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
wangze.ni@connect.ust.hk, leichen@cse.ust.hk
*East China Normal University, Shanghai, China
pcheng@sei.ecnu.edu.cn
The University of New South Wales, Australia
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.

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.

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 [10], we first need to hire a crew chief. After we hire a crew chief, we can go to hire two associate referees. Also, we 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 hiring a scorer, 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).

Furthermore, 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-skills 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.

Example 1. In the example, a spatial crowdsourcing platform has three workers (r1 − r3) and five tasks (t1 − t5) as shown in Figure 1(a) where the dash arrow lines pointing from a
tables and assign a set of workers to fully support a complex

<table>
<thead>
<tr>
<th>Task</th>
<th>Location</th>
<th>Required Skill</th>
<th>Dependency</th>
</tr>
</thead>
<tbody>
<tr>
<td>$t_1$</td>
<td>(4, 1)</td>
<td>$\psi_1$</td>
<td>$\emptyset$</td>
</tr>
<tr>
<td>$t_2$</td>
<td>(2, 2)</td>
<td>$\psi_2$</td>
<td>${t_1}$</td>
</tr>
<tr>
<td>$t_3$</td>
<td>(5, 2)</td>
<td>$\psi_3$</td>
<td>${t_1, t_2}$</td>
</tr>
<tr>
<td>$t_4$</td>
<td>(3, 4)</td>
<td>$\psi_4$</td>
<td>$\emptyset$</td>
</tr>
<tr>
<td>$t_5$</td>
<td>(1, 2)</td>
<td>$\psi_3$</td>
<td>${t_4}$</td>
</tr>
</tbody>
</table>

**TABLE II: Workers’ Detail.**

<table>
<thead>
<tr>
<th>Worker</th>
<th>Location</th>
<th>Skill Set</th>
</tr>
</thead>
<tbody>
<tr>
<td>$w_1$</td>
<td>(2, 1)</td>
<td>${\psi_1, \psi_2}$</td>
</tr>
<tr>
<td>$w_2$</td>
<td>(3, 3)</td>
<td>${\psi_4}$</td>
</tr>
<tr>
<td>$w_3$</td>
<td>(5, 3)</td>
<td>${\psi_1, \psi_2, \psi_3}$</td>
</tr>
</tbody>
</table>

Moreover, each worker $w$ has a set of skills $\Psi = \{\psi_1, \psi_2, \ldots, \psi_r\}$ 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 $s_w + w_w$. In this paper, we first prove that DA-SC problem is NP-hard by reducing it from the maximum 4-dimensional matching (Max4DM) [9]. Thus, the DA-SC problem is intractable and hard to achieve the optimal result. We tackle the DA-SC problem with two approximation algorithms, namely game-theoretic approach and greedy approach, which can 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 formally define 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 $\Psi = \{\psi_1, \psi_2, \ldots, \psi_r\}$ is a universe of $r$ abilities/skills.

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 \( \text{dist}(x, y) \) 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 task assignment 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 \( t = \langle l_t, s_t, w_t, r_{s_t}, D_t \rangle \) 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 \( r_{s_t} \). 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 \( r_{s_t} \) 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 immediate precedent tasks \( D_t \) are completed. Without loss of generality, the graph of the dependency relationships of tasks is acyclic, which means 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

\[
\text{Sum}(M) = |M| = \sum_{w \in W, t \in T} I(w, t)
\]

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 \( r_{s_t} \).

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., \( s_t \leq s_w + w_w \)).

   (2) The worker should be able to arrive at the location of the assigned task before the deadline of the task (i.e., \( w_t - \max\{s_w - s_t, 0\} - ct_w(l_w, l_t) \geq 0 \), here \( ct_w(l_w, l_t) \) is the travel cost from \( l_w \) to \( l_t \)).

3) **Exclusive constraint.** One dependency-aware task can be assigned to at most one worker (i.e., \( \sum_{w \in W} I(w, t) \leq 1 \)) and any worker can be assigned to only one task at each time.

4) **Dependency constraint.** The worker \( w \) can conduct the task \( t \) if and only if the task \( t' \)'s dependent tasks had been assigned. (i.e., \( \prod_{t' \in D_t} a_{t'} = 1 \), where \( a_{t'} = \sum_{w \in W} I(w, t') \)).

**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 maximum 4-dimensional matching (Max4DM) [9], 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 maximum 4-dimensional matching (Max4DM) [9], which can be described as follows:

Given a hyper-graph \( H = (V^1, V^2, V^3, V^4, E) \), where \( V^1, V^2, V^3, \) and \( V^4 \) are four disjoint sets. In addition, each hyper-edge contains exactly one vertex from each set, namely, \( E \subseteq V^1 \times V^2 \times V^3 \times V^4 \). The Max4DM problem is to find a matching of maximal size in \( H \).

For a given Max4DM problem, we can transform it to an instance of DA-SC problem. In this instance, each task’s dependency size is at most 1. Let the set of tasks which do not rely on another task be \( V^2 \). The set of workers who can do the tasks in \( V^2 \) be \( V^1 \). Let the set of tasks which rely on one task in \( V^2 \) be \( T \) and the set of \(|E|\) extra nodes be \( EP \). Let the union set of \( T \) and \( EP \) be \( V^3 \). Let the set of workers who can do the tasks in \( T \) be \( W \) and the set of \(|E|\) extra nodes be \( W^4 \). The quaternions \( (V_1, V_2, V_3, V_4) \) means that the worker \( v_1 \) is valid to task \( v_2 \), the worker \( v_4 \) is valid to task \( v_3 \) and \( v_2 \) is dependent on the task \( v_2 \).

Thus, to maximize the number of assigned worker-and-task pairs in the DA-SC instance is the same as to maximize the number of matching in Max4DM instance. Given this mapping, it is easy to show that the Max4DM instance can be solved if and only if the transformed DA-SC problem can be solved. This way, we can reduce the Max4DM problem to the DA-SC problem. Since the Max4DM is known to be NP-hard [9], DA-SC is also NP-hard, which completes our proof.

Note that, even when we do not consider the dependency between tasks, the DA-SC problem is still NP-hard due to other constraints as shown in the existing work [10].
Thus, due to the NP-hardness of the DA-SC problem, in the subsequent sections, we propose two approximate algorithms, namely greedy and game-theoretic approaches, to efficiently solve DA-SC 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 proposed a greedy-based algorithm, namely DASC_Greedy, 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 $tc_i$ in $TC_b$, we try to find a set of workers $tw$ who can accomplish $tc_i$ by the Hungarian Algorithm[11] (line 5). Then, if the worker set $tw$ can be found, add $\langle tw, tc_i \rangle$ into the candidate set $C$ (line 7). After that, we move the assignment whose associative task set has the largest size from the candidate set $C$ to $M_b$ (line 9). 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 $O(\min\{n_b, m_b\} \cdot m_b)$.

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 I is monotone and submodular.

**Theorem III.1.** $\text{Sum}(M)$ is monotone and submodular.

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

To proving its submodularity, we have:

$\forall \langle t_{w_1}, t_{c_1} \rangle \notin M, \forall \langle t_{w_2}, t_{c_2} \rangle \notin M,\quad \text{Sum}(\hat{M}) - \text{Sum}(M) \leq \text{Sum}(\hat{M}) - \text{Sum}(\hat{M})$  \hspace{1cm} (2)

where $\hat{M} = M \cup \{\langle t_{w_1}, t_{c_1} \rangle\}$ and $\hat{M} = M \cup \{\langle t_{w_2}, t_{c_2} \rangle\}$. Here $tw$ is the set of workers to conduct the associative task set $tc$.

There are two cases:

- $\langle t_{w_1}, t_{c_1} \rangle$ is assigned in $\hat{M}$ and $\hat{M}$, but the size of $t_{c_2}$ in $\hat{M}$ is larger than that in $M$. When we update the associative task sets $TC$ after assigning the associative task set $t_{c_1}$, we will remove the tasks in $t_{c_1}$ from the related associative task sets of $tc_2$. The size of the associative task set $tc_2$ will not increase after assigning another associative task set.

- The worker set $W_b$ is a subset of the worker set $W_b$, where $W_b$ is the active worker set after (before) finishing the associative task set $tc_i$ (i.e., $W_b \subseteq W_b$). If $tc_j$ can be finished by the workers in $W_b$ but cannot be finished by the workers in $W_b$, $tc_j$ will not be assigned in $\hat{M}$. Then, $\text{Sum}(\hat{M}) - \text{Sum}(M) = 0$, but $\text{Sum}(\hat{M}) - \text{Sum}(M) = |tc_j|$.
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 \( \langle W, \{S_w\}_{w \in W}, \{U_w : x \in W \rightarrow S_w \}_{w \in W} \rightarrow \mathbb{R} \rangle \) where \( S_q \) represents all the possible strategies that worker 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 \( w \in W \): \( U_w(s_1, \ldots, s_w, \ldots, s_{|W|}) \leq U_w(s_1, \ldots, s'_w, \ldots, s_{|W|}), \forall s'_w \in S_w. \) 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 \( \Phi : x \in W \rightarrow \mathbb{R} \), called the potential function in potential games, which constitute a special class of strategic games. Let \( \pi_w \) denote the set of strategies followed by all workers except \( w \) (i.e., \( \pi_w = \{s_1, \ldots, s_{w-1}, s_{w+1}, \ldots, s_{|W|}\} \). A potential game is exact if there exists a potential function \( \Phi \), such that for all \( s_w \) and all possible combinations of \( \pi \in x \in \mathbb{N}\setminus\{i\} S_j \), the following condition holds:

\[
U_w(s_w, s_w') - U_w(s_w, s_w) = \Phi(s_w, s_w') - \Phi(s_w', s_w')
\]

proves that for potential games, the best-response framework as shown in Algorithm 2 always converges to a pure Nash equilibrium. Therefore, we can use the best-response framework of Algorithm 2 to obtain the solution for DA-SC problem. Specifically, we propose a game-theoretic algorithm, 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 \( \langle W_b, S_w \in W_b, \{U_w : x \in W \rightarrow S_w \}_{w \in W} \rightarrow \mathbb{R} \rangle \), 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 \( \pi_w \) of the other players, the total utility \( U_w(s_w, \pi_w) \) 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:

\[
U_w(s_w, \pi_w) = \left\{ \begin{array}{ll}
\sum_{f \in D_s} \frac{\alpha}{\alpha + |D_s|} \cdot \frac{s_w}{|D_s|} - \sum_{f \in D_s} \frac{\alpha}{\alpha + |D_s|} \cdot \sum_{s_w' \in S_w} s_w', & D_s \neq \emptyset \\
\frac{1}{nw_w} + \sum_{f \in D_s} \frac{\alpha}{\alpha + |D_s|} \cdot s_w', & D_s = \emptyset
\end{array} \right.
\]

(3)

where \( \alpha \) is the normalization parameter. Because of the dependency constraint, each task’s utility can be obtained if its dependent tasks are assigned, i.e., \( \prod_{f \in D_s} 0_f = 1 \). 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 \( \frac{1}{\alpha} \), and the second part, Utility_Dependency, is the utility of the assignments of its dependent tasks, whose value is \( \frac{1}{\alpha} \). In addition, 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. In addition, 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 $U_w(s_w, s_w)$ 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 summation of all individual worker’s utility (i.e., $\sum_w U_w(s_w, s_w))$. 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. In addition, we define the potential function $\Phi(S)$ as follows:

$$\Phi(S) = -\sum_{w \in W} \sum_{t \notin S_w} \sum_{a_t \in A_t} \frac{\Pi_{f \in D_t(t)} a_f}{(n_w + 1) \cdot (n_b - n_w)}$$

$$= -\sum_{w \in W} \sum_{t \notin S_w} \sum_{a_t \in A_t} \frac{(\alpha - 1) \cdot \Pi_{f \in D_t(t)} a_f}{\alpha \cdot (n_w + 1) \cdot (n_b - n_w)}$$

$$- \sum_{w \in W} \sum_{t \notin S_w} \sum_{a_t \in A_t} \frac{\Pi_{f \in D_t(t)} a_f}{\alpha \cdot (n_w + 1) \cdot (n_b - n_w)}$$

We divide the utility of task $t$ by $(n_w + 1) \cdot (n_b - n_w)$ shares and the utility of task $t$ can be obtained iff $t$ is assigned and its dependent tasks are assigned, i.e., $\Pi_{f \in D_t(t)} a_f = 1$. In addition, for each worker $w \in W$, 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. In addition, the performance of DASC_Game can be improved by some simple 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 [\])

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 $n w_{s_w}$ and $s'_w$ 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, $\overline{n w_{s_w}}$ and $\overline{s'_w}$ 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. Notate that, $n w_{s_w} = \overline{n w_{s_w}} + 1$ and $n w_{s'_w} = \overline{n w_{s'_w}} + 1$. Indeed, when $D_{s_w} \neq \emptyset$ and $D_{s'_w} \neq \emptyset$, we have:

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

$$= \left(\frac{(\alpha - 1) \cdot \Pi_{f \in D_{s_w}(t)} a_f}{\alpha \cdot (n_w + 1)} + \sum_{s'_w \in S_w} \frac{\Pi_{f \in D_{s'_w}(t)} a_f}{\alpha \cdot (n_w + 1)} \cdot |D_t| \cdot |n_b - n_w| \right)$$

$$+ \left(\frac{(\alpha - 1) \cdot \Pi_{f \in D_{s'_w}(t)} a_f}{\alpha \cdot (n_w + 1)} + \sum_{s_w \in S_w} \frac{\Pi_{f \in D_{s_w}(t)} a_f}{\alpha \cdot (n_w + 1)} \cdot |D_t| \right)$$

$$= \left(\frac{(\alpha - 1) \cdot \Pi_{f \in D_{s_w}(t)} a_f}{\alpha \cdot (n_w + 1)} + \sum_{s'_w \in S_w} \frac{\Pi_{f \in D_{s'_w}(t)} a_f}{\alpha \cdot (n_w + 1)} \cdot |D_t| \right)$$

$$+ \sum_{s_w \in S_w} \frac{\Pi_{f \in D_{s_w}(t)} a_f}{\alpha \cdot (n_w + 1)} \cdot |D_t|$$

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

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}) -
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].

Then, we proved DA-SC is an exact potential game. \(\square\)

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 \(\Psi(S) = d \cdot \Phi(S)\), where \(d\) is a constant which depends on the \(nw_s\), such that \(\Psi(S) \in \mathbb{Z}\), \(\forall S\). Obviously, 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 converges to an equilibrium is upper bounded by \(d \cdot \min\{n_b, mb\}\).

**Proof.** The scaled version of DASC Game Algorithm with the potential function as \(\Psi(S) = d \cdot \Phi(S)\) will converge to a Nash equilibrium in the same number of rounds as DASC Game Algorithm. Since \(\Psi(S) \in \mathbb{Z}\), the cost increase in \(\Psi(S)\) after each strategy change of a worker is at least 1. Therefore, the upper bound of the number of rounds is the maximum value, \(\Phi_{max}\), minus the minimum value, \(\Phi_{min}\). We can see that in DASC Game Algorithm, \(\Phi_{max} \leq 0\) and \(\Phi_{min} \geq -d \cdot \min\{n_b, mb\}\). Consequently, the number of rounds to reach an equilibrium is upper-bounded by \(d \cdot \min\{n_b, mb\}\), as stated by the lemma. \(\square\)

Then, we can prove the time complexity of DASC Game Algorithm as follows.

**Lemma IV.2.** The time complexity of DASC Game in batch \(b\) is \(O(d \cdot n_{t} \cdot \min\{n_{b}, m_{b}\})\), where \(m_{t}\) is the number of tasks and \(n_{t}\) is the number of workers in the batch.

**Proof.** As discussed above, the number of iteration is upper bounded by \(O(d \cdot \min\{n_{b}, m_{b}\})\). In addition, 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 \(O(d \cdot n_{t} \cdot \min\{n_{b}, m_{b}\})\). \(\square\)

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. Specifically, the 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, \(U(S) \triangleq \sum_{w \in W_{b}} U_{w}(s_{w}, s_{w}')\), 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 follows:

\[
\frac{1}{nw_{max} + 1} \cdot U(S) \leq |\Phi(S)| \leq \frac{n_{b}}{nw} \cdot (n_{b} - nw) \cdot U(S)
\]

where \(nw = \min\{nw_{min}, n_{b} - nw_{max}\}\), \(nw_{max}\) 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 \(nw_{max} \geq nw_{t}, \forall t \in S_{w},\) we have

\[
\sum_{w \in W_{b}} \sum_{n_{t} \in S_{w}} \Pi_{f \in D_{w}(t)} f_{a} \cdot (nw_{t} + 1) \cdot \Pi_{f \in D_{w}(t)} f_{a}\]

Thus, \(|\Phi(S)| \geq \frac{1}{nw_{max} + 1} \cdot \sum_{w \in W_{b}} \sum_{n_{t} \in S_{w}} \Pi_{f \in D_{w}(t)} f_{a}\)

Since \(nw_{max} \geq nw_{t}, \forall t \in S_{w},\) we have

\[
\sum_{w \in W_{b}} \sum_{n_{t} \in S_{w}} \Pi_{f \in D_{w}(t)} f_{a} \leq \sum_{w \in W_{b}} \sum_{n_{t} \in S_{w}} \Pi_{f \in D_{w}(t)} f_{a} \cdot (nw_{t} + 1) \cdot \Pi_{f \in D_{w}(t)} f_{a}\]

Thus, we have \(|\Phi(S)| \leq \frac{n_{b}}{nw} \cdot (n_{b} - nw) \cdot U(S).\) \(\square\)

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 \(\frac{nw_{min}(nw_{min} - nw_{max})}{nw_{min}(nw_{min} + 1)} \cdot |\Phi(S)|_{min}\), where \(|\Phi(S)|_{min}\) 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 \(U(S^*) \geq U(S), \forall S\) and \(|\Phi(S^{**})| \geq |\Phi(S)|, \forall S\), we have:

\[
U(S^{**}) \geq \frac{nw_{min}(nw_{min} - nw_{max})}{nw_{min}(nw_{min} + 1)} \cdot |\Phi(S^{**})| \geq \frac{nw_{min}(nw_{min} - nw_{max})}{nw_{min}(nw_{min} + 1)} \cdot |\Phi(S)|
\]

Since \(OPT = U(S^*)\), \(PoS = \frac{U(S^{**})}{OPT} \geq \frac{nw_{min}(nw_{min} - nw_{max})}{nw_{min}(nw_{min} + 1)}\).

Next, let \(S^{\#}\) be the strategic configuration of any Nash equilibrium obtained by DASC Game. Thus, we have:

\[
U_{w}(s_{w}, s_{w}') \geq U_{w}(s_{w}, s_{w}')_{w \in W, s_{w}' \in S_{w}}\]

That is, \(\forall w \in S_{w}, \Phi(S^{\#}, s_{w}) - \Phi(S^{\#}, s_{w}') = U_{w}(s_{w}, s_{w}') - U_{w}(s_{w}', s_{w}') \geq 0\). Hence, \(\Phi(S^{\#}, s_{w})\) is a local maximum and \(|\Phi(S^{\#}, s_{w})|\) is a local minimum. Thus, we have:

\[
U(S^{\#}) \geq \frac{nw_{min}(nw_{min} - nw_{max})}{nw_{min}(nw_{min} + 1)} \cdot |\Phi(S^{\#})| \geq \frac{nw_{min}(nw_{min} - nw_{max})}{nw_{min}(nw_{min} + 1)} \cdot |\Phi(S)|_{min}
\]

Since \(U(S^*) \leq \min\{n_{b}, m_{b}\}\), we have:

\[
PoA = \frac{U(S^{\#})}{U(S^*)} \geq \frac{nw_{min}(nw_{min} - nw_{max})}{nw_{min}(nw_{min} + 1)} \cdot |\Phi(S)|_{min}\]
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 tag 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 practiced skills of workers in DA-SC, where each tag is considered as a skill in our experiments. 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 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 $22.209^\circ$ to $22.609^\circ$ and longitude from $113.843^\circ$ to $114.283^\circ$).

For synthetic data, we generate locations of workers and tasks in a 2D data space $[0, 0.5]^2$, following 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. In addition, we simulate the size of each worker’s skill set with the uniform distribution in 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 $[0, 25]$. 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 Algorithm, 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 take. In addition, we propose two baseline methods. The first algorithm, namely Closest, greedily selects worker-and-task pairs with the lowest moving distance for each worker. The second algorithm, namely Random, randomly selects 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, which includes 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 a famous algorithm (g-D&C) in a related 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 condition. Specifically, the parameters used in the experiment is 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 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 [15]. 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 planed 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 gMission. 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 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, 0.05] 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 tasks’ dependency constraints when they allocate tasks and many of 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 achieve the highest score among the six 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, the Greedy achieves much lower running time than the $G\&G$, Game, and GT. The reason is that the $G\&G$, Game, and GT has a large search space and in each round, it searches the whole search space while in each round the 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 costs much more time. Besides, the running time of $g$-$D&C$ is higher than the Greedy and GT. The reason is that the $g$-$D&C$ 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 value range of velocity gets larger, the assignment scores of all the seven 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 the $g$-$D&C$ except the Greedy. As shown in Figure 2(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, the $g$-$D&C$ and GT algorithm 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 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 the six 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. The $G\&G$ 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 the $g$-$D&C$ except the Greedy. As shown in Figure 5(b) the running time of our four approaches increase, when the range of start timestamp gets larger. The reason is that, when the start 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.

**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 the seven 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. The $G\&G$ 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 the $g$-$D&C$ 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. The reason is that, 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**

In order 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 made by the two baseline algorithms and 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 $G&G$ 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 its running time growing constantly.

Effect of the Number of Tasks $m$. Figure 8 illustrates the experimental results on different number, $m$, of tasks from 4K to 6K. In Figure 8(a), when the number of tasks gets larger, the assignment scores of seven approaches increase. The reason is that the increase of $m$ 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 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 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 increase, when the number of tasks gets larger. The reason is that, 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 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 is related to the tasks’ dependency sizes, the $g$-$D&C$’s running time is not as sensitive as our proposed algorithms. When the $|WS|$ is small, $g$-$D&C$ needs more time than other algorithms.

F. Results on gMission Platform

Figure 10 illustrates the seven 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 log in 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 the seven 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. Specifically, the $G&G$ achieves the highest score and the scores of four proposed algorithms are much better than those of two baseline algorithms and the compared algorithm $g$-$D&C$. 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 increase, when the range of waiting time gets larger. The reason is that, 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.

We finally summarize our findings.

- Our four approximate algorithm ($G&G$, $Game$, $GT$, 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.
- The $Greedy$ runs much faster than another three approximate algorithms while can achieve results with a close score compared with them.

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] studied the task assignment in crowdsourcing to finish the tasks more efficiently and accurately.

In spatial crowdsourcing [1], workers are requested to physically move to specific locations to conduct tasks on sites. Based on the ontology [22], 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 [10] 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], [22], [23], [7], [24] 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. [23] 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. [24] 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 proposed 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. To address the DA-SC problem, 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


APPENDIX

A. The derivation of the approximation ratio of DASC_Greedy

Let \( \rho_j(M) = \text{Sum}(M \cup \{tw_j,tc_j\}) - \text{Sum}(M) \) and \( E \) is the universe of assignments. Inspired by the existing work [12], according to the monotonicity and submodularity of \( \text{Sum}(M) \), we have the following lemma.

Lemma A.1. For any assignment \( M \) and \( M' \), it holds,
\[
\text{Sum}(M') - \text{Sum}(M) \leq \sum_{(tw_i,tc_i) \in M'/M} \rho_i(M) - \sum_{(tw_i,tc_i) \in M/M'} \rho_i(M \cup M' - (tw_i,tc_i))
\]

Proof. Based \( \text{Sum}(M) \)'s submodularity, we can have
\[
\rho_{tc}(M) \geq \rho_{tc}(M'), \forall M \subseteq M', tc \in E - M'.
\]
Then, for arbitrary \( M \) and \( M' \) with \( M' - M = \{j_1, \cdots, j_r \} \) and \( M - M' = \{k_1, \cdots, k_q \} \) we have:
\[
\text{Sum}(M \cup M') - \text{Sum}(M)
\]
\[
= \sum_{t=1}^{r} \text{Sum}(M \cup \{j_1, \cdots, j_t \}) - \text{Sum}(S \cup \{j_1, \cdots, j_{t-1} \})
\]
\[
= \sum_{t=1}^{r} \rho_j(M \cup \{j_1, \cdots, j_t \}) \leq \sum_{t=1}^{r} \rho_j(M) = \sum_{j \in M' - M} \rho_j(M)
\]
Similarly, we also have:
\[
\text{Sum}(M \cup M') - \text{Sum}(M')
\]
\[
= \sum_{t=1}^{q} \text{Sum}(M' \cup \{k_1, \cdots, k_t \}) - \text{Sum}(M' \cup \{k_1, \cdots, k_{t-1} \})
\]
\[
= \sum_{t=1}^{q} \rho_k(M' \cup \{k_1, \cdots, k_t \}) \geq \sum_{t=1}^{q} \rho_k(M' \cup M - \{k_t \}) = \sum_{j \in M' - M} \rho_j(M \cup M' - \{j \})
\]
After subtracting above two inequality, we have:
\[
\text{Sum}(M') \leq \text{Sum}(M) + \sum_{k \in M' - M} \rho_k(M \cup M' - \{k \}) - \sum_{j \in M' - M} \rho_j(M) = \sum_{j \in M' - M} \rho_j(M' - \{j \})
\]
In addition, based on the observation that \( 0 \leq \rho_{tc}(M) \leq \psi, \forall M \subseteq E, tc \in E - M, \psi \) is the upper bound of \( \rho_{tc}(M) \), we have:
\[
\text{Sum}(M') \leq \text{Sum}(M) + \sum_{tc \in M' - M} \rho_{tc}(M), \forall M \subseteq E.
\]
Note that \( |M| \leq K \), where \( K = \min \{n_b, m_k \} \). Let \( \text{Sum}(M_{opt}) \) be the value of an optimal solution and \( M' \) be the assignment set which is generated after \( t \) rounds, i.e., \( \text{Sum}(M') = \sum_{i=0}^{t-1} \rho_i \), Suppose the DASC_Greedy Algorithm stops after \( K^* \) rounds, then we have the following lemma.

Lemma A.2. Suppose \( tc_t \) is added by DASC_Greedy Algorithm in the \( t \)-th round, let \( \rho_{t-1} = \rho_{tc_t}(M^{t-1}) \). The corresponding \( \{\rho_i\}_{i=0}^{K^*-1} \) satisfy:
\[
\text{Sum}(M_{opt}) \leq \sum_{t=0}^{t-1} \rho_t + K \cdot \rho_t, t = 0, \cdots, K^* - 1.
\]

Proof. Since \( \rho_j(M) \leq \rho_t, \forall j \in T - S \), according to Lemma A.1, we have:
\[
\text{Sum}(M_{opt}) \leq \text{Sum}(M^t) + \sum_{t \in M_{opt} - M^t} \rho_{tc}(M^t)
\]
\[
\leq \sum_{i=0}^{t-1} \rho_i + \sum_{t \in M_{opt} - M^t} \rho_t
\]
In addition, \( |M_{opt} - M^t| \leq K \), Thus,
\[
\text{Sum}(M_{opt}) \leq \sum_{i=0}^{t-1} \rho_i + \sum_{t \in M_{opt} - M^t} \rho_t \leq \sum_{i=0}^{K^*} \rho_i + K \cdot \rho_t
\]

Let \( M_{greedy} \) be the assignment gotten by DASC_Greedy Algorithm. Then, we have the following theorem to show that our greedy algorithm can provide solution with a theoretic guaranteed bound with respect to the optimal result.

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

Proof. Based on the Lemma A.2, we have
\[
\text{Sum}(M_{opt}) \leq K \cdot \rho_0 = K \cdot (\text{Sum}(M^1))
\]
\[
\text{Sum}(M_{opt}) - \text{Sum}(M^1) \leq K^* \cdot \rho_1 = K \cdot (\text{Sum}(M^2) - \text{Sum}(M^1))
\]
Similarly, we can get the inequalities when \( t = 2, \cdots, K \).
Thus, we have:
\[
\frac{\text{Sum}(M_{opt}) - \text{Sum}(M_{greedy})}{\text{Sum}(M_{opt})} \leq \left( \frac{K-1}{K} \right)^K
\]
Thus, the matching size returned by DASC_Greedy Algorithm is at least \( (1 - \frac{1}{e}) \cdot |M_{opt}| \).

B. Proof of Theore IV.1

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

Proof. Recall from Section IV-A, that it suffices to show that for every worker \( w \), who changes his strategy from the current one \( s_w \) to the best-response \( s'_{w,w} \), and for all possible combinations of the other players' strategies \( \bar{s}_w \) it holds that:
\[
U_w(s_w, \bar{s}_w) - U_w(s_w, s_w) = \Phi(s_w, \bar{s}_w) - \Phi(s_w, s_w)
\]
Suppose \( np_{s_{w\bar{w}}} \) and \( np_{s_{w\bar{w}}} \) are the numbers of workers who are assigned to the tasks \( s_w \) and \( s'_{w} \) in the assignment \( (s_w, s_{w\bar{w}}) \), respectively. Similarly, \( np_{s_{w\bar{w}}} \) and \( np_{s_{w\bar{w}}} \) are the numbers of workers who are assigned to the tasks \( s_w \) and \( s_{w\bar{w}} \) in the assignment \( (s'_{w}, \bar{s}_{w}) \), respectively. Note that, \( np_{s_w} = np_{s_{w\bar{w}}} + 1 \) and \( np_{s_w} = np_{s_{w\bar{w}}} + 1 \). Indeed, when \( D_{s_w} \neq 0 \) and \( D_{s_w} \neq 0 \), we have:
\[ \Phi(s_w, \overline{s_w}) - \Phi(s'_w, \overline{s_w}) = -\left( \frac{(\alpha - 1) \cdot \prod_{f \in D_{s_w}} a_f}{\alpha \cdot (np_{s'_w} + 1)} + \sum_{s'_w \in D_t} \frac{\prod_{f \in D_t \cup \{l\}} a_f}{\alpha \cdot (np_{s'_w} + 1) \cdot |D_t|} \right) \\
+ \left( \frac{(\alpha - 1) \cdot \prod_{f \in D_{s'_w}} a_f}{\alpha \cdot (np_{s'_w} + 1)} + \sum_{s''_w \in D_t} \frac{\prod_{f \in D_t \cup \{l\}} a_f}{\alpha \cdot (np_{s''_w} + 1) \cdot |D_t|} \right) \\
= -\left( \frac{(\alpha - 1) \cdot \prod_{f \in D_{s_w}} a_f}{\alpha \cdot np_{s_w}} + \sum_{s'_w \in D_t} \frac{\prod_{f \in D_t \cup \{l\}} a_f}{\alpha \cdot np_{s'_w} \cdot |D_{s'_w}|} \right) \\
+ \left( \frac{(\alpha - 1) \cdot \prod_{f \in D_{s'_w}} a_f}{\alpha \cdot np_{s'_w}} + \sum_{s''_w \in D_t} \frac{\prod_{f \in D_t \cup \{l\}} a_f}{\alpha \cdot np_{s''_w} \cdot |D_{s''_w}|} \right) \\
= U_w(s_w, \overline{s_w}) - U_w(s'_w, \overline{s_w}) \\
\]

Similarly, when \( D_{s_w} = \emptyset \) and \( D_{s'_w} = \emptyset \); \( 2 \) \( D_{s_w} \neq \emptyset \) 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 A of our technical report [13].

When \( D_{s_w} = \emptyset \) and \( D_{s'_w} \neq \emptyset \), we have:
\[ \Phi(s_w, \overline{s_w}) - \Phi(s'_w, \overline{s_w}) = -\left( \frac{(\alpha - 1) \cdot \prod_{f \in D_{s'_w}} a_f}{\alpha \cdot (np_{s'_w} + 1)} + \sum_{s''_w \in D_t} \frac{\prod_{f \in D_t \cup \{l\}} a_f}{\alpha \cdot (np_{s''_w} + 1) \cdot |D_{s''_w}|} \right) \\
+ \left( \frac{(\alpha - 1) \cdot \prod_{f \in D_{s_w}} a_f}{\alpha \cdot np_{s_w}} + \sum_{s'_w \in D_t} \frac{\prod_{f \in D_t \cup \{l\}} a_f}{\alpha \cdot np_{s'_w} \cdot |D_{s'_w}|} \right) \\
= U_w(s_w, \overline{s_w}) - U_w(s'_w, \overline{s_w}) \]

Then, we proved DA-SC problem is an exact potential game. \( \square \)

### C. Results on Synthetic Data Sets

**Effect of the size of the skill universe \( r \)**. Figure [11] illustrates the experimental results on different size, \( r \), of the skill universe \( \Psi \) from 50 to 90. In Figure [11(a)] the assignment scores of all the seven approaches decrease, when the size of the skill universe gets larger. The reason is that, the increase of \( r \) disperses the tasks/workers over skills. The G&DG achieves the highest score and the scores of four proposed algorithms are much better than those of two baseline algorithms and the compared algorithm \( g-D&C \). As shown in Figure [11(b)] the running time of our four approaches decrease, when the size of the skill universe gets larger. The reason is that, when the size of the skill universe gets larger, the number of tasks that each worker can serve decreases which thus leads to lower complexity of the DA-SC problem and the decrease of the running time.

**Effect of the Range of each Worker’s Skill Size \([sp^-, sp^+]\)**. Figure [12] illustrates the experimental results on different ranges, \([sp^-, sp^+]\), of each worker’s skill set \( |WS| \) from [1, 5] to [1, 30]. In Figure [12(a)] the assignment scores of all the six approaches increase, when the value range of each worker’s skill size gets larger. The reason is that, the increase of \( |WS| \) let each task has more valid workers. The G&DG achieves the highest score and our proposed algorithms achieve better scores than those of two baseline algorithms and the compared algorithm \( g-D&C \). As shown in Figure [12(b)] the running time of our four approaches increases, when the range of each worker’s skill size gets larger. Since the \( g-D&C \)’s running time is mainly dominated by the rounds that the \( g-D&C \) runs which is related to the tasks’ dependency sizes, the \( g-D&C \)’s running time is not as sensitive as our proposed algorithms. When the \( |WS| \) is small, \( g-D&C \) needs more time than other algorithms.

**Effect of the Range of the Maximum Moving Distance \([d^-, d^+]\)**. Figure [13] illustrates the experimental results on different ranges, \([d^-, d^+]\), of each worker’s maximum moving distance \( d \) from \([0.3, 0.32]\) to \([0.38, 0.4]\). In Figure [4(a)] when the value range of maximum moving distance gets larger, the assignment scores of all the seven approaches increase.
moving distance gets larger. The reason is that, when the moving distance increase, each worker has more valid tasks which thus leads to higher complexity of the DA-SC problem and the increase of the running time. However, when the constraint of $d$ is relaxed, the constraints from other parameters prevent the running time keeping growing.

**Effect of the range of the velocity, $[v^-, v^+]$.** Figure 14 illustrates the experimental results on different ranges, $[v^-, v^+]$, of each worker’s velocity $v$ from $[0.001, 0.007]$ to $[0.001, 0.011]$. In Figure 14(a) when the value range of velocity gets larger, the assignment scores of all the seven approaches increase. The reason is that, with the increase of $v$, workers can reach more tasks. As shown in Figure 14(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 higher complexity of the DA-SC problem and the increase of the running time.

**Effect of the range of the start timestamp, $[st^-, st^+]$.** Figure 15 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 15(a), the assignment scores of all the seven 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. The $G&G$ achieves the highest score and the scores of four proposed algorithms are much better than those of two baseline algorithms and the compared algorithm $g$-$D&C$. As shown in Figure 15(b) the running time of our four approaches decreases, when the range of start timestamp gets larger. The reason is that, when the range increases, each worker has less valid tasks which thus leads to lower complexity of the DA-SC problem and the decrease of the running time.

**Effect of the range of the waiting time, $[wt^-, wt^+]$.** Figure 16 illustrates the experimental results on different ranges, $[wt^-, wt^+]$, of each worker’s task’s waiting time $wt$ from $[1, 10]$ to $[1, 18]$. In Figure 16(a), the assignment scores of all the six 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. The $G&G$ achieves the highest score and the scores of four proposed algorithms are much better than those of two baseline algorithms and the compared algorithm $g$-$D&C$. As shown in Figure 16(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 higher complexity of the DA-SC problem and the increase of the running time. Since the $g$-$D&C$’s running time is mainly dominated by the rounds that the $g$-$D&C$ runs which is related to the tasks’ dependency sizes, the $g$-$D&C$’s running time is not as sensitive as our proposed algorithms. When the $[wt^-, wt^+]$ is small, $g$-$D&C$ needs more time than other algorithms.