This work is partially supported by NSFC (No. 61972069, 61836007 and 61832017), and Shenzhen Municipal Science and Technology R&D Funding Basic Research Program (JCYJ20210324133607021).

References

[1] Y. Zhao, K. Zheng, H. Yin, G. Liu, J. Fang, and X. Zhou, "Preference-aware task assignment in spatial crowdsourcing: from individuals to groups," *TKDE*, 2020.

[2] P. Cheng, X. Lian, L. Chen, J. Han, and J. Zhao, "Task assignment on multi-skill oriented spatial crowdsourcing," *TKDE*, vol. 28, no. 8, pp. 2201–2215, 2016.

[3] P. Cheng, X. Lian, L. Chen, and C. Shahabi, "Prediction-based task assignment in spatial crowdsourcing," in *ICDE*, 2017, pp. 997–1008.

[4] T. Song, Y. Tong, L. Wang, J. She, B. Yao, L. Chen, and K. Xu, "Trichromatic online matching in real-time spatial crowdsourcing," in *ICDE*, 2017, pp. 1009–1020.

[5] Y. Tong, L. Wang, Z. Zhou, L. Chen, B. Du, and J. Ye, "Dynamic pricing in spatial crowdsourcing: A matching-based approach," in *SIGMOD*, 2018, pp. 773–788.

[6] Y. Tong, L. Wang, Z. Zimu, B. Ding, L. Chen, J. Ye, and K. Xu, "Flexible online task assignment in real-time spatial data," *PVLDB*, vol. 10, no. 11, pp. 1334–1345, 2017.

[7] Y. Tong, Y. Zeng, Z. Zhou, L. Chen, J. Ye, and K. Xu, "A unified approach to route planning for shared mobility," *PVLDB*, vol. 11, no. 11, p. 1633, 2018.

[8] J. Xia, Y. Zhao, G. Liu, J. Xu, M. Zhang, and K. Zheng, "Profit-driven task assignment in spatial crowdsourcing," in *IJCAI*, 2019, pp. 1914–1920.

[9] Y. Zhao, Y. Li, Y. Wang, H. Su, and K. Zheng, "Destination-aware task assignment in spatial crowdsourcing," in *CIKM*, 2017, pp. 297–306.

[10] G. Ye, Y. Zhao, X. Chen, and K. Zheng, "Task allocation with geographic partition in spatial crowdsourcing," in *CIKM*, 2021, pp. 2404–2413.

[11] L. Kazemi and C. Shahabi, "Geocrowd: enabling query answering with spatial crowdsourcing," in *SIGSPATIAL*, 2012, pp. 189–198.

[12] P. Cheng, X. Lian, Z. Chen, R. Fu, L. Chen, J. Han, and J. Zhao, "Reliable diversity-based spatial crowdsourcing by moving workers," *PVLDB*, vol. 8, no. 10, pp. 1022–1033, 2015.

[13] D. Deng, C. Shahabi, and U. Demiryurek, "Maximizing the number of worker's self-selected tasks in spatial crowdsourcing," in *SIGSPATIAL*, 2013, pp. 324–333.

[14] X. Li, Y. Zhao, X. Zhou, and K. Zheng, "Consensus-based group task assignment with social impact in spatial crowdsourcing," *Data Science and Engineering*, vol. 5, no. 4, pp. 375–390, 2020.

[15] X. Li, Y. Zhao, J. Guo, and K. Zheng, "Group task assignment with social impact-based preference in spatial crowdsourcing," in *DASFAA*, 2020, pp. 677–693.

[16] J. Tang, X. Tang, X. Xiao, and J. Yuan, "Online processing algorithms for influence maximization," in *SIGMOD*, 2018, pp. 991–1005.

[17] X. Chen, Y. Zhao, G. Liu, R. Sun, X. Zhou, and K. Zheng, "Efficient similarity-aware influence maximization in geo-social network," *TKDE*, 2020.

[18] M.-C. Yuen, I. King, and K.-S. Leung, "Task recommendation in crowdsourcing systems," in *Proceedings of the first international workshop on crowdsourcing and data mining*, 2012, pp. 22–26.

[19] H. To, G. Ghinita, and C. Shahabi, "A framework for protecting worker location privacy in spatial crowdsourcing," *PVLDB*, vol. 7, no. 10, pp. 919–930, 2014.

[20] V. V. Vazirani, *Approximation algorithms*, 2013.

[21] D. M. Blei, A. Y. Ng, and M. I. Jordan, "Latent dirichlet allocation," *the Journal of machine Learning research*, vol. 3, pp. 993–1022, 2003.

[22] W. Gong, B. Zhang, and C. Li, "Location-based online task assignment and path planning for mobile crowdsensing," *IEEE Transactions on Vehicular Technology*, vol. 68, no. 2, pp. 1772–1783, 2018.

[23] Q. Tao, Y. Tong, Z. Zhou, Y. Shi, L. Chen, and K. Xu, "Differentially private online task assignment in spatial crowdsourcing: A tree-based approach," in *ICDE*, 2020, pp. 517–528.

[24] J. Han, J. Pei, and M. Kamber, *Data mining: concepts and techniques*, 2011.

[25] W.-Y. Zhu, W.-C. Peng, and L.-J. Chen, "Exploiting mobility for location promotion in location-based social networks," in *DSAA*, 2014, pp. 76–82.

[26] W.-Y. Zhu, W.-C. Peng, L.-J. Chen, K. Zheng, and X. Zhou, "Modeling user mobility for location promotion in location-based social networks," in *SIGKDD*, 2015, pp. 1573–1582.

[27] R. Singhai, S. D. Joshi, and R. K. Bhatt, "A novel discrete distribution and process to model self-similar traffic," in *ICT*, 2007, pp. 167–172.

[28] D. Kempe, J. Kleinberg, and É. Tardos, "Maximizing the spread of influence through a social network," in *SIGKDD*, 2003, pp. 137–146.

[29] W. Chen, C. Wang, and Y. Wang, "Scalable influence maximization for prevalent viral marketing in large-scale social networks," in *SIGKDD*, 2010, pp. 1029–1038.

[30] C. Borgs, M. Brautbar, J. Chayes, and B. Lucier, "Maximizing social influence in nearly optimal time," in *SODA*, 2014, pp. 946–957.

[31] Y. Tang, Y. Shi, and X. Xiao, "Influence maximization in near-linear time: A martingale approach," in *SIGMOD*, 2015, pp. 1539–1554.

[32] A.-A. Stoica and A. Chaintreau, "Fairness in social influence maximization," in *WWW*, 2019, pp. 569–574.

[33] X. Chen, L. Deng, Y. Zhao, X. Zhou, and K. Zheng, "Community-based influence maximization in location-based social network," *WWWJ*, vol. 24, no. 6, pp. 1903–1928, 2021.

[34] F. Chung and L. Lu, "Concentration inequalities and martingale inequalities: a survey," *Internet Mathematics*, vol. 3, no. 1, pp. 79–127, 2006.

[35] L. R. Ford and D. R. Fulkerson, "Maximal flow through a network," *Canadian journal of Mathematics*, vol. 8, pp. 399–404, 2009.

[36] J. Cranshaw, E. Toch, J. Hong, A. Kittur, and N. Sadeh, "Bridging the gap between physical location and online social networks," in *UbiComp*, 2010, pp. 119–128.

[37] E. Cho, S. A. Myers, and J. Leskovec, "Friendship and mobility: user movement in location-based social networks," in *SIGKDD*, 2011, pp. 1082–1090.

[38] A. Likhyani, S. Bedathur, and P. Deepak, "Locate: Influence quantification for location promotion in location-based social networks," in *IJCAI*, 2017, pp. 2259–2265.

[39] Y. Zhao, K. Zheng, Y. Cui, H. Su, F. Zhu, and X. Zhou, "Predictive task assignment in spatial crowdsourcing: a data-driven approach," in *ICDE*, 2020, pp. 13–24.

[40] Z. Wang, Y. Zhao, X. Chen, and K. Zheng, "Task assignment with worker churn prediction in spatial crowdsourcing," in *CIKM*, 2021, pp. 2070–2079.

[41] K. Jung, W. Heo, and W. Chen, "Irie: Scalable and robust influence maximization in social networks," in *ICDM*, 2012, pp. 918–923.

[42] D. Meng, Y. Jia, J. Du, and F. Yu, "Tracking algorithms for multiagent systems," *IEEE Transactions on neural networks and learning systems*, vol. 24, no. 10, pp. 1660–1676, 2013.

[43] Y. Zhao, K. Zheng, J. Guo, B. Yang, T. B. Pedersen, and C. S. Jensen, "Fairness-aware task assignment in spatial crowdsourcing: Game-theoretic approaches," in *ICDE*, 2021, pp. 265–276.

[44] Y. Zhao, J. Guo, X. Chen, J. Hao, X. Zhou, and K. Zheng, "Coalition-based task assignment in spatial crowdsourcing," in *ICDE*, 2021, pp. 241–252.

[45] Y. Zhao, K. Zheng, Y. Li, H. Su, J. Liu, and X. Zhou, "Destination-aware task assignment in spatial crowdsourcing: A worker decomposition approach," *TKDE*, vol. 32, no. 12, pp. 2336–2350, 2019.

[46] Y. Cui, L. Deng, Y. Zhao, B. Yao, V. W. Zheng, and K. Zheng, "Hidden poi ranking with spatial crowdsourcing," in *SIGKDD*, 2019, pp. 814–824.

[47] Y. Tong, L. Chen, Z. Zhou, H. V. Jagadish, L. Shou, and W. Lv, "Slade: A smart large-scale task decomposer in crowdsourcing," *TKDE*, vol. 30, no. 8, pp. 1588–1601, 2018.

[48] Y. Zhao, J. Xia, G. Liu, H. Su, D. Lian, S. Shang, and K. Zheng, "Preference-aware task assignment in spatial crowdsourcing," in *AAAI*, vol. 33, no. 01, 2019, pp. 2629–2636.

[49] Y. Zeng, Y. Tong, L. Chen, and Z. Zhou, "Latency-oriented task completion via spatial crowdsourcing," in *ICDE*, 2018, pp. 317–328.

[50] P. Cheng, L. Chen, and J. Ye, "Cooperation-aware task assignment in spatial crowdsourcing," in *ICDE*, 2019, pp. 1442–1453.

[51] B. Zhao, P. Xu, Y. Shi, Y. Tong, Z. Zhou, and Y. Zeng, "Preference-aware task assignment in on-demand taxi dispatching: An online stable matching approach," in *AAAI*, vol. 33, no. 01, 2019, pp. 2245–2252.

[52] J. Wang, F. Wang, Y. Wang, D. Zhang, L. Wang, and Z. Qiu, "Social-network-assisted worker recruitment in mobile crowd sensing," *TMC*, vol. 18, no. 7, pp. 1661–1673, 2018.