Task Allocation on 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 on each other. Specifically, one task can only be conducted when its dependent tasks have already been finished. 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, moving distances and deadlines with a goal of maximizing the successfully assigned worker-and-task pairs. We prove that the DA-SC problem is NP-hard and thus intractable. Therefore, we propose two approximation algorithms, including greedy and game-theoretic approaches, 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.

PVLDB Reference Format:
Wangze Ni, Peng Cheng, Lei Chen. Task Allocation on Dependency-aware Spatial Crowdsourcing. *PVLDB*, 12(xxx): xxxx-yyyy, 2019.
DOI: https://doi.org/xxx.xxxx/xxx.xxxx

1. 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], TaskRabbit [3], and Waze [4]), 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 [5]), delivering foods (e.g., Uber Eats [6]), and queuing up for a hot restaurant (e.g., TaskRabbit [3]). However, some complex tasks require specific skills to finish, such as assembling furniture, cleaning home and repairing a house.

Previous studies [7, 8, 9] 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, if a requester wants to repair a house, she/he would propose some tasks like painting walls, installing pipe systems, and cleaning. Generally, the pipe systems should be installed before painting walls since some pipes are embedded in the walls. If installing pipe systems after painting, workers have to paint the wall again. Similarly, if the tasks of painting walls and installing pipe systems are not accomplished, the cleaning task could insignificant. In addition, each worker can only do one task at the same time. In spatial crowdsourcing platforms, the workers are free to come and leave and the tasks are keep appearing and being finished. 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 can be not efficient and time-consuming 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.

Inspired by the above observation, 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*$(r_1 - r_3)$ *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 each worker and task are shown in Table 1 and Table 2. For simplicity, in this example we set all workers and tasks appear on the platform at the same time. In addition, the velocity of each worker is fast enough and the maximum moving distance of each worker is large enough to reach the assigned tasks before their deadlines. The goal of the platform is to maximize the total number of valid assigned worker-task pairs such that the assigned worker has the required skill and the dependencies of the task have been satisfied.*

If the platform greedily assigns the nearest skill-satisfied workers

Proceedings of the VLDB Endowment, Vol. 12, No. xxx
ISSN 2150-8097.
DOI: https://doi.org/xxx.xxxx/xxx.xxxx

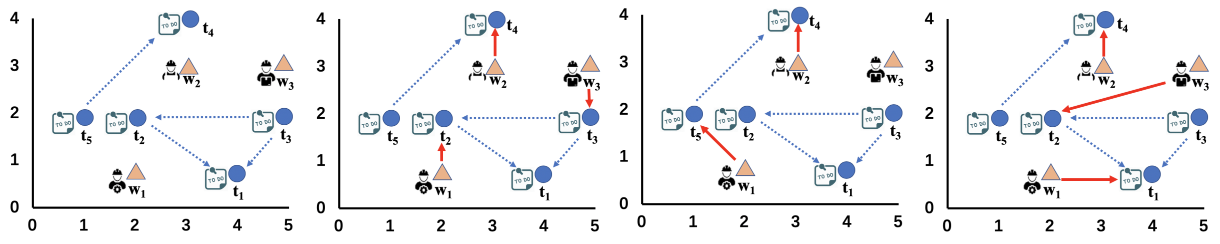

(a) Initial locations (b) Allocation by Greedy (c) Allocation by Team (d) Optimal Allocation

Figure 1: Motivation Example.

Table 1: 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 2: Workers' Detail.

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

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, does not been finished, the assignment (w_1, t_2) is invalid. Similarly, the assignment (w_3, t_3) is also invalid. Thus, the total number of valid assigned worker-task pairs is only 1.

If the platform consider the dependent tasks as one multi-skill task group and try to form a team to do the task group [8], the allocation is shown by the red arrows in Figure 1(c). Specifically, there are two task groups, (t_1, t_2, t_3) and (t_4, t_5). However, for the task group (t_1, t_2, t_3), we cannot find a team that the union of teammates' skill sets can cover the required skills and each worker is just assigned one task. Hence, the platform waives the task group (t_1, t_2, t_3) and forms a team $(w_1, w2)$ to accomplish the task (t_4, t_5). Thus, the total number of valid assigned worker-task pairs is 2.

However, if the platform take dependencies into account and conduct dependent tasks separately, the allocation is shown by the red arrows in Figure 1(d). Each worker is assigned to one task and the dependencies of each assigned task are satisfied. Thus, the total number of valid assigned worker-and-task pairs is 3.

From this example, instead of considering a multi-skill tasks as a unitary task in previous studies [7, 8, 9], we can see that consider a multi-skill task as multiple single-skill subtasks and accomplish them separately while satisfying their dependencies can reach a better result.

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 results 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 only can 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 work in multi-skill oriented spatial crowdsourcing [7, 8, 9] does not consider the dependency of the tasks and assign a set of workers to fully support the required skill set of a task, which is not efficient as some workers need to wait until the dependencies of their tasks are satisfied. Thus, no existing methods are tailored for dependency-aware spatial crowdsourcing problem.

In this paper, we first prove that DA-SC problem is NP-hard by reducing it from the knapsack problem with conflict graphs (KCG) [10]. Thus, the DA-SC problem is intractable and hard to achieve the optimal results. Therefore, we tackle the DA-SC problem with two approximation algorithms, namely *game-theoretic approach* and *greedy approach*, which can efficiently acquire worker-and-task assignment pairs 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 2.
- We propose two batch-based approximation algorithms, greedy and game-theoretic approaches, in Section 4 and Section 3 respectively. They can both guarantees the approximate bound of result in each batch process.
- We conduct extensive experiments on real and synthetic data sets, and show the efficiency and effectiveness of our proposed approaches in Section 5.

In addition, we discuss the related work in Section 6 and conclude in Section 7.

2. PROBLEM DEFINITION

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

2.1 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 $w =< l_w, s_w, w_w, v_w, d_w, WS_w >$, appears on the platform with an initial location i_w at time s_w and waits at most w_w time for assigning a task. In addition, the worker w moves with the 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 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. In addition, each worker 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 dependency-aware spatial crowdsourcing task assignment problem.

2.2 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 $t =< l_t, s_t, w_t, rs_t, D_t >$ denote a task. It appears on the platform with a spatial location l_t at time s_t and needs to be served within w_t time. In addition, the task can be accomplished only by a worker who has the skill rs_t. Moreover, the task is dependent on a set of task 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 $t \in D_t$, it should be assigned before conducting task t. Without loss of generality, the graph of the dependency relationships of tasks is acyclic, which means no tasks are dependent on its subsequent tasks.

2.3 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 worker 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

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

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 (i.e., $ts_t \in WS_w$).

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 be able to arrive at the location of the assigned task before the deadline of the task (i.e., $w_t - \max\{s_w - s_t, 0\} - ct_w(l_w, l_t) \ge 0$, here $ct_w(l_w, l_t)$ 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 at most one task at each time.

4. **Dependency constraint**. The worker w can start 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 $a_{t'} = \sum_{w \in W} I(w, t')$).

In Definition 3, our DA-SC problem aims to assign a worker w to a task t such that: (1) the worker w is able to reach the task t's location before its expired time and has the required skill; (2) the task t's dependent tasks are all assigned; (3) the worker/task has not been assigned with another task/worker; and (4) the number of assigned worker-and-task pairs should be maximized.

Table 3: 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 skill
v_w	The worker w's velocity
d_w	The worker w's maximum moving distance
T	The set of m tasks t
ts_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

2.4 Hardness of DA-SC Problem

With m dependency-aware tasks and n workers, in the worst case, there is an exponential number of possible assignment strategies. In this subsection, we prove that our DA-SC problem is NP-hard, by reducing a well-known NP-hard problem, the *knapsack problem with conflict graphs* (KCG)[10], to our DA-SC problem.

Theorem 2.1. *(Hardness of the DA-SC problem) The Dependency-aware Spatial Crowdsourcing problem is sNP-hard.*

Proof. We prove the theorem by a reduction from the *knapsack problem with conflict graphs (KCG)*[10], which can be described as follows:

Let n be the number of items, each of them with a profit p_j and weight w_j, $j = 1, \cdots, n$, and c the capacity of the knapsack. In addition, the conflict graph $G = (V, E)$ where every vertex corresponds uniquely to one item and an edge $(i, j) \in E$ indicates that items i and j can not be packed together. Moreover, each item can be selected no more than once. The problem is to maximize the sum of the profit of the items in the knapsack so that the sum of the weights is less than or equal to the knapsack's capacity.

We consider the special case of the offline version of the DA-SC problem, where each task and worker appears on the platforms at the same time, their waiting time is large enough and each worker has all skills. In addition, the structure of each task's dependency is a line. In other words, if task t_i and t_j are contained in the task t_l's dependency set, then t_i is contained in the task t_j's dependency set or t_j is contained in the task t_i's dependency set. Next, we construct a new set TC by combining each task with its dependency set, i.e., $TC = \{tc_t | tc_t = \{t_t\} \cup D_t, \forall t \in T\}$. Then, we construct a conflict graph $G' = (V', E')$ as follows:

$$\begin{cases} V' = TC \\ E' = \{(i, j) | tc_i \cap tc_j \neq \emptyset\} \end{cases}$$

We define each task combination's profit and weight are both its cardinality and the capacity of the knapsack is the number of workers. Hence, the objective of our DA-SC problem is transferred to find the task combination allocation which maximize the sum of selected task combinations' profit so that the weights is less than or equal to the knapsack's capacity.

Note that, since the structure of each task's dependency is a line, if $tc_j \cap tc_i \neq \emptyset$, then it must hold that $tc_j \subset tc_i$ or $tc_i \subset tc_j$. Therefore, the task combination allocation we get satisfying the conflict constraint guarantees that each task $t \in T$ is selected no more than once.

Since the KCG problem is equivalent to that of the aforementioned special case, we can reduce the KCG problem to the special case of the offline version of the DA-SC problem. Therefore, the offline version of the DA-SC problem is NP-hard. □

Thus, due to the NP-hardness of the DA-SC problem, in the subsequent sections, we first present a general framework for DA-SC, then propose two approximate algorithms, namely *greedy* and *game-theoretic* approaches, to efficiently solve DA-SC with proven approximation ratios for each batch process.

Table 3 summarizes the commonly used symbols.

3. GREEDY APPROACH

In this section, we proposed a batch-based algorithm, namely *DASC_Greedy Algorithm*, to quickly get a bounded result based on the submodular function [11]. Instead of taking all dependent tasks as a task group, in *DASC_Greedy Algorithm*, we just combine the task and its dependency set as one task combination. Then, the algorithm iteratively select the biggest task combination that can be assigned and remove the assigned tasks from the left task combinations. By this way, when we allocate tasks, we not only consider the assignment on individual task but also consider the effect on tasks whose dependency sets contain the assigned task.

3.1 Task Combination

In DASC_Greedy Algorithm, we solve the DA-SC problem by task combinations, instead of individual tasks. Consider task t_i and its dependencies is one task combination, tc_i. According to dependencies, we can transform tasks which are dependent into one task combination as follows:

$$tc_i = \{t_j | t_j \in D_i\}$$

Each task combination cannot be assigned partially. And obviously, after assigning one task combination tc_i, we should update other related task combinations which contain the subset of task combination tc_i. We remove these subsets out of these task combinations. For instance, there is a task t_4 and its dependency is $\{t_1, t_2, t_3\}$. In addition, t_2 is t_3's dependency, but t_1 and t_2 are independent, Thus, there are four task combinations, $\{\{t_1\}, \{t_2\}, \{t_2, t_3\}, \{t_1, t_2, t_3, t_4\}\}$. If we select task combination $\{t_1, t_2, t_3, t_4\}$, we cannot select any task combination $\in \{\{t_1\}, \{t_2\}, \{t_2, t_3\}\}$ again. However, if we select task combination $\{t_2, t_3\}$, we still can select task combination $\{t_1\}$ or $\{t_1, t_4\}$, but not $\{t_2\}$ or $\{t_1, t_2, t_3, t_4\}$. In addition, if we assign the task combination $\{t_2\}$ to work firstly, the updated task combinations are $\{\{t_1\}, \{t_3\}, \{t_1, t_3, t_4\}\}$.

3.2 DASC_Greedy Algorithm

Algorithm 1 shows the pseudo code of our *DASC_Greedy* Algorithm, where we greedily select the workers-combination assignment pair with the largest cardinality of the task combination each time.

Initially, we generate task combination set TC_p (line 1). Then, for each round, we would select one task combination with the largest cardinality and the whole combination can be accomplished (line 2-10). Firstly, for each left task combination, we try to find a set of workers who can accomplish the task combination (line 5). There are many ways can achieve this function, in this paper, we use a simple heuristic greedy function which will be presented shortly. Then, if the set can be found, then add it into the candidate set C (line6-7). After that, we add the assignment of the task combination with the largest cardinality from the candidate set C to M_p (line 8). Next, we update the task combination set TC_p and the worker set W_p (line 9).

Algorithm 1: DASC_Greedy Algorithm

```
Input: A set $W_p$ of $n_p$ workers and a set $T_p$ of $m_p$ tasks
Output: An assignment $M_p$
1  generate the task combination set $TC_p$;
2  repeat
3      $C = \emptyset$;
4      foreach combination $tc \in TC_p$ do
5          find a set of workers who can accomplish the task
           combination;
6          if the set can be found then
7              add the assignment into the candidate set $C$;
8      add the combination $tc$ with the largest cardinality from
       the candidate set $C$ to $M_p$;
9      update $TC_p$ and $W_p$;
10 until there is no combination can be assigned
11 return $M_p$
```

3.3 Theory Analyses

We first discuss the running time of *DASC_Greedy* Algorithm.

Lemma 3.1. *The time complexity of* DASC_Greedy *Algorithm is* $O(\min\{n_p, m_p\} \cdot m_p \cdot |D|_{max} \cdot n_p \cdot |WS|_{max})$, *where* $|D|_{max}$ *is the maximum cardinality of one task's dependency set and* $|WS|_{max}$ *is the maximum cardinality of one worker's skill set.*

Proof. The loop will be ended until there is no task combination can be assigned. There are three cases: (1) the tasks are all assigned; (2) the workers are all assigned; (3) there are workers or tasks left, but these pairs cannot satisfy all constraints. Thus, the number of round can be bounded by $O(\min\{n_p, m_p\})$. In each round, there are $O(m_p)$ task combinations should be scanned (line 4). For each task combination, there are $O(|D|_{max})$ tasks need to find a worker to do the task. In order to find the workers, we have to scan the worker set W_p whose cardinality is $O(n_p)$ and check the skills that workers have which are bounded by $O(|WS|_{max})$. Thus, the time complexity of *DASC_Greedy* Algorithm is $O(\min\{n_p, m_p\} \cdot m_p \cdot |D|_{max} \cdot n_p \cdot |WS|_{max})$. □

After proving we can get the resulting solution within polynomial time, next we discuss how good the resulting solution is. Firstly, we prove that the objective function of DA-SC problem in Equation 1 is monotone and submodular. Then, we can transform the objective function as follows:

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

where $|tc|$ is the cardinality of the task combination.

Theorem 3.1. *$Sum(M)$ is monotone and submodular.*

Proof. Since $|tc|$ is always positive, it is easy to see that $Sum(M)$ is monotone. To proving its submodularity, we have that:

$$\forall tc_i \notin M, \forall tc_j \notin M,$$
$$Sum(\widehat{M}) - Sum(\widetilde{M}) \leq Sum(\overline{M}) - Sum(M)$$

where $\widehat{M} = M \cup tc_i \cup tc_j$, $\widetilde{M} = M \cup tc_i$ and $\overline{M} = M \cup tc_j$.

We prove the inequality by contradiction. Suppose to the contrary that $S(I \cup tc_i \cup tc_j) - S(I \cup tc_i) > S(I \cup tc_j) - S(I)$. There are two cases:

1. tc_j is assigned in $\widehat{M}$ and $\overline{M}$ and the cardinality of tc_j in $\widehat{M}$ is larger than that in $\overline{M}$. Since when we update the combination set TC after assigning the task combination tc_i, we only remove some tasks from some combinations. The cardinality of the task combination tc_j will not increase after assigning one another combination.

2. tc_j is assigned in $\widehat{M}$ and tc_j is not assigned in $\overline{M}$. In other words, we cannot find a set of workers to accomplish the whole task combination tc_j in $\overline{M}$ while we can find a set of workers to accomplish it in $\widehat{M}$. However, the worker set $\widehat{W_p}$ after assigning the task combination tc_i is a subset of the worker set $\overline{W_p}$ before assigning the task combination. In other words, if there is a set of workers in $\widehat{W_p}$ who can totally accomplish the task combination tc_j, then the set of workers is also in $\overline{W_p}$.

Thus, we prove that the inequality holds. □

Let $\rho_j(M) = Sum(M \cup tc_j) - Sum(M)$ and E is the universe of assignments. According to the monotonicity and submodularity of $Sum(M)$, we have the following lemma.

Lemma 3.2. *For any assignment M and M′, it holds,* $Sum(M') \leq Sum(M) + \sum_{tc \in M'-M} \rho_{tc}(M) - \sum_{tc \in M-M'} \rho_{tc}(M \cup M' - tc)$

Proof. Based $Sum(M)$'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:

$$\begin{aligned}
&Sum(M \cup M') - Sum(M)\\
=&\sum_{t=1}^{r}[Sum(M \cup \{j_1, \cdots, j_t\}) - Sum(S \cup \{j_1, \cdots, j_{t-1}\})]\\
=&\sum_{t=1}^{r} \rho_{j_t}(M \cup \{j_1, \cdots, j_{t-1}\})\\
\leq&\sum_{t=1}^{r} \rho_{j_t}(M) = \sum_{j \in M'-M} \rho_j(M)
\end{aligned}$$

Similarly, we also have:

$$\begin{aligned}
&Sum(M \cup M') - Sum(M')\\
=&\sum_{t=1}^{q}[Sum(M' \cup \{k_1, \cdots, k_t\}) - Sum(M' \cup \{k_1, \cdots, k_{t-1}\})]\\
=&\sum_{t=1}^{q} \rho_{k_t}(M' \cup \{k_1, \cdots, k_{t-1}\})\\
\geq&\sum_{t=1}^{q} \rho_{k_t}(M' \cup M - \{k_t\}) = \sum_{j \in M-M'} \rho_j(M \cup M' - \{j\})
\end{aligned}$$

By subtracting above two inequality, we have:

$$\begin{aligned}
&Sum(M')\\
\leq&Sum(M' \cup M) - \sum_{tc \in M-M'} \rho_{tc}(M \cup M' - tc)\\
\leq&Sum(M) + \sum_{tc \in M'-M} \rho_{tc}(M) - \sum_{tc \in M-M'} \rho_{tc}(M \cup M' - tc)
\end{aligned}$$

□

In addition, based on the observation that $0 \leq \rho_{tc}(M) \leq \psi$, $\forall M \subseteq E, tc \in E - M$, where ψ is the upper bound of $\rho_{tc}(M)$, we have

$$Sum(M') \leq Sum(M) + \sum_{tc \in M'-M} \rho_{tc}(M), \forall M, M' \subseteq E.$$

Note that $|M| \leq K$, where $K = \min\{n_p, m_p\}$. Let $Sum(M_{opt})$ be the value of an optimal solution and M^t be the assignment set which is generated after t rounds, i.e., $Sum(M^t) = \sum_{i=0}^{t-1} \rho_i$. Suppose the *DASC_Greedy* Algorithm stops after K^* rounds, then we have the following lemma.

Lemma 3.3. *Suppose tc_t is added by* DASC_Greedy *Algorithm in t^{th} round, let $\rho_{t-1} = \rho_{tc_t}(M^{t-1})$. The corresponding $\{\rho_i\}_{i=0}^{K^*-1}$ satisfy*

$$Sum(M_{opt}) \leq \sum_{i=0}^{t-1} \rho_i + K \cdot \rho_t, t = 0, \cdots, K^* - 1$$

Proof. Since $\rho_j(M) \leq \rho_t, \forall j \in T - S$, according to Lemma 3.2, we have:

$$\begin{aligned}
Sum(M_{opt}) &\leq Sum(M^t) + \sum_{tc \in M_{opt}-M^t} \rho_{tc}(M^t)\\
&\leq \sum_{i=0}^{t-1} \rho_i + \sum_{tc \in M_{opt}-M^t} \rho_t
\end{aligned}$$

In addition, $|M_{opt} - M^t| \leq K$. Thus,

$$Sum(M_{opt}) \leq \sum_{i=0}^{t-1} \rho_i + \sum_{tc \in M_{opt}-M^t} \rho_t \leq \sum_{i=0}^{t-1} \rho_i + K \cdot \rho_t$$

□

Let M_{greedy} be the assignment gotten by *DASC_Greedy* Algorithm. Then, we have the following theorem.

Theorem 3.2. *The matching size returned by* DASC_Greedy *Algorithm is at least $(1 - \frac{1}{e}) \cdot |M_{opt}|$.*

Proof. Based on the Lemma 3.3, we have

$$Sum(M_{opt}) \leq K \cdot \rho_0 = K \cdot (Sum(M^1))$$

$$Sum(M_{opt}) - Sum(M^1) \leq K \cdot \rho_1 = K \cdot (Sum(M^2) - Sum(M^1))$$

Similarly, we can get the inequalities when $t = 2, \cdots, K$. Thus, we have

$$\frac{Sum(M_{opt} - Sum(M_{greedy}))}{Sum(M_{opt})} \leq \left(\frac{K-1}{K}\right)^K$$

Thus, the matching size returned by *DASC_Greedy* Algorithm is at least $(1 - \frac{1}{e}) \cdot |M_{opt}|$. □

4. GAME THEORETIC APPROACH

From Example 1, we can observe that the task assigned to one worker will not only influence the utility of this worker but also influence the utility of other workers. In other words, the utility of each task-and-worker pair is estimated by the task and worker of this pair and the dependent task-and-worker pairs. Thus, there are two challenges need to be solved. The first is how to estimate the influence from other assignments and the second is how to make assignments with high total utility.

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
Assign a random strategy to each player
repeat
    foreach player $v \in V$ do
        compute $v$'s best strategy wrt the other players' strategies
        let $v$ follow his best strategy
until Nash equilibrium//no player has changed his strategy
return the strategy of each player
```

Algorithm 3: DASC_Game Algorithm

```
Input: A set $W_p$ of $n_p$ workers and a set $T_p$ of $m_p$ tasks
Output: An assignment $M_p$
foreach each worker $w \in W$ do
    assign $w$ with a random task $t \in S_w$;
repeat
    foreach worker $w \in W$ do
        $maxUtility = -\infty$;
        foreach task $t \in S_w$ do
            compute worker $w$'s individual utility function $U_w(s_w, \overline{s_w})$;
            if $U_w(s_w, \overline{s_w}) > maxUtility$ then
                $maxUtility = U_w(s_w, \overline{s_w})$;
                $s_w = t$;
until Nash equilibrium
According to the Nash equilibrium, get the assignment set $I$;
return $I$
```

In this section, we develop a batch-based solution based on a game theoretic framework to solve the above issues. We process the requests batch by batch. In addition, We model the problem as a game, where each worker corresponds to a player: his goal is to find the task that maximize his own utility. We will show latter, the game reaches an equilibrium, i.e., a local maximum where no worker has incentive to deviate.

4.1 Game Theory

Algorithm 2 illustrates a common framework for studying the dynamic process of decision-making in a strategic game [12]. In *strategic games*, players compete with each other over the same resources in order 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 the 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 [13], Nash points that a strategic game has a pure Nash equilibrium, if there exist a specific choice of strategies $s_w \in S_w$ such that the following condition is true for all $w \in W$:

$$U_w(s_1, \cdots, s_w, \cdots, s_{|W|}) \leq U_w(s_1, \cdots, s'_w, \cdots, s_{|W|}), \forall s'_w \in S_w$$

In other words, no player has incentive to deviate from his current strategy.

In order to express the objective functions of all the workers, [14] proposed a single function $\Phi : \times_{w \in W} S_w \to \mathbb{R}$, called the *potential function*, in *potential games*, which constitute 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 $\overline{s_i} \in \times_{j \in N \setminus \{i\}} S_j$, the following condition holds:

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

[14] proves that for potential games, the best-response dynamics of Algorithm 2 always converge to a pure Nash equilibrium. Therefore, we can use the best-response framework of Algorithm 2 to obtain the solution for DA-SC problem.

4.2 Game-Theoretic Algorithm

We model the DA-SC problem as a game, which is represented by a tuple $\langle W_p, S_{w\,w \in W_p}, \{U_w : \times_{w \in W_p} S_w \to \mathbb{R}\}_{w \in W_p}\rangle$, where W_p is the worker set in the batch b. W_p contains the workers who appear in the previous batches and still wait for assignments and the new workers who just appear in this batch. Similarly, T_p denotes the task set in the batch b. Specifically, each worker $w \in W_p$ has a set of strategies $S_w \in T_p$. 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 np_{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 sum of (i) the shared utility of task s_w, and (ii) the contribution to the tasks which are depended on s_w. The goal of each worker $w \in W$ is to find the task s_w that maximize his own total utility as expressed by Equation 2.

$$U_w(s_w, \overline{s_w}) \begin{cases} \frac{(\alpha-1)\cdot\prod_{f \in D_{s_w}} a_f}{\alpha\cdot np_{s_w}} + \sum\limits_{s_w \in D_t} \frac{\prod_{f \in D_t \cup \{t\}} a_f}{\alpha\cdot|D_t|\cdot np_{s_w}}, & D_{s_w} \neq \emptyset \\ \frac{1}{np_{s_w}} + \sum\limits_{s_w \in D_t} \frac{\prod_{f \in D_t \cup \{t\}} a_f}{\alpha\cdot|D_t|\cdot np_{s_w}}, & D_{s_w} = \emptyset \end{cases} \tag{2}$$

where α is the normalization parameter.

Significantly, we have the observation that the objective function of DA-SC problem in Equation 1 is equal to the sum of all individual worker utility, i.e., $Sum(M) = \sum_{w \in W} U_w(s_w, \overline{s_w})$. This decomposition of the DA-SC objective into a sum of individual utility functions provides a natural motivation for modeling DA-SC as a game, since workers' own goals are considered for reaching a global solution. In addition, we define the potential function $\Phi(S)$ as follows:

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

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, each iteration computes, for each worker w, the utility of assigning worker w with each possible tasks and assign the task with the maximum utility to the worker w. The iteration terminates when there is no task change for any worker during a round. Then, according to $s_w \forall w \in W_p$, we get the assignment set M_p(line 12). Specifically, we filter out the assignments that the task's dependencies are not fully satisfied and randomly select one worker to do the task when there are more than one worker try to do the task.

Normalization Issues. In some cases, the first term and the second term in Equation 2 may not be comparable. For instance, In some complex cases, one task may be in thousands tasks' dependency set.Then many workers are likely to be assigned to the tasks who are thousands tasks' dependency set regardless the number of workers who are assigned to the same tasks because the total utility is dominated by the second term. To avoid these cases, we use the normalization parameter α to guarantee that the second term's range is $[0, 1]$, i.e., it should hold that $\sum_{s_w \in D_t} \frac{1}{\alpha \cdot |D_t|} \leq 1$. In other words, $\alpha \geq \frac{ND_{max}}{D_{max}}$, where ND_{max} is the maximum number of dependency sets that the same task belongs to and D_{max} is the maximum cardinality of one task's dependency set.

4.3 Theory Analyses

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

Theorem 4.1. *DA-SC problem constitutes an exact potential game.*

Proof. Recall from Section 4.1, 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, 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 np_{s_w} and $np_{s'_w}$ 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, $\overline{np_{s_w}}$ and $\overline{np_{s'_w}}$ 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, $np_{s_w} = \overline{np_{s_w}} + 1$ and $\overline{np_{s'_w}} = np_{s'_w} + 1$. Indeed, when $D_{s_w} \neq \emptyset$ and $D_{s'_w} \neq \emptyset$, we have:

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

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 A of our technical report [15].

Then, we proved DA-SC problem is an exact potential game. □

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

Lemma 4.1. *The number of rounds required by* DASC_Game *Algorithm in each batch converges to an equilibrium is upper bounded by* $d \cdot \min\{n_p, m_p\}$.

Proof. The scaled version of *DASC_Game* Algorithm with potential function

$$\Phi_{\mathbb{Z}}(S) = d \cdot \Phi(S)$$

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. Therefore, the upper bound of the number of rounds is the maximum value, Φ_{max}, minus the minimum value, Φ_{min}. We can easily see that in *DASC_Game* Algorithm, $\Phi_{max} \leq 0$ and $\Phi_{min} \geq -d \cdot \min\{n_p, m_p\}$. Consequently, the number of rounds to reach an equilibrium is upper-bounded by $d \cdot \min\{n_p, m_p\}$, as stated by the lemma. □

Thus, we can prove the time complexity of *DASC_Game* Algorithm.

Lemma 4.2. *The time complexity of* DASC_Game *Algorithm in each batch is* $O(d \cdot |D|_{max} \cdot |S|_{max} \cdot n_p \cdot \min\{n_p, m_p\})$, *where* $|D|_{max}$ *is the maximum cardinality of one task's dependency set and* $|S|_{max}$ *is the maximum cardinality of one worker's candidate task set.*

Proof. As discussed above, the number of iteration is upper bounded by $O(d \cdot \min\{n_p, m_p\})$. In addition, the best response of a user w 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). Therefore, the time complexity is $O(d \cdot |D|_{max} \cdot |S|_{max} \cdot n_p \cdot \min\{n_p, m_p\})$. □

After proving we can find the Nash equilibrium within polynomial time, next 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 social optimum and the PoA of a game is the ratio between the worst value among its equilibriums and the social optimum.

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

Lemma 4.3.

$$\frac{1}{2\cdot np_{max}}\cdot U(S)\leq|\Phi(S)|\leq\frac{n_p}{\overline{np}\cdot(n_p-\overline{np})}\cdot U(S)$$

where $\overline{np}=\min\{np_{min},n_p-np_{max}\}$*,* np_{max} *is the maximum number of workers who are assigned to the same task and* np_{min} *is the minimum number of workers who are assigned to the same task.*

Proof. Since $np_t\geq 1$, $\frac{1}{np_t+1}\geq\frac{1}{2\cdot np_t}$. Then,

$$\begin{aligned}&\sum_{w\in W_p}\sum_{\substack{t\in\cup S_w\\ \wedge t\neq s_w}}\frac{\prod_{f\in D_t\cup\{t\}}a_f}{(np_t+1)\cdot(n_p-np_t)}\\ =&\sum_{t\in\cup S_w}\frac{\prod_{f\in D_t\cup\{t\}}a_f}{np_t+1}\\ \geq&\sum_{t\in\cup S_w}\frac{\prod_{f\in D_t\cup\{t\}}a_f}{2\cdot np_t}\\ \geq&\frac{1}{2\cdot np_{max}}\cdot\sum_{t\in\cup S_w}\prod_{f\in D_t\cup\{t\}}a_f\end{aligned}$$

Thus, $|\Phi(S)|\geq\frac{1}{2\cdot np_{max}}\cdot U(S)$. In addition, we have:

$$\begin{aligned}&\sum_{w\in W_p}\sum_{\substack{t\in\cup S_w\\ \wedge t\neq s_w}}\frac{\prod_{f\in D_t\cup\{t\}}a_f}{(np_t+1)\cdot(n_p-np_t)}\\ \leq&\sum_{w\in W_p}\sum_{\substack{t\in\cup S_w\\ \wedge t\neq s_w}}\frac{\prod_{f\in D_t\cup\{t\}}a_f}{np_t\cdot(n_p-np_t)}\\ \leq&\sum_{w\in W_p}\sum_{t\in\cup S_w}\frac{\prod_{f\in D_t\cup\{t\}}a_f}{np_t\cdot(n_p-np_t)}\\ =&\sum_{t\in\cup S_w}\frac{n_p\cdot\prod_{f\in D_t\cup\{t\}}a_f}{np_t\cdot(n_p-np_t)}\\ \leq&\frac{n_p}{\overline{np}\cdot(n_p-\overline{np})}\cdot\sum_{t\in\cup S_w}\prod_{f\in D_t\cup\{t\}}a_f\end{aligned}$$

Thus, we have $|\Phi(S)|\leq\frac{n_p}{\overline{np}\cdot(n_p-\overline{np})}\cdot U(S)$. □

We match the task set T_p and worker set W_p in each batch just considering the skill by solving bipartite graph problem. The number of the assigned tasks whose dependencies are fully satisfied is represented by m_p^*.

Theorem 4.2. *In* DASC_Game*, the* PoS *of each batch is bounded by* $\frac{\overline{np}\cdot(n_p-\overline{np})}{2\cdot n_p\cdot np_{max}}$*. In addition, the* PoA *of each batch is bounded by* $\frac{(\alpha-1)\cdot m_p^*}{\alpha\cdot\min\{n_p,m_p\}}$.

Proof. Let S^* be 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 4.3 and since $U(S^*)\geq U(S),\forall S$ and $|\Phi(S^{**})|\geq|\Phi(S)|,\forall S$, we have:

$$\begin{aligned}U(S^{**})&\geq\frac{\overline{np}\cdot(n_p-\overline{np})}{n_p}\cdot|\Phi(S^{**})|\\ &\geq\frac{\overline{np}\cdot(n_p-\overline{np})}{n_p}\cdot|\Phi(S^{*})|\\ &\geq\frac{\overline{np}\cdot(n_p-\overline{np})}{2\cdot n_p\cdot np_{max}}\cdot U(S^*)\end{aligned}$$

Since $OPT=U(S^*)$, $PoS=\frac{U(S^{**})}{OPT}\geq\frac{\overline{np}\cdot(n_p-\overline{np})}{2\cdot n_p\cdot np_{max}}$.

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

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

Choosing s'_w to be the task with the smallest number of workers who are assigned to the same task (i.e., $s'_w=arg\min_{s_w\in S_w}np_{s_w}$), we have:

$$\begin{aligned}&U(S^{\#})\\ =&\sum_{w\in W_p}U_w(s_w^{\#},\overline{s_w^{\#}})\geq\sum_{w\in W_p}U_w(s'_w,\overline{s_w^{\#}})\\ =&\sum_{w\in W_p}\left(\frac{(\alpha-1)\cdot\prod_{f\in D_{s_w}}a_f}{\alpha\cdot np_{s_w}}+\sum_{s_w\in D_t}\frac{\prod_{f\in D_t\cup\{t\}}a_f}{\alpha\cdot|D_t|\cdot np_{s_w}}\right)\\ &+\sum_{\substack{w\in W_p\\ D_{s_w}=\emptyset}}\frac{\prod_{f\in D_{s_w}}a_f}{\alpha\cdot np_{s_w}}\\ \geq&\sum_{w\in W_p}\frac{(\alpha-1)\cdot\prod_{f\in D_{s_w}}a_f}{\alpha\cdot np_{s_w}}\\ \geq&\frac{(\alpha-1)\cdot m_p^*}{\alpha}\end{aligned}$$

Since $U(S^*)\leq\min\{n_p,m_p\}$, we have:

$$PoA=\frac{U(S^{\#})}{U(S^*)}\geq\frac{(\alpha-1)\cdot m_p^*}{\alpha\cdot\min\{n_p,m_p\}}$$

□

5. EXPERIMENTAL STUDY

5.1 Data Sets

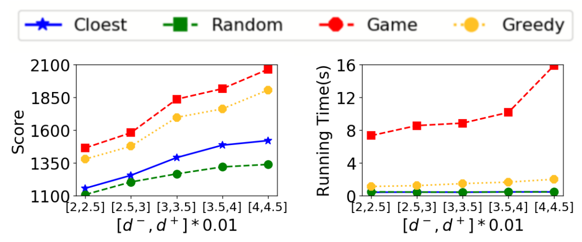
We use both real and synthetic data to test our proposed DA-SC approaches. Specifically, for real data, we use Meetup data set from [16], which was crawled from meetup.com between Oct. 2011 and Jan. 2012. There are 5,153,886 users, 5,183,840 events, and 97,587 groups in the Meetup data set, where each 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. We use the locations and tags of users in the Meetup data set to initialize the locations and the practiced skills of workers in our DA-SC problem, where each tag is considered as a skill in our experiments. In addition, we utilize the locations and tags of groups which create the events to initialize the locations, the required skills and the dependencies of tasks in our experiments. 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

Table 4: Experimental Settings on Real Data.

Parameters	Values
the start time range $[st^-, st^+]$	[0, 50], [0, 100], **[0, 150]**, [0, 200], [0, 250]
the waiting time range $[wt^-, wt^+]$	[1, 3], [3, 5], **[5, 7]**, [7, 9], [9, 11]
the velocity range $[v^-, v^+] * 0.01$	[0.1, 0.5], [0.5, 1], **[1, 1.5]**, [1.5, 2], [2, 2.5]
the distance range $[d^-, d^+] * 0.01$	[2, 2.5], [2.5, 3], **[3, 3.5]**, [3.5, 4], [4, 4.5]

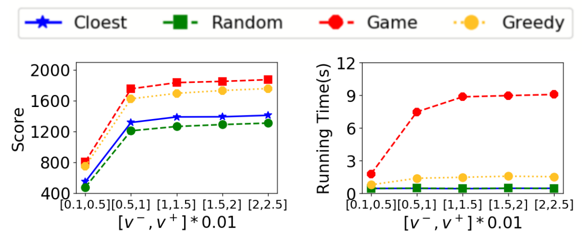
Table 5: Experimental Settings on Synthetic Data.

Parameters	Values
the number, m, of workers	1K, 3K, **5K**, 8K, 10K
the number, n, of tasks	1K, 3K, **5K**, 8K, 10K
the start time range $[st^-, st^+]$	[0, 25], [0, 50], **[0, 75]**, [0, 100], [0, 125]
the waiting time range $[wt^-, wt^+]$	[1, 5], [5, 10], **[10, 15]**, [15, 20], [20, 25]
the velocity range $[v^-, v^+] * 0.01$	[1, 2], [2, 3], **[3, 4]**, [4, 5], [5, 6]
the distance range $[d^-, d^+] * 0.1$	[3, 4], [4, 5], **[5, 6]**, [6, 7], [7, 8]
the dependency size range	[0,50], [0,60], **[0, 70]**, [0, 80], [0, 90]
the number, r, of skill universe	500, 1000, **1500**, 2000, 2500
the skill set range for each worker	[1, 5], [1, 10], [1, 15], [1, 20], [1, 25]



(a) Score (b) Running Time

Figure 2: Effect of the Moving Distance Range $[d^-, d^+]$ (Real)



(a) Score (b) Running Time

Figure 3: Effect of the Velocity Range $[v^-, v^+]$ (Real)

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 $114.283°$), in which we obtain 1,282 tasks and 3,525 workers.

For real data, we generate tasks' dependency based on events' groups. Specifically, we consider one event as a task group and the tag set of the event as the required skill set of this task group. Then, we generate the dependent tasks in each task group. Each task requires one skill in the required skill set of the task group and its dependency set is the subset of this task group. Without loss of generality, the structure of tasks' dependencies is one directed acyclic graph.

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 cardinality of the skills' universe to mimic a wide scale of real-world application scenarios. In addition, we simulate the cardinality 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 cardinality of each task's dependency with the uniform distribution within the range from $[0, 50]$ to $[0, 90]$.

For both real and synthetic data sets, we simulate the velocity of each worker with the uniform distribution within the range $[v^-, v^+]$ from $[0.01, 0.02]$ to $[0.05, 0.06]$. 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 $[wt^-, wt^+]$.

5.2 Approaches and Measurements

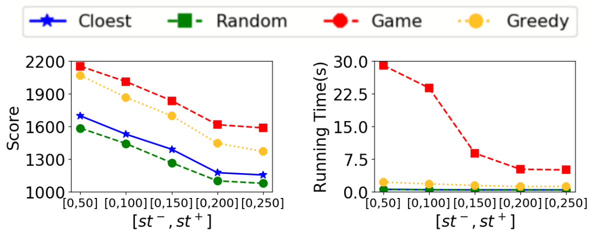
We conduct experiments to evaluate the effectiveness and efficiency of our two approaches, which includes *DASC_Game* and *DASC_Greedy*, in terms of the cardinality of the valid assignment and the running time. As proved in Section 2.4, the DA-SC problem is NP-hard, thus it is infeasible to calculate the optimal result as the ground truth. Alternatively, we compare our approaches with 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.

Table 4 and Table 5 depict our experimental settings on real data and synthetic data, where the default values of parameters are in bold font. In each set of experiments, we vary one parameter, while setting other parameters to their default values. For each experiment, we report the running time and the assignment score (the cardinality of the valid assignment) of our tested approaches. All our experiments were run on an Intel i7 CPU @2.2 GHz with 16GM RAM in Java.

5.3 Results on Real Datesets

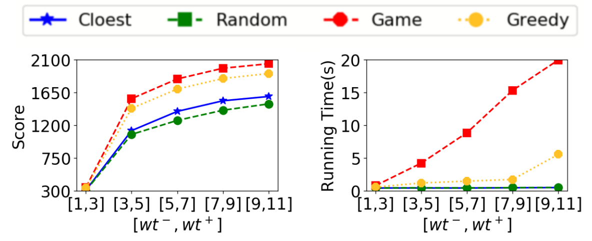
Effect of the Range of the Maximum Moving Distance $[d^-, d^+]$. Figure 2 illustrates the experimental results on different ranges, $[d^-, d^+]$, of each worker's maximum moving distance d from $[0.02, 0.025]$ to $[0.04, 0.045]$. In Figure 2(a), the assignment scores of all the four approaches increase, 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 *Game Algorithm* achieves the highest score and the scores of two proposed algorithms are much better than those of two baseline algorithms. As shown in Figure 2(b), the running times of our two approaches increase, when the range of maximum moving distance gets larger. The reason is that, when the maximum 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. Specifically, *Greedy Algorithm* achieves much lower running time than *Game Algorithm*.

Effect of the Range of the Velocity $[v^-, v^+]$. Figure 3 illustrates the experimental results on different ranges, $[v^-, v^+]$, of each worker's velocity v from $[0.001, 0.005]$ to $[0.02, 0.025]$. In Figure 3(a), when the value range of velocity gets larger from $[0.001, 0.005]$ to $[0.005, 0.01]$, the assignment scores of all the four approaches first increase; then, they almost keep stable for the velocity range varying from $[0.005, 0.01]$ to $[0.02, 0.02]5$. The reason is that, at beginning, with the increase of v, workers can reach more tasks on time. Nevertheless, workers are also constrained by other constraints, e.g., maximum moving distance. As shown in Figure 3(b), the running times of our two approaches increase, when the range of velocity gets larger. The reason is that, when the velocity 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 v is relaxed, the constraints



(a) Score (b) Running Time

Figure 4: Effect of the Start Time Range $[st^-, st^+]$ (Real)



(a) Score (b) Running Time

Figure 5: Effect of the Waiting Time Range $[wt^-, wt^+]$ (Real)

from other parameters prevent the running time keeping growing.

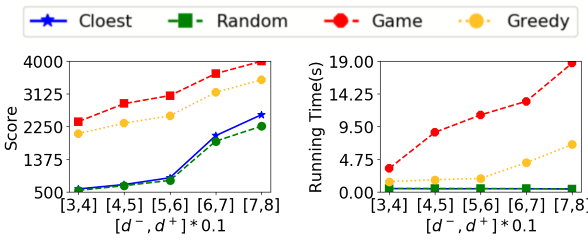
Effect of the Range of the Start Timestamp $[st^-, st^+]$. Figure 4 illustrates the experimental results on different ranges, $[st^-, st^+]$, of each worker's/task's start timestamp *st* from $[0, 50]$ to $[0, 250]$. In Figure 4(a), the assignment scores of all the four approaches decrease, when the 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 *Game Algorithm* achieves the highest score and the scores of two proposed algorithms are much better than those of two baseline algorithms. As shown in Figure 4(b), the running times of our two approaches decrease, when the range of start timestamp gets larger. The reason is that, when the range increase, each worker has less valid tasks which thus leads to lower complexity of the *DA-SC* problem and the decrease of the running time. Specifically, *Greedy Algorithm* achieves much lower running time than *Game Algorithm*. In addition, *Game Algorithm* is more sensitive to the range of start timestamp than *Game Algorithm*, because when the range is bigger, the search space decreases dramatically.

Effect of the Range of the Waiting Time $[wt^-, wt^+]$. Figure 5 illustrates the experimental results on different ranges, $[wt^-, wt^+]$, of each worker's/task's waiting time wt from $[1, 3]$ to $[9, 11]$. In Figure 5(a), the assignment scores of all the four 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 *Game Algorithm* achieves the highest score and the scores of two proposed algorithms are much better than those of two baseline algorithms. As shown in Figure 5(b), the running times of our two 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 higher complexity of the *DA-SC* problem and the increase of the running time. Specifically, *Greedy Algorithm* achieves much lower running time than *Game Algorithm*.

5.4 Results on Synthetic Date sets

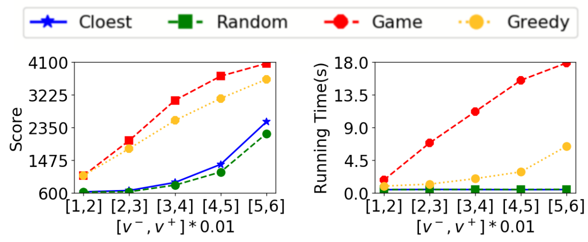
Effect of the Range of the Maximum Moving Distance $[d^-, d^+]$. Figure 6 illustrates the experimental results on different ranges, $[d^-, d^+]$, of each worker's maximum moving distance d from $[0.3, 0.4]$ to $[0.7, 0.8]$. In Figure 6(a), the assignment scores of all the four approaches increase, when the value range of maximum moving distance gets larger. The reason is that, the increase of d enlarges the access range of workers. The *Game Algorithm* achieves the highest score and the scores of two proposed algorithms are much better than those of two baseline algorithms. As shown in Figure 6(b), the running times of our two approaches increase, when the range of maximum moving distance gets larger. The reason is that, when the maximum 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. Specifically, *Greedy Algorithm* achieves much lower running time than *Game Algorithm*.

Effect of the Range of the Velocity $[v^-, v^+]$. Figure 7 illustrates the experimental results on different ranges, $[v^-, v^+]$, of each worker's velocity v from $[0.01, 0.02]$ to $[0.05, 0.06]$. In Figure 7(a), the assignment scores of all the four approaches increase, when the value range of velocity gets larger. The reason is that, the increase of d let workers can reach more tasks on time. The *Game Algorithm* achieves the highest score and the scores of two proposed algorithms are much better than those of two baseline algorithms. As shown in Figure 7(b), the running times of our two approaches increase, when the range of velocity gets larger. The reason is that, when the velocity 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. Specifically, *Greedy Algorithm* achieves much lower running time than *Game Algorithm*.



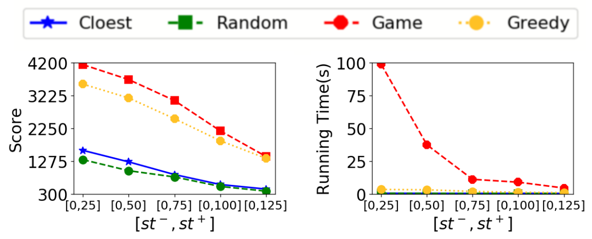
(a) Score (b) Running Time

Figure 6: Effect of the Moving Distance Range $[d^-, d^+]$ (Synthetic)



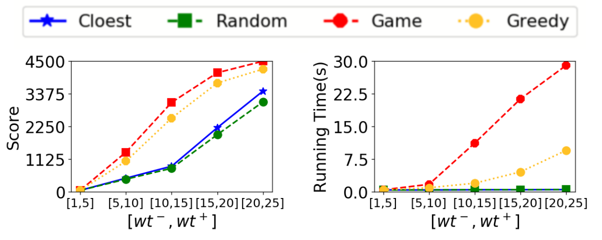
(a) Score (b) Running Time

Figure 7: Effect of the Velocity Range $[v^-, v^+]$ (Synthetic)



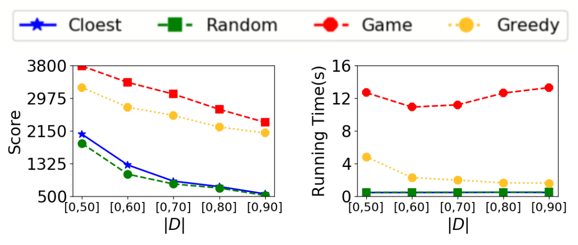
(a) Score (b) Running Time

Figure 8: Effect of the Start Time Range $[st^-, st^+]$ (Synthetic)



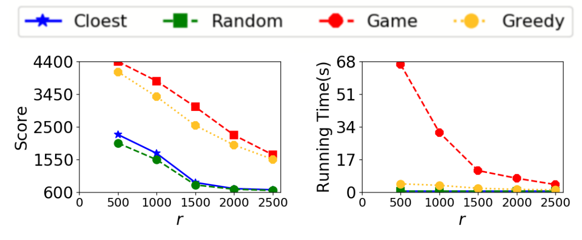
(a) Score (b) Running Time

Figure 9: Effect of the Waiting Time Range $[wt^-, wt^+]$ (Synthetic)



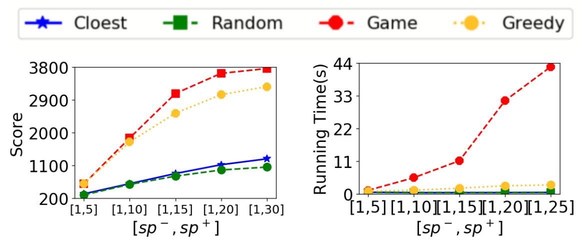
(a) Score (b) Running Time

Figure 10: Effect of the Dependency Size Range (Synthetic)



(a) Score (b) Running Time

Figure 11: Effect of the Cardinality of the Skill Universe (Synthetic)



(a) Score (b) Running Time

Figure 12: Effect of each Worker's Skill Size Range (Synthetic)

Effect of the Range of the Start Timestamp $[st^-, st^+]$. Figure 8 illustrates the experimental results on different ranges, $[st^-, st^+]$, of each worker's/task's start timestamp st from $[0, 25]$ to $[0, 125]$. In Figure 8(a), the assignment scores of all the four 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 *Game Algorithm* achieves the highest score and the scores of two proposed algorithms are much better than those of two baseline algorithms. As shown in Figure 8(b), the running times of our two approaches decrease, when the range of start timestamp gets larger. The reason is that, when the range increase, each worker has less valid tasks which thus leads to lower complexity of the *DA-SC* problem and the decrease of the running time. Specifically, *Greedy Algorithm* achieves much lower running time than *Game Algorithm*. In addition, *Game Algorithm* is more sensitive to the range of start timestamp than *Game Algorithm*, because when the range is bigger, the search space decreases dramatically.

Effect of the Range of the Waiting Time $[wt^-, wt^+]$. Figure 9 illustrates the experimental results on different ranges, $[wt^-, wt^+]$, of each worker's/task's waiting time wt from $[1, 5]$ to $[20, 25]$. In Figure 9(a), the assignment scores of all the four 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 *Game Algorithm* achieves the highest score and the scores of two proposed algorithms are much better than those of two baseline algorithms. As shown in Figure 9(b), the running times of our two 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 higher complexity of the *DA-SC* problem and the increase of the running time. Specifically, *Greedy Algorithm* achieves much lower running time than *Game Algorithm*.

Effect of the Range of dependency set size. Figure 10 illustrates the experimental results on different ranges, of each task's dependency set size $|D|$ from $[0, 50]$ to $[0, 90]$. In Figure 10(a), the assignment scores of all the four 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 is more sensitive to the increase, whose score decrease sharply with the increase of $|D|$. As shown in Figure 10(b), the running time of *Game Algorithm* keeps stable because the increase of $|D|$ does not change its search space. But the running time of *Greedy Algorithm* decrease with the increase of $|D|$, because of the number of accomplished tasks decreases.

Effect of the Cardinality of the Skill Universe r. Figure 11 illustrates the experimental results on different cardinality, r, of the skill universe Ψ from 500 to 2500. In Figure 11(a), the assignment scores of all the four approaches decrease, when the cardinality of the skill universe gets larger. The reason is that, the increase of r disperses the tasks/workers over skills. The *Game Algorithm* achieves the highest score and the scores of two proposed algorithms are much better than those of two baseline algorithms. As shown in Figure 11(b), the *Game Algorithm* is very sensitive to the change of r, since with the decrease of r, each worker has more valid tasks can accomplish and the algorithm's search space become larger.

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 four 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 *Game Algorithm* achieves the highest score and the scores of two proposed algorithms are much better than those of two baseline algorithms. As shown in Figure 12(b), the running times of our two approaches increase, when the range of each worker's skill size gets larger. Specifically, the *Game Algorithm* is very sensitive to the change of $|WS|$, since with the increase of $|WS|$, each worker has more valid tasks can accomplish and the algorithm's search space become larger.

Effect of the Number of Tasks m. Figure 13 illustrates the experimental results on different number, m, of tasks from $1K$ to $10K$. In Figure 13(a), the assignment scores of all the four approaches increase, when the number of tasks gets larger. The reason is that, the increase of m let each worker has more valid tasks. Specifically, the two baseline algorithms are not sensitive to the in-

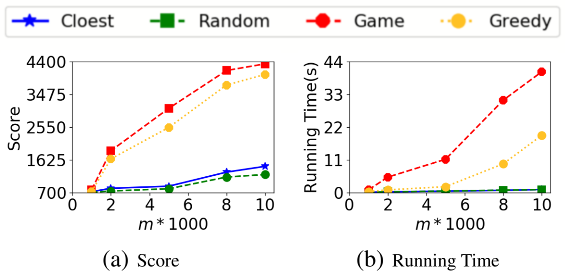


Figure 13: Effect of the Number of Tasks (Synthetic)

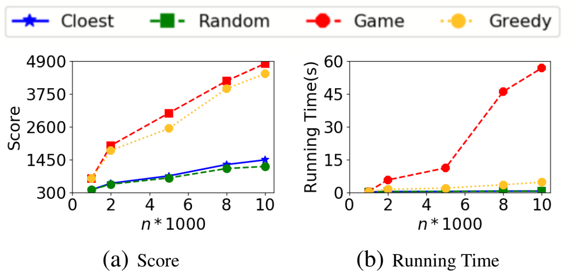


Figure 14: Effect of the Number of Workers (Synthetic)

crease of m since they do not consider the dependency constraints when they allocate the tasks and their many allocations are invalid. As shown in Figure 13(b), the running times of our two approaches increase, when the number of tasks gets larger. The reason is that, when the number of tasks 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. Specifically, *Greedy Algorithm* achieves much lower running time than *Game Algorithm*.

Effect of the Number of Workers n. Figure 14 illustrates the experimental results on different number, n, of workers from $1K$ to $10K$. In Figure 14(a), the assignment scores of all the four approaches increase, when the number of workers gets larger. The reason is that, the increase of n let each task has more valid workers. Specifically, the two baseline algorithms are not sensitive to the increase of n since they do not consider the dependency constraints when they allocate the tasks and their many allocations are invalid. As shown in Figure 14(b), the *Game Algorithm* is very sensitive to the increase of n, since with the increase of n, there are more workers and the algorithm's search space become larger.

5.5 Summary on Experiment Results

We finally summarize our findings.

1. Both two proposed algorithms are efficient and they process tasks' dependency constraints well, which results in a larger assignment set size.
2. Both two proposed algorithms works well on real and synthetic data sets.
3. The running time of the *Game Algorithm* keeps stable with the increase of each task's dependency size. In other words, it is more economic that implementing *Game Algorithm* in the complex scenarios than implementing it in the simple scenarios.
4. The *Greedy Algorithm* runs much faster than the *Game Algorithm* while its result is similar with the *Game Algorithm*.

6. RELATED WORK

Crowdsourcing has attracted considerable attention due to its high practicality for real-world applications. Without considering the location constraint, previous work [17, 18, 19] studied the task assignment in crowdsourcing to finish the tasks more efficiently and accurately. In addition, exiting studies [20, 21] focused on selecting a set of proper workers for a particular task such that the expected accuracy of the result is optimal. In addition, the existing work [22] in general crowdsourcing only use the location information as a parameter to better infer the results.

In spatial crowdsourcing [1], workers are requested to physically move to specific locations to conduct tasks on sites. Based on the ontology [23], 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 [24] 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, 23, 25, 8, 26] 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. [25] 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. [26] 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 [7, 8] 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.

7. 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 cardinality 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. Finally, we demonstrate the effectiveness of the proposed approaches with extensive experiments on both real and synthetic data sets.

8. REFERENCES

[1] L. Kazemi and C. Shahabi, "Geocrowd: enabling query answering with spatial crowdsourcing," in *Proceedings of the 20th international conference on advances in geographic information systems*, pp. 189–198, ACM, 2012.

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

[3] "[online] TaskRabbit." `https://www.taskrabbit.com/`, 2019.

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

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

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

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

[8] 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.

[9] H. Rahman, S. Thirumuruganathan, S. B. Roy, S. Amer-Yahia, and G. Das, "Worker skill estimation in team-based tasks," *Proceedings of the VLDB Endowment*, vol. 8, no. 11, pp. 1142–1153, 2015.

[10] U. Pferschy and J. Schauer, "The knapsack problem with conflict graphs.," *J. Graph Algorithms Appl.*, vol. 13, no. 2, pp. 233–249, 2009.

[11] G. L. Nemhauser, L. A. Wolsey, and M. L. Fisher, "An analysis of approximations for maximizing submodular set functionsi," *Mathematical programming*, vol. 14, no. 1, pp. 265–294, 1978.

[12] N. Armenatzoglou, H. Pham, V. Ntranos, D. Papadias, and C. Shahabi, "Real-time multi-criteria social graph partitioning: A game theoretic approach," in *Proceedings of the 2015 ACM SIGMOD International Conference on Management of Data*, pp. 1617–1628, ACM, 2015.

[13] J. F. Nash *et al.*, "Equilibrium points in n-person games," *Proceedings of the national academy of sciences*, vol. 36, no. 1, pp. 48–49, 1950.

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

[15] "[online] Technical Report." `https://www.fieldagent.net`, 2019.

[16] 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 *Proceedings of the 18th ACM SIGKDD international conference on Knowledge discovery and data mining*, pp. 1032–1040, ACM, 2012.

[17] R. Boim, O. Greenshpan, T. Milo, S. Novgorodov, N. Polyzotis, and W.-C. Tan, "Asking the right questions in crowd data sourcing," in *2012 IEEE 28th International Conference on Data Engineering*, pp. 1261–1264, IEEE, 2012.

[18] Y. Zheng, J. Wang, G. Li, R. Cheng, and J. Feng, "Qasca: A quality-aware task assignment system for crowdsourcing applications," in *Proceedings of the 2015 ACM SIGMOD International Conference on Management of Data*, pp. 1031–1046, ACM, 2015.

[19] J. Fan, G. Li, B. C. Ooi, K.-l. Tan, and J. Feng, "icrowd: An adaptive crowdsourcing framework," in *Proceedings of the 2015 ACM SIGMOD International Conference on Management of Data*, pp. 1015–1030, ACM, 2015.

[20] C. C. Cao, J. She, Y. Tong, and L. Chen, "Whom to ask?: jury selection for decision making tasks on micro-blog services," *Proceedings of the VLDB Endowment*, vol. 5, no. 11, pp. 1495–1506, 2012.

[21] Y. Zheng, R. Cheng, S. Maniu, and L. Mo, "On optimality of jury selection in crowdsourcing," in *Proceedings of the 18th International Conference on Extending Database Technology, EDBT 2015*, OpenProceedings. org., 2015.

[22] H. Hu, Y. Zheng, Z. Bao, G. Li, J. Feng, and R. Cheng, "Crowdsourced poi labelling: Location-aware result inference and task assignment," in *2016 IEEE 32nd International Conference on Data Engineering (ICDE)*, pp. 61–72, IEEE, 2016.

[23] H. To, C. Shahabi, and L. Kazemi, "A server-assigned spatial crowdsourcing framework," *ACM Transactions on Spatial Algorithms and Systems*, vol. 1, no. 1, p. 2, 2015.

[24] D. Deng, C. Shahabi, and U. Demiryurek, "Maximizing the number of worker's self-selected tasks in spatial crowdsourcing," in *Proceedings of the 21st acm sigspatial international conference on advances in geographic information systems*, pp. 324–333, ACM, 2013.

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

[26] Y. Tong, J. She, B. Ding, L. Wang, and L. Chen, "Online mobile micro-task allocation in spatial crowdsourcing," in *2016 IEEE 32Nd international conference on data engineering (ICDE)*, pp. 49–60, IEEE, 2016.

APPENDIX

A. PROOF OF THEORE 4.1

Theorem 4.1. DA-SC problem constitutes an exact potential game.

Proof. Recall from Section 4.1, 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, 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 np_{s_w} and $np_{s'_w}$ 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, $\overline{np_{s_w}}$ and $\overline{np_{s'_w}}$ 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, $np_{s_w} = \overline{np_{s_w}} + 1$ and $\overline{np_{s'_w}} = np_{s'_w} + 1$. Indeed, when $D_{s_w} \neq \emptyset$ and $D_{s'_w} \neq \emptyset$, we have:

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

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 A of our technical report [15].

When $D_{s_w} = \emptyset$ and $D_{s'_w} \neq \emptyset$, we have:

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

When $D_{s_w} \neq \emptyset$ and $D_{s'_w} = \emptyset$, we have:

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

When $D_{s_w} = \emptyset$ and $D_{s'_w} = \emptyset$, we have:

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

Then, we proved DA-SC problem is an exact potential game. □