Task Allocation in Dependency-aware Spatial Crowdsourcing

Wangze Ni †, Peng Cheng *, Lei Chen †, Xuemin Lin #,*
†The Hong Kong University of Science and Technology, Hong Kong, China
*East China Normal University, Shanghai, China
#The University of New South Wales, Australia
{wniab, leichen}@cse.ust.hk, pcheng@sei.ecnu.edu.cn, lxue@cse.unsw.edu.au

***Abstract*—Ubiquitous smart devices and high-quality wireless networks enable people to participate in spatial crowdsourcing tasks easily, which require workers to physically move to specific locations to conduct their assigned tasks. Spatial crowdsourcing has attracted much attention from both academia and industry. In this paper, we consider a spatial crowdsourcing scenario, where the tasks may have some dependencies among them. Specifically, one task can only be dispatched when its dependent tasks have already been assigned. In fact, task dependencies are quite common in many real-life applications, such as house repairing and holding sports games. We formally define the dependency-aware spatial crowdsourcing (DA-SC), which focuses on finding an optimal worker-and-task assignment under the constraints of dependencies, skills of workers, moving distances and deadlines to maximize the successfully assigned tasks. We prove that the DA-SC problem is NP-hard and thus intractable. Therefore, we propose two approximation algorithms, including a greedy approach and a game-theoretic approach, which can guarantee the approximate bounds of the results in each batch process. Through extensive experiments on both real and synthetic data sets, we demonstrate the efficiency and effectiveness of our DA-SC approaches.**

***Index Terms*—Spatial Crowdsourcing, Task Assignment, Approximate Algorithm**

I. INTRODUCTION

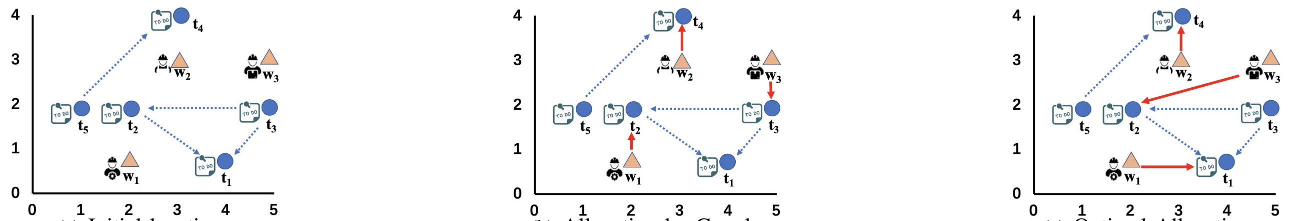
Recently, with the popularity of smart devices and high-speed wireless networks, a new kind of crowdsourcing systems, namely spatial crowdsourcing [1], become ubiquitous (e.g., Uber [2], and Waze [3]), and attract attention from both academia and industry. Specifically, in spatial crowdsourcing systems, workers are requested to physically move to particular locations to conduct their assigned tasks. The spatial tasks can be simple tasks that every normal worker can conduct, such as taking photos of a landmark (e.g., street view of Google Maps [4]), and delivering foods (e.g., Uber Eats [5]). However, some complex tasks require specific skills to finish, such as wall painting, cleaning home and refereeing a sports game.

Wangze Ni and Lei Chen are partially supported by the Hong Kong RGC CRF Project C6030-18GF, the National Science Foundation of China (NSFC) under Grant No. 61729201, Science and Technology Planning Project of Guangdong Province, China, No. 2015B010110006, Hong Kong ITC ITF grants ITS/044/18FX and ITS/470/18FX, Didi-HKUST joint research lab grant, Microsoft Research Asia Collaborative Research Grant, and Wechat Research Grant. Peng Cheng is supported by Shanghai Pujiang Program 19PJ1403300. Xuemin Lin is supported by 2018YFB1003504, NSFC61232006, ARC DP180103096 and DP170101628. Corresponding author: Peng Cheng.

Previous studies [6], [7], [8] in multi-skill/complex tasks oriented spatial crowdsourcing focus on assigning a set of multi-skilled workers to a given complex task such that the required skills of the task can be fully covered by the union of the skill sets of the assigned workers. However, in practice, each complex task can be a combination of multiple subtasks and there are dependencies between the subtasks. For instance, when we want to hold an orienteering race, we have to arrange a set of checkpoints following a specific order. Since the clue of the next checkpoint should be put in the last checkpoint, the tasks of setting each checkpoint should be conducted following the order. Another example is that when we want to hold a sports game, like a basketball game, we have to hire many workers in different roles. For a basketball game, to make the game fairer and avoid the argument, we need to hire referees. According to the official basketball game regulation [9], we first need to hire a crew chief. After that, we can go to hire two associate referees. We also need to hire a scorer, a timekeeper, and a 24-second shot clock operator. Before hiring these officials, we should firstly hire a chairperson. After that, we can hire an assistant scorer. If the previous solutions [6], [7], [8] are used directly, the dependency between the subtasks will cause the efficiency of the assignment to become very low (i.e., some assigned workers need to wait until the dependencies of their subtasks are satisfied).

Further, the workers are free to come and leave and the tasks dynamically appear in spatial crowdsourcing platforms. If a requester creates the subtasks one-by-one satisfying their dependencies, he/she needs to keep on monitoring the progresses of the subtasks and submits proper subtasks to the platform once their dependent subtasks are all accomplished, which is not efficient with respect to time and requires a lot of effort from the requesters in the highly dynamic spatial crowdsourcing scenarios. Thus, a dependency-aware spatial crowdsourcing platform is needed to efficiently assign multi-skill workers to dependency-aware tasks such that they can be optimally conducted. In this paper, we consider an important problem in the spatial crowdsourcing system, namely *dependency-aware spatial crowdsourcing* (DA-SC), which assigns workers to those dependency-aware tasks, with the constraints of skills, dependencies, moving distance, and deadlines of spatial tasks.

In the sequel, we will illustrate the DA-SC problem through a motivation example.



(a) Initial locations (b) Allocation by Greedy (c) Optimal Allocation

Fig. 1: Motivation Example.

TABLE I: Tasks' Detail.

Task	Location	Required Skill	Dependency
t_1	(4, 1)	ψ_1	$\emptyset$
t_2	(2, 2)	ψ_2	$\{t_1\}$
t_3	(5, 2)	ψ_3	$\{t_1, t_2\}$
t_4	(3, 4)	ψ_4	$\emptyset$
t_5	(1, 2)	ψ_3	$\{t_4\}$

TABLE II: Workers' Detail.

Worker	Location	Skill Set
w_1	(2, 1)	$\{\psi_1, \psi_2\}$
w_2	(3, 3)	$\{\psi_4\}$
w_3	(5, 3)	$\{\psi_1, \psi_2, \psi_3\}$

Example 1. *In the example, a spatial crowdsourcing platform has three workers $(r_1 - r_3)$ and five tasks $(t_1 - t_5)$ as shown in Figure 1(a), where the dash arrow lines pointing from a task to its dependent tasks. The details about workers and tasks are shown in Table I and Table II. For simplicity, in this example we set all workers and tasks appear on the platform at the same time. And, the maximum moving distance of each worker is large enough and the moving speed of each worker is fast enough to reach the assigned tasks before their deadlines. The goal of the platform is to maximize the total number of finished tasks, whose assigned workers have the required skill, and dependencies have been satisfied.*

If the platform greedily assigns the nearest skill-satisfied workers to the tasks but does not consider the dependencies, the allocation is shown by the red arrows in Figure 1(b). Note that, since t_1, which is t_2's dependency, has not been assigned, the assignment (w_1, t_2) is invalid. Similarly, the assignment (w_3, t_3) is also invalid. Then, only one task can be conducted.

However, if the platform takes dependencies into account, the allocation is shown by the red arrows in Figure 1(c). Each worker is assigned to a task and the dependencies of each assigned task are satisfied. Thus, 3 tasks can be conducted.

Motivated by the example above, in this paper, we formalize the DA-SC problem, which focuses on efficiently assigning proper workers to dependency-aware spatial tasks, under the constraints of dependencies, skills, moving distances and deadlines, and the total number of the assigned worker-and-task pairs is maximized.

However, to achieve an optimal dependency-aware assignment result is not easy in DA-SC, where the tasks and workers are highly dynamic and their constraints are required to be all satisfied. Moreover, the dependencies between tasks make the problem more complex, since one task can only be conducted when its dependent tasks are all accomplished and finishing one critical task may lead to its subsequent tasks become ready to be conducted. Previous works in multi-skill oriented spatial crowdsourcing [6], [7], [8] do not consider the dependency of the tasks and assign a set of workers to fully support a complex task (consisting of a set of dependency-aware single-worker tasks) requiring a certain skill set, which is not efficient as some workers need to wait until the dependencies of their tasks are satisfied. Thus, no existing methods can be used to solve the dependency-aware spatial crowdsourcing problem efficiently.

In this paper, we first prove that the DA-SC problem is NP-hard by reducing it from the subset-sum-optimization problem [10]. Thus, the DA-SC problem is intractable and hard to achieve the optimal result. We tackle the DA-SC problem with two approximation algorithms, DASC_Greedy and DASC_Game, which efficiently acquire near-optimal results with proven approximation ratios for each batch process under the constraints of skills, dependencies, distance, and deadlines.

To summarize, our main contributions are listed as follows:

- We formalize the dependency-aware spatial crowdsourcing (DA-SC) problem and prove it is NP-hard in Section II.
- We propose two batch-based approximation algorithms, greedy and game-theoretic approaches, in Section III and Section IV respectively. Both of the approaches have guaranteed approximate bounds for the final assignment results.
- We conduct extensive experiments on real as well as synthetic data sets and show the efficiency and effectiveness of our proposed approaches in Section V.

For the rest of the paper, we discuss the related work in Section VI and conclude in Section VII.

II. Problem Definition

In this section, we present the formal definition of the dependency-aware spatial crowdsourcing (DA-SC) problem.

A. Heterogeneous Workers

We first define the heterogeneous workers in spatial crowdsourcing applications. Assume that $\Psi = \{\psi_1, \psi_2, \cdots, \psi_r\}$ is a universe of r abilities/skills.

Definition 1. (Heterogeneous Workers) A worker, denoted by $w = \langle l_w, s_w, w_w, v_w, d_w, WS_w \rangle$, appears on the platform with an initial location l_w at timestamp s_w and waits at most w_w time for assigning a task. In addition, the worker w moves with velocity v_w and has a maximum moving distance d_w. Moreover, each worker w has a set of skills $WS_w \subseteq \Psi$.

In Definition 1, the heterogeneous worker w locates at a spatial place l_w at timestamp s_w and will wait w_w time for an assignment. In other words, the worker w no longer provides services on the platform after the timestamp $s_w + w_w$. Moreover, each worker w is associated with a set of skills WS_w and can move dynamically in any direction with the velocity v_w. For simplicity, in this paper, we use Euclidean distance as our distance function, which is denoted as $dist(x, y)$ for the distance between locations x and y. Note that, our proposed approaches can also be used with other distance functions (e.g., road-network distance) to solve the DA-SC problem.

B. Dependency-aware Spatial Tasks

Next, we define dependency-aware tasks in the spatial crowdsourcing system, which are constrained by task locations, deadlines, skill requirements, and dependencies.

Definition 2. (Dependency-aware Spatial Tasks) Let $t = \langle l_t, s_t, w_t, rs_t, D_t\rangle$ denote a task. It appears on the platform with a spatial location l_t at timestamp s_t and needs to be served within w_t time. In addition, the task can be accomplished only by a worker who has skill rs_t. Moreover, the task is dependent on a set of tasks D_t.

A task requester creates a dependency-aware spatial task t, which requires a worker with skill rs_t physically moving to a specific location l_t and starting the service before the expiration time $s_t + w_t$. When the requesters propose the tasks, meanwhile, they can designate the dependency relationships between the tasks. For any task t, it can be conducted only after its precedent tasks D_t are conducted. Without loss of generality, the graph of the dependency relationships of tasks is acyclic, i.e., no tasks are dependent on its subsequent tasks.

C. The Dependency-aware Spatial Crowdsourcing Problem

We formally define the *dependency-aware spatial crowdsourcing* (DA-SC) problem as follows.

Definition 3. (Dependency-aware Spatial Crowdsourcing Problem) Given a set of workers W and a set of tasks T, the DA-SC problem is to obtain an assignment M among W and T to maximize the number of assigned worker-and-task pairs

$$Sum(M) = |M| = \sum_{w \in W, t \in T} I(w, t) \tag{1}$$

where $I(w, t) = 1$ if the pair (w, t) is matched in the assignment M, and otherwise $I(w, t) = 0$, such that the following constraints are satisfied:

1) **Skill constraint**. In order to accomplish the task t, the worker w must have the required skill ts_t.
2) **Deadline constraint**. For any worker-task pair (w, t), it should satisfy the following two deadline conditions. (1) The task should appear before the worker leaves the platform, (i.e., $s_t \le s_w + w_w$). (2) The worker should arrive at the location of the assigned task before the deadline of the task (i.e., $w_t - \max\{s_w - s_t, 0\} - ct_w(l_w, l_t) \ge 0$, $ct_w(l_w, l_t)$ is the travel cost from l_w to l_t).
3) **Exclusive constraint**. One task can be assigned to at most one worker (i.e., $\sum_{w \in W} I(w, t) \le 1$) and any worker can be assigned to only one task at each time.
4) **Dependency constraint**. The worker w can conduct the task t iff the task t's dependent tasks had been assigned. (i.e., $\prod_{t' \in D_t} a_{t'}$ = 1, where $a_{t'} = \sum_{w \in W} I(w, t')$).

TABLE III: Symbols and Descriptions.

Symbol	Description
Ψ	The universe of r abilities/skills Ψ_r
W	The set of n workers w
WS_w	The set of worker w's skills
v_w	The worker w's velocity
d_w	The worker w's maximum moving distance
T	The set of m tasks t
rs_t	The task t's required skill
D_t	The set of task t's dependent tasks
$l_w(l_t)$	The location of a worker(task)
$s_w(s_t)$	The start timestamp of a worker(task)
$w_w(w_t)$	The waiting time of a worker(task)
$ct_w(x, y)$	The time cost of the worker w's moving from x to y

D. Hardness of DA-SC Problem

In this subsection, we prove that our DA-SC problem is NP-hard, by reducing a well-known NP-hard problem, the subset-sum-optimization (SSO) problem [10], to our DA-SC problem.

Theorem II.1. *(Hardness of the DA-SC problem) The DA-SC problem is NP-hard.*

Proof. We prove the theorem by a reduction from the SSO problem [10], which can be described as follows: Given a set of positive integers, $S = \{x_1, \cdots, x_n\}$, and a positive integer t, the SSO problem aims to find a subset $\overline{S}$ of S whose sum N is as large as possible but no greater than t.

For a given SSO problem, we can transform it to an instance of DA-SC problem as following: Generate n set of tasks $S = \{s_1, s_2, \cdots, s_n\}$ where each set s_i contains x_i tasks. Each task t_i in each set s_j in S requires a worker who has skill Φ_1 to conduct it. There is another n set of tasks $DS = \{ds_1, ds_2, \cdots, ds_n\}$ where each set ds_i contains D tasks. D is a large enough positive integer constant. The dependency of each task in ds_i in DS is s_i in S. Each task t_i in each set ds_j in DS requires a worker who has skill Φ_2 to conduct it. Besides, there are t workers who only have skill Φ_1 and enough workers who only have skill Φ_2. Suppose there are several sets s_i in S can be completely assigned and the number of tasks in these sets is N'. Thus, the objective of the transformed DA-SC problem instance can be represented by maximize the number of tasks $\overline{N} = N' \cdot (D + 1) + (t - N') = N' \cdot D + t$. Since D and t are constant, to maximize $\overline{N}$, we have to maximize N'. Thus, the maximum number of assigned worker-and-task pairs in the transformed DA-SC problem instance is the same as to maximize the sum of subsets in SSO problem. Given this mapping, it is easy to show that the SSO problem instance can be solved if and only if the transformed DA-SC problem instance can be solved. This way, we can reduce the SSO problem to the DA-SC problem. Since the SSO is known to be NP-hard [10], DA-SC is also NP-hard, which completes our proof. $\square$

Thus, due to the NP-hardness of the DA-SC problem, in the subsequent sections, we propose two approximate algorithms, namely DASC_Greedy and DASC_Game approaches, to efficiently solve DA-SC problem with proven approximation ratios for each batch process. Specifically, the spatial crowdsourcing platforms assign workers to tasks batch-by-batch for every constant time interval (e.g., 5 seconds). In each batch process, the server applies our approximate algorithms to assign workers to tasks under the constraints of dependencies, skills, moving distances and deadlines such that the number of finished tasks is maximized in the current batch process. Table III summarizes the commonly used symbols.

III. Greedy Approach

In this section, we propose a greedy approach to quickly achieve bounded results. We combine each task and its dependent tasks as an associative task set and iteratively assign the biggest associative task set to workers.

A. Associative Task Set

We denote task t_i and its dependent tasks as one associative task set tc_i (i.e., $tc_i = \{t_i\} \cup D_i$). For instance, in Example 1 task t_4 and its dependent task t_5 group into an associative task set $\{t_4, t_5\}$. In *DASC_Greedy*, every time we will assign a set of workers to an associative task set tc_i such that the tasks in tc_i can be fully finished by the assigned workers. Note that after assigning one associative task set tc_i, we will update other related associative task sets which contain any tasks in tc_i. In particular, we will remove these tasks in tc_i from these related associative task sets. For instance, in Example 1 there are five associative task sets $\{\{t_1\}, \{t_1, t_2\}, \{t_1, t_2, t_3\}, \{t_4\}, \{t_4, t_5\}\}$. If we select the associative task set $\{t_1\}$ to work firstly, the rest associative task sets will be updated to $\{t_2\}, \{t_2, t_3\}, \{t_4\}, \{t_4, t_5\}\}$.

B. DASC_Greedy Algorithm

Algorithm 1 shows the pseudo-code of our *DASC_Greedy* Algorithm, where we greedily select the associative task set with the largest size in each iteration.

Initially, we generate associative task sets TC_b (line 1). Then, in each iteration, we select one associative task set which can be accomplished by active workers and has the largest size (line 2-10). Specifically, for each associative task set tc in TC_b, we try to find a set of workers tw who can accomplish tc by the Hungarian Algorithm [11] (line 5). Then, if the worker set tw can be found, add $\langle tw, tc\rangle$ into the candidate set C (line6-7). After that, we move the assignment whose associative task set has the largest size from the candidate set C to M_b (line 8). Next, we update the associative task sets in TC_b and the worker set W_b (line 9).

C. Theoretic Analyses

We first discuss the running time of DASC_Greedy.

Lemma III.1. *The time complexity of* DASC_Greedy *is* $O(\min\{n_b, m_b\} \cdot m_b)$.

Proof. The loop will be ended until there is no associative task set can be assigned. Thus, the number of iterations (lines 2-10) can be bounded by $O(\min\{n_b, m_b\})$. In each iteration, there are $O(m_b)$ associative task sets should be scanned (line 4). Since the number of candidate workers of each associative task set and the number of tasks in each associative task set are both negligible compared with m_b and n_b, the time complexity of DASC_Greedy is $O(\min\{n_b, m_b\} \cdot m_b)$. □

Algorithm 1: DASC_Greedy Algorithm

```
Input: A set $W_b$ of $n_b$ workers and a set $T_b$ of $m_b$ tasks in
       the batch $b$
Output: An assignment $M_b$ in the batch $b$
1  generate the associative task sets $TC_b$;
2  repeat
3      $C = \emptyset$;
4      foreach associative task set $tc \in TC_b$ do
5          run the Hungarian Algorithm [11] to find a set of
             workers $tw$ who can accomplish the associative
             task set $tc$;
6          if $tw \neq \emptyset$ then
7              add the assignment $\langle tw, tc\rangle$ into $C$;
8          end
9      end
10     move the assignment $\langle tw, tc\rangle$ whose associative task
         set $tc$ has the largest size from $C$ to $M_b$;
11     update $TC_b$ and $W_b$;
12 until no more associative task set can be assigned
13 return $M_b$
```

This lemma shows that DASC_Greedy can get the results within polynomial time. Next, we discuss how good the results are. Firstly, we prove that the objective function of the DA-SC problem in Equation 1 is monotone and submodular.

Theorem III.1. *$Sum(M)$ is monotone and submodular.*

Proof. Let $|tc|$ be the size of the associative task set tc. Then, the objective function (as shown in Equation 1) can be rewritten to $Sum(M) = \sum_{w \in W, t \in T} I(w,t) = \sum_{tc \in M} |tc|$. Since $|tc|$ is always positive, $Sum(M)$ is monotone.

To proving its submodularity, we have:

$$\forall \langle tw_i, tc_i\rangle \notin M, \forall \langle tw_j, tc_j\rangle \notin M,$$
$$Sum(\widehat{M}) - Sum(\widetilde{M}) \leq Sum(\overline{M}) - Sum(M) \quad (2)$$

where $\widehat{M} = M \cup \{\langle tw_i, tc_i\rangle, \langle tw_j, tc_j\rangle\}$, $\widetilde{M} = M \cup \{\langle tw_i, tc_i\rangle\}$ and $\overline{M} = M \cup \{\langle tw_j, tc_j\rangle\}$. Here tw is the set of workers to conduct the associative task set tc.

There are two cases:

- $\langle tw_i, tc_i\rangle$ is assigned in $\widehat{M}$ and $\widetilde{M}$, but the size of tc_j in $\widehat{M}$ is not larger than that in $\overline{M}$. When we update the associative task sets TC after assigning the associative task set tc_i, we will remove the tasks in tc_i from the related associative task sets of tc_i. The size of the associative task set tc_j will not increase after assigning another associative task set.
- The worker set $\widehat{W_b}$ is a subset of the worker set $\overline{W_b}$, where $\widehat{W_b}$ ($\overline{W_b}$) is the active worker set after (before) finishing the associative task set tc_i (i.e., $\widehat{W_b} \subseteq \overline{W_b}$). If tc_j can be finished by the workers in $\overline{W_b}$ but cannot be finished by the workers in $\widehat{W_b}$, tc_j will not be assigned in $\widehat{M}$. Then, $Sum(\widehat{M}) - Sum(\widetilde{M}) = 0$, but $Sum(\overline{M}) - Sum(M) = |tc_j|$.

Thus, Equation 2 has been proven. In conclusion, $Sum(M)$ is monotone and submodular. □

Since $Sum(M)$ is monotone and submodular, according to [12], we have the below theorem. For the details, please refer to Appendix A of our technical report [13].

Theorem III.2. *The matching size returned by* DASC_Greedy *Algorithm is at least* $(1 - \frac{1}{e}) \cdot |M_{opt}|$.

IV. Game Theoretic Approach

In this section, we develop a game theoretic based framework, namely DASC_Game, to further improve the results achieved by DASC_Greedy. Specifically, we model the DA-SC problem as a strategic game, where each worker corresponds to a player: his goal is to find the task that maximizes his own utility. We first introduce the general knowledge of game theory and show that the strategic game can reach an equilibrium (i.e., a local optimal result where no worker has the incentive to deviate from his/her assigned task). Then, we propose a game-theoretic approach with theory analyses on its converge speed and quality of results.

A. Game Theory

Algorithm 2 illustrates a common framework for studying the dynamic process of decision-making in a strategic game [14]. In *strategic games*, players compete with each other over the same resources to optimize their individual objective functions. Under this framework, each player always tries to choose a *strategy* that maximizes his own utility without taking the effect of his choice on the other players' objectives into consideration. The input of the framework is a strategic game, which can be formally represented by a tuple $\langle W, \{S_w\}_{w \in W}, \{U_w : \times_{w \in W} S_w\}_{w \in W} \to \mathbb{R}\rangle$ where S_q represents all the possible tasks that worker w can take during the game to optimize his function U_w. The optimization of U_w depends on w's own strategy, as well as the strategies of the other workers. In [15], Nash points that a strategic game has a pure Nash equilibrium, if there exists a specific choice of strategies $s_w \in S_w$ such that the following condition is true for all $w \in W$: $U_w(s_1, \cdots, s_w, \cdots, s_{|W|}) \le U_w(s_1, \cdots, s'_w, \cdots, s_{|W|}), \forall s'_w \in S_w$. In other words, no player has the incentive to deviate from his current strategy.

To express the objective functions of all the workers, [16] proposed a single function $\Phi : \times_{w \in W} S_w \to \mathbb{R}$, called the *potential function* in *potential games*, which constitute a special class of strategic games. Let $\overline{s_w}$ denote the set of strategies followed by all workers except w (i.e., $\overline{s_w} = \{s_1, \cdots, s_{w-1}, s_{w+1}, \cdots, s_{|W|}\}$). A potential game is *exact* if there exists a potential function Φ, such that for all s_w and all possible combinations of $\overline{s_w}$, it holds $U_w(s_w, \overline{s_w}) - U_w(s'_w, \overline{s_w}) = \Phi(s_w, \overline{s_w}) - \Phi(s'_w, \overline{s_w})$. It is proved in [16] that for potential games, the best-response framework always converges to a pure Nash equilibrium. Thus, we can use the Algorithm 2 to obtain the solution for the DA-SC problem. We propose a game-theoretic approach, namely DASC_Game, and analyze its converge speed and result's quality.

B. Game-Theoretic Algorithm

We model the DA-SC problem as a game, which is represented by a tuple $\langle W_b, S_{w w \in W_b}, \{U_w : \times_{w \in W_b} S_w \to \mathbb{R}\}_{w \in W_b}\rangle$, where W_b is the worker set in the batch b. W_b contains the workers who appear in the previous batches but still wait for assignments and the new workers who just come in this batch. Similarly, T_b denotes the task set in batch b. Specifically, each worker $w \in W_b$ has a set of strategies $S_w \in T_b$. Let $s_w \in S_w$ be a specific strategy of worker w, which represents the task in this batch that the worker w can do, and nw_{s_w} is the number of workers who also try to do the task s_w. Given s_w and the strategies $\overline{s_w}$ of the other players, the total utility $U_w(s_w, \overline{s_w})$ of w is the summation of (i) the shared utility of task s_w, and (ii) the contribution to the tasks which are dependent on s_w. The goal of each worker $w \in W$ is to find the task s_w that maximizes his own total utility as expressed by Equation 3:

Algorithm 2: Best Response Framework

```
  Input: Strategic game
         $\langle V, \{S_v\}_{v \in V}, \{C_v : \times_{u \in V} S_u\}_{v \in V} \to \mathbb{R}\rangle$
  Output: Nash equilibrium
1 Assign a random strategy to each player
2 repeat
3     foreach player $v \in V$ do
4         compute $v$'s best strategy wrt the other players' strategies
5         let $v$ follow his best strategy
6     end
7 until Nash equilibrium//no player has changed his strategy
8 return the strategy of each player
```

$$U_w(s_w, \overline{s_w}) = \begin{cases} \frac{(\alpha-1)\cdot\prod_{f \in D_{s_w}} a_f}{\alpha \cdot nw_{s_w}} + \sum_{s_w \in D_t} \frac{\prod_{f \in D_t \cup \{t\}} a_f}{\alpha \cdot |D_t| \cdot nw_{s_w}}, & D_{s_w} \neq \emptyset \\ \frac{1}{nw_{s_w}} + \sum_{s_w \in D_t} \frac{\prod_{f \in D_t \cup \{t\}} a_f}{\alpha \cdot |D_t| \cdot nw_{s_w}}, & D_{s_w} = \emptyset \end{cases} \quad (3)$$

where α is the normalization parameter. Because of the dependency constraint, each task's utility can be obtained iff its dependent tasks are assigned, i.e., $\prod_{f \in D_t} a_f = 1$. Since each task's utility is affected by the assignment of itself and the assignments of its dependent tasks, we divide the utility of each task t into two parts. The first part, namely *Utility_Self*, is the utility of the assignment of itself, whose value is $\frac{\alpha-1}{\alpha}$, and the second part, *Utility_Dependency*, is the utility of the assignments of its dependent tasks, whose value is $\frac{\alpha-1}{\alpha}$. And, the task t has $|D_t|$ dependent tasks, thus, we divide the *Utility_Dependency* into $|D_t|$ shares and add each share to each dependent task. In other words, each task's utility is formed by two parts, the first part is the left utility of itself and the extra utilities from the tasks whose dependency sets contain this task. And, since each task may be assigned to multiple workers in the *Best Response* procedure, we divide the utility of the task t into nw_t shares where nw_t is the number of workers who try to do the same task t, and each worker who tries to do this task has one share of the utility. Hence, in Equation 3, the first term is the shared *Utility_Self* and the second term is the sum of the shared *Utility_Dependency* from the tasks whose dependency sets contain this task s_w. For a worker w conducting task t, *Utility_Self* represents his/her own utility on conducting task t and *Utility_Dependency* indicates the potential future utility of conducting the tasks depending on task t. Thus, for a reasonable worker w, maximizing his/her total utility $U_w(s_w, \bar{s_w})$ is his/her rational choice.

Significantly, we have the observation that the objective function of DA-SC problem in Equation 1 is equal to the

```
Algorithm 3: DASC_Game Algorithm
Input: A set $W_b$ of $n_b$ workers and a set $T_b$ of $m_b$ tasks in
       the batch $b$
Output: An assignment $M_b$ in the batch $b$
1  foreach each worker $w \in W$ do
2      assign $w$ with a random task $t \in S_w$;
3  end
4  repeat
5      foreach worker $w \in W$ do
6          $maxUtility = -\infty$;
7          foreach task $t \in S_w$ do
8              compute worker $w$'s individual utility function
               $U_w(s_w, \overline{s_w})$;
9              if $U_w(s_w, \overline{s_w}) > maxUtility$ then
10                 $maxUtility = U_w(s_w, \overline{s_w})$;
11                 $s_w = t$;
12             end
13         end
14     end
15 until Nash equilibrium
16 According to the Nash equilibrium, get the assignment set
   $I$;
17 return $M_b$
```

summation of all individual worker's utility (i.e., $Sum(M) = \sum_{w \in W} U_w(s_w, \overline{s_w})$). This decomposition of the DA-SC objective into the summation of individual utility functions provides a natural motivation for modeling DA-SC as a game. And, we define the potential function $\Phi(S)$ as follows:

$$\begin{aligned}\Phi(S) =& - \sum_{w \in W_b} \sum_{\substack{t \in \cup S_w \\ \wedge t \neq s_w}} \frac{\prod_{f \in D_t \cup \{t\}} a_f}{(nw_t + 1) \cdot (n_b - nw_t)} \\ =& - \sum_{w \in W_b} \sum_{\substack{t \in \cup S_w \\ \wedge t \neq s_w}} \frac{(\alpha - 1) \cdot \prod_{f \in D_t \cup \{t\}} a_f}{\alpha \cdot (nw_t + 1) \cdot (n_b - nw_t)} \\ & - \sum_{w \in W_b} \sum_{\substack{t \in \cup S_w \\ \wedge t \neq s_w \wedge D_t = \emptyset}} \sum_{t \in D_l} \frac{\prod_{f \in D_l \cup \{l\}} a_f}{\alpha \cdot (nw_t + 1) \cdot |D_l| \cdot (n_b - nw_t)} \\ & - \sum_{w \in W_b} \sum_{\substack{t \in \cup S_w \wedge \\ t \neq s_w \wedge D_t = \emptyset}} \frac{\prod_{f \in D_t \cup \{t\}} a_f}{\alpha \cdot (nw_t + 1) \cdot (n_b - nw_t)}\end{aligned}$$

We divide the utility of task t by $(nw_t{+}1){\cdot}(n_b{-}nw_t)$ shares and the utility of task t can be obtained iff t is assigned and its dependent tasks are assigned, i.e., $\prod_{f \in D_t \cup \{t\}} a_f = 1$. And, for each worker $w \in W_b$, we calculate the sum of one utility share of each task t in the strategy universe $\cup S_w$ except the task that the worker w selects to conduct, i.e., s_w. Then, we define the potential function $\Phi(S)$ as the negative of the sum of these value. Recall that, each task's value can be separated into two parts, thus, we can represent the potential function $\Phi(S)$ by three terms. The first term is the sum of the shared *Utility_Self*, the second term is the sum of the shared *Utility_Dependency* and the third term is the sum of the left *Utility_Self* for the tasks whose dependency sets are empty.

Algorithm 3 shows the pseudo-code of our *DASC_Game* Algorithm. Initially, *DASC_Game* assigns each worker with a random task t from worker's possible task set S_w (line 1 – 2). Then, it starts the best-response procedure (lines 3 – 11). Specifically, in each iteration for each worker w, it computes the utility of assigning worker w with each possible tasks, then assigns the task associated with the maximum utility to the worker w. The iteration terminates when no worker changes his/her assigned task during a round. Then, according to the selected task of each worker, we can get the assignment set M_b (line 12). Note that, we finally remove the assignments of that the tasks whose dependencies are not fully satisfied. And for the task which more than one worker want to do, we randomly select one worker to the task. And, the performance of *DASC_Game* can be improved by some heuristics. Specifically, in Line 2, instead of a random initialization, we can run the *DASC_Greedy* for the initialization. We examine the effect of this heuristic in the experimental evaluation (Section V).

C. Theoretic Analyses

We first prove that DA-SC game is an exact potential game.

Theorem IV.1. *DA-SC problem constitutes an exact potential game.*

Proof. As shown in Section IV-A, it suffices to have that for every worker w, who changes his strategy from the current one s_w to his/her best-response s'_w, and for all possible combinations of the other players' strategies $\overline{s_w}$ it holds that:

$$U_w(s_w, \overline{s_w}) - U_w(s'_w, \overline{s_w}) = \Phi(s_w, \overline{s_w}) - \Phi(s'_w, \overline{s_w})$$

Suppose nw_{s_w} and $nw_{s'_w}$ are the numbers of workers who are assigned to the tasks s_w and s'_w in the assignment $(s_w, \overline{s_w})$, respectively. Similarly, $\overline{nw_{s_w}}$ and $\overline{nw_{s'_w}}$ are the numbers of workers who are assigned to the tasks s_w and s'_w in the assignment $(s'_w, \overline{s_w})$, respectively. Note that, $nw_{s_w} = \overline{nw_{s_w}} + 1$ and $\overline{nw_{s'_w}} = nw_{s'_w} + 1$. Indeed, when $D_{s_w} \neq \emptyset$ and $D_{s'_w} \neq \emptyset$, we have:

$$\begin{aligned}&\Phi(s_w, \overline{s_w}) - \Phi(s'_w, \overline{s_w}) \\ =& -\Big(\frac{(\alpha - 1) \cdot \prod_{f \in D_{s'_w}} a_f}{\alpha \cdot (nw_{s'_w} + 1)} + \sum_{s'_w \in D_l} \frac{\prod_{f \in D_l \cup \{l\}} a_f}{\alpha \cdot (nw_{s'_w} + 1) \cdot |D_l|}\Big) \\ & + \Big(\frac{(\alpha - 1) \cdot \prod_{f \in D_{s_w}} a_f}{\alpha \cdot (\overline{nw_{s_w}} + 1)} + \sum_{s_w \in D_l} \frac{\prod_{f \in D_l \cup \{l\}} a_f}{\alpha \cdot (\overline{nw_{s_w}} + 1) \cdot |D_{s_w}|}\Big) \\ =& \Big(\frac{(\alpha - 1) \cdot \prod_{f \in D_{s_w}} a_f}{\alpha \cdot nw_{s_w}} + \sum_{s_w \in D_l} \frac{\prod_{f \in D_l \cup \{l\}} a_f}{\alpha \cdot nw_{s_w} \cdot |D_{s_w}|}\Big) \\ & - \Big(\frac{(\alpha - 1) \cdot \prod_{f \in D_{s'_w}} a_f}{\alpha \cdot \overline{nw_{s'_w}}} + \sum_{s'_w \in D_l} \frac{\prod_{f \in D_l \cup \{l\}} a_f}{\alpha \cdot \overline{nw_{s'_w}} \cdot |D_{s'_w}|}\Big) \\ =& U_w(s_w, \overline{s_w}) - U_w(s'_w, \overline{s_w})\end{aligned}$$

Similarly, when (1) $D_{s_w} = \emptyset$ and $D_{s'_w} \neq \emptyset$; (2) $D_{s_w} \neq \emptyset$ and $D_{s'_w} = \emptyset$; (3) $D_{s_w} = \emptyset$ and $D_{s'_w} = \emptyset$, we can have the same result: $\Phi(s_w, \overline{s_w}) - \Phi(s'_w, \overline{s_w}) = U_w(s_w, \overline{s_w}) - U_w(s'_w, \overline{s_w})$. Due to the space limitation, we do not show the details here. For the full proof, please refer to Appendix B of our technical report [13]. □

Since DA-SC problem is an exact potential game, and the set of strategic configurations S is finite, a Nash equilibrium can be reached after workers changing their strategies a finite number of times. For simplicity, we prove the upper bound for the number of rounds required to reach the convergence of DASC_Game Algorithm by a scaled version of the problem where the objective function takes integer values. Specifically, we assume an equivalent game with potential function $\Phi_{\mathbb{Z}}(S) = d{\cdot}\Phi(S)$, where d is a constant which depends on the

nw_{s_w} such that $\Phi_{\mathbb{Z}}(S) \in \mathbb{Z}, \forall S$. This does not scale with the size of the problem. Then, we can prove the following lemma.

Lemma IV.1. *The number of rounds required by* DASC_Game *Algorithm in each batch to converge to an equilibrium is upper bounded by* $\mathcal{O}(d \cdot \min\{n_b, m_b\})$.

Proof. The scaled version of *DASC_Game* Algorithm with the potential function as $\Phi_{\mathbb{Z}}(S) = d \cdot \Phi(S)$ will converge to a Nash equilibrium in the same number of rounds as *DASC_Game* Algorithm. Since $\Phi_{\mathbb{Z}} \in \mathbb{Z}$, the cost increase in $\Phi_{\mathbb{Z}}$ after each strategy change of a worker is at least 1. Thus, the upper bound of the number of rounds is the maximum value, Φ_{max}, minus the minimum value, Φ_{min}. Since $\Phi_{max} \le 0$ and $\Phi_{min} \ge -d \cdot \min\{n_b, m_b\}$, the number of rounds to reach an equilibrium is upper-bounded by $d \cdot \min\{n_b, m_b\}$. □

Then, we prove the algorithm's time complexity as follows.

Lemma IV.2. *The time complexity of* DASC_Game *in batch* b *is* $O(d \cdot n_b \cdot \min\{n_b, m_b\})$.

Proof. As proved, the number of iteration is upper bounded by $O(d \cdot \min\{n_b, m_b\})$. And, we need to run the best response for n_b workers. For each worker w, the best response requires computing the individual utility function for at most $|S|_{max}$ possible tasks (line 4-6). Furthermore, the time cost of computing individual utility function is $O(|D|_{max})$ (line 7). However, compared with n_b and m_b, $|S|_{max}$ and $|D|_{max}$ are negligible. Thus, the time complexity is $O(d \cdot n_b \cdot \min\{n_b, m_b\})$. □

After proving that DASC_Game can converge within polynomial time, we discuss how good the resulting solution is. Usually, researchers use social optimum (OPT), price of stability (PoS), and price of anarchy (PoA) to evaluate the quality of an equilibrium [14]. A game's OPT is the solution that yields the optimal values to all the objective functions so that their total utility is maximum. The PoS of a game is the ratio between the best value among its equilibrium and the global optimum and the PoA of a game is the ratio between the worst value among its equilibriums and the global optimum.

Let $U(S)$ denotes the summation of all workers' utility, $U(S) \triangleq \sum_{w \in W_b} U_w(s_w, \overline{s_w})$, and recall that $U(S)$ is equal to the *DA-SC* object function. Then, we have the following lemma to bound the potential function $\Phi(S)$.

Lemma IV.3. *The potential function* $\Phi(S)$ *is bounded as* $\frac{1}{nw_{max}+1} \cdot U(S) \le |\Phi(S)| \le \frac{n_b}{\overline{nw} \cdot (n_b - \overline{nw})} \cdot U(S)$*, where* $\overline{nw} = \min\{nw_{min}, n_b - nw_{max}\}$*,* nw_{max} *is the maximum number of workers who are assigned to the same task and* nw_{min} *is the minimum number of workers who are assigned to the same task.*

Proof. Since $nw_{max} \ge nw_t, \forall t \in \cup S_w$, we have

$$\sum_{w \in W_b} \sum_{\substack{t \in \cup S_w \\ \wedge t \neq s_w}} \frac{\prod_{f \in D_t \cup \{t\}} a_f}{(nw_t + 1) \cdot (n_b - nw_t)}$$

$$= \sum_{t \in \cup S_w} \frac{\prod_{f \in D_t \cup \{t\}} a_f}{nw_t + 1} \ge \frac{1}{nw_{max} + 1} \cdot \sum_{t \in \cup S_w} \prod_{f \in D_t \cup \{t\}} a_f$$

Thus, $|\Phi(S)| \ge \frac{1}{nw_{max}+1} \cdot U(S)$. And, we have:

$$\sum_{w \in W_b} \sum_{\substack{t \in \cup S_w \\ \wedge t \neq s_w}} \frac{\prod_{f \in D_t \cup \{t\}} a_f}{(nw_t + 1) \cdot (n_b - nw_t)} \le \sum_{w \in W_b} \sum_{\substack{t \in \cup S_w \\ \wedge t \neq s_w}} \frac{\prod_{f \in D_t \cup \{t\}} a_f}{nw_t \cdot (n_b - nw_t)}$$

$$\le \sum_{w \in W_b} \sum_{t \in \cup S_w} \frac{\prod_{f \in D_t \cup \{t\}} a_f}{nw_t \cdot (n_b - nw_t)} = \sum_{t \in \cup S_w} \frac{n_b \cdot \prod_{f \in D_t \cup \{t\}} a_f}{nw_t \cdot (n_b - nw_t)}$$

$$\le \frac{n_b}{\overline{nw} \cdot (n_b - \overline{nw})} \cdot \sum_{t \in \cup S_w} \prod_{f \in D_t \cup \{t\}} a_f$$

Thus, we have $|\Phi(S)| \le \frac{n_b}{\overline{nw} \cdot (n_b - \overline{nw})} \cdot U(S)$. □

Based on the potential function's bounds, we can prove the bounds of the equilibrium obtained by *DASC_Game*.

Theorem IV.2. *In* DASC_Game*, the PoS of each batch is bounded by* $\frac{\overline{nw} \cdot (n_b - \overline{nw})}{n_b \cdot (nw_{max}+1)}$*. And, the PoA of each batch is bounded by* $\frac{\overline{nw} \cdot (n_b - \overline{nw})}{n_b \cdot \min\{n_b, m_b\}} \cdot \widehat{|\Phi(S)|}_{min}$*, where* $\widehat{|\Phi(S)|}_{min}$ *is the minimum of* $\Phi(S)$*'s local minimums.*

Proof. Let S^* be the optimal set of strategies in this batch that maximize $U(S)$, and let $OPT = U(S^*)$. Further, let S^{**} be the set of strategies that yields the minimum of the potential function $\Phi(S)$, i.e., the best Nash equilibrium. From the Lemma IV.3 and since $U(S^*) \ge U(S), \forall S$ and $|\Phi(S^{**})| \ge |\Phi(S)|, \forall S$, we have:

$$U(S^{**}) \ge \frac{\overline{nw} \cdot (n_p - \overline{nw})}{n_p} \cdot |\Phi(S^{**})| \ge \frac{\overline{nw} \cdot (n_b - \overline{nw})}{n_b} \cdot |\Phi(S^*)|$$

$$\ge \frac{\overline{nw} \cdot (n_b - \overline{nw})}{n_b \cdot (nw_{max} + 1)} \cdot U(S^*)$$

Since $OPT = U(S^*)$, $PoS = \frac{U(S^{**})}{OPT} \ge \frac{\overline{nw} \cdot (n_b - \overline{nw})}{n_b \cdot (nw_{max}+1)}$.

Next, let $S^{\#}$ be the strategic configuration of any Nash equilibrium obtained by *DASC_Game*. Thus, we have:

$$U_w(s_w^{\#}, \overline{s_w^{\#}}) \ge U_w(s'_w, \overline{s_w^{\#}}), \forall w \in W, \forall s'_w \in S_w$$

That is, $\forall s'_w \in S_w$, $\Phi(s_w^{\#}, \overline{s_w^{\#}}) - \Phi(s'_w, \overline{s_w^{\#}}) = U_w(s_w^{\#}, \overline{s_w^{\#}}) - U_w(s'_w, \overline{s_w^{\#}}) \ge 0$. Hence, $\Phi(s_w^{\#}, \overline{s_w^{\#}})$ is a local maximum and $|\Phi(s_w^{\#}, \overline{s_w^{\#}})|$ is a local minimum. Thus, we have:

$$U(S^{\#}) \ge \frac{\overline{nw} \cdot (n_b - \overline{nw})}{n_b} \cdot |\Phi(S^{\#})| \ge \frac{\overline{nw} \cdot (n_b - \overline{nw})}{n_b} \cdot \widehat{|\Phi(S)|}_{min}$$

Since $U(S^*) \le \min\{n_b, m_b\}$, we have:

$$PoA = \frac{U(S^{\#})}{U(S^*)} \ge \frac{\overline{nw} \cdot (n_b - \overline{nw})}{n_b \cdot \min\{n_b, m_b\}} \cdot \widehat{|\Phi(S)|}_{min}$$

□

V. Experimental Study

A. Data Sets

We use both real and synthetic data to test our proposed DA-SC approaches. Specifically, for real data, we use Meetup data set from [17], which was crawled from meetup.com between Oct. 2011 and Jan. 2012. There are 5,153,886 users, 5,183,840 events, and 97,587 groups in the Meetup data set, where each user is associated with a location and a set of tags, each group is associated with a set of tags, and each event is associated with a location and a group who creates the event. The tags of a group are treated as the tags of the events created by the group. We use the locations and tags of users in the Meetup data set to initialize the locations and the skills of workers in DA-SC, where each tag is considered as a skill. In addition, we utilize the locations and tags of events to initialize the locations and the required skills of tasks in our experiments. Each task will depend on other tasks who contain more than two common tags with it and are created before it. Since

TABLE IV: Experimental Settings on Real Data.

Parameters	Values
the start time range $[st^-, st^+]$	[0, 10], [0, 15], **[0, 20]**, [0, 25], [0, 30]
the waiting time range $[wt^-, wt^+]$	[1, 8], [1, 9], **[1, 10]**, [1, 11], [1, 12]
the velocity range $[v^-, v^+] * 0.001$	[1, 3], [1, 5], **[1, 7]**, [1, 9], [1, 11]
the distance range $[d^-, d^+] * 0.01$	[2, 2.5], [2.5, 3], **[3, 3.5]**, [3.5, 4], [4, 4.5]

workers are unlikely to move between two distant cities to conduct one spatial task, and the constraints of expired time and maximum moving distance also prevent workers from moving too far, we only consider those user-and-event pairs located in one city. Specifically, we select one famous and popular city, Hong Kong, and extract Meetup records from the area of Hong Kong (with latitude from $22.209°$ to $22.609°$ and longitude from $113.843°$ to $114.283°$).

For synthetic data, we generate locations of workers and tasks in a 2D data space $[0, 0.5]^2$, with the uniform distribution. We vary the number of workers, tasks, and the size of the skills' universe to mimic a wide scale of real-world application scenarios. And, we simulate the size of each worker's skill set with the uniform distribution within the range $[sp^-, sp^+]$ from $[1, 5]$ to $[1, 25]$. Besides, we set the size of each task's dependency with the uniform distribution within the range from $[0, 0]$ (i.e., there is no dependency between tasks) to $[0, 20]$. For each task t, we randomly add tasks generated before t and their dependency set into t's dependency set until the size of t's dependency set reaches the generated value.

For both real and synthetic data sets, we simulate the velocity of each worker with the uniform distribution within the range $[v^-, v^+]$ from $[0.001, 0.003]$ to $[0.001, 0.011]$. For the maximum moving distance of each worker, we generate it following the uniform distribution within the range $[d^-, d^+]$. For temporal constraints, we set each worker's and each task's start time and waiting time following the uniform distribution within the range $[st^-, st^+]$ and $[wt^-, wt^+]$.

B. Approaches and Measurements

In the beginning, we conduct an experiment on small scale data sets. The main purpose is to investigate the effectiveness of approximation methods regarding the optimal solution. We propose a depth-first search algorithm, namely DFS, which exactly enumerates all possible assignments to find the optimal assignment. Each level in the search tree represents a worker in the worker set and the children of one node are the valid tasks that the worker represented by the next level can do. In addition, we propose two baseline methods. The first algorithm, namely Closest, greedily selects a worker-and-task pair with the lowest moving distance for each worker. The second algorithm, namely Random, randomly selects a worker-and-task pair for each worker.

After that, we conduct experiments on large scale data sets to evaluate the effectiveness and efficiency of our two approaches, DASC_Greedy and DASC_Game, in terms of the size of the valid assignment and the running time. As proved in Section II-D, the DA-SC problem is NP-hard, thus it is infeasible to calculate the optimal result as the ground truth in large scale datasets. Alternatively, we compare our approaches with two aforementioned baseline methods (*Closest* and *Random*) as well as an existing algorithm (*g*-D&C) in a related

TABLE V: Experimental Settings on Synthetic Data.

Parameters	Values
the number, r, of skill universe	50, 60, **70**, 80, 90
the dependency size range	[0,0], [0,5], **[0, 10]**, [0, 15], [0, 20]
the skill set range for each worker	[1, 5], [1, 10], **[1, 15]**, [1, 20], [1, 25]
the number, m, of workers	3K, 4K, **5K**, 6K, 7K
the number, n, of tasks	4K, 4.5K, **5K**, 5.5K, 6K
the start time range $[st^-, st^+]$	[0, 10], [0, 15], **[0, 20]**, [0, 25], [0, 30]
the waiting time range $[wt^-, wt^+]$	[1, 10], [1, 12], **[1, 14]**, [1, 16], [1, 18]
the velocity range $[v^-, v^+] * 0.001$	[1, 7], [1, 8], **[1, 9]**, [1, 10], [1, 11]
the distance range $[d^-, d^+] * 0.01$	[30, 32], [32, 34], **[34, 36]**, [36, 38], [38, 40]

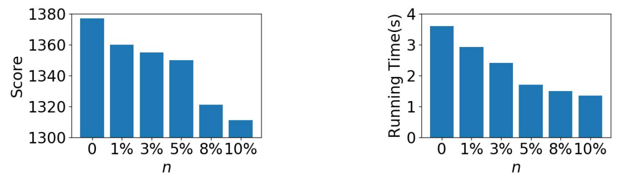
paper [6]. The *g*-D&C algorithm keeps dividing the problem into *g* subproblems on each level until finally, the number of tasks in each subproblem is 1 (which can be solved by the greedy algorithm on each one-task subproblem). Here, the parameter *g* is estimated by a cost model to minimize the computing cost. Note that, for *DASC_Game*, if we strictly set the termination condition as there is no worker changed his strategy in the last iteration, the converging speed is very slow. In practice, we usually set a threshold of the utility updating ratio instead. In other words, if the utility updating ratio in a round is lower than the threshold, we terminate the iteration in *DASC_Game*. Although with the increase of the threshold, the running time of *DASC_Game* will decrease, the score of *DASC_Game* will decrease too. Thus, we have to select a proper threshold to trade off the score and the running time.

We run an experiment on the real datasets with different termination conditions. Specifically, the parameters used in the experiment are the default values as shown in Table IV. We vary the value of the threshold from 0 (there is no worker changed his strategy in the last iteration) to 10%. As shown in Figure 2(a), when the threshold is larger than 5%, the score decreases sharply. And Figure 2(b) illustrates that, with the increase of the threshold, the running time decrease. Thus, to trade off the score and the running time, we set the threshold as 5%.

In the following experiments, we examine the *DASC_Game* with the strict termination condition, namely *Game*, and the *DASC_Game* with the threshold as 5%, namely *GT*. In addition, we examine the heuristic that using *DASC_Greedy* for *DASC_Game*'s initialization, namely *G&G*. For simplicity, we represent the *DASC_Greedy* by *Greedy*.

In addition, we tested the performance of our proposed algorithms on a real spatial crowdsourcing platform, namely gMission [18]. In particular, gMission is a general laboratory application, which records the active users' trajectories and pushes tasks to users based on their spatial locations. The users claim the skills they have in their profiles on the gMission and when they log in the gMission, they propose that when they planned to log off and do not accept the assignments. In order to hold sports meetings or sports games, some schools and companies in Shaoxing City, Zhejiang Province, China published some spatial tasks in the gMisssion. The task requesters claim each tasks' dependent tasks and the deadline of the tasks' assignments. We estimate each user's movement speed by their recent trajectories. In our experiments, we test the algorithms' performance with different tasks' waiting time range. Specifically, the tasks' waiting time range in our experiment is varied from [2,6] to [6,14].

Table IV and Table V show our experimental settings on real



(a) Score (b) Running Time

Fig. 2: Effect of the Threshold

TABLE VI: Experimental Results on Small-Scale Datasets

Algorithm	Score	Running Time (ms)
DFS	17	955514.9
G&G	17	1.9
Game	17	1.9
GT	17	1.9
Greedy	16	1.5
g-D&C	16	1.7
Closest	13	1.1
Random	12	1.4

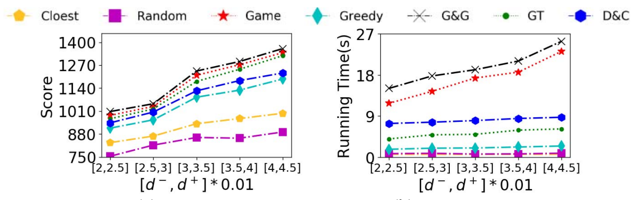
data and synthetic data, where the default values of parameters are in bold font. In each experiment, we vary one parameter and set other parameters to their default values. Since *Game*, *GT*, and *Random* algorithms use the random number, we run those algorithms for 10 times for each experiment and report the result with the average score. And for each experiment, we report the running time and the assignment score (the number of the valid assigned worker-and-task pairs) of our tested approaches. All our experiments were run on an Intel i7 CPU @2.2 GHz with 16GM RAM in Java.

C. Results on Small-Scale Datasets

We run the experiment on a small scale synthetic data sets. Specifically, we set the number of workers is 20, the number of tasks is 40, and the size of the skill universe is 10. In addition, the range of each worker's skill set's size is [1,3] and the range of each task's dependency set's size is [0, 8]. Other parameters are the default values of the experimental settings on synthetic data (Table V). Table VI shows the score and the running time of each algorithm. We can see that the scores of four proposed methods follow the bound which we proved in Section III and Section IV. The compared algorithm *g*-D&C has a similar performance with we proposed greedy algorithm *Greedy*. Besides, the two baseline algorithms' score is much worse than the two proposed methods. In addition, we can see that the two proposed methods cost very little running time while the *DFS Algorithm* spends vast amounts of time. Thus, we can draw a conclusion that *DFS Algorithm* is only suitable for small scale data sets and our proposed algorithms have good acceptances compared with the optimal results.

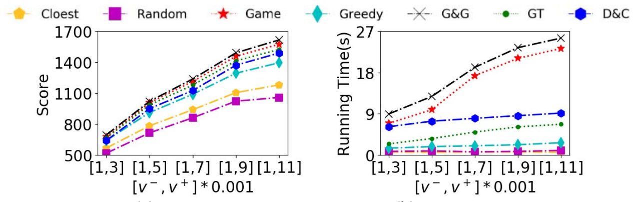
D. Results on Real Datasets

Effect of the Range of the Maximum Moving Distance $[d^-, d^+]$. Figure 3 illustrates the experimental results on different ranges, $[d^-, d^+]$, of each worker's maximum moving distance d from $[0.02, 0.025]$ to $[0.04, 0.045]$. In Figure 3(a), the assignment scores of all the seven approaches increase, when the value range of maximum moving distance gets larger. The reason is that the increase of d enlarges the access range of workers. The scores of four proposed algorithms are much better than those of two baseline algorithms. It is reasonable because the baseline algorithms do not consider



(a) Score (b) Running Time

Fig. 3: Effect of the Moving Distance Range $[d^-, d^+]$ (Real)

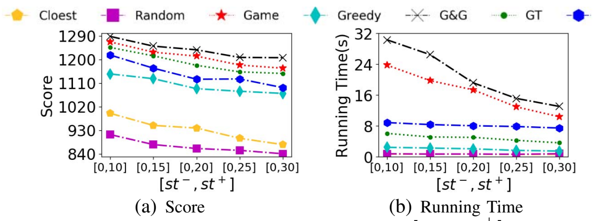


(a) Score (b) Running Time

Fig. 4: Effect of the Velocity Range $[v^-, v^+]$ (Real)

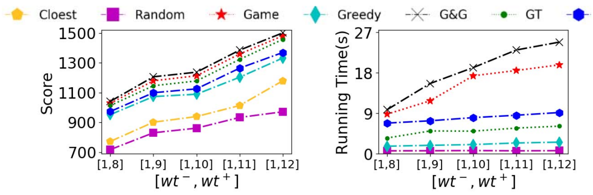
tasks' dependency constraints when they allocate tasks and many of the assignments are invalid. In addition, the *G&G*, *Game*, and *GT* achieve higher scores than the *Greedy Algorithm*. The reason is that each worker has multiple skills and for each skill, the number of workers who has the skill is different. When the *Greedy* assigns a task with a worker, it just selects a worker who has this skill and can reach the task on time. In other words, the *Greedy* does not guarantee that the assigned worker is the optimal choice for this task. Maybe the optimal choice is assigning another worker to do this task and assigning this worker to another task whose required skill can only be satisfied by this worker. However, the *G&G*, *Game*, and *GT* can avoid this case. By checking all the strategies that each worker can take, the *G&G*, *Game*, and *GT* can assign the worker to the task whose required skill is the skill that few workers have. The proposed algorithms achieve better results than the compared algorithm *g*-D&C, except the *Greedy* Algorithm. The reason is that in the Meetup dataset, the users and events which have the same tags are usually in close locations and the dependency between tasks is simply. The advantage of *g*-D&C algorithm in matching makes up the shortcoming in the dependency process. Besides, *G&G* achieves the highest score among the seven algorithms. The reason is that, compared with the random initialization in *Game* and *GT*, the *G&G* can reach a Nash equilibrium with better quality.

As shown in Figure 3(b), the running time of our four DA-SC approaches increase, when the range of maximum moving distance gets larger. The reason is that when the maximum moving distance increases, each worker has more valid tasks which thus leads to the higher complexity of the *DA-SC* problem and the increase of the running time. Specifically, *Greedy* achieves much lower running time than the *G&G*, *Game*, and *GT*. The reason is that *G&G*, *Game*, and *GT* have large search space and in each round, it searches the whole search space while in each round *Greedy* just searches an associate task set with the largest size and can be fully assigned with a group of workers. In addition, the running time of *G&G* is the highest. The reason is that, compared with the random initialization in *Game* and *GT*, its initialization



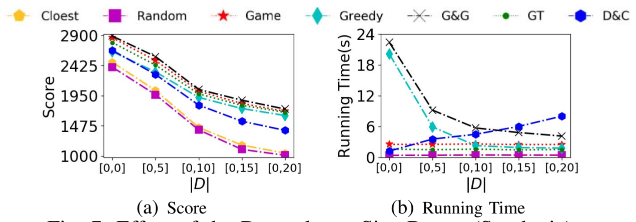
(a) Score (b) Running Time

Fig. 5: Effect of the Start Time Range $[st^-, st^+]$ (Real)



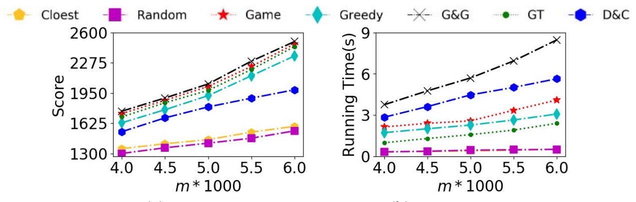
(a) Score (b) Running Time

Fig. 6: Effect of the Waiting Time Range $[wt^-, wt^+]$ (Real)



(a) Score (b) Running Time

Fig. 7: Effect of the Dependency Size Range (Synthetic)



(a) Score (b) Running Time

Fig. 8: Effect of the Number of Tasks (Synthetic)

costs much more time. Besides, the running time of ***g**-D&C* is higher than *Greedy* and *GT*. The reason is that ***g**-D&C* has to filter the invalid assignments which do not satisfy the dependency constraint after each round and repeatedly make the assignments for the left tasks and workers.

Effect of the Range of the Velocity $[v^-, v^+]$**.** Figure 4 illustrates the experimental results on different ranges, $[v^-, v^+]$, of each worker's velocity v from $[0.001, 0.003]$ to $[0.001, 0.011]$. In Figure 4(a), when the range of velocity gets larger, the assignment scores of all approaches increase. The reason is that, with the increase of v, workers can reach more tasks on time. The proposed algorithms have better results than ***g***-D&C except the Greedy. As shown in Figure 4(b), the running time of our four approaches increase, when the range of velocity gets larger. The reason is that, when the velocity increases, each worker has more valid tasks which thus leads to the higher complexity of the DA-SC problem and the increase of the running time. Specifically, G&G, Game, and GT are more sensitive. Because, when the velocity increases, each worker's strategy space is larger, which dramatically increases the algorithms' running time.

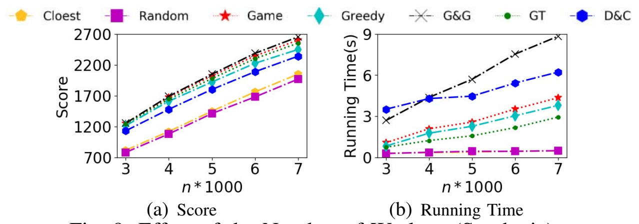
Effect of the Range of the Start Timestamp $[st^-, st^+]$**.** Figure 5 illustrates the experimental results on different ranges, $[st^-, st^+]$, of each worker's/task's start timestamp st from $[0, 10]$ to $[0, 30]$. In Figure 5(a), the assignment scores of all approaches decrease, when the value range of start timestamp gets larger. The reason is that the increase of st's range disperses the tasks/workers over time and thus each task has less valid workers who can reach the task timely. *G&G* achieves the highest score and the scores of four proposed algorithms are much better than those of two baseline algorithms. The proposed algorithms also have better results than ***g**-D&C* except *Greedy*. As shown in Figure 5(b), the running time of our seven approaches decrease, when the range of start timestamp gets larger. The reason is that, when the range increase, each worker has fewer valid tasks which thus leads to lower complexity of the *DA-SC* problem and the decrease of the running time. Specifically, *G&G*, *Game*, and *GT* are more sensitive. The reason is that, when the range increases, each worker's strategy space is smaller, which dramatically decreases the algorithms' running time.

Effect of the Range of the Waiting Time $[wt^-, wt^+]$**.** Figure 6 illustrates the experimental results on different ranges, $[wt^-, wt^+]$, of each worker's/task's waiting time wt from $[1, 8]$ to $[1, 12]$. In Figure 6(a), the assignment scores of all approaches increase, when the value range of waiting time gets larger. The reason is that the increase of wt let each worker can reach more tasks timely. G&G achieves the highest score and the scores of four proposed algorithms are much better than those of two baseline algorithms. The proposed algorithms also have better results than ***g***-D&C except the Greedy. As shown in Figure 6(b), the running time of our four approaches increase, when the range of waiting time gets larger. Because, when the waiting time increase, each worker has more valid tasks which thus leads to the higher complexity of the DA-SC problem and the increase of the running time.

E. Results on Synthetic Datasets

To examine the effects of the size of the skill universe, the number of workers, the number of tasks, the size of each task's dependency set, and the size of each worker's skill set, we generate the synthetic dataset and run the experiments on it. We also test the effects of the number of skill universe, the skill set range of workers, the start time range, the waiting time range, the velocity range and the distance range on the synthetic data sets. Due to the space limitation, please refer to Appendix C of our technical report [13] for details.

Effect of the range of dependency set size, $|D|$**.** Figure 7 illustrates the experimental results on different ranges, of each task's dependency set size $|D|$ from $[0, 0]$ (i.e., tasks are all independent) to $[0, 20]$. In Figure 7(a), the assignment scores of all the seven approaches decrease, when the value range of each task's dependency gets larger. The reason is that the increase of $|D|$ let the tasks' dependency constraints are more difficult to satisfy. Specifically, the two baseline algorithms and the compared algorithm ***g**-D&C* are more sensitive to the increase and their score decrease sharply with the increase of $|D|$. The reason is that, with the increase of $|D|$, the tasks' dependency constraints are more difficult to satisfy and the assignments made by the two baseline algorithms and



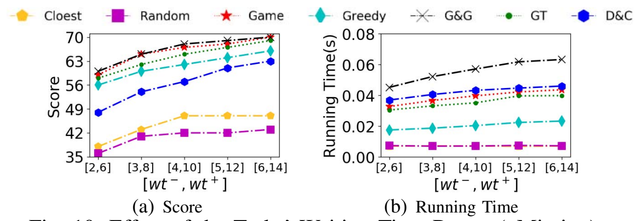
(a) Score (b) Running Time

Fig. 9: Effect of the Number of Workers (Synthetic)

the compared algorithm ***g**-D&C* which do not consider the dependency constraints are more likely be invalid. As shown in Figure 7(b), the running time of *Game* and *GT* keeps stable when the value range of each task's dependency gets larger, because the change of dependency's size does not influence the search space. Because the time cost of generating and updating associative task sets in *Greedy* and *G&G* is only related to the task number and when the dependency size is small, the number of associative task sets decrease slowly during the algorithm's running, when the dependency size is small, the running time of *G&G* and *Greedy* is high. But when the dependency size is bigger, their running time decreases sharply. When the dependency size grows, the dependency constraints are more difficult to satisfy and ***g**-D&C* has to run more rounds, which leads to that its running time grows constantly.

Effect of the Number of Tasks m. Figure 8 illustrates the experimental results on the different number, m, of tasks from $4K$ to $6K$. In Figure 8(a), when the number of tasks increases, the assignment scores of seven approaches increase. The reason is that the increase in m let each worker has more valid tasks. But the two baseline algorithms and the compared algorithm ***g***-D&C are not as sensitive to the change of the number of tasks as our proposed algorithms do. The reason is that since the two baseline algorithms and the ***g***-D&C do not consider the dependency constraint when they allocate the tasks, the increase of the number of tasks that each worker can do also decreases the probability that the assigned task's dependency constraint is satisfied. As shown in Figure 8(b), the running time of our four approaches increases, when the number of tasks increases. Because, when the number of tasks increases, each worker has more valid tasks which thus leads to the higher complexity of the DA-SC problem and the increase of the running time.

Effect of the Number of Workers n. Figure 9 illustrates the experimental results on the different number, n, of workers from $3K$ to $7K$. In Figure 9(a), the assignment scores of all the seven approaches increase, when the number of workers gets larger. The reason is that the increase of n let each task has more valid workers. As shown in Figure 9(b), the running time of our four approaches increases, when the number of workers gets larger. Since the ***g**-D&C*'s running time is mainly dominated by the rounds that the ***g**-D&C* runs which are related to the tasks' dependency sizes, the ***g**-D&C*'s running time is not as sensitive as our proposed algorithms and its running time is higher than the running time of our algorithms when the number of workers is small.



(a) Score (b) Running Time

Fig. 10: Effect of the Tasks' Waiting Time Range (gMission)

F. Results on gMission Platfrom

Figure 10 illustrates the tested algorithms' performance over the real gMission platform on different ranges, $[wt^-, wt^+]$, of each task's waiting time wt from $[2, 6]$ to $[6, 14]$ minutes. In this experiment, during the 15 minutes, 71 workers login the platform and 167 tasks are proposed. In addition, the workers' waiting time is in the range of 1 to 4 minutes and their estimated velocity is in range of 5 km/h to 67 km/h. Besides, the workers' maximum moving distance is in the range of 2 km to 8 km and the tasks' dependency size is in the range of 0 to 4.

In Figure 10(a), the assignment scores of all tested approaches increase, when the value range of waiting time gets larger. The reason is that the increase of wt enable that each worker can reach more tasks timely. Specifically, *G&G* achieves the highest score and the scores of four proposed algorithms are much better than those of Closest, Random, and ***g**-D&C*. Since the dataset is small, there is no big difference between the results of *G&G* and *Game*. Since the *GT* is terminated before reaching an equilibrium, its result is slightly worse than the *Game*. As shown in Figure 10(b), the running time of our four approaches increases, when the range of waiting time gets larger. The reason is that, when the waiting time increases, each worker has more valid tasks which thus leads to the higher complexity of the *DA-SC* problem and the increase of the running time.

We finally summarize our findings.

- Our four approximate algorithms (*G&G*, *Game*, *GT*, and *Greedy*) can achieve results with much more valid worker-and-task pairs compared with that of baseline algorithms and ***g**-D&C* in all scenarios.
- The heuristic of utilizing *DASC_Greedy* to initialize *DASC_Game* is effective while it costs much time.
- Greedy runs much faster than the other three approximate algorithms while it can achieve similar scores.

VI. Related Work

Crowdsourcing has attracted considerable attention due to its high practicality for real-world applications. Without considering the location constraint, previous work [19], [20], [21], [22] studied the task assignment in crowdsourcing to finish the tasks more efficiently and accurately.

In spatial crowdsourcing [23], workers are requested to physically move to specific locations to conduct tasks on sites. Based on the ontology [24], from the perspective of the publishing models, task assignment in spatial crowdsourcing can be classified into two groups: worker selected tasks (WST) mode and server assigned tasks (SAT) mode. Specifically, for the WST mode, spatial tasks are broadcast to all the workers or group of close workers, then the workers select preferred tasks by themselves. In prior work [25] in task assignment in SAT mode, the author proposed methods to design a travel route for each worker such that the worker can finish as many tasks as possible before the deadlines. In the contrast, in WST mode, the workers reveal their real time locations to the server, then the server will assign the suitable tasks to workers. Since the server has the control of the task assignment in WST mode, it is more convenient to optimize the targeted goal and many existing studies [1], [24], [26], [7], [27], [28] in task assignment in spatial crowdsourcing are using the WST mode.

In particular, Kazemi et al. [1] studied the problem of maximizing the number of assigned problem under the constraint of the working areas and the capacities of workers and the deadlines of tasks. With considering the reliability of workers, Cheng et al. [26] tackle the problem of reliable diversity-based spatial crowdsourcing problem, which assigns a set of workers to each task such that the spatial/temporal diversity and the reliability score of the answers to the task is optimized. Tong et al. [27] studied the online task assignment for spatial crowdsourcing, in which the server needs to assign the most suitable worker to each task when the workers and tasks are coming one-by-one and the platform has no information about the future tasks or workers. Previous studies on multi-skill oriented spatial crowdsourcing [6], [7] tackle the problem through finding a set of workers and the union of their skill sets can fully support the requirement of the assigned complex task. However, the existing methods do not consider the dependencies between the subtasks such that some assigned workers need to wait until their subtasks are ready to conduct, which makes that the existing methods are not efficient. In this paper, we take the dependencies between tasks into consideration and propose tailored approximation algorithms to efficiently and effectively solve the DA-SC problems with theory bounds of the number of the assigned worker-and-task pairs for each batch process, which had not been studied before.

VII. Conclusion

In this paper, we studied the *Dependnecy-Aware Spatial Crowdsourcing* (DA-SC) problem, a new problem of batch-based task allocation in spatial crowdsourcing, where tasks have been allocated following the dependency constraints. We proposed two approximation algorithms, including greedy and game-theoretic approaches. Specifically, we defined the *task combination* and design a submodule function in the greedy approach, which greedily allocates a task combination with the largest size and guarantees the approximate bounds of the results in each batch process. In addition, we propose a game-theoretic approach to further increase the number of the assigned worker-and-tasks. We conclude that our proposed two solutions are effective and efficient in extensive experiments on both real and synthetic data sets.

References

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

[2] "[online] Uber." https://www.uber.com, 2019.

[3] "[online] Waze." https://www.waze.com/, 2019.

[4] "[online] Google Maps' Street View." https://www.google.com/maps/views/streetview, 2019.

[5] "[online] Uber Eats." https://www.ubereats.com, 2019.

[6] P. Cheng, X. Lian, *et al.*, "Task assignment on multi-skill oriented spatial crowdsourcing," *IEEE TKDE*, vol. 28, no. 8, pp. 2201–2215, 2016.

[7] D. Gao, Y. Tong, J. She, T. Song, L. Chen, and K. Xu, "Top-k team recommendation and its variants in spatial crowdsourcing," *Data Science and Engineering*, vol. 2, no. 2, pp. 136–150, 2017.

[8] H. Rahman, S. Thirumuruganathan, *et al.*, "Worker skill estimation in team-based tasks," *PVLDB*, vol. 8, no. 11, pp. 1142–1153, 2015.

[9] "[online]technical officials in basketball game." https://australia.basketball/technical-officials/.

[10] D. S. Johnson, "Approximation algorithms for combinatorial problems," *Journal of computer and system sciences*, vol. 9, no. 3, pp. 256–278, 1974.

[11] H. W. Kuhn, "The hungarian method for the assignment problem," *Naval research logistics quarterly*, vol. 2, no. 1-2, pp. 83–97, 1955.

[12] S. Khuller, A. Moss, and J. S. Naor, "The budgeted maximum coverage problem," *Information processing letters*, vol. 70, no. 1, pp. 39–45, 1999.

[13] "[online] Technical Report." https://cspcheng.github.io/pdf/DASC.pdf.

[14] N. Armenatzoglou, H. Pham, V. Ntranos, D. Papadias, and C. Shahabi, "Real-time multi-criteria social graph partitioning: A game theoretic approach," in *ACM SIGMOD*, pp. 1617–1628, 2015.

[15] J. F. Nash *et al.*, "Equilibrium points in n-person games," *PNAS*, vol. 36, no. 1, pp. 48–49, 1950.

[16] D. Monderer and L. S. Shapley, "Potential games," *Games and economic behavior*, vol. 14, no. 1, pp. 124–143, 1996.

[17] X. Liu, Q. He, Y. Tian, W.-C. Lee, J. McPherson, and J. Han, "Event-based social networks: linking the online and offline social worlds," in *ACM SIGKDD*, pp. 1032–1040, 2012.

[18] Z. Chen, R. Fu, Z. Zhao, Z. Liu, L. Xia, L. Chen, P. Cheng, C. C. Cao, Y. Tong, and C. J. Zhang, "gmission: A general spatial crowdsourcing platform," *Proceedings of the VLDB Endowment*, vol. 7, no. 13, pp. 1629–1632, 2014.

[19] R. Boim, O. Greenshpan, T. Milo, S. Novgorodov, N. Polyzotis, and W.-C. Tan, "Asking the right questions in crowd data sourcing," in *ICDE*, pp. 1261–1264, 2012.

[20] Y. Zheng, J. Wang, G. Li, R. Cheng, and J. Feng, "Qasca: A quality-aware task assignment system for crowdsourcing applications," in *ACM SIGMOD*, pp. 1031–1046, 2015.

[21] J. Fan, G. Li, B. C. Ooi, K.-l. Tan, and J. Feng, "icrowd: An adaptive crowdsourcing framework," in *ACM SIGMOD*, pp. 1015–1030, 2015.

[22] Y. Zeng, Y. Tong, L. Chen, and Z. Zhou, "Latency-oriented task completion via spatial crowdsourcing," in *2018 IEEE 34th International Conference on Data Engineering (ICDE)*, pp. 317–328, IEEE, 2018.

[23] Y. Tong, Z. Zhou, Y. Zeng, L. Chen, and C. Shahabi, "Spatial crowdsourcing: a survey," *The VLDB Journal*, pp. 1–34, 2019.

[24] H. To, C. Shahabi, and L. Kazemi, "A server-assigned spatial crowdsourcing framework," *ACM TSAS*, 2015.

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

[26] 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.

[27] Y. Tong, J. She, *et al.*, "Online mobile micro-task allocation in spatial crowdsourcing," in *ICDE*, pp. 49–60, IEEE, 2016.

[28] Y. Tong, Y. Zeng, Z. Zhou, L. Chen, J. Ye, and K. Xu, "A unified approach to route planning for shared mobility," *Proceedings of the VLDB Endowment*, vol. 11, no. 11, pp. 1633–1646, 2018.