Hindawi
Wireless Communications and Mobile Computing
Volume 2020, Article ID 8822251, 15 pages
https://doi.org/10.1155/2020/8822251

Research Article

Social-Aware Task Allocation in Mobile Crowd Sensing

Weiping Zhu,[1] **Wenzhong Guo**,[1,2,3] **and Zhiyong Yu**[1,2,3]

[1]*College of Mathematics and Computer Science, Fuzhou University, Fuzhou 350116, China*
[2]*Fujian Provincial Key Laboratory of Networking Computing and Intelligent Information Processing, Fuzhou University, Fuzhou 350116, China*
[3]*Key Laboratory of Spatial Data Mining and Information Sharing, Ministry of Education, Fuzhou 350003, China*

Correspondence should be addressed to Wenzhong Guo; guowenzhong@fzu.edu.cn

Received 26 March 2020; Revised 4 September 2020; Accepted 13 September 2020; Published 14 October 2020

Academic Editor: Kim-Kwang Raymond Choo

Task allocation is a significant issue in crowd sensing, which trades off the data quality and sensing cost. Existing task allocation works are based on the assumption that there is plenty of users available in the candidate pool. However, for some specific applications, there may be only a few candidate users, resulting in the poor completion of tasks. To tackle this problem, in this paper, we investigate the task allocation problem with the assistance of social networks. We select a subset of users; if a user can not complete the task, he can propagate the task information to his friends. The object of this problem is to maximize the expected number of completed tasks. We prove that the task allocation problem is an NP-hard and submodular problem and then propose a native greedy selection (NGS) algorithm, which selects the user with maximum margin gain in each round. To improve the efficiency of the NGS algorithm, we further propose a fast greedy selection algorithm (FGS), which selects the user who can actually complete the maximum number of tasks. Experimental results show that although FGS gets slightly worse results in terms of the expected number of completed tasks, it can greatly reduce the running time of seed selection.

1. Introduction

In recent years, with the popularity of mobile devices and the development of various communication technologies, mobile crowd sensing (MCS) [1–3] has emerged as a promising sensing paradigm, in which mobile users leverage their carry-on devices to collect and upload the sensing data. By collecting and analyzing sensing data from a group of users, the platform can provide ubiquitous services. Due to its advantages such as low deployment cost and wide spatio-temporal coverage, numerous practical MCS applications have been rolled out in many areas, such as environmental monitoring [4], smart transportation [5], and emergency alarming [6].

A mobile crowd sensing process can be divided into four stages: task creation, task allocation, task execution, and data integration [7]. Particularly, task allocation is an important stage, which trades off the sensing quality and sensing cost [3, 8]. Different from traditional wireless sensor networks [9], task allocation in MCS should select eligible users to complete tasks. So far, there have been numerous related works on task allocation [10–12]. Among these works, the platform directly selects users from a large pool of candidate users to maximize the sensing quality under limited cost or minimize the sensing cost while guaranteeing certain data quality.

With the development of MCS, there may be plenty of applications using this scheme to collect data. For some specific applications, there may not be enough available candidate users. The reasons can be listed as follows: on the one hand, in the early stage of a new application, there are a few registered users available to allocate tasks. On the other hand, due to the privacy concern, users do not open the location service, so the platform cannot get their locations. This situation makes the platform hard to recruit enough users, and the existing candidate users can complete only a small number of tasks, resulting in a low completion ratio of tasks.

Fortunately, with the development of social networks, "word-of-mouth" effect has played an important role in propagating and sharing information, giving an alternative

way to allocate mobile sensing tasks. The social relationship is built if there is a connection between two users. If users know the service is requested by his friends, he may be glad to participate in the sensing tasks. There are several advantages of allocating tasks with the assistance of social networks. From the perspective of the platform, it only needs to recruit a few seeds to complete or propagate the task information, which can reduce the sensing cost. Furthermore, users may be glad to help their friends to complete sensing tasks due to their friendship. So the sensing enthusiasm will be increased compared to the traditional "push" task allocation model. From the perspective of users who have not registered in the platform, they do not need to share their real-time locations, thus reducing the risk of privacy leakage. Inspired by this, this paper attempts to allocate the sensing tasks with the assistance of social networks. Concretely, we select a small number of users to allocate the tasks. If the users can reach the target location at the required time, then, the task can be completed by the selected users. Otherwise, the selected users propagate the task information to his friends, who may be in the target location at the required time, different from [13, 14], which select seeds to propagate the task information in the social network to maximize the spatiotemporal coverage of an area. This paper considers that seeds propagate task information to their friends only when the task is not completed.

Considering some location-based tasks in the platform, for example, monitoring the traffic dynamic information of a specific road, reporting the air pollution of the target location, this type of task requires users to arrive in the target location to collect data. However, the platform only knows a little part of users' location information. To allocate these tasks, an effective way for the platform is selecting some users as seeds to complete or propagate these tasks. These seeds not only complete these tasks that they can complete but also propagate the task information that they cannot complete to their friends and expect their friends can help to complete the sensing tasks. For the latter case, the completion probability of these tasks depends on the probability of friends in the target location and the probability that the seeds propagate the task information to him.

The task allocation problem mentioned above is similar to the location aware influence maximization problem in social networks to some extent, which is to select k users in the target location to maximize the expected number of influenced users. However, there are mainly two differences between these two problems: (1) location constraints of seeds and their influenced users. In the influence maximization problem, both seeds and influenced users should be in the target location, while in the task allocation problem, seeds are not necessarily in the target location but influenced users should be in the target location (in case only propagate once). (2) performance quantification of seeds. In the influence maximization problem, a seed is more valuable if he can influence more additional users, while in the task allocation problem, a seed is more valuable if he can complete more additional tasks by influencing users or by himself.

According to the analysis above, the key challenge is how to identify the seeds that can maximize the expected number of completed tasks. Compared to the traditional task allocation mode in MCS, we consider not only the actual number of completed tasks but also the propagation ability to tasks that they cannot complete. With the object and challenges mentioned above, the main contributions of this work can be concluded as follows:

(1) We formulate a social-aware task allocation problem (SATA), in which the platform allocates the tasks considering the social influence of users. If the selected user can not complete the task, he propagates the task information to his friends and expects to get sensing data from friends. The SATA problem is aimed at selecting k users as seeds to maximize the expected number of completed tasks

(2) We prove that the problem is an NP-hard and submodular problem. Then, we propose a native greedy-based selection algorithm (NGS) to solve the problem. The NGS algorithm selects the user with the largest margin gain in each round

(3) Considering the low efficiency of NGS, we further propose the fast greedy selection algorithm (FGS). FGS divides the seed selection process into two stages. In the first stage, FGS gives the priority to the user who can complete the maximum number of remaining tasks. For the second stage, in which no one can complete any task, FGS selects the user with maximum propagation performance from candidate users

(4) We conduct extensive experiments using two real-world datasets, which contain users' social relationship and mobility traces. The experimental results show that the performance of the FGS algorithm is slightly worse than that of the NGS algorithm. However, the FGS algorithm runs more efficiently than the NGS algorithm

The remainder of this paper is organized as follows. In Section 2, the related works about task allocation are presented. In Section 3, the problem of SATA is described in detail. Then, we analyze the problem and introduce our proposed algorithms in Section 4. In Section 5, we evaluate the performance of proposed algorithms and baseline algorithms. Finally, the conclusions are presented in Section 6.

2. Related Work

Task allocation has become an important problem in MCS and drawn considerable attention from researchers. There have been numerous studies on task allocation, which can be divided into the following two types.

2.1. Single Task Allocation. In the early state of MCS, several systems have been designed for single task allocation. For example, in [15, 16], Reddy et al. considered the location, time constraints, and habits of users and proposed a coverage-based framework to select proper users to maximize spatial coverage. Zhang et al. proposed a task allocation framework, which is aimed at maximizing the coverage

quality under budget constraint [17]. Wang et al. [18] predicted the mobility of vehicles and selected participants to minimize the overall recruitment cost. Another work [19] defined a new coverage metric, namely, "t-sweep k-coverage" and proposed two methods to select the smallest set of candidate users to satisfy the predefined requirements.

2.2. Multiple Task Allocation. With the development of crowdsensing, there may be many tasks to be allocated in the platform. Researchers made efforts to study the allocation for multitasks. Though the optimization and constraints may be similar to single-task allocation, several factors should be considered to deal with multiple task allocation.

Firstly, from the perspective of spatiotemporal aspects, different tasks vary in spatial and temporal granularity and other properties. Based on this consideration, Wang et al. formulated a multitask allocation problem, which considered heterogeneous spatial and temporal granularity. Users can complete multiple tasks without changing their locations. The authors proposed a two-stage allocation algorithm to solve this problem [11]. Li et al. [20] proposed an online task allocation algorithm for dynamic heterogeneous tasks to minimize the number of users while guaranteeing a certain level of coverage.

Secondly, from the perspective of task properties, our previous work [21] considered the type of sensing tasks and heterogeneity of users' sensing devices and proposed a Particle Swarm Optimization (PSO) algorithm to select eligible users to maximize the number of completed tasks. Inversely, a recent work [12] considered that different tasks can share the same data property under some special circumstances and proposed a triple-layer task allocation framework, which considers not only the temporal and spatial correction but also the data property of sensing data. Jiang et al. [22] considered the skills that tasks required and allocated tasks to a natural user group, in which members can cooperate to complete the complex task. If the users in an assigned group do not possess all required skills, they can cooperate with other contextual groups. The authors selected a principal group with the maximum self-crowdsourcing value and allocated the tasks to the group. Then, the authors proposed two greedy-based approaches to select an assistant group according to the circumstance of principal group. Wang et al. [23] combined the content information with context information of tasks and unified all the factors together to measure the preference score of a user to a task. Then, the user with the maximum matching probability is selected to complete tasks.

Thirdly, from the perspective of global optimization, the optimal total quality of tasks does not always guarantee the optimal quality for every task. To solve this problem, Wang et al. [24] considered the task-specific sensing quality thresholds and proposed a descent greedy approach to select a set of users to complete tasks. Considering the total time consumption for multiple tasks depends on the time consumption of the last completed task. The authors in the study [25] considered the time-sensitive tasks and proposed a cooperation scheme to minimize the maximum completion time of tasks. Ni et al. [26] investigated the dependency-aware task allocation problem with the consideration of skills, moving distances, and deadlines. To allocate these tasks efficiently, the authors grouped these tasks into associative task sets and proposed two algorithms to solve this problem. Song et al. [27] considered that the existing task matching strategies may cause a misaligned task coverage problem, which indicates that some popular tasks can find enough users but the less popular task cannot assigned successfully. To solve this problem, the authors proposed the *cTaskMat* framework. It learns users' task preferences and their attitudes on task attributes. Then, it migrates certain qualified users to less popular tasks for increasing task coverage and reducing the sensing cost.

These works assumed that there are enough candidate users and the platform can push the task information to the eligible users. However, for those new applications, there have not been many users registered in the platform, resulting in the poor completion ratio of sensing tasks. Thus, a new task allocation scheme is urgently needed to solve this dilemma.

To solve this problem, Wang et al. [13] proposed a task allocation framework, which propagates the task information leveraging the social network to maximize the wide-area coverage. Lu and Zhu [14] proposed a hybrid user recruitment framework, which divides the recruitment process into two phases. In the offline phase, it recruits users to propagate task information in communication and social networks. In the online phase, it incentivizes users to move to uncovered subareas and fulfil tasks based on subarea clustering. However, in these two studies, the object is to maximize the coverage of the whole area, which can be divided into several fine-grained subareas. The selected seeds and their influenced users will cover all subareas that they pass even those subareas already covered, which may cause data redundancy. In this paper, seeds propagate the task information to their friends only when the task is not completed. Thus, it is promising to reduce the data redundancy. Peng et al. [28] selected eligible users for crowdsourcing tasks based on social relation cognition. Different from this work, we considered the propagation of location-based crowd sensing tasks, which is common in the real life and agent to be explored.

3. System Model and Problem Formulation

In this section, we present the overview of the crowdsensing system with limited users and then formulate the problem of SATA, in which platform selects mobile users to complete or propagate the information of tasks. Additionally, for the ease of presentation, we list the notations frequently used in this paper in Table 1.

3.1. System Model. In this paper, we advocate a mobile crowd system as shown in Figure 1, which consists of three characters: service requesters, platform, and mobile users. Their roles are defined as follows:

(1) Service requesters: when the service requesters want to know some useful information (e.g., the driver wants to know the traffic condition of the target road where he is going to), the requesters specify the requests as a set of keywords, including the location and time requests of the task and the quality of sensing data. Then, service requesters upload the sensing

Table 1: The summary of notation.

Parameters	Meanings
$G = (U, E)$	Social network
u, v, w	Users in U
(u, v)	Edge from user u to user v
$N(u)$	Neighbor set of user u
S	The selected seed set
k	Number of selected seeds
L	The location set
loc_j	The location of task t_j
tim_j	The required time of task t_j
R	The range of tasks to recruit user
$\text{sim}(u, v)$	The *Jaccard* similarity of user u and user v
$\text{pos}(u, j)$	The geographical performance of user u in the location loc_j at time tim_j
$\text{pos}_j(u, v)$	The possibility of user v complete task t_j under influence of user u
$\Phi(S)$	Expected number of completed tasks of user set S

requests to the platform and expect to get the sensing data from the platform.

(2) Platform: the major function of the platform is allocating the sensing task to the eligible mobile users according to their spatiotemporal characteristics and other capacities that tasks required.

Considering the task sets to be allocated in the platform, denoted as $T = \{t_1, t_2, \cdots, t_n\}$, where t_j is the j^{th} task. Each task can be characterized by a tuple with three elements: $\{\text{loc}_j, \text{tim}_j, R\}, 1 \le j \le n,$ where loc_j is the location of task t_j and tim_j is the required time of task t_j. To improve the completion rate of tasks, if the distance between one user and the task t_j is less than R at time tim_j, the platform regards that this user can complete the task t_j.

(3) Mobile users: mobile users are the main force to complete tasks. After receiving the task information from the platform, mobile users collect sensing data and upload it to the platform. In this paper, we define mobile users from the social aspect. Let $G = (U, E)$ denote the social network, in which U represents the set of users $U = \{u_1, u_2, \cdots, u_m\}$ and E represents the relationship among users. For any two users u and v in U, we regard they are connected if $(u, v) \in E$ and can share the task information with each other with a certain probability.

According to the main roles in the system model, the social-aware task allocation problem (SATA) in this paper can be described as follows: the service requesters send the service requests to the platform and expect to get the service from the platform. To provide timely service, the platform launches the task information and selects eligible users to collect sensing data. However, due to the "cold effect," there are only a few users available to perform the tasks. Under this circumstance, the platform selects a part of users as seeds to complete tasks or propagate the information with the assistance of social networks under the specific rule to ask their friends for help. The detail of this rule is presented in the next section. The object of the platform is to maximize the expected number of completed tasks. It is worthy to note that if there are multiple seeds propagated the information to the common friends in the social network, we choose the influence path with maximum probability.

3.2. Propagation Model. As mentioned above, the SATA problem is similar to the location aware influence maximization problem in the social network, which is aimed at selecting a subset from candidate users with a cardinality of k to maximize the influence spread in the target location under a certain propagation model. The indicator of influence spread for selected seeds is usually defined as the number of influenced users. To simulate the propagation process among users, we employ the independent cascade (IC) model, which is widely used in information propagation. In the IC model, every edge in the social network is associated with a weight to measure the propagation possibility from one node to his neighbor. The process of a traditional IC model can be described as follows:

(1) In the initial stage, all nodes in the social network are in the inactive state

(2) If a node u is selected as seed, then, it will get a chance to influence its neighbor nodes, which has not been selected as seeds

(3) The process continues until selected seed set reaches to the required cardinality

According to the process of the IC model, the task information propagation in our problem can be described as follows: if the seed cannot complete the task, he should propagate the task information to his friends with a certain probability and expect friends can complete the sensing task. To measure the probability of a nonseed can complete tasks under the influence of seeds, we define two major factors that affect task completion.

(1) Geographical performance: when a user can not complete this task himself, he should propagate the task to his friends who might be in the target location. We adopt the empirical statistical model to measure the geographical performance of users. The possibility that a user visits the location loc_j at the time tim_j is modeled by the frequency of visiting in the location loc_j at the time tim_j. Mathematically, it can be computed as follows:

$$\text{pos}(u, j) = \frac{n_u(\text{loc}_j, \text{tim}_j)}{\sum_{\text{loc}_j \subseteq L} n_u(\text{loc}_j, \text{tim}_j)}, \quad (1)$$

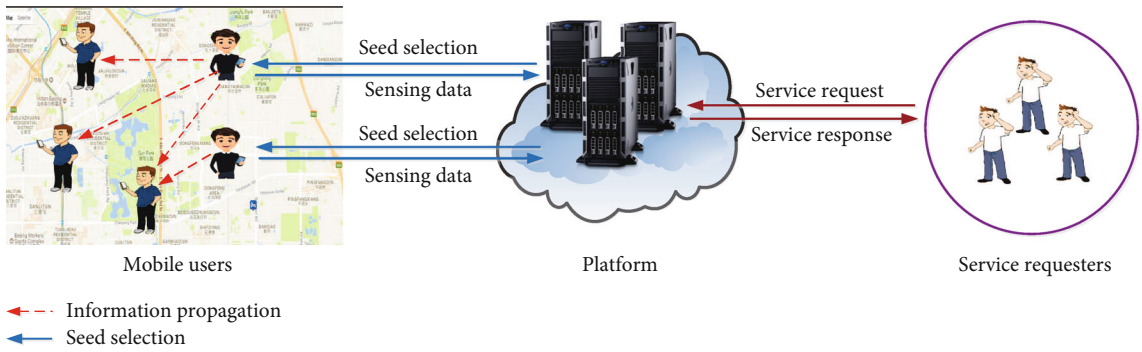


Figure 1: The framework of the MCS system.

where L is all the location set and $n_u(\text{loc}_j, \text{tim}_j)$ denotes the check-in times of user u in the location loc_j at the time tim_j in the historical check-in records.

(2) Social similarity: intuitively, users prefer to share information with close friends. So the social relationship between two users should be considered in the process of propagation. In this paper, we adopt *Jaccard* coefficient to measure the social similarity of two users

$$\text{sim}(u, v) = \frac{|N(u) \cap N(v)|}{|N(u) \cup N(v)|}, \tag{2}$$

where $N(u)$ is the neighbor set of user u. Larger similarity indicates that two users have a closer social relationship and can share task information with higher possibility.

By considering the factors of geographical performance and social similarity, the probability that nonseed v can complete the task t_j after influenced by a seed u can be calculated as follows:

$$\text{pos}_j(u, v) = \text{pos}(v, j) * \text{sim}(u, v). \tag{3}$$

Since selected seeds may have some common friends. During the information propagation, these common friends are influenced by multiple seeds simultaneously. It is worthy to note that the possibility that these common friends complete tasks is not simply computed by the utility summation of selected seeds. Alternatively, we regard a nonseed to complete the task with the probability that influenced by the seed who has maximum social similarity with him.

3.3. Problem Formulation. According to the definitions above, we can formulate the problem: given a social network and a set of tasks, the platform tries to allocate these tasks with the assistance of social networks. Specially, the platform selects k users as seeds. These seeds may only satisfy the spatiotemporal requirement of part of tasks. For those tasks that they cannot complete, they propagate the task information to their friends according to the rule defined in the last section.

Thus, the possibility of selected users to complete the task t_j can be computed from two perspectives: if u in the location of t_j at the required time, he can complete the task. Otherwise, u propagates the task information to his friends. Assume that friends are willing to complete the task after receiving the propagation from his friends if they are in the target location. So the completion possibility of a task can be computed by Equation (3).

For the selected seed set, their utility to a task t_j can be computed as follows: if it exists a seed that can complete the task t_j; the complete possibility is equal to 1. Otherwise, the selected seeds propagate the task information to their friends. For the latter case, it is inappropriate to compute the utility summation of nonseeds influenced by the selected users. Instead, the completion possibility of a task can be measured by the possibility that at least one friend can complete under the influence of selected seeds. Thus, the formula is listed as follows:

$$p(S, t_j) = \begin{cases} 1, & \exists u \in S, \text{pos}(u, j) = 1, \\ 1 - \prod_{w \in N(u)} \left(1 - \text{pos}_j(u, w)\right), & \text{others}. \end{cases} \tag{4}$$

This task allocation problem is aimed at maximizing the expected number of completed tasks under a limited number of seeds and propagating task information with the assistance of social networks. Based on the above definitions, the problem can be formulated as follows:

$$\begin{aligned} \max: \quad & \sum_{j=1}^{n} p(S, t_j) \\ \text{s.t.} \quad & |S| = k. \end{aligned} \tag{5}$$

4. Theoretical Analysis and Algorithm Design

In this section, we first prove that the SATA problem is an NP-hard problem. Then, we further prove that it is also a submodular problem. For the ease of presentation, we use the $\Phi(S)$ as the expected number of completed tasks of user set S in the following sections.

4.1. NP-Hard

Theorem 1. *The SATA problem is an NP-hard problem.*

Proof. The SATA problem is aimed at selecting k seeds to maximize the expected number of completed tasks, which can be transferred into the maximum coverage problem. The traditional maximum coverage problem can be described as follows: given a collection set $S = \{S_1, S_2, \cdots, S_m\}$ and a number k, the object is to find a subset $S' \subset S$, such that $|S'| \prec k$ and the number of covered elements in S is maximized. In our problem, we can treat the union set of tasks T and users U as the set S, i.e., $S = T \cup U$. Because every user can cover himself, selecting k user to cover the set S is equivalent to covering the set T. So our problem is a special case of maximum coverage problem, which is a well-known NP-hard problem. Thus, SATA is also an NP-hard problem.

4.2. Submodular Problem

Theorem 2. *Giving the seed set S, the $\Phi(S)$ is a submodular function. More specifically, for two seed sets A and B, $A \subset B$, a new user $u \in U/B$, the following equation holds:*

$$\Phi(A \cup \{u\}) - \Phi(A) \geq \Phi(B \cup \{u\}) - \Phi(B). \tag{6}$$

Proof. We prove the theorem from two properties, monotonic property and submodular property. For the sake of simplicity, we discuss the circumstance that $|B| - |A| = 1$ and $B - A = \{u'\}$.

(1) Monotonic property: since $A \subset B$, so the sensing task completed by the set A is also completed by the set B. For those tasks that have not been completed by the set A, user u' can complete tasks or propagate these tasks to his friends. According to the definition of propagation performance and Equation (4), the following equation holds:

$$\Phi(B) - \Phi(A) = \sum_{j=1}^{n} P\left(\left\{u'\right\}, t_j\right) \geq 0. \tag{7}$$

Therefore, $\Phi(S)$ is a nondecreasing function.

(2) Submodular property: Given a new user u, for those tasks that u can complete, two user sets have the same increment. So the following equation holds:

$$(\Phi(B \cup \{u\}) - \Phi(B)) - (\Phi(A \cup \{u\}) - \Phi(A)) = 0. \tag{8}$$

For those tasks that user u cannot complete, u propagates the task information to his friends. To analyze this situation elaborately, we divide this situation into three cases:

(1) Case 1: user u is independent to set A and set B; i.e., there are no common friends between u and two user sets. When adding the new user u into the two sets, both two union sets only increase the performance of u. So Equation (8) holds.

(2) Case 2: user u has common friends u' with set B but no common friends with set A. For the sake of simplification, we discuss there is only one task t_j to be propagated. Assume that u and u' have a common friend v. v completes the t_j with the probability of p_1 under the influence of u, and with the probability of p_2 under the influence of u'. The performance that u' influences other friends keeps unchanged. However, the performance that it influences user v should be recomputed; then we have:

$$\begin{aligned}
&(\Phi(B \cup \{u\}) - \Phi(B)) - (\Phi(A \cup \{u\}) - \Phi(A)) \\
&= \Phi(B) - \Phi\left(u'\right) + \Phi\left(u, u'\right) - \Phi(B) - \Phi(u) \\
&= \Phi\left(u, u'\right) - \Phi\left(u'\right) - \Phi(u) \\
&= \max\{p_1, p_2\} - p_2 - p_1 < 0.
\end{aligned} \tag{9}$$

(3) Case 3: user u has common friends with sets A and B. In this case, the possibility of common friends being influenced changes equally. Equation (8) holds.

By combining Equations (7), (8), and (9), we can conclude that $\Phi(S)$ is a submodular function.

4.3. Algorithm Design. According to the analysis above, we can extend the native greedy selection algorithm to solve the SATA problem, which can return at least $(1 - 1/e)$-approximation ratio of the optimal result and is widely used to solve the influence maximization problem [29]. Algorithm 1 shows the details of the native greedy selection algorithm. It selects the seed with the maximum margin gain in terms of the expected number of completed tasks in each round. Then, the completed tasks should be deleted least reallocated in the next round. This process continues until the number of the selected seeds reaches to k.

However, the NGS algorithm has some drawbacks. It should compute the performance of all candidate users to all tasks. So the running time of the NGS algorithm increases with the number of tasks. Different from the influence maximization problem, which simply computes the influence summation of selected users, according to the definition of the SATA problem, we aim to maximize the expected number of completed tasks, the most important part of which is the number of tasks that are actually completed. If a task can be completed by a seed, we need not to propagate the task information. Thus, a user who can complete more tasks should get higher priority than those who complete fewer tasks and propagate more tasks to friends. The advantage of this strategy is that after deleting the completed tasks from the task list, the number of tasks to be allocated in the next round will decrease, so it can reduce the running time to traverse task list.

Input: G: social network, k: seed set size,
T: task set, U: candidate seeds.
Output: a set of k seeds.
1: $S \longleftarrow \phi$;
2: **while** $|S| < k$
3: **for** $u \in U/S$ **do**
4: $u \longleftarrow \arg\max \{\Phi(S \cup \{u\}) - \Phi(S)\}$;
5: $S \longleftarrow S \cup \{u\}$;
6: **end for**
7: delete the completed tasks from T;
8: **end while**
9: **return** S;

Algorithm 1:Native greedy selection algorithm (NGS).

According to the analysis above, we further propose the FGS algorithm which considers the number of tasks that seeds can actually complete. In the FGS algorithm, we divide the selection process into two stages: in the first stage, the candidate seeds can complete part of sensing tasks. We select the seed which can complete the maximum number of tasks. In the second stage, the rest candidate seeds cannot complete any remaining tasks. In this situation, we change the object to the propagation utility of seeds to the rest of tasks and select the seed with maximum propagation margin gain. Because FGS deletes more completed tasks in the first stage, there is only part of tasks to be allocated in the second stage, so it is more efficient. The details of FGS can be shown in Algorithm 2.

4.4. Complexity Analysis. In this section, we analyze the time complexity of the proposed algorithms. According to Algorithm 1 and Algorithm 2, we should traverse the remaining users in each round and compute their utility to the remaining tasks. This process continues until the cardinality of the selected seed set reaches to k. So the time complexity of NGS and FGS are $O(kmn)$. It is worthy to note that FGS has the same time complexity with NGS; however, FGS deletes more completed tasks in each round and reduces the running time in the next round. Thus, FGS is faster than NGS.

5. Evaluation

In this section, we evaluate the performance of our proposed algorithms. We firstly present the detailed description of datasets and some basic experiment settings. Secondly, we introduce the baseline algorithms for evaluation. Finally, the detailed results of the proposed algorithms and baseline algorithms are presented and analyzed.

5.1. Datasets. We adopt two widely used real-world datasets, Brightkite and Gowalla datasets [30], to evaluate the performance of the proposed algorithms and baseline algorithms. These two datasets contain both the mobility trace of users and the relationships among users.

The Brightkite dataset is a service provider where users shared their locations by check-in, and the friendship network can be collected using the public API. The dataset consists of 58228 nodes and 214078 edges. There are 4491143 check-in records during Apr. 2008-Oct. 2010. In our simulation experiments, we select the check-in records during Apr. 2008-Mar. 2009 to train the geographical performance of users and then randomly select the check-in locations on Apr.1, 2009, as the locations of sensing tasks. The required time of tasks is also randomly set in this day. The part of tasks can be shown in Figure 2(a).

Input: G: social network, k: seed set size,
T: task set, U: candidate seeds.
Output: a set of k seeds.
1: $S \longleftarrow \phi$, flag = False
2: **while** $|S| < k$ and flag == False **do**
3: **for** $u \in U/S$ **do**
4: $u \longleftarrow \arg\max \{complete\_num\}$
5: **if** $\max \{complete\_num\} == 0$ **then**
6: flag = True;
7: **break;**
8: **end if**
9: $S \longleftarrow S \cup \{u\}$;
10: **end for**
11: delete the completed tasks from T;
12: **end while**
13: **if** $|S| < k$ **then**
14: $u \longleftarrow \arg\max \{\Phi(S \cup \{u\}) - \Phi(S)\}$;
15: $S \longleftarrow S \cup \{u\}$;
16: **end if**
17: **return** S;

Algorithm 2: Fast greedy selection algorithm (FGS).

Similar to the Brightkite, Gowalla is also a location-based network that users share their locations by checking in. The friendship network was collected using their API. There are 196591 nodes and 950327 edges. The dataset collects 6442890 check-in records during Feb. 2009-Oct. 2010. In this paper, we select the check-in records during Feb. 2009-Jan. 2010 to train the geographical performance of users and then randomly select locations on Feb.1, 2010, as the locations of sensing tasks. The required time of tasks is also randomly set in this day. The part of tasks can be shown in Figure 2(b).

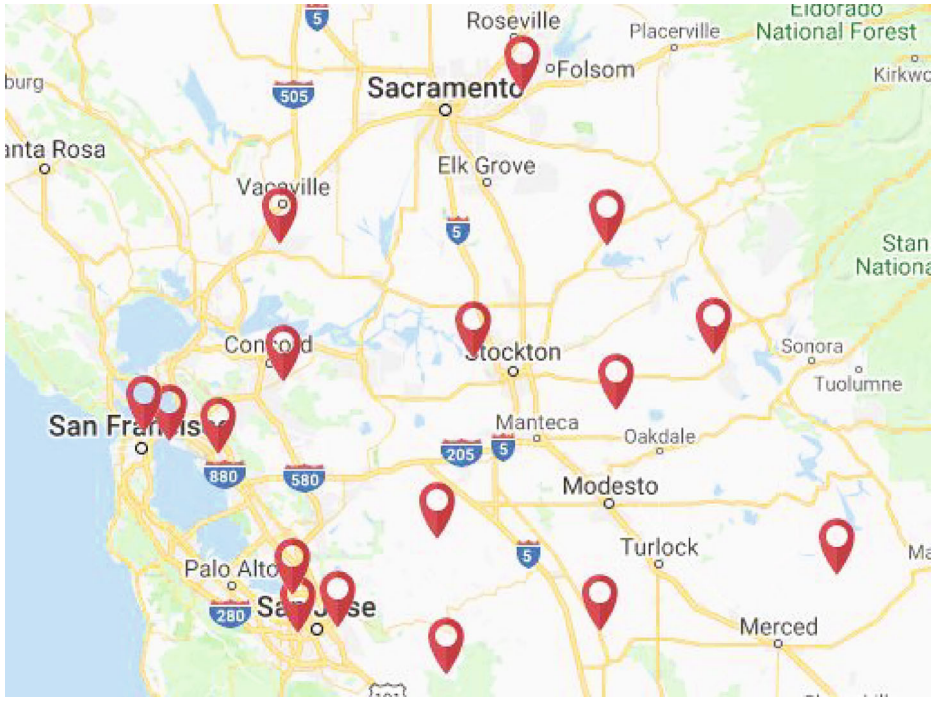
5.2. Baseline Algorithms. To compare the performance of the proposed algorithms, several baseline methods are designed as follows:

(1) Propagation-based greedy selection (PGS): this method selects the seed with the largest propagation performance iteratively until the number of seeds reaches the limitation.

(2) Degree-based greedy selection (DGS): this method simply selects the seeds with the largest degree, which is aimed at propagating the task information to more friends.

(3) Random selection (RS): this method randomly selects k users as seeds to complete and propagate task

(a) Part of tasks in Brightkite

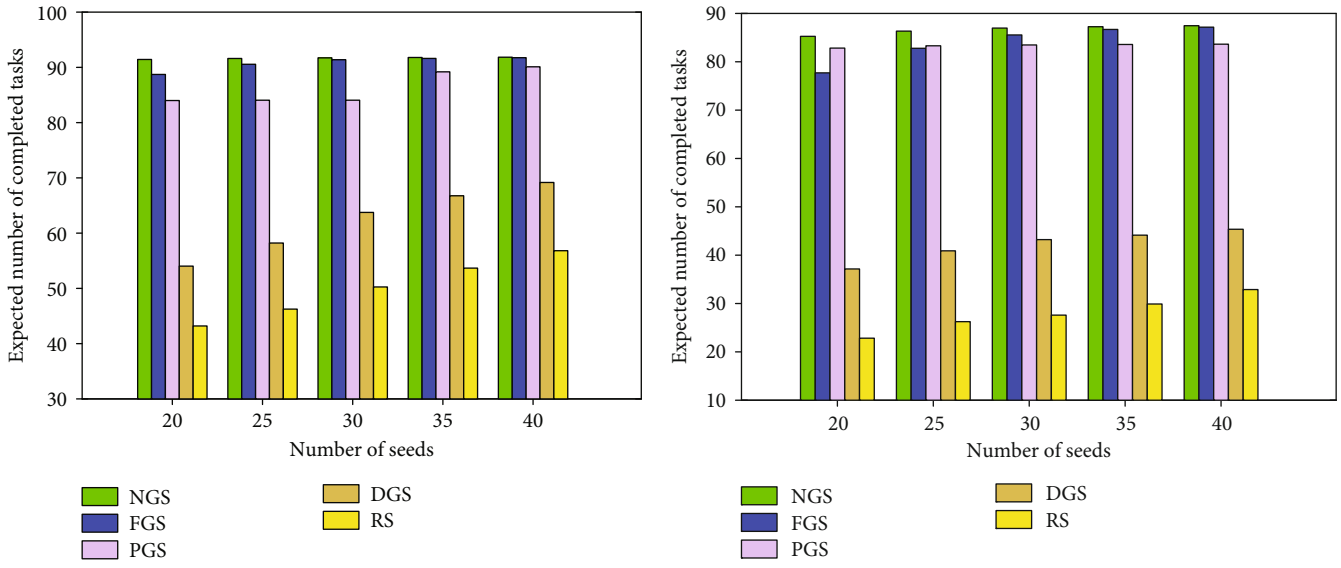


(b) Part of tasks in Gowalla

Figure 2: Part of tasks in two datasets.

information. To reduce the error caused by randomness, we run the RS algorithm 30 times and compute the average expected number of completed tasks as the final result.

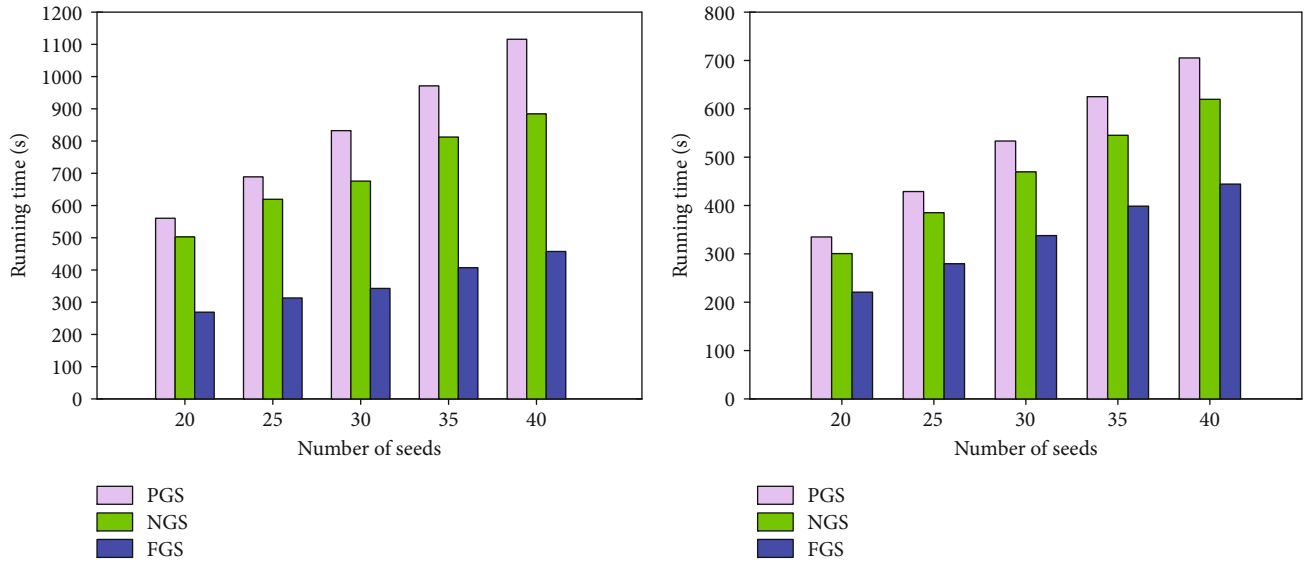
In the following simulations, all the experiments are conducted on a PC with 3.10 GHz CPU and 16 GB memory. We compare two indicators of algorithms: the expected number of completed tasks and running time. In terms of efficiency of algorithms, because the DGS selects the users with top k degree while the degree of users keeps unchanged in the whole seed selection process, so target seeds can be easily selected. For RS, it randomly selects k users as seeds without any heuristic strategies. Compared to the NGS, PGS, and FGS, these two algorithms can quickly identify the target users. It does not make sense to discuss the running time of these two algorithms. So we only compare the running time of the NGS, PGS, and FGS.



(a) Expected number of completed tasks in Brightkite

(b) Expected number of completed tasks in Gowalla

Figure 3: Performance comparison on the number of seeds.



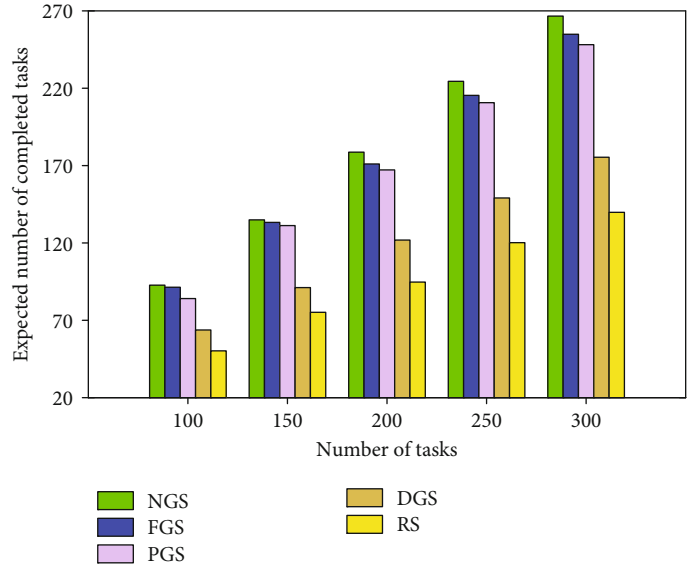
(a) Running time in Brightkite

(b) Running time in Gowalla

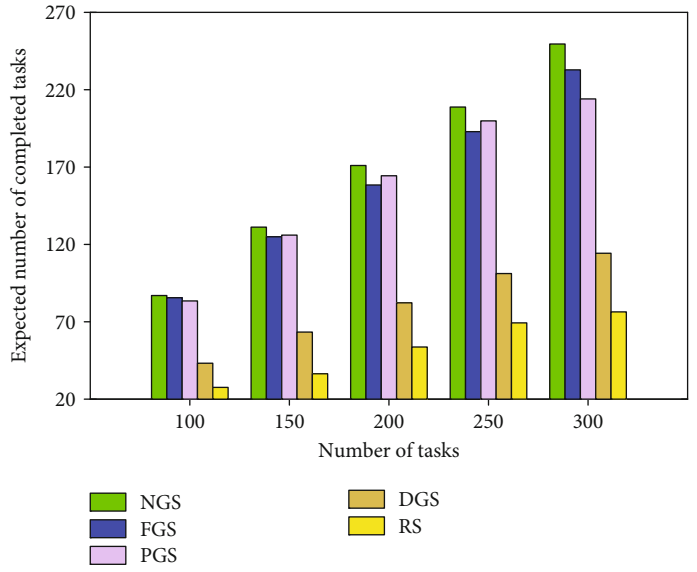
Figure 4: Running time comparison on the number of seeds.

5.3. Impact of Number of Seeds. The main objective of the SATA problem is to maximize the expected number of completed tasks. To measure the impact of the seed number on the expected number of completed tasks, the related parameters are set as follows: for the Brightkite dataset, we set $m = 100$, $n = 100$, and $R = 10$ and change the number of seeds from 20 to 40 with an increment of 5. For the Gowalla dataset, we set $R = 5$, while other parameters are the same with those of the Brightkite dataset. Figures 3 and 4 show the results of our proposed algorithms and baseline algorithms.

From Figure 3, we can find that with the increment of seed number, the expected number of completed tasks also increases. This is reasonable because more seeds can complete or propagate more tasks. NGS outperforms other algorithms while the increment of NGS is smaller than those of others. Because our task allocation problem is a submodular problem and has the diminishing return property, which matches the theoretical result in Theorem 2, FGS firstly selects the seed user who can actually complete the maximum number of tasks. However, these seeds may have limited capacity to propagate the tasks that they cannot actually complete. So the performance is a little worse than NGS. PGS just focuses on the seed with the largest propagation performance and ignores the number of actual completed tasks.



(a) Expected number of completed tasks in Brightkite



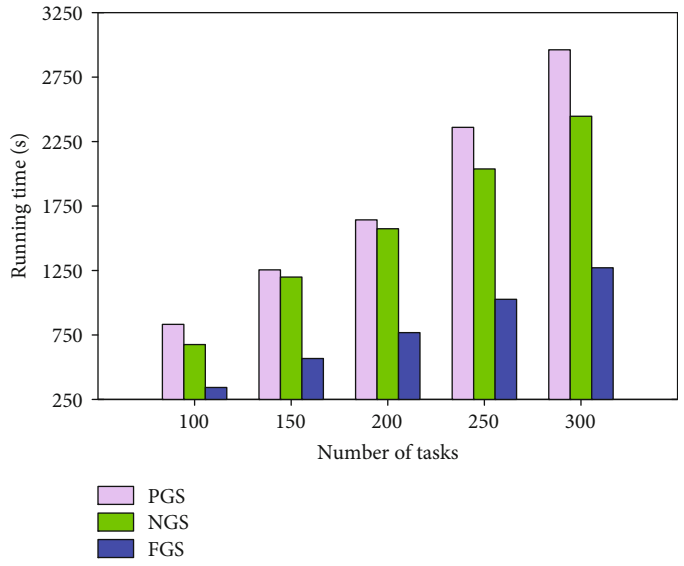
(b) Expected number of completed tasks in Gowalla

Figure 5: Performance comparison on the number of tasks.

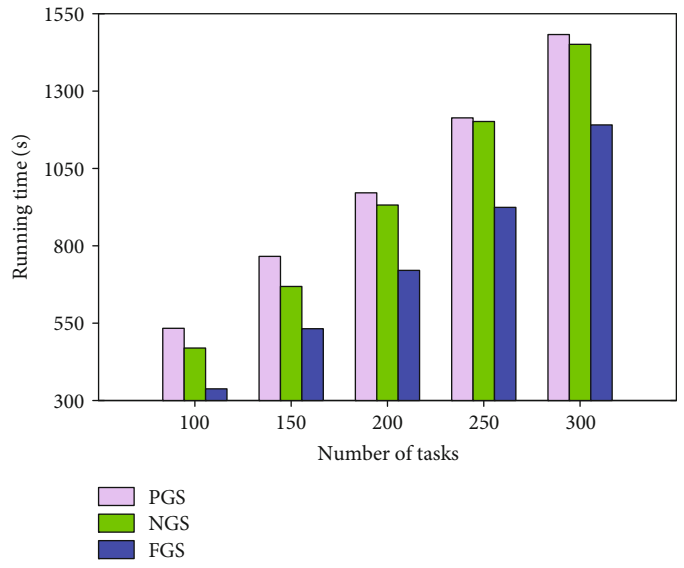
So it gets worse results. For DGS, it firstly selects a user with the maximum degree, which is aimed at propagating the task information to more friends and ignores the number of tasks that seeds can actually complete. So it gets worse results than FGS and NGS algorithm. RS selects seeds randomly and does not consider any heuristic strategy in the process of seed selection, so it gets the worst results in terms of the expected number of completed task.

From Figure 4, we can see that the running time of three algorithms increases with the increment of seed number. Because selecting more seeds needs more rounds to traverse the candidate users, PGS consumes the most time among the three algorithms. The reason is that the seeds selected by the PGS have the maximum propagation performance. However, they may only complete a small part of tasks in each round, leaving most of tasks to be allocated in the next round. Thus, it needs more time in every round. In contrast, FGS firstly selects the user who can complete most tasks; with the increment of seeds, more and more tasks can be completed. The number of tasks to be allocated is greatly decreased in the next round. So it is faster than the NGS and PGS. In summary, for the impact of the number of selected seeds, it is worthy to note that the expected number of completed tasks in FGS is slightly worse than that in NGS; however, the time consumption of FGS is far less than those of NGS and PGS.

5.4. Impact of Number of Tasks. In this section, we conduct experiments to illustrate the impact of the number of tasks on the performance of algorithms. The related parameters



(a) Running time in Brightkite



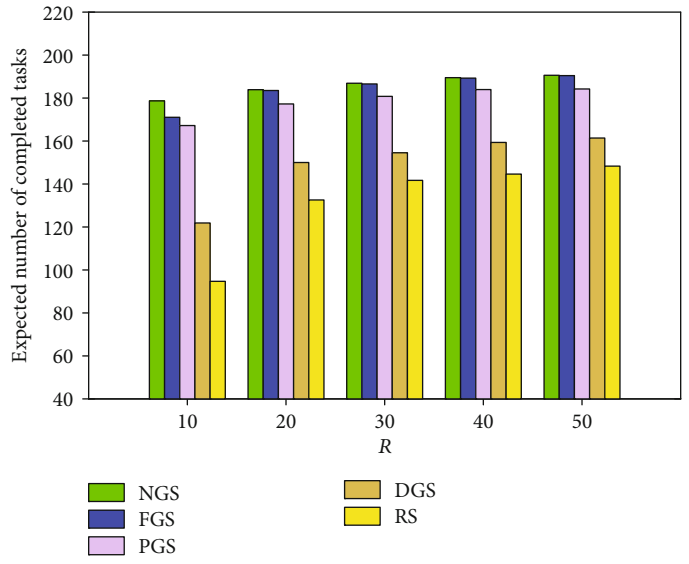
(b) Running time in Gowalla

Figure 6: Running time comparison on the number of tasks.

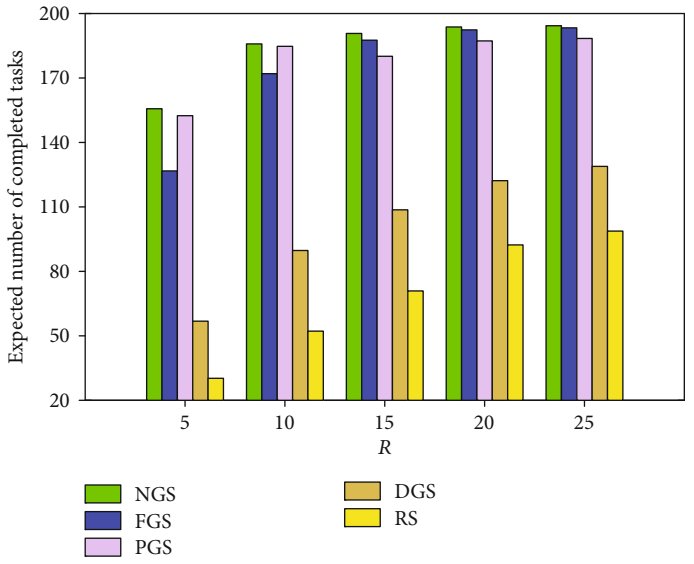
are set as follows: for the Brightkite dataset, we set $m = 100$, $R = 10$, and $k = 30$ and change the number of tasks from 100 to 300 with an increment of 50. For the Gowalla dataset, we set $R = 5$, while other parameters are the same with those of the Brightkite dataset. The results of our proposed algorithms and baseline algorithms are shown in Figures 5 and 6.

As shown in Figure 5, the expected number of completed tasks increases with the number of tasks. It indicates that seeds can complete and propagate more tasks even if the number of seeds is limited. NGS gets better results compared with other algorithms, and the difference among them becomes more and more significant with the increase of tasks. FGS focuses on the number of actually completed tasks. Under the limited candidate user resource, the number of tasks that seeds can complete is also limited; the actual completion ratio decreases with the increment of the number of total tasks. In this situation, the seeds selected by FGS have the weaker capacity to propagate the tasks that they cannot complete. However, the NGS considers both the number of tasks that users can complete and the capacity to propagate the task information that they cannot complete. So it gets better results. PGS gets worse results than NGS and FGS in the Brightkite dataset. For the Gowalla dataset, PGS gets better results than FGS when the task number is 200 and 250. DGS selects the user with the maximum degree as seed to propagate the task, so the growth rate decreases with the increase of task number.

According to Figure 6, the running time of algorithms increases with the number of tasks. Because more tasks need



(a) Expected number of completed tasks in Brightkite



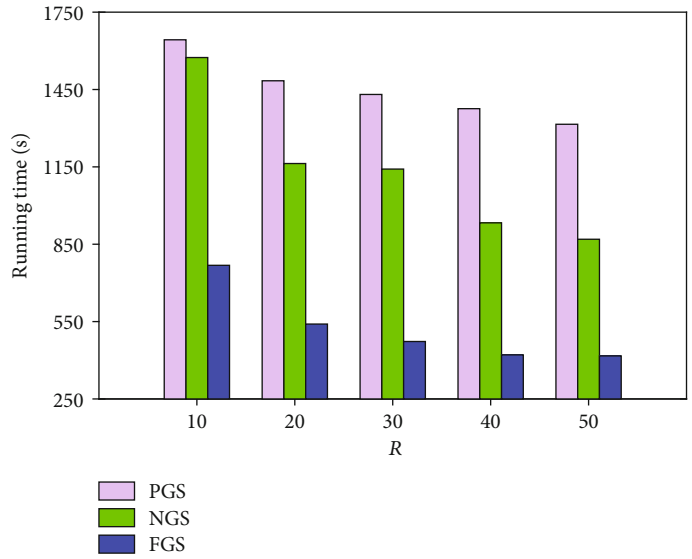
(b) Expected number of completed tasks in Gowalla

Figure 7: Performance comparison on R.

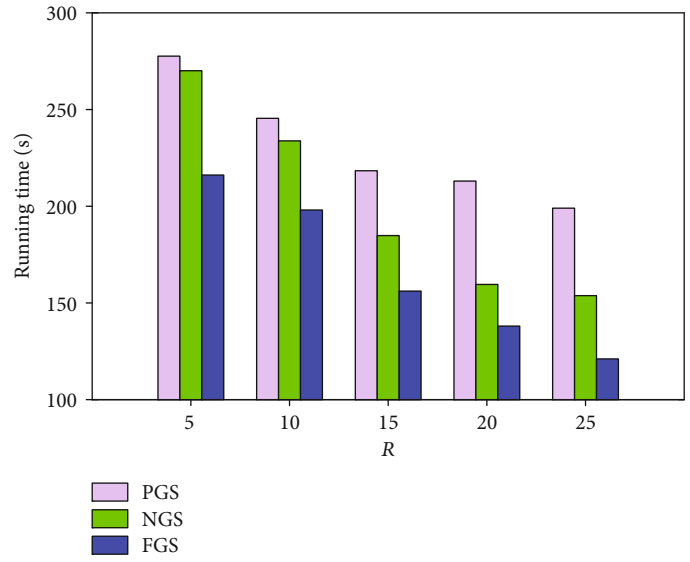
more time for seeds to propagate task information, PGS consumes the most time among three algorithms, because PGS needs to propagate more task information due to the low completion ratio of selected users in the previous rounds. In the Brightkite dataset, FGS costs nearly half running time compared with NGS. In the Gowalla dataset, though FGS costs less running time than NGS, the advantage in running time becomes less obvious with the increment of the number of tasks.

5.5. Impact of R. In this line of simulations, we conduct experiments to illustrate the impact of parameter R to the performance of algorithms, where R is the seed recruiting range for every task. The related parameters are set as follows: for the Brightkite dataset, we set $m = 100$, $n = 200$, and $k = 30$ and change R from 10 to 50 with an increment of 10. For the Gowalla dataset, we set $k = 10$ and change R from 5 to 25 with an increment of 5. The results of the expected number of completed tasks and running time are shown in Figures 7 and 8.

From Figure 7, we can see that with the increment of R, the expected number of completed tasks increases. The advantage of NGS becomes more and more obvious. The reason can be listed as follows: on the one hand, seeds can complete more tasks within the larger range, even though the number of seeds is limited. On the other hand, seeds can propagate the task information to more friends if the geographical range is enlarged. So the expected number of completed tasks increases with the increment of R. However, for the Gowalla dataset, PGS gets better results compared to FGS when $R = 5$ and $R = 10$. Because PGS firstly selects the user with the maximum propagation



(a) Running time in Brightkite



(b) Running time in Gowalla

Figure 8: Running time comparison on R.

performance, if the selected user just completed a small part of tasks, it may get lower result in terms of the expected number.

According to Figure 8, the running time of three algorithms decreases with the increment of R. This can be explained that enlarging the range of users to complete tasks, the number of actually completed tasks increases, seeds selected by three algorithms can complete more task, and there is no need to propagate the task information to friends; thus, the time to propagate the task information decreases. Furthermore, the efficiency of FGS is prominent with the increment of R. Compared to the NGS and PGS, the seeds firstly selected by FGS can complete more tasks. The number of tasks to be allocated decreases significantly in the next round, so the running time decreases with the increment of recruit range.

6. Conclusion

In this paper, we study the social-aware task allocation problem (SATA) in mobile crowd sensing, which is aimed at selecting a part of users as seeds to complete or propagate tasks. Firstly, we illustrate the task information propagation under the IC model with the consideration of geographical performance and social similarity factors. Secondly, we prove that the SATA problem is an NP-hard and submodular problem. To solve this problem, we devise two greedy-based algorithms (NGS and FGS) to select seeds. NGS selects seeds with the maximum margin gain in terms of the expected number of completed tasks. FGS selects seeds which can actually complete the maximum number of tasks. The experimental results show that FGS gets slightly worse results than NGS; however, it greatly reduces the running time.

Data Availability

The source data used in the study is available in the following website: http://snap.stanford.edu/data/.

Conflicts of Interest

The authors declare that there are no conflicts of interest regarding the publication of this paper.

Acknowledgments

This work was partially supported by the National Natural Science Foundation of China (Nos. U1705262, 61772136, and 61672159).

References

[1] B. Guo, Z. Wang, Z. Yu et al., "Mobile crowd sensing and computing: the the review of an emerging human-powered sensing paradigm," *ACM Computing Surveys*, vol. 48, no. 1, pp. 1–31, 2015.

[2] W. Guo, W. Zhu, Z. Yu, J. Wang, and B. Guo, "A survey of task allocation: contrastive perspectives from wireless sensor networks and mobile crowdsensing," *IEEE Access*, vol. 7, pp. 78406–78420, 2019.

[3] J. Wang, L. Wang, Y. Wang, D. Zhang, and L. Kong, "Task allocation in mobile crowd sensing: state-of-the-art and future opportunities," *IEEE Internet of Things Journal*, vol. 5, no. 5, pp. 3747–3757, 2018.

[4] J. Chen and J. Yang, "Maximizing coverage quality with budget constrained in mobile crowd-sensing network for environmental monitoring applications," *Sensors*, vol. 19, no. 10, article 2399, 2019.

[5] S. Morishita, S. Maenaka, D. Nagata et al., "Sakurasensor: quasi-real time cherry-lined roads detection through participatory video sensing by cars," in *UbiComp '15: Proceedings of the 2015 ACM International Joint Conference on Pervasive and Ubiquitous Computing*, pp. 695–705, New York, NY, USA, September 2015.

[6] T. Ludwig, C. Reuter, and V. Pipek, "What you see is what I need: mobile reporting practices in emergencies," *ECSCW 2013: Proceedings of the 13th European Conference on Computer Supported Cooperative Work, 21-25 September 2013, Paphos, Cyprus*, O. Bertelsen, L. Ciolfi, M. Grasso, and G. Papadopoulos, Eds., , pp. 181–206, Springer, London, 2013.

[7] D. Zhang, L. Wang, H. Xiong, and B. Guo, "4w1h in mobile crowd sensing," *IEEE Communications Magazine*, vol. 52, no. 8, pp. 42–48, 2014.

[8] B. Guo, Y. Liu, L. Wang, V. O. K. Li, J. C. K. Lam, and Z. Yu, "Task allocation in spatial crowdsourcing: current state and future directions," *IEEE Internet of Things Journal*, vol. 5, no. 3, pp. 1749–1764, 2018.

[9] W. Guo, J. Li, G. Chen, Y. Niu, and C. Chen, "A pso-optimized real-time fault-tolerant task allocation algorithm in wireless sensor networks," *IEEE Transactions on Parallel and Distributed Systems*, vol. 26, no. 12, pp. 3236–3249, 2014.

[10] Y. Liu, B. Guo, Y. Wang, W. Wu, Z. Yu, and D. Zhang, "Taskme: multi-task allocation in mobile crowd sensing," in *UbiComp '16: Proceedings of the 2016 ACM International Joint Conference on Pervasive and Ubiquitous Computing*, pp. 403–414, New York, NY, USA, September 2016.

[11] L. Wang, Z. Yu, Q. Han, B. Guo, and H. Xiong, "Multi-objective optimization based allocation of heterogeneous spatial crowdsourcing tasks," *IEEE Transactions on Mobile Computing*, vol. 17, no. 7, pp. 1637–1650, 2018.

[12] E. Wang, Y. Yang, J. Wu, K. Lou, D. Luan, and H. Wang, "User recruitment system for efficient photo collection in mobile crowdsensing," *IEEE Transactions on Human-Machine Systems*, vol. 50, no. 1, pp. 1–12, 2019.

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

[14] A.-q. Lu and J.-h. Zhu, "Worker recruitment with cost and time constraints in mobile crowd sensing," *Future Generation Computer Systems*, vol. 112, pp. 819–831, 2020.

[15] S. Reddy, K. Shilton, J. Burke, D. Estrin, M. Hansen, and M. Srivastava, "Using context annotated mobility profiles to recruit data collectors in participatory sensing," in *Location and Context Awareness. LoCA 2009. Lecture Notes in Computer Science, vol 5561*, T. Choudhury, A. Quigley, T. Strang, and K. Suginuma, Eds., pp. 52–69, Springer, Berlin, Heidelberg, 2009.

[16] S. Reddy, D. Estrin, and M. Srivastava, "Recruitment framework for participatory sensing data collections," in *Pervasive Computing. Pervasive 2010. Lecture Notes in Computer Science, vol 6030*, P. Floréen, A. Krüger, and M. Spasojevic, Eds., pp. 138–155, Springer, Berlin, Heidelberg, 2010.

[17] M. Zhang, P. Yang, C. Tian et al., "Quality-aware sensing coverage in budget-constrained mobile crowdsensing networks," *IEEE Transactions on Vehicular Technology*, vol. 65, no. 9, pp. 7698–7707, 2015.

[18] X. Wang, W. Wu, and D. Qi, "Mobility-aware participant recruitment for vehicle-based mobile crowdsensing," *IEEE Transactions on Vehicular Technology*, vol. 67, no. 5, pp. 4415–4426, 2017.

[19] Z. Yu, J. Zhou, W. Guo, L. Guo, and Z. Yu, "Participant selection for t-sweep k-coverage crowd sensing tasks," *World Wide Web*, vol. 21, no. 3, pp. 741–758, 2018.

[20] H. Li, T. Li, and Y. Wang, "Dynamic participant recruitment of mobile crowd sensing for heterogeneous sensing tasks," in *2015 IEEE 12th International Conference on Mobile Ad Hoc and Sensor Systems*, pp. 136–144, Dallas, TX, USA, October 2015.

[21] W. Zhu, W. Guo, Z. Yu, and H. Xiong, "Multi-task allocation to heterogeneous participants in mobile crowd sensing," *Wireless Communications and Mobile Computing*, vol. 2018, Article ID 7218061, 10 pages, 2018.

[22] J. Jiang, B. An, Y. Jiang, C. Zhang, Z. Bu, and J. Cao, "Group-oriented task allocation for crowdsourcing in social networks," *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, pp. 1–16, 2019.

[23] Z. Wang, J. Zhao, J. Hu et al., "Towards personalized task-oriented worker recruitment in mobile crowdsensing," *IEEE Transactions on Mobile Computing*, 2020.

[24] J. Wang, Y. Wang, D. Zhang et al., "Multi-task allocation in mobile crowd sensing with individual task quality assurance," *IEEE Transactions on Mobile Computing*, vol. 17, no. 9, pp. 2101–2113, 2018.

[25] H. Wang, D. Zhao, H. Ma, and L. Ding, "Min-max planning of time-sensitive and heterogeneous tasks in mobile crowd sensing," in *2018 IEEE Global Communications Conference*

(GLOBECOM), pp. 1–7, Abu Dhabi, United Arab Emirates, December 2018.

[26] W. Ni, P. Cheng, L. Chen, and X. Lin, "Task allocation in dependency-aware spatial crowdsourcing," in *2020 IEEE 36th International Conference on Data Engineering (ICDE)*, pp. 985–996, Dallas, TX, USA, April 2020.

[27] S. Song, Z. Liu, Z. Li, T. Xing, and D. Fang, "Coverage-oriented task assignment for mobile crowdsensing," *IEEE Internet of Things Journal*, vol. 7, no. 8, pp. 7407–7418, 2020.

[28] Z. Peng, X. Gui, J. An, R. Gui, and Y. Ji, "Tdsrc: A task-distributing system of crowdsourcing based on social relation cognition," *Mobile Information Systems*, vol. 2019, Article ID 7413460, 12 pages, 2019.

[29] Y. Yang, Y. Xu, E. Wang, K. Lou, and D. Luan, "Exploring influence maximization in online and offline double-layer propagation scheme," *Information Sciences*, vol. 450, pp. 182–199, 2018.

[30] E. Cho, S. A. Myers, and J. Leskovec, "Friendship and mobility: user movement in location-based social networks," in *KDD '11: Proceedings of the 17th ACM SIGKDD international conference on Knowledge discovery and data mining*, pp. 1082–1090, New York, NY, USA, August 2011.