Coalition-based Task Assignment in Spatial Crowdsourcing

Yan Zhao
Department of Computer Science
Aalborg University
Aalborg, Denmark
yanz@cs.aau.dk

Jiannan Guo
China Mobile Cloud Centre
Suzhou, China
guojiannan@cmss.chinamobile.com

Xuanhao Chen
University of Electronic Science
and Technology of China
Chengdu, China
xhc@std.uestc.edu.cn

Jianye Hao
Tianjin University
Tianjin, China
jianye.hao@tju.edu.cn

Xiaofang Zhou
University of Queensland
Brisbane, Australia
zxf@itee.uq.edu.au

Kai Zheng*
University of Electronic Science
and Technology of China
Chengdu, China
zhengkai@uestc.edu.cn

***Abstract*—With the fast-paced development of mobile networks and the widespread usage of mobile devices, Spatial Crowdsourcing (SC), which refers to assigning location-based tasks to moving workers, has drawn increasing attention in recent years. One of the critical issues in SC is task assignment that allocates tasks to appropriate workers. In this paper, we propose a novel SC problem, namely Coalition-based Task Assignment (CTA), where the spatial tasks (e.g., house removals, furniture installation) may require more than one workers (forming a coalition) to cooperate in order to maximize the overall rewards of workers. To tackle the CTA problem, we design both greedy method and equilibrium-based method. In particular, the greedy method aims to form a set of worker coalitions greedily to perform the tasks, in which we introduce an acceptance possibility to find the high-value task assignments. In the equilibrium-based algorithm, workers form coalitions in sequence and update their strategy (i.e., selecting a best-response task) at their turn, in order to maximize their own utility (i.e., reward of the coalition they stay in) until Nash equilibrium is reached. Since the equilibrium point obtained by the best-response approach is not unique and optimal in terms of total rewards, we further propose a simulated annealing scheme to find a better Nash equilibrium. The extensive experiments demonstrate the efficiency and effectiveness of the proposed methods on both real and synthetic datasets.**

***Index Terms*—coalition, task assignment, spatial crowdsourcing**

I. Introduction

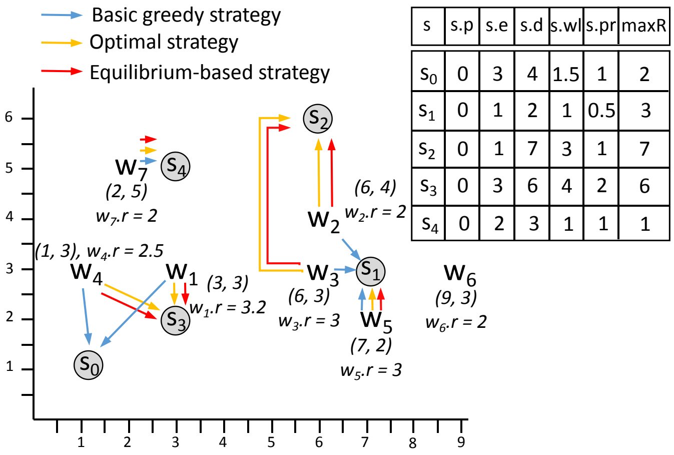
Spatial Crowdsourcing (SC) is a new class of crowdsourcing that has enabled people to move as multi-modal sensors collecting and sharing various types of high-fidelity spatio-temporal data instantaneously. Specifically, task requesters can issue spatial tasks to the SC server, and then the server employs smart device carriers as workers to physically travel to the specified locations and accomplish these tasks, referred to as *task assignment*.

A number of existing studies place their focus on single task assignment, in which each task can only be assigned to a single worker [1]–[5]. Inevitably, however, there exist some SC applications, in which an individual worker cannot efficiently conduct the task by herself, e.g., house removals, major furniture installations, monitoring traffic condition for an area, and holding a barbecue party [6]–[8]. Therefore, workers have to form a coalition to jointly complete these complex tasks that exceed the capabilities of individual workers. Moreover, for each worker, she may prefer to collaborate with other workers for reputation or economic purpose.

In this paper, we investigate the task assignment of SC under such a problem setting, namely Coalition-based Task Assignment (CTA). To be more specific, given a set of workers and a set of tasks, it aims at assigning a stable worker coalition for each task to achieve the highest total rewards. Some recent work has explored the multiple task assignment approaches that allow each task to be assigned to multiple workers [9]–[13]. But workers can conduct the tasks independently without cooperating, which is different from our problem. The most related work is [6] concerning the collaboration-aware task assignment, where workers are required to cooperate and accomplish the tasks jointly for achieving high total cooperation quality scores. However, they assume that workers always voluntarily conduct tasks, which is deemed unrealistic in practice since workers may have no motivation to perform the assigned tasks unless they receive satisfactory rewards. Besides, [6] only aims to find a single Nash equilibrium point without further exploration of more optimal equilibrium points that may exist. In this paper, we target a more realistic setting, in which workers will be given incentives if they can cooperate with each other to complete tasks, and aim to achieve a better Nash equilibrium with higher total rewards.

We will first illustrate the CTA problem through a motivation example in Figure 1, which involves seven workers (indicated as $\{w_1, ..., w_7\}$) and five tasks (indicated as $\{s_0, ..., s_4\}$). Each worker is associated with her current location and reachable distance (marked as $w.r$). Each task, published and expired at different time instances, is labelled with its

* Corresponding author: Kai Zheng.



s	s.p	s.e	s.d	s.wl	s.pr	maxR
s_0	0	3	4	1.5	1	2
s_1	0	1	2	1	0.5	3
s_2	0	1	7	3	1	7
s_3	0	3	6	4	2	6
s_4	0	2	3	1	1	1

Fig. 1. Running Example

workload ($s.wl$) and reward information (i.e., penalty rate $s.pr$ and maximum reward $s.maxR$). The reward of each task can be obtained by a reward pricing model depicted in Section II. The problem is to assign tasks to the suitable workers so as to maximize the total reward. For the sake of simplicity, we assume that all the workers share the same velocity, for which the travel time between two locations can be estimated with their Euclidean distance. In SC, it is an intuitive move to greedily assign the nearby workers to tasks (in order to obtain the maximal actual reward for each task) without violating the spatio-temporal constraint (i.e., the assigned tasks should be located in the reachable range of the corresponding workers and workers can arrive in the locations of assigned tasks before the deadline of tasks), referred to as basic greedy algorithm. Therefore, we can obtain a task assignment, $\{< s_0, \{w_1, w_4\} >, < s_1, \{w_2, w_3, w_5\} >, < s_4, \{w_7\} >\}$ (shown in blue arrow lines in Figure 1), with the overall reward of 5.76. Nevertheless, this assignment method leaves s_2 and s_3 (that have a high reward) unassigned, which may decrease the overall reward. Applying the optimal task assignment strategy, we can achieve the optimal task assignment, $\{< s_1, \{w_5\} >, < s_2, \{w_2, w_3\}, < s_3, \{w_1, w_4\} >, < s_4, \{w_7\} >\}$ (depicted in yellow arrow lines in Figure 1), the total reward of which is 12.26. However, the existing optimal task assignment algorithm tends to apply the techniques (e.g., dynamic programming) with huge computational cost [5], [14], which makes it unfit for practical applications.

Under the context of this distributed task allocation, there is a necessity for the workers to form coalitions with sufficient cumulative time or capabilities across the coalition members to accomplish the assigned tasks. In order to tackle this problem, we propose two novel coalition-based task assignment algorithms, i.e., greedy algorithm and equilibrium-based algorithm, for the purpose of achieving high total rewards. More specifically, the greedy algorithm is a non-reducing reward allocation strategy that incentivises workers to enlarge a coalition for more total rewards. By considering the time utilization ratio (measured by workers' workload and travel time) of workers and rate of return (measured by tasks' actual reward and maximal reward) of tasks, the greedy algorithm adopts an acceptance possibility to find the high-value task assignments (i.e., task assignments ensuring high time utilization ratio for workers and high rate of return for tasks). However, the greedy algorithm cannot guarantee the stability of the formed worker coalitions. Therefore, we further propose a task assignment method from a game-theoretic perspective. In particular, converting the CTA problem into a multi-player game, we develop an equilibrium-based algorithm by applying the best-response method with sequential and asynchronous updates of workers' strategies, which reaches the pure Nash equilibrium. The Nash equilibrium holds that if workers are all in Nash equilibrium, they are closely interconnected and the formed worker coalitions are stable, i.e., no worker can improve their utility by a unilateral change to their coalitions when other workers persist in their existing coalitions. As the task assignment result generated by the best-response method is locally optimal and there may exist multiple Nash equilibriums, we further propose the simulated annealing optimization strategy that can coordinate workers to obtain a better Nash equilibrium. In Figure 1, the overall reward generated by the equilibrium-based algorithm is 12.26, which is the maximal reward. This is due to the fact that there is only one equilibrium point (the optimal point) in such a small-scale example.

The contributions can be summarized as follows:

1) We formulate a novel task assignment in SC, namely Coalition-based Task Assignment (CTA), where workers need to interact with others by forming worker coalitions to conduct the corresponding tasks.

2) A greedy task allocation approach is developed to efficiently assign tasks, in which an acceptance possibility is introduced to find the high-value task assignments.

3) We develop a game-theoretic solution, wherein the Nash equilibrium is found based on the best-response approach. We also introduce a simulated annealing strategy to further improve the assignment when multiple Nash equilibriums exist.

4) As demonstrated by the experiments, our proposed algorithms can effectively and efficiently form stable worker coalitions for tasks, which achieve nearly optimal total reward. In particular, our equilibrium-based method with the simulated annealing strategy can obtain up to 98% of the maximal reward and its CPU cost is considerably lower than that of the optimal task assignment approach, achieving an excellent trade-off between effectiveness and efficiency.

II. Problem Statement

In this section, we briefly introduce a set of preliminary concepts and formulate our problem. Table I lists the major notations used throughout the paper.

Definition 1 (Spatial Task): A spatial task, denoted as s = $<$ $s.l$, $s.p$, $s.e$, $s.d$, $s.wl$, $s.maxR$, $s.pr$ $>$, is a task to be performed at location $s.l$, published at time $s.p$, expected to be finished at time $s.e$ and will expire at deadline $s.d$, where $s.l : (x, y)$ is a point in the 2D space. Each task is also labelled with a required workload $s.wl$ to finish task s by a

TABLE I
SUMMARY OF NOTATIONS

Notation	Definition
s	Spatial task
$s.l$	Location of spatial task s
$s.p$	Publish time of spatial task s
$s.e$	Expected completion time of spatial task s
$s.d$	Deadline of spatial task s
$s.wl$	Workload of spatial task s
$s.maxR$	Maximum reward of spatial task s
$s.pr$	Penalty rate of spatial task s
w	Worker
$w.l$	Location of worker w
$w.r$	Reachable radius of worker w
$AW(s)$	Available worker set of task s
t_{now}	The current time
$t(a,b)$	Travel time from location a to location b
$d(a,b)$	Travel distance from location a to location b
$w.RS$	Reachable task set of worker w
$WC(s)$	Worker coalition for task s
$R_{WC(s)}$	Reward of worker coalition $WC(s)$ by finishing s
$w.wl(WC(s))$	Worker w's workload when performing s in $WC(s)$
$MWC(s)$	Minimal worker coalition for task s
$\mathcal{A}$	A spatial task assignment
$\mathcal{A}.R$	Total reward for task assignment $\mathcal{A}$

normal worker (we simply use the time required to finish a task to denote $s.wl$ in our work). $s.maxR$ denotes the maximum reward the requester of task s can offer and $s.pr$ is a penalty rate, which establishes a correlation between the completion time and reward.

Definition 2 (Worker): A worker, denoted as $w = < w.l, w.r >$, is a person who is able to perform spatial tasks only if she is paid. A worker can be in an either online or offline mode. A worker is online when she is ready to accept tasks and she is offline when she is unavailable to perform tasks. An online worker is associated with her current location $w.l$ and her reachable circular range with $w.l$ as the center and $w.r$ as the radius, in which w can accept assignments of tasks.

In our work, a worker is able to handle only one task (i.e., her capacity is 1) at a certain time instance, whether on her own or as a part of a coalition that makes joint effort on the completion of that task, which is reasonable in practice. A worker can be assigned a task only when she is online and is not performing any tasks. Once a task is assigned to a certain worker, the worker is considered as being offline until she completes the assigned task.

Due to the constraint of workers' reachable range and tasks' expiration time, each task can be completed only by a small subset of workers, called *available worker set*.

Definition 3 (Available Worker Set): The available worker set for a task s, denoted as $AW(s)$, is a set of workers that satisfy the following two conditions: $\forall w \in AW(s)$,

1) $t_{now} + t(w.l, s.l) < s.d$, and

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

where t_{now} is the current time, $t(a,b)$ is the travel time from location a to location b and $d(a,b)$ is the travel distance from location a to b. The above two conditions guarantee a worker can travel from her origin to the location of her reachable task s directly before it expires. If worker w is available for task s, i.e., $w \in AW(s)$, we say s is a reachable task of w and denote the reachable task set of w as $w.RS$.

Definition 4 (Worker Coalition): Given a task s to be assigned and its available worker set $AW(s)$, the worker coalition for task s, denoted as $WC(s)$, is a subset of $AW(s)$ such that all the workers in $WC(s)$ have enough time to complete task s together before it expires, i.e., $\sum_{w \in WC(s)} (s.d - (t_{now} + t(w.l, s.l))) \geq s.wl$.

Taking Figure 1 as a case, the available worker set of task s_1 is $\{w_2, w_3, w_5, w_6\}$, where all of the available workers can arrive at $s_1.l$ before $s_1.d$ and s_1 is reachable for them. $\{w_3\}$, $\{w_2, w_3\}$ and $\{w_2, w_3, w_6\}$ are worker coalitions for task s_1 since all the workers in each coalition can cooperate together to finish task s_1 before $s_1.d$.

As for task reward, taking longer time for completing a task (including waiting time for its assignment and task duration, i.e., from a task's publish time to its finish time) increases the probability of the task failure in the SC environment, and thus reduces the rewards for workers. Considering the constraints on the tasks' expected completion time, deadline, required workload and budget (i.e., the maximum reward the requester can offer), we adopt the Reward Pricing Model (RPM) [7], which can effectively quantify the temporal constraints of tasks, and is an important incentive mechanism to motivate workers to finish the assigned tasks on time. Specifically, RPM takes a single task s and one of its worker coalition into account, focusing on the task completion time and real reward (i.e., the requester's real payment for the task), as depicted in Figure 2.

With the main time constraints of the task (i.e., task's publish time $s.p$, expected completion time $s.e$ and deadline $s.d$), penalty rate $s.pr$ and maximum reward $s.maxR$, the RPM can be expressed as a formula shown below:

$$R_{WC(s)} = \begin{cases} s.maxR, & s.p \leq s.t_e \leq s.e \\ s.maxR - s.pr* \\ (s.t_e - s.e), & s.e < s.t_e \leq s.d \\ 0, & s.t_e > s.d, \end{cases} \tag{1}$$

where $R_{WC(s)}$ represents task s's real reward that the task requester offers for workers in $WC(s)$ and $s.t_e$ indicates s's completion time with its worker coalition $WC(s)$. From Equation 1 we can see, if a task can be completed before its expected time, workers will obtain the maximum reward. Without loss of generality, Equation 1 models the penalty rate linearly when the task cannot be finished before its expected completion time but can be finished before its deadline. For example, in a house removal scenario, the task requester is happy to pay the maximum rewards to workers if the job can be finished before 17:00. After 17:00, she may be a little bit disappointed but still would like to pay reduced reward based on a penalty rate. However, she will be too unsatisfied to pay any reward if the task cannot be completed by the hard deadline, e.g., by midnight today.

In order to calculate $s.t_e$, we denote $s.t_s$ as the start time (i.e., time of assignment) of s, $T_{WC(s)}$ as the task duration of s (i.e., elapsed time from assignment to completion), $w.wl(WC(s)) > 0$ as the workload contribution (measured by time) of w when task s is performed by $WC(s)$. From Figure 3, which illustrates the workload allocation of worker coalition $\{w_2, w_3\}$ for task s_1 (where workers and tasks are

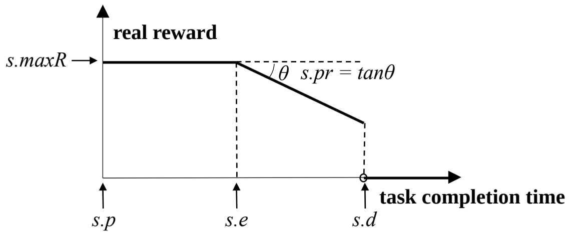


Fig. 2. Task Reward Pricing Model

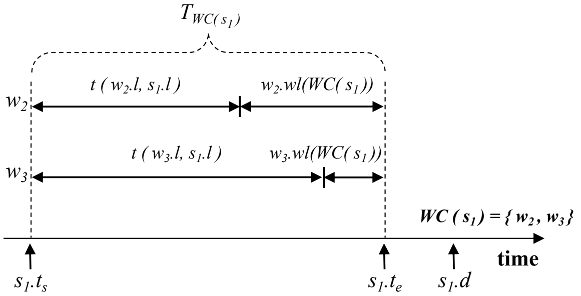


Fig. 3. Workload Allocation of Worker Coalition $\{w_2, w_3\}$ for Task s_2

from the running example in Figure 1), it is easily understandable that, $\forall w \in WC(s)$, the task duration is equal to w's travel time plus her workload contribution, i.e.,

$$T_{WC(s)} = t(w.l, s.l) + w.wl(WC(s)),\\ \forall w \in WC(s), w.wl(WC(s)) > 0. \tag{2}$$

By summing up the right side over all workers in coalition $WC(s)$, we have

$$T_{WC(s)} = \frac{\sum_{w \in WC(s)} t(w.l, s.l) + \sum_{w \in WC(s)} w.wl(WC(s))}{|WC(s)|}. \tag{3}$$

Given the fact that $s.wl = \sum_{w \in WC(s)} w.wl(WC(s))$, it comes to

$$T_{WC(s)} = \frac{\sum_{w \in WC(s)} t(w.l, s.l) + s.wl}{|WC(s)|}. \tag{4}$$

Finally, $s.t_e$ can be calculated as $s.t_e = s.t_s + T_{WC(s)}$, and each worker's workload can be calculated as $w.wl(WC(s)) = T_{WC(s)} - t(w.l, s.l)$. From the perspective of worker coalition, we assume that the goal of a worker joining a coalition is to increase the total reward of the coalition, which will accordingly lead to a satisfying reward and reputational award for this worker.

Since we require $w.wl(WC(s)) > 0$, if a worker's travel time exceeds the task duration, i.e., $t(w.l, s.l) \geq T_{WC(s)}$, then the worker w has no contribution to task s and should be removed from $WC(s)$. Additionally, in Figure 2, when a worker coalition $WC(s)$ can cooperate to finish a task s before its expected completion time $s.e$, which means they obtain the maximum reward ($s.maxR$) of this task, adding more workers into $WC(s)$ cannot lead to a higher reward. In other words, more workers in $WC(s)$ does not necessarily mean earlier completion time, or more total reward. This observation motivates the notion of minimal worker coalition.

Definition 5 (Minimal Worker Coalition): A worker coalition $WC(s)$ for task s is minimal (denoted by $MWC(s)$) if none of its subsets can obtain a reward that is equal to $R_{WC(s)}$.

In Figure 1, although $\{w_3\}$, $\{w_2, w_3\}$ and $\{w_2, w_3, w_6\}$ are all worker coalitions for task s_1, $\{w_2, w_3, w_6\}$ is not a minimal worker coalition since $\{w_2, w_3\}$ can generate the same reward with $\{w_2, w_3, w_6\}$, i.e., $R_{\{w_2,w_3\}} = R_{\{w_2,w_3,w_6\}}$. For task s_4, both $\{w_7\}$ and $\{w_1, w_7\}$ are its worker coalitions, but only $\{w_7\}$ is its minimal worker coalition. This is because w_7 can obtain the maximum reward of s_4 when conducting s_4 by herself without the need of others' collaboration.

Definition 6 (Task Completion): Given a task s and one of its minimal worker coalitions $MWC(s)$, task s is completed once all the workers in coalition $MWC(s)$ cooperate to finish s's workload, i.e., $\sum_{w \in MWC(s)} w.wl(MWC(s)) = s.wl$.

Definition 7 (Spatial Task Assignment): Given a set of workers W and a set of tasks S, a spatial task assignment, denoted by $\mathcal{A}$, consists of a set of $< task, MWC >$ pairs in the form of $< s_1, MWC(s_1) >$, $< s_2, MWC(s_2) >$,...,$< s_{|S|}, MWC(s_{|S|}) >$, where $MWC(s_1) \bigcap MWC(s_2) \bigcap ... \bigcap MWC(s_{|S|}) = \emptyset$.

Let $\mathcal{A}.R$ denote the total reward for task assignment $\mathcal{A}$, i.e., $\mathcal{A}.R = \sum_{s \in S, <s, MWC(s)> \in \mathcal{A}} R_{MWC(s)}$ (where $R_{MWC(s)}$ can be calculated by Equation 1), and $\mathbb{A}$ denote all the possible ways of assignments.

Problem Statement: Given a set of online workers W and a set of tasks S in a time instance, the CTA problem aims to find the global optimal assignment $\mathcal{A}_{opt}$, such that the total reward can be maximized, i.e., $\forall \mathcal{A}_i \in \mathbb{A}$, $\mathcal{A}_i.R \leq \mathcal{A}_{opt}.R$.

Lemma 1: The CTA problem is NP-hard.

Proof 1: The lemma is proved through a reduction from the 0-1 knapsack problem. A 0-1 knapsack problem can be described as follows: given a set C with n items, in which each item $c_i \in C$ is labelled with a weight m_i and a value v_i, the 0-1 knapsack problem is to identify a subset C' of C that maximizes $\sum_{c_i \in C'} v_i$ subjected to $\sum_{c_i \in C'} m_i \leq M$, where M indicates the maximum weight capacity.

For a given 0-1 knapsack problem, it can be transformed into an instance of CTA problem in the following. We give a task set S with n tasks, in which each task s_i is associated with the publish time $s_i.p = 0$, the expected completion time $s_i.e = 1$, the deadline $s_i.d = 1$, the workload $s_i.wl = m_i$, and the maximum reward $s_i.maxR = v_i$. In addition, we give a worker set W with M workers, where each worker w_i is allowed to complete 1 workload only. All of the tasks and workers are situated at the same location. Under this circumstance, in order to get the reward v_i for each task s_i, the system needs to assign m_i workers to task s_i.

Given this mapping, we can show that the transformed CTA problem can be solved, if and only if the 0-1 knapsack problem can be solved. As 0-1 knapsack problem is known to be NP-hard [15], CTA problem is also NP-hard.

III. Greedy Approach

In this section, we design a greedy algorithm, which encourages worker coalitions to obtain more rewards. This algorithm is based on the consensus that the nearby workers selected to perform a task can generate a higher reward since the reward is non-increasing over time (i.e., it keeps stable with the maximum payoff offered by the task requester at first and then gets lower in the Reward Pricing Model in Section II) and the nearby workers are capable to deal with more workloads to obtain more rewards. Moveover, considering the time utilization ratio (measured by workers' workload and travel time) of workers and rate of return (measured by tasks' actual reward and maximal reward) of tasks, we introduce an acceptance possibility to find the high-value task assignments (i.e., task assignments ensuring high time utilization ratio for workers and high rate of return for tasks).

Algorithm 1: Greedy Approach

```
   Input: Worker set $W$, task set $S$, acceptance threshold $\eta$
   Output: Task assignment: $\mathcal{A}$
1  $\mathcal{A} \leftarrow \emptyset$;
2  for each $s \in S$ do
3      Obtain the available worker set $AW(s)$ from $W$;
4      $WC(s) \leftarrow \emptyset$; $R_{WC(s)} \leftarrow 0$; $R' \leftarrow 0$;
5      for the nearest worker $w \in AW(s)$ do
6          $R' \leftarrow R_{WC(s)\cup\{w\}}$;
7          /*$R_{WC(s)\cup\{w\}}$ is computed based on Equation 1.*/
8          $AW(s) \leftarrow AW(s) - \{w\}$;
9          if $R' = 0$ and $AW(s) = \emptyset$ then
10             break;
11         if $R' = 0$ and $AW(s) \neq \emptyset$ then
12             $WC(s) \leftarrow WC(s) \cup \{w\}$;
13         if $R' > R_{WC(s)}$ then
14             $WC(s) \leftarrow WC(s) \cup \{w\}$;
15             $R_{WC(s)} \leftarrow R'$;
16         else
17             $MWC(s) \leftarrow WC(s)$;
18             $R_{MWC(s)} \leftarrow R_{WC(s)}$;
19             $\mathcal{A} \leftarrow \mathcal{A} \cup < s, MWC(s) >$;
20             break;
21     Calculate the acceptance possibility $AP_{MWC(s)}$ based on
       Equation 6;
22     if $AP_{MWC(s)} < \eta$ then
23         $\mathcal{A} \leftarrow \mathcal{A} - < s, MWC(s) >$;
24         $S = S - \{s\}$;
25     else
26         $W = W - MWC(s)$;
27         $S = S - \{s\}$;
28 return $\mathcal{A}$.
```

Algorithm 1 outlines the major procedure of the greedy approach, which takes the worker set W, task set S and an acceptance threshold η as input and outputs a task assignment result $\mathcal{A}$. The algorithm starts with the calculation of the available worker set $AW(s)$ for each task s (line 3). After initialization of the current worker coalition (i.e., $WC(s) \leftarrow \emptyset$), the corresponding reward obtained by $WC(s)$ (i.e., $R_{WC(s)} \leftarrow 0$, $R_{WC(s)}$ is also the total reward of s), and a temporary variable (i.e., $R' \leftarrow 0$, line 4), for each task $s \in S$, the algorithm generates the minimal worker coalition $MWC(s)$ by choosing the nearest workers who can contribute a higher overall reward for s and assigns the worker coalition $MWC(s)$ to s (line 5–20). Specifically, by adding the nearest worker $w \in AW(s)$ into the current worker coalition $WC(s)$, we can compute the reward obtained by coalition $WC(s) \cup \{w\}$ based on Equation 1, i.e., $R_{WC(s)\cup\{w\}}$ (line 6). Then we judge whether adding worker w can increase the total actual reward of s by performing the following actions:

1) if task s cannot be completed by workers in coalition $WC(s) \cup \{w\}$ (i.e., $R' = 0$) and there are no available workers (i.e., $AW(s) = \emptyset$), task s cannot be assigned to a suitable coalition (line 9–10);

2) if task s cannot be completed by workers in coalition $WC(s) \cup \{w\}$ (i.e., $R' = 0$) but there are enough available workers (i.e., $AW(s) \neq \emptyset$), we add worker w into the current worker coalition $WC(s)$ (line 11–12);

3) if adding worker w into $WC(s)$ can increase the reward obtained by $WC(s)$ (i.e., $R' > R_{WC(s)}$), worker w can be added into $WC(s)$ and the reward obtained by $WC(s)$ can be accordingly updated, i.e., $R_{WC(s)} \leftarrow R'$ (line 13–15);

4) otherwise (i.e., when $R' \neq 0$ and adding worker w into $WC(s)$ cannot increase the reward obtained by $WC(s)$), we can obtain the minimal worker coalition $MWC(s)$, the corresponding reward $R_{MWC(s)}$ for s and the updated task assignment, i.e., $\mathcal{A} \leftarrow \mathcal{A} \cup < s, MWC(s) >$ (line 16–20).

After assigning a worker coalition $MWC(s)$ to task s, we calculate the acceptance possibility $AP_{MWC(s)}$ (that means the possibility that a task assignment $< s, MWC(s) >$ is accepted) in the following:

$$\begin{aligned} AP_{MWC(s)} =& \alpha \frac{\sum_{w \in MWC(s)} w.wl(MWC(s))}{\sum_{w \in MWC(s)} \big(t(w.l, s.l) + w.wl(MWC(s))\big)} \\ &+ (1-\alpha)\frac{R_{MWC(s)}}{s.maxR}, \end{aligned} \tag{5}$$

where α is a parameter controlling the contribution of the time utilization ratio of workers (i.e., $\frac{\sum_{w \in MWC(s)} w.wl(MWC(s))}{\sum_{w \in MWC(s)} \big(t(w.l,s.l)+w.wl(MWC(s))\big)}$) and the rate of return of tasks (i.e., $\frac{R_{MWC(s)}}{s.maxR}$), $w.wl(MWC(s)$ denotes the workload contribution (measured by time) of w when task s is performed by $MWC(s)$, $t(w.l, s.l)$ denotes the travel time from location $w.l$ to location $s.l$. $R_{MWC(s)}$ is the actual reward that $MWC(s)$ can obtain by performing task s, and $s.maxR$ is the maximal reward of s. Combining Equation 3, Equation 5 can be represented in the following:

$$\begin{aligned} AP_{MWC(s)} =& \alpha \frac{s.wl}{|MWC(s)| T_{MWC(s)}} \\ &+ (1-\alpha)\frac{R_{MWC(s)}}{s.maxR}, \end{aligned} \tag{6}$$

where $s.wl = \sum_{w \in MWC(s)} w.wl(MWC(s))$ denotes the required workload of task s, $|MWC(s)|$ denotes the number of worker coalition $MWC(s)$, and $T_{MWC(s)}$ is the task duration of s (i.e., elapsed time from assignment to completion).

In case that the acceptance possibility of task assignment $< s, MWC(s) >$ is less than a given threshold η ($0 \leq \eta \leq 1$), i.e., $AP_{MWC(s)} < \eta$, Algorithm 1 will quit performing task s, which means task s fails to be assigned (line 23-24). η can be specified by task requesters or SC platforms. Otherwise (i.e., $AP_{MWC(s)} \geq \eta$), the task assignment $< s, MWC(s) >$ is regarded as high-value and workers in $MWC(s)$ are assigned to perform task s. As a result, workers in $MWC(s)$ and task

s can be removed from the worker set W to be assigned and task set S to be assigned (line 26-27). Finally, Algorithm 1 will obtain a suitable task assignment result (line 28).

It is easy to see the time complexity of Algorithm 1 is $O(|S| \cdot |W| \cdot |maxAW|)$, where $|S|$ is the number of tasks, $|W|$ is the number of workers, and $|maxAW|$ is the maximum number of available workers among all the tasks, i.e., $|maxAW| = \max_{s \in S} |AW(s)|$.

In Figure 1, the greedy algorithm with acceptance possibility can obtain a task assignment, $\{< s_1, \{w_2, w_3, w_5\} >, < s_3, \{w_1, w_4\} >, < s_4, \{w_7\} >\}$, with the reward of 8.53, in which we set α as 0.5 and η as 0.4 .

IV. Equilibrium-based Approach

Although the greedy algorithm can efficiently find a task assignment result, it cannot guarantee the stability of the formed worker coalitions. The fundamental nature of CTA problem is that each worker needs to choose a task to conduct by interacting with other workers during the process of task assignment, suggesting that the task selection for a worker depends on the decisions taken by the other workers. Such interdependent decisions can be modeled by game theory, where workers can be treated as independent players involved in a game. On the basis of this, the CTA system can be considered as a *multi-player game*.

To be more specific, our problem can be modeled as an exact potential game, which has at least one Nash equilibrium in pure strategy (a.k.a. pure Nash equilibrium) [16]. Then we employ the best-response algorithm, one of the most basic tools in exact potential games as it is efficient in addressing the conflicts arising among the players [17]. With the best-response dynamics, players are required to have their strategies updated sequentially and asynchronously on the basis of their best-response utility functions conditioned on the strategies of the other players in a myopic manner, which finally achieves a pure Nash equilibrium. A Nash equilibrium represents a state of the game where any single worker is incapable to improve their utility by making a unilateral shift from the assigned coalition to other coalitions when other workers stay in their assigned coalitions. This suggests that workers will voluntarily select the assigned tasks when they have freedom to do so. In such situation, the formed worker coalitions are regarded as stable coalitions. Nevertheless, this Nash equilibrium achieved by the best-response algorithm may be far from optimum as there can be many equilibrium points. In order to resolve this problem, we introduce a Simulated Annealing (SA) strategy into the best-response dynamics, which finds a better Nash equilibrium corresponding to the approximately optimal task assignment. With the help of SA strategy, the updating process has a better chance to realize a better Nash equilibrium with higher total rewards. Finally, we analyze the feasibility of our solutions.

A. Game Modeling and Nash Equilibrium

We first formulate our CTA problem as an n-player strategic game, $\mathcal{G} =< W, \mathbb{ST}, \mathbb{U} >$, which is comprised of players, strategy spaces and utility functions. It is specified as follows:

1) $W = \{w_1, ..., w_n\}$ $(n \geq 2)$ represents a finite set of workers playing the role as the game players. In the rest of the paper, we will use player and worker interchangeably when the context is clear.

2) $\mathbb{ST} = \{ST_i\}_{i=1}^{n}$ is the overall strategy set of all the players, i.e., the strategy space of the game. ST_i is the finite set of strategies available to worker w_i, which contains w_i's reachable task set and null task (that means w_i does not choose any tasks to conduct), denoted as $ST_i = \{w_i.RS, null\}$ (where $w_i.RS$ indicates the reachable task set of worker w_i and $null$ represents the null task).

3) $\mathbb{U} = \{U_i\}_{i=1}^{n}$ denotes the utility functions of all the players, and $U_i : \mathbb{ST} \rightarrow \mathbb{R}$ is the utility function of player w_i. For every joint strategy $\vec{st} \in \mathbb{ST}$, $U_i(\vec{st}) \in \mathbb{R}$ represents the utility of player w_i, which can be calculated as follows:

$$\begin{aligned} U_i(\vec{st}) = & R_{MWC(s) \cup \{w_i\}} - R_{MWC(s)} \\ & - (R_{MWC(s_0)} - R_{MWC(s_0) - \{w_i\}}), \end{aligned} \tag{7}$$

where $R_{MWC(s) \cup \{w_i\}}$ is the total reward obtained by coalition $MWC(s) \cup \{w_i\}$, $MWC(s) \cup \{w_i\}$ is the new worker coalition including $MWC(s)$ and $\{w_i\}$, $MWC(s_0) - \{w_i\}$ denotes the worker coalition (where w_i is removed from $MWC(s_0)$), s_0 denotes the task that is currently assigned to worker w_i, $MWC(s)$ and $MWC(s_0)$ are worker coalitions that w_i is willing to join and currently staying in, respectively. When the context of $\vec{st}$ is clear, we use U_i to denote $U_i(\vec{st})$.

In a strategic game, a policy profile $\pi^* = (\pi_1^*, ..., \pi_n^*)$ (where $\pi_i^* : ST_i \rightarrow [0, 1]$ is a probability distribution over ST_i) is called a Nash equilibrium with mixed strategies (a.k.a. mixed Nash equilibrium) if and only if for any $w_i \in W$, it holds that:

$$U_i(\pi^*) \geq \max_{\pi_i' \in \Sigma_i} U_i\left(\pi_1^*, \ldots, \pi_{i-1}^*, \pi_i', \pi_{i+1}^*, \ldots, \pi_n^*\right), \tag{8}$$

where Σ_i denotes the policy space of player w_i. For any given policy profile $\pi = (\pi_1, ..., \pi_n)$, $U_i(\pi) = \sum_{\vec{st} \in \mathbb{ST}} \pi_i(\vec{st}) U_i(\vec{st})$. The Nash equilibrium is a pure Nash equilibrium (i.e., Nash equilibrium with pure strategy) only when players play deterministic strategies, which means the probability of one strategy worker w_i can choose from ST_i is 1 while the rest strategies from ST_i are 0.

As proved by [18], every game with a finite number of players and a finite strategy set has a mixed Nash equilibrium, which only implies stable probability distributions over profiles rather than the fixed play of a particular joint strategy profile. This uncertainty is unacceptable in our scenario where each worker needs to have a definite strategy, i.e., selecting a task to conduct or doing nothing. Therefore, we next prove that our CTA game has pure Nash equilibrium, wherein each player can choose a strategy in a deterministic manner.

Given a joint strategy $\vec{st} = (st_1, ..., st_n) \in \mathbb{ST}$, st_i (i.e., $s \in w_i.RS$ or $null$) represents the strategy chosen by player w_i $(0 < i \leq n)$. As for player w_i, a joint strategy $\vec{st_i} \in \mathbb{ST}$ can also be denoted by $(st_i, \vec{st}_{-i})$, where $\vec{st}_{-i} = (st_1, ..., st_{i-1}, st_{i+1}, ..., st_n) \in \mathbb{ST}_{-i}$ is the joint strategies of all the other players.

Lemma 2: The CTA game has pure Nash equilibrium.

Proof 2: To prove Lemma 2, there is a need to prove the CTA game as an Exact Potential Game (EPG) that has a global potential function onto which the incentive of all the players can be mapped. For the EPG, the best-response framework always converges to a pure Nash equilibrium for countable strategies [16].

In the following part, we introduce the definition of EPG and show that the CTA game is an EPG.

Definition 8 (Exact Potential Game): A strategic game, $\mathcal{G} =< W, \mathbb{ST}, \mathbb{U} >$, is an Exact Potential Game (EPG) if there exists a function, $\Phi : \mathbb{ST} \rightarrow \mathbb{R}$, such that for all $\vec{st}_i \in \mathbb{ST}$, it holds that, $\forall w_i \in W$,

$$U_i(st'_i, \vec{st}_{-i}) - U_i(st_i, \vec{st}_{-i}) = \Phi(st'_i, \vec{st}_{-i}) - \Phi(st_i, \vec{st}_{-i}), \quad (9)$$

where st'_i and st_i are the strategies that can be selected by worker w_i, $\vec{st}_{-i}$ is the joint strategy of the other workers except for worker w_i, and the function Φ is called an exact potential function for game $\mathcal{G}$.

Lemma 3: CTA is an Exact Potential Game (EPG).

Proof 3: We define a potential function as $\Phi(\vec{st}) = \sum_{s \in S} R_{MWC(s)}$, which represents the total rewards for all the tasks in S. Then it can be obtained that,

$$\begin{aligned}
&\Phi(st'_i, \vec{st}_{-i}) - \Phi(st_i, \vec{st}_{-i}) \\
=&(R_{MWC(s_k)\cup\{w_i\}} + R_{MWC(s_g)} + \sum_{s \in S - s_k - s_g} R_{MWC}(s)) \\
&- (R_{MWC(s_k)} + R_{MWC(s_g)\cup\{w_i\}} + \sum_{s \in S - s_k - s_g} R_{MWC}(s)) \\
=&(R_{MWC(s_k)\cup\{w_i\}} - R_{MWC(s_k)}) \\
&- (R_{MWC(s_g)\cup\{w_i\}} - R_{MWC(s_g)}) \\
=&\Big(R_{MWC(s_k)\cup\{w_i\}} - R_{MWC(s_k)} \\
&- (R_{MWC(s_0)} - R_{MWC(s_0)-\{w_i\}})\Big) \\
&- \Big(R_{MWC(s_g)\cup\{w_i\}} - R_{MWC(s_g)} \\
&- (R_{MWC(s_0)} - R_{MWC(s_0)-\{w_i\}})\Big) \\
=&U_i(st'_i, \vec{st}_{-i}) - U_i(st_i, \vec{st}_{-i}),
\end{aligned} \quad (10)$$

where the tasks selected in strategies st'_i and st_i are s_k and s_g respectively. In accordance with Definition 9, the strategic game of the CTA problem is an exact potential game, for which the CTA game has a Nash equilibrium in pure strategy.

Let st^*_i denote the best strategy that player w_i can make response to the strategy combination $\vec{st}_{-i}$ of others. Therefore, the utility $U_i(st^*_i, \vec{st}_{-i})$ is maximized for a given $\vec{st}_{-i}$. A pure Nash equilibrium is reached by the joint strategy $\vec{st}^* = (st^*_1, ..., st^*_n)$, as a result of which no player can have any gain in their utility by making change to their strategy unilaterally [19].

B. Best-response Approach

As our CTA game has pure Nash equilibrium, we adopt the best-response approach to solve it, which generates a number of stable worker coalitions to perform the tasks by reaching the pure Nash equilibrium. Specifically, the designed best-response algorithm consists of players taking turns to adapt their strategies based on the most recent known strategies of the others, which ends up reaching the Nash equilibrium that is a locally optimal task assignment. A general framework of the best-response approach is illustrated in Algorithm 2.

Algorithm 2: Best-response Approach

```
   Input: Worker set $W$, task set $S$
   Output: Task assignment: $\mathcal{A}$
1  $\mathcal{A} = \emptyset$;
2  for each task $s \in S$ do
3      Obtain the available worker set $AW(s)$ from $W$ and randomly
       assign an available worker, stored in $MWC(s)$, to $s$, where
       $\bigcap_{MWC(s)} = \emptyset$;
4      for each worker $w_i \in W$ do
5          if $w_i$ is assigned to a task $s$ then
6              $w_i.st = s$;
7          else
8              $w_i.st = null$;
9      $\mathcal{A} = \mathcal{A} \cup < s, MWC(s) >$;
10 $k = 1$;
11 repeat
12     for each worker $w_i \in W$ do
13         find the best-response task $s^*$ for $w_i$;
14         /*$s^*$ can be obtained by Equation 11*/
15         if $U_i(\vec{st}) \leq 0$ then
16             continue;
17         else if $w_i.st = null$ then
18             $w_i.st = s^*$;
19             $MWC(s^*) = MWC(s^*) \cup \{w_i\}$;
20             else
21                 if workers in $\{MWC(w_i.st) - \{w_i\}\}$ cannot
                   complete task $w_i.st$ before its deadline then
22                     for each worker
                       $w_j \in \{MWC(w_i.st) - \{w_i\}\}$ do
23                         $w_j.st = null$;
24                     $MWC(w_i.st) = \emptyset$;
25                 else
26                     $MWC(w_i.st) = MWC(w_i.st) - \{w_i\}$;
27                 $w_i.st = s^*$;
28                 $MWC(s^*) = MWC(s^*) \cup \{w_i\}$;
29     $k = k + 1$;
30 until $W.\vec{st}^k = W.\vec{st}^{k-1}$;
31 /*$W.\vec{st}^k$ denotes the strategies of all the workers in the $k$th iteration*/
32 update $\mathcal{A}$;
33 return $\mathcal{A}$.
```

Given a worker set W and a task set S to be assigned, the task assignment $\mathcal{A}$ is initialized as $\emptyset$ (line 1). The algorithm first randomly chooses an available worker for each task, obtains the corresponding strategy (i.e., a reachable task or doing nothing) for each worker, and updates the task assignment $\mathcal{A}$ accordingly (line 2–9). Then the algorithm iteratively adjusts each worker's strategy to her best-response strategy that maximizes the reward increase in her coalition (as defined in Equation 11) based on the current joint strategies of others until a Nash equilibrium (i.e., no one changes her strategy) is found (line 11–30). At each iteration, only one worker is allowed to select her best-response strategy and the game is supposed to be played in sequence.

To be specific, for each worker $w_i \in W$, we first find the best-response task s^* with the maximal reward increase, which can be calculated in Equation 11.

$$\begin{aligned} s^* =& argmax_{s \in w_i.RS} U_i(\vec{st}) \\ =& argmax_{s \in w_i.RS} \big(R_{MWC(s) \cup \{w_i\}} - R_{MWC(s)} \\ & - (R_{MWC(s_0)} - R_{MWC(s_0) - \{w_i\}})\big). \end{aligned} \tag{11}$$

When there is no best-response task for worker w_i based on the current task assignment, w_i makes no change to her strategy (line 15–16). For the worker selecting a best-response task, we check her current strategy as follows:

1) in the event that her current strategy is doing nothing, i.e., $w_i.st = null$, we assign her the best-response task (i.e., $w_i.st = s^*$) and update the minimal worker coalition for task s^* (line 17–19);

2) in case that her current strategy involves a task (marked as $w_i.st$), which means w_i is assigned a task $w_i.st$ in coalition $MWC(w_i.st)$, the strategies of the other workers in coalition $MWC(w_i.st)$ are updated based on whether they are able to complete task $w_i.st$ together on time (line 21–26). Subsequently, the strategy and worker coalition of w_i are updated (line 27–28).

Finally, we update the task assignment $\mathcal{A}$ according to the Nash equilibrium (line 32). The time complexity of Algorithm 2 is $O(|S| \cdot |W|^2 + |W| \cdot |maxRS| \cdot K)$, where $|S|$ is the number of tasks, $|W|$ is the number of workers, $|maxRS|$ is the maximum number of reachable tasks among all the workers (i.e., $|maxRS| = \max_{w \in W} |w.RS|$), and K is the number of iterations to adjust each worker's best-response strategy until a Nash equilibrium is achieved.

C. Simulated Annealing based Optimization Strategy

Although the pure Nash equilibrium calculated by the best-response algorithm can generate an acceptable task assignment result with stable worker coalitions, it is a local optima of the CTA problem and is not necessarily unique. Under the situations where multiple pure Nash equilibriums exist (i.e., the structure of the problem space is not smooth), it is desirable to obtain a better one than the one generated by the best-response algorithm. Simulated Annealing (SA) is a stochastic optimization procedure. It takes random walks through the problem space at successively lower temperatures, looking for points with better results (generated by the objective function) than the current local optimal point. Inspired by the success achieved by SA in solving discrete optimization problems [20], we employ it to search for better approximation to the global optimal task assignment.

In particular, when each worker updates her strategy st_i sequentially based on the given $\vec{st}_{-i}$ to maximize the utility function $U_i(st_i, \vec{st}_{-i})$, the workers may reach a Nash equilibrium that is a stable state. Considering that the search space is discrete (i.e., the strategy sets $\mathbb{ST} = \{ST_i\}_{i=1}^n$ are discrete), the Simulated Annealing (SA) [21] can be applied in the process of updating each worker's strategy in order for a better local optimum. SA is regarded as an efficient probabilistic scheme for game updating to solve discrete optimization problems, evolving a discrete-time inhomogenous Markov chain, $x(k) = (st_1, ..., st_n)$. In our work, the state $x(k) = (st_1, ..., st_n)$ is the strategy combination of the workers at the kth iteration in Algorithm 2. For worker w_i, the strategy st_i can keep the current task s_0 or make change to one of the other reachable tasks (i.e., $w_i.RS - \{s_0\}$). For simulation of the heat (randomness), it is assumed that worker w_i is able to change her current strategy at random by using one of the other reachable tasks with an identical probability $P_{st_i, st'_i} = 1/|w_i.RS|$, where $st'_i = s$ $(s \in w_i.RS - \{s_0\})$ or $st'_i = null$. Every single worker can update her strategy sequentially in line with the following rules.

1) If $U_i(st'_i, \vec{st}_{-i}) \geq U_i(st_i, \vec{st}_{-i})$, then $x(k+1) = (st'_i, \vec{st}_{-i})$.

2) If $U_i(st'_i, \vec{st}_{-i}) < U_i(st_i, \vec{st}_{-i})$, then $x(k+1) = (st'_i, \vec{st}_{-i})$ with probability

$$\begin{aligned} \mathcal{P} &= \exp\Big\{\frac{U_i(st'_i, \vec{st}_{-i}) - U_i(st_i, \vec{st}_{-i})}{Tem(k)}\Big\}, \\ &= \exp\Big\{\frac{\Phi_i(st'_i, \vec{st}_{-i}) - \Phi_i(st_i, \vec{st}_{-i})}{Tem(k)}\Big\}, \end{aligned} \tag{12}$$

where $Tem(k)(> 0)$ denotes the temperature at the kth iteration, which is in decline gradually throughout the updating process; otherwise, $x(k+1) = x(k) = (st_i, \vec{st}_{-i})$.

By adhering to the above rules, we can update line 15–28 in Algorithm 2. The detailed pseudo-code is omitted due to space limit. Formally, the transition probability can be computed in Equation 13.

$$\begin{aligned} &\mathbb{P}\left[x(k+1) = \left(st'_i, \vec{st}_{-i}\right) | x(k) = \left(s_0, \vec{st}_{-i}\right)\right] \\ &= \tfrac{1}{|w_i.RS|} \exp\left\{\tfrac{\min\left(0, U_i\left(st'_i, \vec{st}_{-i}\right) - U_i\left(st_i, \vec{st}_{-i}\right)\right)}{Tem(k)}\right\}. \end{aligned} \tag{13}$$

The function $Tem(k) : \mathbb{N} \rightarrow (0, \infty)$ is non-increasing, called cooling schedule, where $\mathbb{N}$ is the set of positive integers. From Equation 13 it can be seen that the strategy selection is almost random when $Tem(k)$ is large while a better strategy with larger utility has a greater likelihood to be chosen when $Tem(k)$ approaches zero. Though the allowance of task selection with a smaller utility contributes to a decline in the total utility, such "irregular" strategy selections have a potential to facilitate a better Nash equilibrium (i.e., a better task assignment), which is validated by the experiments in Section V.

D. Convergence Analysis

The question of convergence to a Nash equilibrium has attracted a great deal of attention in the game theory field [22]. Therefore, we subsequently prove the convergence of our solution to a pure Nash equilibrium point where no worker is incentivised to unilaterally deviate.

Lemma 4: The best-response algorithm converges to a pure Nash equilibrium.

Proof 4: As depicted in Equation 10, the utilities of all the workers are mapped onto the potential function (i.e., Φ), suggesting that the individually-made adjustment to her strategy by each worker will result in a change to her utility and to the potential function with the same amount. For a potential game, each worker has her strategy updated sequentially for maximal utility by the best-response algorithm, and the potential function will reach a local maximum (i.e., Nash equilibrium) accordingly, wherein the best-response dynamic is equivalent to a local search on the potential function of a potential game. [23] has proven that in any finite potential game, sequential updates with best-response dynamic always converge to a Nash equilibrium.

In terms of convergence time, Fabrikant et al. [24] show that identifying a pure Nash equilibrium in a potential game is Polynomial Local Search (PLS)-complete when the best response of each player can be found in polynomial time. In our CTA game, each worker w_i has only $|w_i.RS|$ strategies corresponding to her reachable tasks $w_i.RS$, where $|w_i.RS|$ is not large in practical applications due to the spatio-temporal constraints of workers and tasks. Each worker can pick up one task out of her reachable tasks that maximizes the existing utility in polynomial time. Therefore, the convergence to a Nash equilibrium is fairly quick.

Lemma 5: The best-response algorithm with simulated annealing optimization converges to a pure Nash equilibrium when the cooling schedule is regulated.

Proof 5: With integration of the simulated annealing strategy into the best-response algorithm, randomness is added into the update process of workers' strategies, i.e., worker w_i can change her current strategy on a random basis by using one of the other reachable tasks with probabilities. Specifically, the process is "heated" up before "cooling" down, helping the potential function to avoid a local optimum (obtained by the best-response algorithm) and converge to another better Nash equilibrium, in which the cooling schedule ought to be regulated such that the process will eventually "freeze" (i.e., converge). [25] demonstrates that the convergence of simulated annealing strategy can be guaranteed when the cooling schedule is set as $Tem(k) = \frac{\beta}{log(k)}$ (where $\beta \geq D^*$ is a positive constant). If a joint strategy $\vec{st}$ has a path to the optimal joint strategy $\vec{st}^*$, D is defined as the depth such that the smallest value of the potential Φ along the path is $\Phi(\vec{st}) - D$. D^* denotes the maximum depth of the path starting from any joint strategy $\vec{st}$ and ending at the final joint strategy $\vec{st}^*$ if $\vec{st}$ has a path to $\vec{st}^*$.

The convergence of the best-response algorithm with simulated annealing is certain to proceed at a slower pace than that of the best-response algorithm. In Section V, we provide experimental evidence for this statement and demonstrate a pure Nash equilibrium can effectively be calculated in finite time, as stated in our Lemma 4 and Lemma 5.

V. Experiment

In this section, we evaluate the performance of the proposed methods on both real and synthetic datasets. All the experiments are implemented on an Intel (R) Xeon (R) CPU E5-2650 v2 @ 2.60 GHz with 128 GB RAM.

A. Experimental Setup

The experiments are carried out on two datasets, which are *gMission dataset* (marked as GM) and *synthetic dataset* (marked as SYN). To be specific, gMission is an open source SC platform [26], in which each task is associated with its location, deadline and reward (regarded as its maximal reward), and each worker is associated with her location and reachable radius. The publish time of all the tasks is set to 0. As the gMission data is not associated with the workload, expected completion time and penalty rate of tasks, we uniformly generate the attributes from the range of $[\frac{2}{5} \cdot s.d, 2 \cdot s.d]$, $[\frac{2}{5} \cdot s.d, \frac{3}{5} \cdot s.d]$ and $[0, \frac{s.maxR}{s.d - s.e}]$ (to guarantee the actual reward obtained by workers to be non-negative), respectively, where $s.d$ is the deadline of s, and $s.maxR$ is the reward of each task s. It is a common practice in experimental settings of SC platforms to generate the attributes in a uniform manner [2], [27] since it can demonstrate the effects of those attributes on more fair basis.

TABLE II
Experiment Parameters

Parameter	Value
Number of tasks $\lvert S\rvert$ (GM)	100, $\underline{200}$, 300, 400, 500,
Number of tasks $\lvert S\rvert$ (SYN)	1K, $\underline{2K}$, 3K, 4K, 5K
Number of workers $\lvert W\rvert$ (GM)	100, 200, 300, 400, $\underline{500}$
Number of workers $\lvert W\rvert$ (SYN)	1K, 2K, 3K, 4K, $\underline{5K}$
Reachable distance of workers r (SYN)	1km, 2km, 3km, $\underline{4km}$, 5km
Expected completion time of tasks e (SYN)	2h, $\underline{4h}$, 6h, 8h, 10h
Time between expected completion time and deadline of tasks $d - e$ (GM)	2h, 4h, $\underline{6h}$, 8h, 10h

For synthetic dataset, based on the observation from real dataset (i.e., gMission) that the location of workers/tasks is uniformly distributed in space, we also generate the location of workers/tasks following a uniform distribution. The maximal reward of each task is set following a Gaussian distribution since it is influenced by complex variables in real world. The workload is uniformly generated from the range of [2,10]. Other settings (i.e., the publish time and penalty rate of each task) in synthetic dataset are set in the same way with the gMission dataset.

We study and compare the performance of the following algorithms:

1) OTA: Optimal Task Assignment (OTA) algorithm based on the tree-decomposition technique. OTA first finds all the minimal worker coalitions for each task by utilizing the dynamic programming technique, and then applies the tree-decomposition-based algorithm [5] to identify the optimal task assignment with maximal rewards.

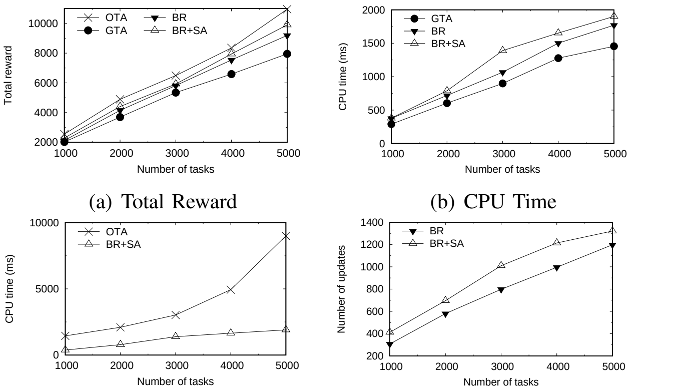
2) GTA: Greedy Task Assignment (GTA) algorithm, where α is set as 0.5, and the acceptance threshold η is set as 0.4.

3) BR: Best-Response (BR) approach.

4) BR+SA: our Best-Response approach with Simulated Annealing optimization (BR+SA), where the cooling schedule is set as $Tem(k) = \frac{1}{log(k+1)}$ and k denotes the kth iteration of the algorithm.

Three main metrics are compared among the above algorithms, i.e., *Total reward*, *CPU time* and the *Number of updates*, for finding the final task assignments, where *Number of updates* denotes the number of strategy updates made by workers. Table II shows our experimental settings, where the default values of all parameters are underlined.

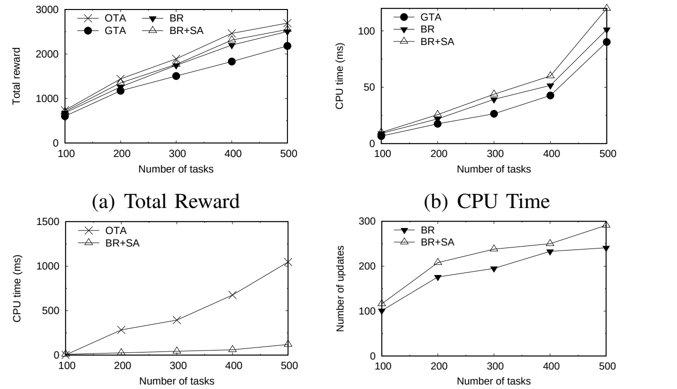
B. Experimental Results



(a) Total Reward (b) CPU Time

(c) CPU Time (d) Number of Updates

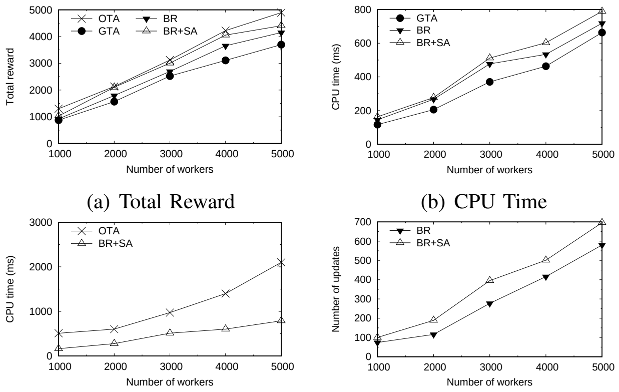
Fig. 4. Effect of $|S|$ on Synthetic Dataset



(a) Total Reward (b) CPU Time

(c) CPU Time (d) Number of Updates

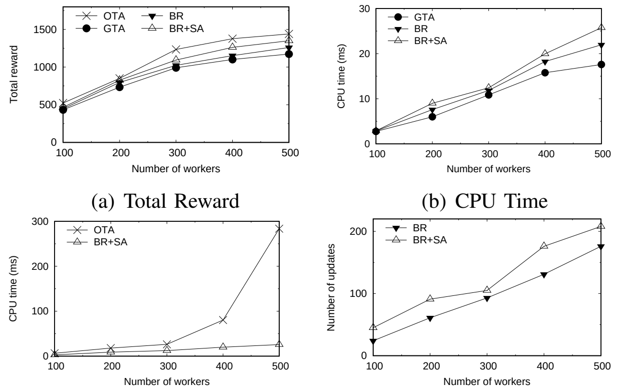
Fig. 5. Effect of $|S|$ on gMission Dataset



(a) Total Reward (b) CPU Time

(c) CPU Time (d) Number of Updates

Fig. 6. Effect of $|W|$ on Synthetic Dataset



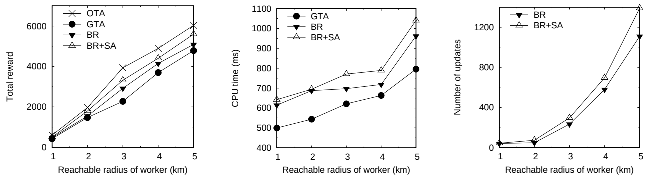
(a) Total Reward (b) CPU Time

(c) CPU Time (d) Number of Updates

Fig. 7. Effect of $|W|$ on gMission Dataset

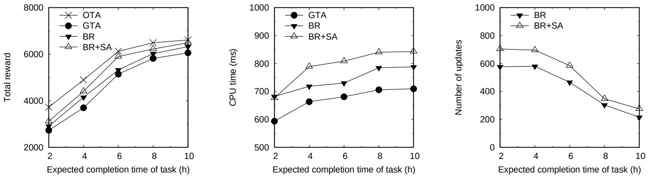
a) Effect of $|S|$*:* To study the scalability of all the algorithms, we generate 5 datasets containing 1000 to 5000 (100 to 500) tasks by random selection from the synthetic dataset (gMission dataset). As shown in Figure 4(a) and 5(a), the total reward of all the methods exhibits a similar increasing trend when $|S|$ grows. Since OTA is the optimal task assignment algorithm, it achieves the highest total reward, followed by BR+SA, BR and GTA, in both synthetic dataset and gMission dataset. BR+SA can obtain at most 96% of the maximal reward, and its reward is consistently higher than that of BR (by up to 8%), which demonstrates the superiority of the simulated annealing optimization strategy. BR and GTA can achieve up to 93% and 82% of the optimal reward, respectively. In Figure 4(b), 4(c), 5(b) and 5(c), despite the CPU cost of all methods being on the rise as $|S|$ increases, our proposed algorithms (including GTA, BR and BR+SA) deliver a clearly superior performance to OTA. OTA deteriorates at a significantly faster pace in respect of efficiency. As expected, GTA is the fastest algorithm, which can improve efficiency by $11\% - 32\%$ ($23\% - 39\%$) compared with BR (BR+SA), while it generates smaller reward when compared to the other algorithms. BR+SA can provide an excellent trade-off between effectiveness (second to OTA) and efficiency (only $30\% - 64\%$ slower than GTA). In Figure 4(d) and 5(d), workers update their strategies more frequently in BR+SA than in BR, which is attributed to the fact that BR+SA allows a worker to randomly change her current strategies whereas a worker in BR can update her current strategies only when there exists a best-response strategy. We also observe that both BR and BR+SA have an increasing number of updates as $|S|$ grows. The reason behind it is self-evident, that is, with more tasks to be assigned, each worker receives more reachable tasks (i.e., available strategies) such that she has more chance to update her strategies.

b) Effect of $|W|$*:* Next we study the effect of $|W|$, the number of workers to be assigned. As illustrated in Figure 6(a), 6(b), 7(a) and 7(b), BR and BR+SA can achieve a higher global reward than GTA while sacrificing some efficiency. However, the computing efficiency of BR and BR+SA is acceptable. Though the total reward of OTA is undoubtedly the highest, it is time-consuming as shown in Figure 6(c) and 7(c). More specifically, BR+SA can get up to 98% of the maximal reward and its CPU cost is significantly less than that of OTA, i.e., the CPU cost of BR+SA is only $10\% - 53\%$ of OTA's. By contrast, the reward obtained by BR (GTA) is only $71\% - 93\%$ ($67\% - 86\%$) of the maximal reward. From Figure 6(d) and 7(d) we can see that the number of updates shows an upward trend with the varying $|W|$ since there are more workers needing to change their strategies when the number of workers to be assigned gets larger. It is noteworthy that the number of workers' strategy updates is much lower than the number of workers to be assigned. This can be accounted for by various reasons. The first one is that some of the workers stick to their initial assigned



(a) Total Reward (b) CPU Time (c) Number of Updates

Fig. 8. Effect of r on Synthetic Dataset



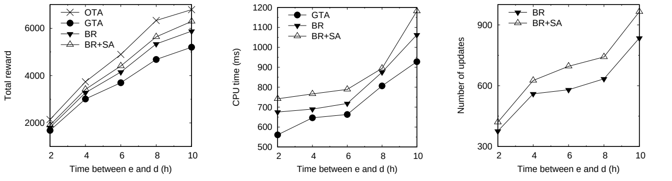
(a) Total Reward (b) CPU Time (c) Number of Updates

Fig. 9. Effect of e on Synthetic Dataset

tasks (both BR and BR+SA algorithms give each worker an initial strategy, i.e., one of her reachable tasks or a null task, refer to Algorithm 2) without making any strategy updates. The second reason is that there exist some workers having no reachable tasks, which means their strategies are always null tasks such that the number of their strategy updates is 0. To save space, in the following experiments, we will neither present the CPU cost of OTA that is really huge, nor report the results of gMission dataset, which show similarity to those of the synthetic dataset.

c) Effect of r: Figure 8 describes the effect of workers' reachable radius, r, on the performance of all the algorithms by changing it from 1 km to 5 km. With the increase of workers' reachable radius, workers have more reachable tasks, as a result of which they have more chance to select the tasks with higher rewards, which explains the rising trends of the total rewards with growing r in Figure 8(a). Accordingly, the CPU cost of all the approaches grows as r varies in Figure 8(b) since workers have to search more reachable tasks to find the suitable ones. Besides, as we can see from Figure 8(c), the number of updates using BR+SA algorithm is always higher than that using BR algorithm, regardless of r.

d) Effect of e: We next study how the expected time of tasks affects the performance of all the methods. Obviously, in Figure 9(a), the total rewards of all the methods gradually increase with the increasing e since larger e implies that more tasks can reach the maximal rewards. OTA still gets the maximal rewards, and BR+SA outperforms BR and GTA. It is noticeable, however, that all the methods tend to maintain stability when $e > 8h$, which may be due to the fact that a majority of the tasks can be completed before $8h$ to achieve their own maximal rewards. In terms of CPU time, the tendency of all the methods is ascending at first and stable afterwards. This is because initially, with larger e, each worker tends to have more reachable tasks and searching these tasks needs more CPU time. Then, as the expected completion time of tasks continues to be extended, the number of reachable tasks for each worker will keep stable due to other spatio-temporal constraints (e.g., workers' reachable distance) such that the CPU time maintains stability. In Figure 9(c), the number of updates for BR and BA+SA decreases as e increases, since workers are more likely to select their satisfied tasks to avoid strategy updates with the larger e.



(a) Total Reward (b) CPU Time (c) Number of Updates

Fig. 10. Effect of $d - e$ on Synthetic Dataset

e) Effect of $d - e$: In the final set of experiments, we study the effect of $d - e$. Not surprisingly, as can be seen in Figure 10(a) and 10(b), all the approaches lead to the higher rewards with more CPU time when the deadlines are more relaxed. As expected, the larger $d - e$ means on average each worker has more reachable tasks, which can contribute to more total rewards and cost more CPU time. Another observation is that, the performance gap between BR-related approaches and GTA algorithm in terms of total reward is also increasing. This is due to the fact, when applying the BR-related algorithms, the total reward is more sensitive to the average number of available worker sets for each task that increases with $d - e$. In such circumstances, the benefits of BR-related approaches become more significant. In addition, the numbers of updates of both BR and BR+SA show an upward trend as $d - e$ grows, depicted in Figure 10(c).

Summary: The take-away message of our empirical study can be summarized as follows:

1) OTA achieves the maximum rewards but sacrifices a great deal of efficiency.

2) BR+SA achieves good balance between efficiency and effectiveness (second to OTA).

3) Although GTA is the most efficient algorithm, it performs worse than other methods in effectiveness.

VI. Related Work

Recent studies in Spatial Crowdsourcing (SC) make great efforts to devise algorithms to solve the task assignment problems [3], [28]–[31]. Nevertheless, these aforementioned studies adopt task assignment methods in a centralized control way without considering the coordination among workers, which incur a substantial amount of computational cost especially in large-scale SC scenarios. Therefore, they greatly increase the system implementation difficulty and human efforts of applying these methods in a practical situation.

Additionally, a majority of the SC researches have been focusing on assigning each task to a single nearby worker based on various system optimization goals. However, in practice, it is inevitable to encounter complex tasks (such as monitoring traffic condition and cleaning rooms) that require a group of workers to conduct, called multiple task assignment mode. In this work we will go further in this direction to address the coalition-based task assignment problem by considering the reward and stability of worker coalitions. In our problem setting, workers are required to form a coalition to perform a task through collaboration. Establishing worker coalitions is an important manner for workers' coordination and cooperation in SC, by which workers can enhance their capability to perform tasks and obtain more utility.

The closest related research to ours is [6], which designs a game theoretic approach to solve the cooperative task assignment problem. However, it differs from our work in terms of the objectives and problem setting. Firstly, [6] aims to maximize the total cooperation quality scores of assignments, while our goal is to maximize the overall rewards for workers. Besides, an implicit assumption made in [6] is that workers are willing to voluntarily perform the tasks assigned to them. Nevertheless, in practice, workers are likely to be reluctant to perform the assigned tasks without actual payments or credits as they have various participation cost (e.g., mobile device battery energy cost) [7], especially for the complex tasks that need a group of workers to conduct together. In our problem, we take workers' rewards into consideration in order to arouse workers' enthusiasm about performing tasks. Moreover, [6] only obtains a Nash equilibrium from the multiple Nash equilibriums through the best-response method, while we aim to achieve a better Nash equilibrium with higher total rewards through a combination of the simulated annealing scheme and the best-response method.

VII. Conclusion

In this paper, we study a novel problem, called Coalition-based Task Assignment (CTA), in spatial crowdsourcing, where an individual worker may not be able to accomplish a task independently since completing the task alone exceeds the capability of this worker. Therefore, workers are required to form stable coalitions with sufficient cumulative capabilities (or time) to finish the tasks. As the CTA problem is proved to be NP-hard, we propose different algorithms (including greedy algorithm and equilibrium-based algorithm) to efficiently and effectively assign tasks to maximize the overall rewards. The first algorithm assigns tasks to the nearby workers greedily and adopts an acceptance possibility to find the high-value task assignments. The equilibrium-based algorithm hybrids the best-response strategy and simulated annealing strategy to find a Nash equilibrium, which is an approximately optimal task assignment. As demonstrated by the extensive empirical study, our proposed solutions can significantly improve the efficiency and effectiveness of task assignment.

Acknowledgment

This work is partially supported by Natural Science Foundation of China (No. 61972069, 61836007 and 61832017), and Sichuan Science and Technology Program under Grant 2020JDTD0007.

References

[1] B. Zhao, P. Xu, Y. Shi, Y. Tong, Z. Zhou, and Y. Zeng, "Preference-aware task assignment in on-demand taxi dispatching: An online stable matching approach," in *AAAI*, 2019, pp. 2245–2252.

[2] Y. Tong, J. She, B. Ding, and L. Wang, "Online mobile micro-task allocation in spatial crowdsourcing," in *ICDE*, 2016, pp. 49–60.

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

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

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

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

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

[8] P. Cheng, X. Lian, L. Chen, J. Han, and J. Zhao, "Task assignment on multi-skill oriented spatial crowdsourcing," *TKDE*, vol. 28, no. 8, pp. 2201–2215, 2015.

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

[10] Y. Zhao, K. Zheng, Y. Li, H. Su, J. Liu, and X. Zhou, "Destination-aware task assignment in spatial crowdsourcing: A worker decomposition approach," *TKDE*, 2019.

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

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

[13] Y. Zhao, J. Xia, G. Liu, H. Su, D. Lian, S. Shang, and K. Zheng, "Preference-aware task assignment in spatial crowdsourcing," in *AAAI*, 2019, pp. 2629–2636.

[14] Y. Tong, Z. Zhou, Y. Zeng, L. Chen, and C. Shahabi, "Spatial crowdsourcing: A survey," *VLDBJ*, 2019.

[15] V. V. Vazirani, *Approximation algorithms*. Springer Science and Business Media, 2013.

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

[17] F. D. and T. J., "Game theory," *MIT Press*, 1991.

[18] J. F. Nash, "Equilibrium points in n-person games," *Proceedings of the National Academy of Sciences of the United States of America36*, pp. 48–49, 1950.

[19] R. B. Myerson, "Game theory: Analysis of conflict," *Harvard University Press*, 1997.

[20] D. E. Kaufman and R. L. Smith, "Fastest paths in time-dependent networks for intelligent vehicle-highway systems application," *IVHSJ*, 1993.

[21] S. Kirkpatrick, C. D. Gelatt, and M. P. Vecchi, "Optimization by simulated annealing," *Science*, vol. 220, p. 671680, 1983.

[22] D. Fudenberg and D. K. Levine, *The Theory of Learning in Games*. MIT Press, 1998.

[23] N. Nisan, T. Roughgarden, E. Tardos, and V. V. Vazirani, "Algorithmic game theory," *MIT Press*, 2007.

[24] A. Fabrikant, C. Papadimitriou, and K. Talwar, "The complexity of pure nash equilibria," *STOC*, pp. 604–612, 2004.

[25] Y. Liu, D. Liang, and R. J. Marks, "Common control channel assignment in cognitive radio networks using potential game theory," in *WCNC*, 2013.

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

[27] S. R. B. Gummidi, T. B. Pedersen, and X. Xie, "Transit-based task assignment in spatial crowdsourcing," in *SSDBM*, 2020.

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

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

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

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