Composite Task Selection with Heterogeneous Crowdsourcing

Jianhui Zhang[†‡], Zhi Li[†], Xiaojun Lin[*], Feilong Jiang[†]

[†]College of Computer Science and Technology, Hangzhou Dianzi University, Hangzhou 310018 China.
[*]School of Electrical and Computer Engineering, Purdue University, IN 47907-2035, U.S.A.
[‡]Corresponding author Email: jhzhang@ieee.org

***Abstract*—A common feature among many crowdsourcing applications is to decompose the huge or complex tasks into some small sub-ones, which require some users with different skills to implement. The kind of tasks are composite and called Composite Tasks (CTs) , which are said to be completed and return reward only after all of their sub-tasks are finished successfully. Meanwhile, users may have various capabilities to implement diverse sub-tasks (STs) with corresponding cost so users are heterogeneous. In this paper, we study the problem of how users choose the STs to maximize their payoff (reward minus cost) when there are multiple such CTs. This payoff maximization problem with multiple CTs and heterogenous users turns out to be NP-completed. We then propose a Local Composite Task Selection (LCTS) algorithm to help the users choose their sub-task strategies. Its convergence and complexity are analyzed theoretically. For comparison, we design a centralized Composite Task Selection (CTS) algorithm and a Low Cost and Random sub-task selection (LCR) algorithm as benchmarks. Numerical results suggest that the LCTS algorithm achieves a similar payoff and task completion ratio to the CTS when the number of users is large. The performance of LCTS is highly over LCR on both of the payoff and the task completion ratio. The results also illustrate the quick convergence of the LCTS algorithm.**

***Index Terms*—Composite Task; Crowdsourcing; Mobile Crowdsensing; Game Theory**

I. Introduction

Today's smartphone and wearable devices have powerful computing capability and are embedded with a rich set of sensors, such as cameras and GPS. Thus, such smart devices can implement various kinds of tasks to support crowdsourcing applications [1][2]. One key advantage of crowdsourcing is to decompose a big complex task into some Sub-Tasks (STs) and to distribute them across a large number of individual users [3]. In many applications, each big complex task is composite and finished only after all of its STs are implemented. We refer to this scenario as *composite task crowdsourcing* (CTC).

Application examples for CTC include SmartPhoto [4], composite event coverage [5], multi-skill spatial crowdsourcing [6]. For instance, SmartPhoto needs some users to take pictures for the same objective from different directions, and then the complete view of the objective is constructed with these directional pictures [4]. Similarly, in composite event coverage, each event composes of some atomic information. A sensor network has to assign sensor nodes with different types of sensors to cover the event [5]. Finally, in multi-skills spatial crowdsourcing, each user has multiple different kinds of skills, and can participate in different location-dependent tasks. Each task need one or more users to implement [6]. All of these applications suggest that the tasks are composite. Furthermore, users have different skills or their smart devices may be embedded with different sensors, and thus are *heterogenous*. On the other side, to implement task usually needs user to spend cost, such as users' cost on the road to task in the location-dependent task selection/assignment scenarios [7][8][9]. The example in Figure 1 shows that the heterogenous users take ways with different distance to implement the STs of two Composite Tasks (CTs). It takes them different costs to finish the STs.

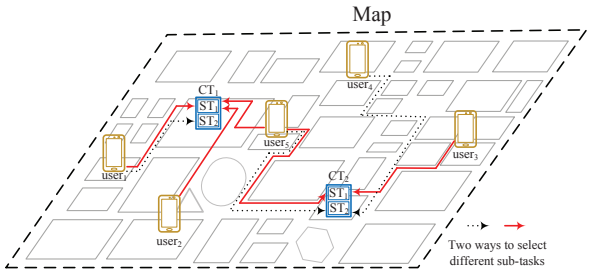


Fig. 1. Each CT has two STs. Five users have several possible ways to implement the STs, ST_1 and ST_2 of the CT_1 and CT_2. Each way has its own cost different from others. If all STs of one CT are completed, the users completing the STs get reward.

An important question for CTC is *how to help users select STs while accounting for their cost and heterogenous skills, and considering the unique feature that they can obtain payoff only after the whole task is finished*. Some recent works have studied the task allocation or selection problem with crowdsourcing, and their goal is to maximize payoff or minimize cost. However, they did not consider composite tasks where the payoff is only obtained after all STs are completed. For example, Cheung *et al.* considered the task selection problem for heterogeneous users with different initial locations, movement costs, movement speeds, and reputation levels. The payoff is shared when multiple users select the same task [8]. He *et al.* considered sensing quality maximization for location-dependent task scheduling with crowdsensing by considering multiple independent sensing measurements for each task [7]. Xiao *et al.* determined a task assignment strategy to minimize the average makespan when users conduct multiple complex

978-1-5090-6599-8/17/$31.00 ©2017 IEEE

computation and sensing tasks [9]. There is another group of works focusing on incentive mechanism design for task allocation while considering the truthfulness or preserving the privacy of users' locations [2][3].

Different from the previous works, which focus on the selection or assignment of location-dependent tasks without considering *task composite* and *user heterogeneity*, we study the location-dependent CT selection by heterogenous users. Since each CT is said to be completed only after all its STs are implemented, it requires users to implement the CTs collaboratively so as to obtain their payoff. Each user has several choice for the STs of each CT and cannot select STs independently. The joint consideration of location dependency, user heterogeneity, and CTs motivates the design of a good solution very challenging.

In this paper, we solve the location-dependent CT selection problem through *local cooperation* among users, who are heterogeneous in their skills or sensor types. We show the problem is NP-complete, and formulate the cooperation problem among users as a local cooperative Composite Task Selection Game (CTSG). A Localized CT Selection algorithm (LCTS) is proposed for each user so that he can collaborate with his neighbors to compute the strategies. Each user only needs limited information on the CT selection strategies and the cost of his neighbors. As *performance benchmarks*, we propose two algorithms: the heuristic greedy Centralized Task Selection algorithm (CTS) and the Low Cost and Random ST selection (LCR) algorithm. CTS computes the approximate centralized solution with a lower complexity by assuming that all users are controlled by a service provider. LCR is a simple and distributed algorithm, by which users choose the CT with the minimum cost and then implements one ST of the CT randomly. In summary, the contributions of our work are as follows:

- CT selection modeling. Since the completion of CT depends on the implementation of all its STs, we present the CT selection model to measure the completeness of the CT implementation while users are heterogeneous in their skills.
- Complexity and Game Formulation. We formulate the CT selection with crowdsourcing as an optimization problem and prove its NP-completeness. It is modeled as a CT selection game, and is proved to be an exact potential game, which indicates the existence of at least one Nash Equilibrium (NE), *i.e.*, at least one sub-optimal solution to the problem.
- Localized solution. This paper designs the LCTS algorithm, which provides a suboptimal solution to the CT selection problem in polynomial time. We prove that LCTS converges into an NE in at most $|A_i||B_i|$ iterations theoretically, where $|A_i|$ and $|B_i|$ are the total numbers of user u_i's neighbors and skills respectively.
- As the performance benchmarks, we propose two other algorithms: CTS and LCR. Numerical results show that the payoff of the LCTS can be quite close to CTS and much better than LCR, and that LCTS can converge quickly.

Most symbols used in this paper are summarized in Table I.

TABLE I
SYMBOL AND MEANING

Sym.	Description	Sym.	Description
a	CT	A	CT set
b	ST	B	ST set/Skill set
c	cost	B_i	User u_i's skill set
C	Overall cost	N	# of CTs
u	User	K	# of STs
s	Strategy	V	Set of users
S	Strategy set	$\mathbf{s}$	Strategy profile
J	Neighboring users	$g(*)$	Reward function

II. PRELIMINARIES AND PROBLEM FORMULATION

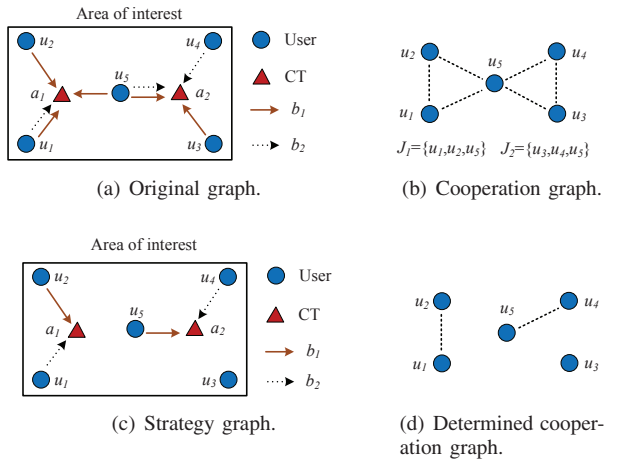
A. System Model

Assume that a service provider offers the location and reward information of the CTs to users in an application area. The users have different skills to implement STs of these CTs with some cost.They will obtain the reward after all STs of a CT is completed. We consider a set of CTs arbitrarily located in a target area, and put no constraint on the distribution of their locations. Let A denote the set of N CTs, $A = \{a_j, j = 1, \cdots, N\}$, and V denote the set of M users $V = \{u_i, i = 1, \cdots, M\}$. Each CT a_j composes of a set of K STs, $a_j = \{b_1^j, b_2^j, \cdots, b_K^j\}$, which can be different for other CTs. Each ST requires one skill to implement so b_k indicates the skill to finish the ST either. Let B denote the set of all skills needed to implement all STs of all CTs, *i.e.*, $B = \bigcup_{a_j \in A} a_j$. Each user u_i can implement one or more kinds of location-dependent STs. Let B_i the set of STs the user can implement, and we have $B_i \subseteq B$. u_i costs $c_k^i(a_j)$ to implement ST b_k^j of CT a_j.

B. Cooperative Graph

Users need to know the cooperative relation with each other so as to find their CT selection strategies. We construct a new graph, called *cooperative graph*, to describe the relationship among users. We can use the original graph to represent the possible ways that users can choose in practical scenario as the example in Figure 1. The Figure 2(a) shows the instance of the original graph. In this figure, each CT has two STs and the brown full line arrow means that user has skill to implement the first type of ST b_1 while the black dash line arrow means that user has skill to implement the second type of ST b_2. The arrow between user and CT represents that the user may obtain positive payoff to implement one ST of the CT. For instance, user u_1 has skill to implement the first and second STs of the CT a_1 with positive payoff. On the other hand, user u_2 has skill to implement the first ST of the CT a_1 with positive payoff. Notice that Figure 2(a) includes all CTs, all users and all of their possible selections with positive payoff.

We characterize the relation among the users by a graph, called *cooperation graph*, as the example in Figure 2(b). We



(a) Original graph. (b) Cooperation graph.

(c) Strategy graph. (d) Determined cooperation graph.

Fig. 2. Users with heterogeneous skills implement two CTs. Each CT has two STs. The users having the skills to implement the STs from a common CT can be neighbors, and they form a cooperation graph as shown in Figure 2(b).

call the users who can implement the STs of a common CT a_j as neighbors, and combine them into a set denoted by J_j. By connecting the neighboring users with dash lines, we obtain the cooperation graph. For example, users u_1, u_2 and u_5 can implement a_1's STs in Figure 2(a), and they can be a_1's neighbors in Figure 2(b) and denoted by a set $J_1 = \{u_1, u_2, u_5\}$. With the cooperation graph, each user can easily find others to cooperate with. We denote the set of user u_i's neighbors by $J_i' = J_i/\{u_i\}$.

After users determine their strategies to select STs, we can obtain a strategy graph. Figure 2(c) shows the instance of the strategy graph obtained from the original graph in Figure 2(a). From the strategy graph, we can figure out the determined cooperative graph as shown in Figure 2(d). The determined cooperative graph is introduced to describe the relation among users after the strategies are taken.

C. Composite Task Selection Problem

Each user can obtain some payoff after the CT is finished completely. This "payoff after completion" model differentiates our work from the previous studies in the literature. In this paper, the payoff equals to the reward minus the cost. Let $r(b_k^j \rightarrow u_i) = 1$ denote that the ST b_k^j of CT a_j is implemented by user u_i, and 0 otherwise. To check whether a CT is *completely implemented* or not, the reward $g(a_j)$ for CT a_j is given by the following equation:

$$g(a_j) = \beta \bigwedge_{b_k^j \in a_j} r(b_k^j \rightarrow u_i) \quad \exists u_i \in J_j \tag{1}$$

where $\bigwedge$ is the truth-functional operator of logical conjunction in logic and mathematics, and means that $g(a_j)$ is equal to a positive value β only after all $r(b_k^j \rightarrow u_i)$ are true. Otherwise, $g(a_j) = 0$. $\beta > 0$ is a constant.

Normally, users are willing to obtain positive payoff. Not all STs can bring positive payoff because it feedbacks limited reward to complete CT. We give the following definitions to define the tasks returning positive payoff.

Definition 1: (Available ST). A ST b_k of CT a_j is said to be available for user u_i if its reward β is over u_i's cost $c_k^i(a_j)$, and denoted by a pair, $s_i = (b_k^j, u_i)$, where $b_k^j \in a_j$ and $u_i \in V$.

Obviously, each user must set the strategy to select available task in order to obtain positive payoff. With a slight abuse of notation, we also use the term s_i to indicate user u_i's strategy to select the ST b_k^j of the CT a_j, *i.e.*, $r(b_k^j \rightarrow u_i) = 1$. Each ST belongs to one CT in A. Let S_i denote the set of all available selection strategies of user u_i.

$$S_i = \{(b_k^j, u_i), j = 1, \cdots, N; k = 1, \cdots, K\} \subset A \times B \tag{2}$$

We have $s_i \in S_i$. Let $\mathbf{s}$ denote the strategy profile describing all users' strategies, *i.e.*, $\mathbf{s} = (s_1, s_2, \cdots, s_M) \in \times_{u_i \in V} S_i$. Let $\mathbf{s}_{-i}$ denote the strategy profile for all users except user u_i, *i.e.*, $\mathbf{s}_{-i} = \{s_1, \cdots, s_{i-1}, s_{i+1}, \cdots, s_M\}$.

When users adopt the strategy file $\mathbf{s}$, according to Equation (1), the reward $g(a_j, \mathbf{s})$ for each user u_i implementing ST of CT a_j is as the following equation:

$$g_i(a_j, \mathbf{s}) = \begin{cases} g(a_j), & r(b_k^j \rightarrow u_i) = 1 \text{ and } b_k^j \in a_j \\ 0, & \text{otherwise} \end{cases} \tag{3}$$

To take the strategy file $\mathbf{s}$, the cost for user u_i to implment ST b_k^j is denoted by $c_k^i(a_j, \mathbf{s})$, where $b_k^j \in a_j$. The overall cost under $\mathbf{s}$ is:

$$C(\mathbf{s}) = \sum_{a_j \in A} c(a_j, \mathbf{s}) = \sum_{a_j \in A} \sum_{u_i \in J_j, b_k \in a_j} c_k^i(a_j, \mathbf{s}) \tag{4}$$

Recall the definition given in Equation (1) and (3). We can calculate the overall reward from two sides: CT and user, and then have the following equation to cumulate the overall reward.

$$G(\mathbf{s}) = \sum_{a_j \in A} g(a_j, \mathbf{s}) = \sum_{u_i \in V} \frac{1}{\gamma_j} g_i(a_j, \mathbf{s}) \tag{5}$$

where γ_j is the number of users to implement the CT a_j. For a common CT, its reward $g_i(a_j, \mathbf{s})$ is counted for γ_j times so we add the parameter $\frac{1}{\gamma_j}$. The goal of this paper is to find a strategy profile so as to maximize the overall payoff as given in the following equation:

$$\max_{\mathbf{s} \in S} \; R(\mathbf{s}) = G(\mathbf{s}) - C(\mathbf{s}) \tag{6}$$

where $S = \times_{u_i \in V} S_i$. The CT Selection Problem (CTSP) is how to find the strategy profile so that the objective in Equation (6) can be maximized.

III. Localized Composite Task Selection Game

This section shows the hardness of the CT selection problem, and then maps the problem to a CT Selection Game (CTSG). We propose a localized algorithm based on the game to handle the problem.

A. Problem Hardness

The hardness of the CTSP given in Equation (6) can be proved by reducing it to the minimum set cover problem, which is NP-complete [10]. Given a universe set V_u and a collection $\mathbf{V}$ of sets whose union equals to the universe set, the minimum set cover problem is to find the smallest sub-collection of $\mathbf{V}_{\min} \in \mathbf{V}$ to cover the universe set V_u. We may consider a CTSP problem by reducing the number of STs of each CT to one. The universe set becomes the union of the CT and users sets, *i.e.*, $V_u = A \cup V$ in our analysis. The elements in the set A are the STs and cannot be selected to cover other elements. In other words, only a part of elements in the universe set V_u can be selected so the simplified CTSP is still more difficult than the minimum set cover problem. When considering each CT has multiple STs, it's equivalent to that each element in A requires to be covered multiple times, which is the set *k*-cover problem [11]. The problem has been proved to be NP-complete. We thus have the following lemma.

Lemma 1: The CTSP in Equation (6) is NP-complete.

B. Composite Task Selection Game

Since the CTSP problem is NP-complete, it is difficult to find a solution with both polynomial execution time and optimal result. We formulate the problem as the CTSG, in which users act as players to select their strategies by local negotiation. They aim to maximize their payoff, *i.e.*, reward minus cost. The equilibria and convergence properties of the game are analyzed.

The game involves the competition and negotiation among neighboring users. Recall that each user selects available CTs to implement, *i.e.*, selecting the available strategies in S_i. It competes for the ST with other users when they select the same ST according to the reward given in Equation (3). For example, u_1 and u_2 compete for the ST b_1 of the CT a_1 in Figure 2(a). Only one of them can succeed. On the other hand, neighboring users need to cooperate with each other to complete all STs of a CT. For example, u_3 and u_4 select the STs b_1 and b_2 of the CT a_2 respectively in Figure 2(a) so that a_2 can be completed. Given the CT set A, the strategy space S, reward in Equation (3), we give the definition of the game as follows.

Definition 2 (Composite Task Selection Game (CTSG)): In the CTSG, each user u_i selects his available STs to implement by cooperating with his neighbors in order to maximize the payoff.

For the game, we define the utility function for each user u_i as follows:

$$U_i(\mathbf{s}) = \frac{1}{\gamma_j}\{g_i(a_j, \mathbf{s}) + \sum_{u_j \in J_j/\{u_i\}} g_j(a_j, \mathbf{s})\} - c_k^i(a_j, \mathbf{s}) \quad (7)$$

where the first term is the achievable payoff of individual user u_i as defined in Equation (3), the second term is the aggregated reward received by his neighbors, and the third terms is u_i's cost under strategy s_i. In other words, each user considers not only his own strategy but also his neighbors' strategies. Then, the goal of the CTSG is expressed as follows:

$$\max_{\mathbf{s} \in S} \sum_{u_i \in V} U_i(\mathbf{s}) \quad (8)$$

It's easy to see that the sum of all users' utility given in Equation (7) is equal to $G(\mathbf{s}) - C(\mathbf{s})$ according to Equation (4), Equation (5) and the definition in Equation (3). Therefore, the objective in Equation (8) is same with that in Equation (6).

In the CTSG, each user adopts the better response dynamic scheme by updating his strategies and negotiating with his neighbors so as to maximize his payoff. Specifically, the user repeats the following process: choosing the ST with better utility in Equation (7), and negotiating with his neighbors to complete all STs of the CTs. What users negotiate with each other is their strategies and cost. The process can lead to convergence and at least the suboptimal solution if the game possesses the *finite improvement property* [12], *i.e.*, the best response updates always converge to a pure NE within a finite number of updates regardless of the initial strategy profile or the users updating order.

Definition 3 (NE): A strategy profile $\mathbf{s}^*$ is called NE if and only if, for each player u_i and an arbitrary strategy s_i in his strategy space, the following inequality is always satisfied.

$$g_i(\mathbf{s}^*) \geq g_i(s_i, \mathbf{s}^*_{-i}) \quad (9)$$

If the game is an exact potential game, it has the finite improvement property [8][12].

Theorem 2: The CTSG is an exact potential game.

Proof: We construct constructs the potential function as follows:

$$\psi(\mathbf{s}) = \sum_{u_i \in V} \{g_i(s_i, s_{J_i'}) - c(s_i)\} \quad (10)$$

where $g_i(s_i, s_{J_i'})$ is individual payoff given in Equation (3), and J_i' is the set of u_i's neighbors.

We separate Equation (10) into two parts in order to show the proof clearly.

$$\psi^p(\mathbf{s}) = \sum_{u_i \in V} g_i(s_i, s_{J_i'}) \quad (11)$$

and

$$\psi^c(\mathbf{s}) = -\sum_{u_i \in V} c(s_i) \quad (12)$$

Considering player u_i who unilaterally changes his strategy from s_i to s_i', where $s_i, s_i' \in S$, the change in the potential function caused by this unilateral change is given by:

$$\begin{aligned}
&\psi^p(s_i, \mathbf{s}_{-i}) - \psi^p(s_i', \mathbf{s}_{-i}) = \sum_{u_j \in V} \frac{1}{\gamma_j}\{g_j(s_j, s_{J_j'}) \\
&- g_j(s_j', s_{J_j'})\} = \frac{1}{\gamma_j}\{g_i(s_i, s_{J_i'}) + \sum_{u_j \in J_i'} g_j(s_j, s_{J_j'}) \\
&+ \sum_{u_j \in V \setminus J_i', u_j \neq u_i} g_j(s_j, s_{J_j'}) - \{g_i(s_i', s_{J_i'}) + \sum_{u_j \in J_i'} g_j(s_j', s_{J_j'}) \\
&+ \sum_{u_j \in V \setminus J_i', u_j \neq u_i} g_j(s_j', s_{J_j'})\}\}
\end{aligned} \quad (13)$$

Because each player's strategy only affects the payoff of his own and neighbors, we have the following equality:

$$\sum_{u_j \in V \setminus J_i, u_j \neq u_i} \{g_j(s_j, s_{J'_j}) - g_j(s'_j, s_{J'_j})\} = 0 \qquad (14)$$

On the other hand, the cost of each user does not depend on other users' strategies besides his own. For the second part in Equation (12), we obviously have:

$$\begin{aligned}\psi^c(s_i, s_{J'_i}) - \psi^c(s'_i, s_{J'_i}) &= -\sum_{u_i \in V} c(s_i, s_{J'_i}) + \sum_{u_i \in V} c(s'_i, s_{J'_i}) \\ &= -c(s_i) + c(s'_i) \qquad (15)\end{aligned}$$

By Equations (13), (14) and (15), we have the following equality:

$$\psi(s_i, \mathbf{s}_{-i}) - \psi(s'_i, \mathbf{s}_{-i}) = U_i(s_i, \mathbf{s}_{-i}) - U_i(s'_i, \mathbf{s}_{-i}) \qquad (16)$$

■

Theorem 2 indicates that the CTSG has at least one pure NE. In particular, any local and global optimal solutions of the CT selection problem result in at least one pure strategy for NE [12].

C. LCTS Algorithm

In this subsection, we design an iterative algorithm to attain the NE of our CTSG by using local information based on the best response dynamic scheme. This paper considers the pure strategy and leaves the mixed strategy as future work.

Definition 4 (Feasible strategy): Suppose that user u_i and his neighbors take a strategy profile $\mathbf{s}$. If $\mathbf{s}$ can implement at least one CT completely, then any strategy $s \in \mathbf{s}$ is called feasible strategy.

It's easy to see that the feasible strategy space S' is smaller than the available strategy space S. A feasible strategy of each user depends on his own and neighbors' strategies. For example, suppose that a CT has two STs b_1 and b_2, and there are two users u_1 and u_2. If u_1 makes decision to implement b_1, and u_2 makes decision to implement b_2, their decisions correspond to feasible strategy since all STs are completed. If both u_1 and u_2 make decisions to implement one of b_1 and b_2, their decisions are not feasible strategy.

The idea of the localized algorithm is that each user exchanges information with his neighbors to find the feasible strategy. They update their strategy repeatedly to find better strategies until a predefined iteration limit τ_{max}. The iteration limitation τ_{max} is set to be large enough so that LCTS can converge. By the algorithm, each user switches repeatedly between two states: initial and decided ones. In the *initial state*, each user can cooperate with all neighbors by choosing any strategy in his feasible strategy space S'. In the *decided state*, it chooses a certain strategy to select a certain ST and does not negotiate with any other neighbors any more unless it finds better strategy or comes back to the initial state. To support the CT selection by crowdsourcing, the algorithm needs a service provider who help users negotiate with each other. When the user claims a ST of a CT to implement, it sends the cost to implement the ST to the service provider. The service provider offers the information of other users which also claim the CT to the user. The algorithm is summarized below, and its convergence to NE and complexity are analyzed thereafter.

Algorithm 1 Localized CT Selection Algorithm (LCTS)

Input: Cost $c(b_j)$, $b_j \in B_i$. Locations of CTs A_i near u_i.
Output: u_i's strategy.

1: User u_i checks available CTs a_j, $a_j \in A$, their STs b_k^j, and neighbors J'_i as Definition 1;
2: u_i sets his strategy being the initial state;
3: **while** $\tau < \tau_{max}$ **do**
4: Obtain strategy s_i of his neighbors $u_j \in J_i$ from the service provider;
5: Choose one feasible strategy s_i so that his payoff U_i by Equation (7) is maximized;
6: **if** $U_i > 0$ **then**
7: Choose the strategy s_i, and accordingly set himself to be decided state;
8: Report s_i and the correspding cost $c_k^i(a_j)$ to the service provider;
9: **else**
10: Set his strategy as the initial state.
11: **end if**
12: **end while**

By Algorithm 1, each user only needs the information on the location, the cost and the affordable STs of his neighbors from the service provider so the algorithm is localized. Each ST can be allocated to only one user so users would not share reward with others for completing a common ST. Otherwise, the payoff would be decreased. This is because, when more than one users complete one common ST, the cost must be increased while the reward does not.

Theorem 2 guarantees that LCTS possesses the finite improvement property. We then analyze how long it takes for all CTs to converge to a pure NE by Algorithm 1. The following theorem ensures that LCTS can compute the response update in polynomial time.

Theorem 3: For the CTSG, the number of iterations to converge to NE by Algorithm 1 is less than $|A|^2|B|$.

Proof: According to Algorithm 1, in each iteration, each user u_i tries to choose the strategy that maximizes his own payoff, *i.e.* the first case in the line 6 of the algorithm, or to wait for his neighbors to choose their strategies that maximize their payoff, *i.e.* the second case in the line 9 of the algorithm. By checking the size of the feasible strategy space S', we can see that an arbitary user u_j has at most $|A_j||B_j|$ strategies to select, where $|*|$ is the set cardinality. Hence, u_i runs at most $|A_i||B_i|$ steps in the second case. In the first case, u_i waits his neighbors at most $\sum_{u_j \in J_i} |A_j||B_j|$ steps. Therefore, Algorithm 1 will take at most $\sum_{u_i \in V} \sum_{u_j \in J_i \cup \{u_i\}} |A_j||B_j| \leq \sum_{u_i \in V} |A||B| = |A|^2|B|$ iterations to converge to an NE. ■

By Theorem 2, the value of τ_{max} can be at most $|A|^2|B|$. Simulation results show that its value is constrained by both the numbers of CTs and users, and be much less than $|A|^2|B|$.

IV. Centralized Composite Task Selection

As a benchmark, this section considers the CTSP, where the service provider can control every user. The goal of the service provider is different from users', and seeks to maximize the number of completely-implemented CTs. Lemma 1 proved that the CTSP problem in Equation (6) is NP-complete. We next design a heuristic greedy centralized algorithm for the centralized problem and show that how far its cost is away from the optimal solution.

Recall that the reward is paid to users only after all STs of a CT are implemented as measured by Equation (1). Thus, the idea of the centralized algorithm is to give priority to the CT with a higher payoff. The utility to implement each CT a_j is:

$$U(a_j) = \sum_{u_i \in J_j} \{g_i(a_j, \mathbf{s}) - c_k^i(a_j, \mathbf{s})\} \tag{17}$$

The greedy centralized CT selection algorithm is given below.

Algorithm 2 Centralized CT Selection Algorithm (CTS)

Input: CT A, user set V and their locations.
Output: Strategy profile $\mathbf{s}$.

```
Set s empty.
while A ≠ ∅ and V ≠ ∅ do
    Calculate the aggregated payoff for each CT a_j according to Equation (17), and obtain strategies for the neighboring users of a_j.
    Sort all CTs in the descending order of the aggregated payoff, and obtain a list.
    Let a_j be the first item of the list, and s_i be the corresponding strategies of its neighboring users, u_i ∈ J_j.
    if U(a_j) > 0 then
        Add s_i, u_i ∈ J_j, into s.
        Delete a_j from A, and the skills of users in J_j which implement a_j.
    else
        Delete a_j from A.
    end if
end while
```

To maximize the overall payoff in Equation (6), Algorithm 2 selects the CT returning maximal utility, which contains two parts, reward and cost in each iteration. Recall the reward rule in Equation (1), which indicates that each CT returns the reward β if it is completely implemented. Thus, the reward is same for any algorithm (includes the optimal solution), while the cost can be quite different under different algorithms. So we are interested in the cost caused by the centralized algorithm. The cost of Algorithm 2 can be theoretically bounded by the following theorem.

Theorem 4: Let C_{opt} and C_{CTS} be the overall cost achieved by the optimal solution and Algorithm 2 for the problem in Equation (6). We have $C_{CTS} < N(1 + \ln N)C_{opt}$.

Proof: To prove the theorem, we introduce the parameter $f = \frac{C}{w}$, where w_k is the number of users to implement CTs till the k^{th} iteration and C is the payoff. f is average payoff per user till the k^{th} iteration. Denote the cost of the optimal solution by C_{OPT} and w for the optimal solution by w_{OPT}. Since each user can have at most K types of skills, each CT requires at least one user to implement. Till k^{th} iteration, w_{OPT} has at least k users. Denote the average cost of CTS by f_{CTS}^k at the k^{th} iteration. We have:

$$f_{CTS}^k \leq \frac{C_{OPT}}{w_{OPT}} \leq \frac{C_{OPT}}{k}$$

Assume that CTS needs γ^k ($\gamma^k \leq K$) users to implement the CT selected by it till k^{th} iteration. The cost of CTS is:

$$\begin{aligned} C_{CTS} &= \sum_{k=1}^{N} \gamma^k f_{CTS}^k \leq N \sum_{k=1}^{N} f_{CTS}^k \\ &\leq N \sum_{k=1}^{N} \frac{C_{OPT}}{w_{OPT}} \leq N \sum_{k=1}^{N} \frac{C_{OPT}}{k} \end{aligned}$$

Since $1 + \frac{1}{2} + \frac{1}{3} + \cdots + \frac{1}{N} < 1 + \ln N$, we have $C_{CTS} \leq N(1 + \ln N)C_{OPT}$. ■

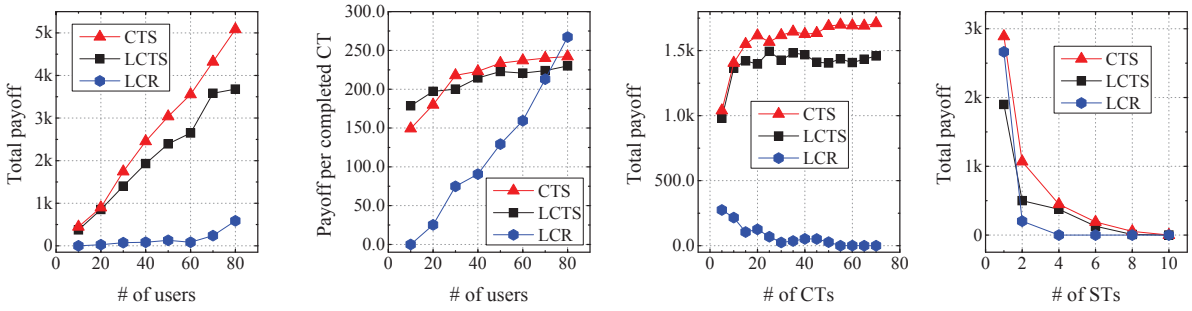
There are totally $|A|$ CTs so Algorithm 2 takes $|A|$ iterations in the "while" loop. In each iteration, each CT has $|B|$ STs, which has at most $|V|$ users to select it. The computation complex of Algorithm 2 is given in the following theorem.

Theorem 5: The computation complexity of Algorithm 2 is $O(|A||B||V|)$.

V. Performance Evaluations

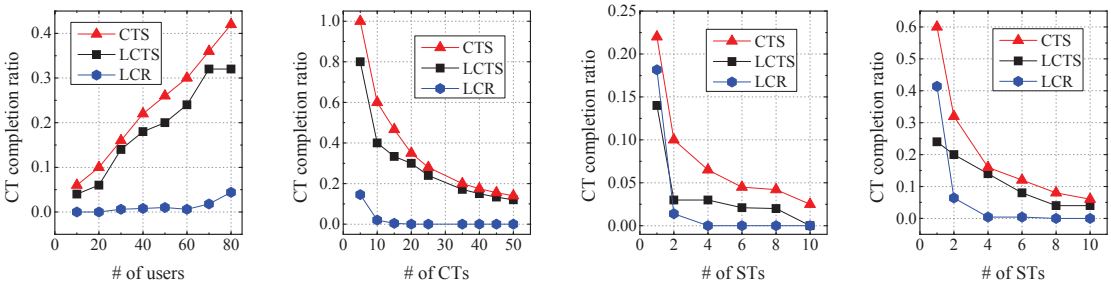
This section evaluates the performance of our proposed LCTS algorithm by comparing with the benchmarks: CTS and LCR. We numerically analyze the impact of various system parameters on the user payoff, the convergence, and the task completion ratio under the three algorithms. As the example in Figure 1, we set the field to a 200m×200m area, and the cost to 1 unit per meter. The locations of all CTs and users are randomly selected in the field. The cost for each user to complete a ST is given by a linear function of the distance between the his location to the task location, at the rate of 1 unit per meter. If a CT is completely implemented, the reward is 300 units. There are totally 10 types of STs. Each user has the skill to randomly implement 4 types of them. In the different cases of the numerical simulation, the number of STs is set to be 1∼10, the number of CTs is 5∼80 and the number of users is 10∼80. We show the numerical results in Figure 3∼6, where each point is the average result of 50 independent simulations with random locations of CTs and users. Each user is randomly assigned a diverse number of skills to implement the STs, and can only select one ST to implement. Each strategy is feasible when it can obtain reward over cost, *i.e.*, positive payoff.

In the following figures, the terms " # of STs" and "STs/CT" mean the number of STs that each CT has. The task completion ratio is the fraction of the completed CTs over all CTs. The title of sub-figures shows the simulation setting. For example, 50 CTs and 4 STs/CT means that the data in



(a) 50 CTs and 4 STs/CT. (b) 50 CT and 4 STs/CT. (c) 30 users and 4 STs/CT. (d) 10 users and 50 CTs.

Fig. 3. Payoff under different numbers of users, CTs and STs.



(a) 50 CTs and 4 STs/CT (b) 40 users and 6 STs/CT. (c) 50 CTs and 10 users (d) 50 CTs and 30 users

Fig. 4. Completion ratio under different numbers of CTs, users and STs.

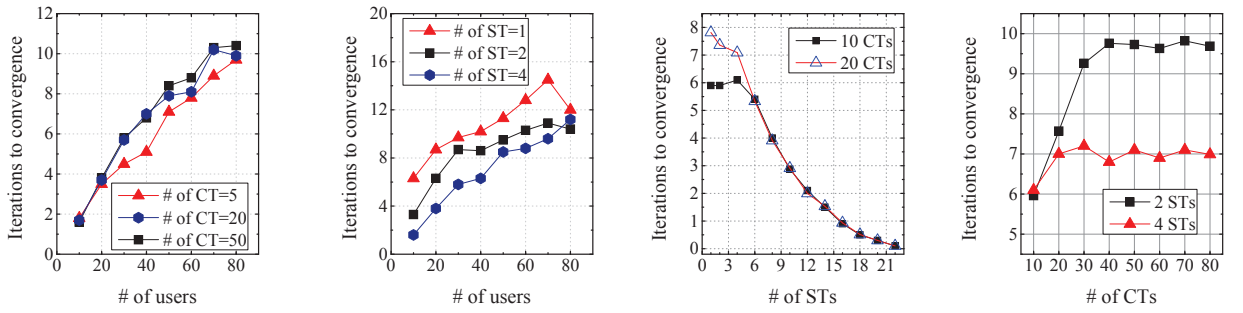
the figure is the results of the numerical simulation when the number of CTs is 50, and each CT has 4 STs.

Payoff. Figure 3 shows the total and average payoff obtained by algorithms LCTS, CTS and LCR, respectively, under different numbers of users, CTs and STs. Figure 3(a) shows that the overall payoff of LCTS, CTS and LCR increases with the number of users. LCTS achieves the similar total payoff to CTS. As shown in Figure 3(c), the average payoff per completed CT of LCTS is 92.12% of that of CTS when the number of users is over 30. In contrast, the overall payoff of LCR is much lower as show in Figure 3(a). Recall that LCR always choose the CT with minimal cost. The number of CTs finished by LCR is also very small. Due to this reason, its average payoff per completed CT increases quickly as Figure 3(b).

The overall payoff does not always increase with the number of CTs under both CTS and LCTS as shown in Figure 3(c). When the number of users is fixed and the number of CTs increases, the total payoff does not increases any more after the number of CTs is up to a certain value, such as 40 in Figure 3(c). Increasing the number of CTs has a negative effect on the performance on LCR. When the number of CTs is over 30, the total payoff of LCR decreases almost to zero as shown in the figure. Because users have different skills and their location is random, each CT needs at least K users to implement. The results in both Figure 3(a) and 3(c) illustrate that increasing the number of user can help increase the total payoff.

Figure 3(d) shows that the total payoff decreases with the number of STs by the three algorithms when the number of user and CTs are fixed. It's interesting to see that the total payoff of LCR is closer to CTS than LCTS when each CT has only single ST. When the number of STs increases, the total payoff of LCR decreases to be lower than the other two algorithms. This figure suggests that algorithms that do not consider CTs, *i.e.*, each task has a single ST, and have been widely researched in previous works, are not suitable for the case with CTs.

Task completion ratio. In Figure 4, we plot the task completion ratio against the different number of CTs, users and STs. The task completion ratio refers to the proportion of the completed CTs in all CTs. Corresponding to the total payoff in Figure 3(a), Figure 4(a) shows that one can increase the completion ratio of both CTS, LCTS and LCR by increasing the number of users since there are more users to work on the tasks. The completion ratio of LCTS is quite close to that of LCTS, and much higher than that of LCR. It increases with the number of users when fixing the number of CTs. The ratio by LCTS is 77.2% of that of CTS on average under different numbers of users. Increasing the number of CTs has a negative impact on the three algorithm as shown in Figure 4(b). LCTS attains the performance close to CTS while the completion ratio of LCR goes down to zero quickly. The ratio by LCTS is 81.18% of that by CTS on average under different numbers



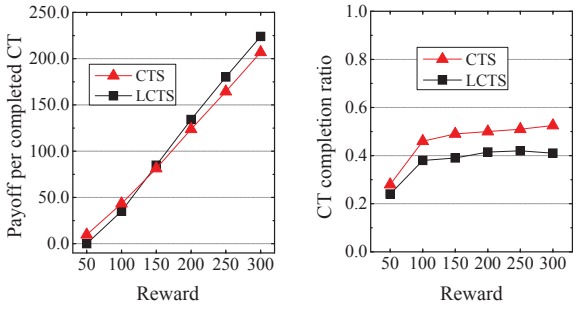
(a) Each CT has 4 STs. (b) The number of CTs is 50 (c) The number of users is 40. (d) The number of users is 40.

Fig. 5. Convergence of LCTS.

of CTs in Figure 4(b). When the number of CTs increases, more users are needed. Hence, a smaller number of CTs are completed. Due to a similar reason, the increase of the number of STs also worsens the ratio as shown in Figure 4(c). The completion ratios of three algorithms fall down to zero. We enhance the result by changing the number of users from 10 to 30 as shown in Figure 4(d), which leads to the conclusion similar to Figure 4(c). The difference is that the completion ratio of LCTS is closer to CTS when the number of users is increased.

Convergence. This block analyzes the convergence of LCTS. We do not report the convergence of other algorithms since both CTS and LCR need not update their strategies repeatedly due to no negotiation among users. Figure 5 illustrates the convergence of the LCTS. From these figures, we can see: 1) LCTS converges quickly and in less than 12 iterations in most cases, which is much less than the theoretical bound $|A|^2|B|$ as given in Theorem 3. This is because of the limited reward. Only some CTs can bring users with positive payoff. 2) When the number of users is fixed, the number of iterations for convergence decreases with the number of the STs that each CT has. This is because the more STs that each CT has, the less CT can be finished with a fixed number of users. 3) As the number of CTs increasing, the number of iterations increases first, and keeps stable thereafter. When the number of CTs is relatively small, the number of CTs that can be finished increases since there are enough users who are able to implement them. When the number of CTs is relatively large, the number of CTs that can be finished is determined by the numbers of users and the number of STs that each CT has. When the numbers of the two later ones are fixed, the number of iterations also keep somewhat stable.

Reward. In the previous experiment, the amount of reward is fixed, *i.e.*, 300 units. We evaluate the performance of the algorithms CTS and LCTS when each CT has various rewards. In the experiment, there are totally 20 CTs and 40 users and each CT has 4 STs. Figure 6(a) shows that an increasing reward can improve the payoff for each completed CT. Increasing reward can also improve the completion ratio when the amount of reward is low. Figure 6(b) indicates that the completion ratios under both CTS and LCTS increases with the reward when the reward is less than 150. After this point, the completion ratio flattens. The underlying reason is that, when the reward continues increasing, the completion ratio is constrained by the numbers of users, CTs and STs. In this experiment, 40 users can finish at most 10 CTs since each CT has 4 STs. Thus, one cannot increase the completion ratio indefinitely by additional reward since it is constrained also by the numbers of CTs and users.



(a) Payoff per completed CT. (b) Completion Ratio

Fig. 6. The performance of CTS and LCTS with increasing reward. There are 20 CTs and 40 users. Each CT has 4 STs.

VI. Related Work

A. Task Selection Application

Applications on crowdsourcing develop quickly in recent years [13], such as VTrack [14] and SignalGuru [15]. Many applications require users to cooperatively implement tasks.

With VTrack, users can use the sensors in their smartphones to sense location data, and then collaboratively estimate travel time [14]. By SignalGuru, users opportunistically detect current traffic signals with cameras in their smartphone, collaboratively communicate and learn traffic signal schedule patterns, and predict their future schedule [15]. Urban air pollution sensing focuses on the air quality measurement, such as N-SMARTS [16] and Air Sensor [17]. This kind of project requires a large number of participants to cooperate so that the air quality over a large area can be measured and the measurement can be precise. The construction of fine-grained

noise maps using uploaded data captured by users smartphone microphones, such as Ear-Phone [18] and NoiseTube [19]. In these application examples, users cooperatively implement a small part of the job so that the whole complex task is finished.

B. Task Assignment Methods

In recent years, the location-dependent task assignment for crowdsourcing has received great attention due to rich sensors in smart phone [7][6][20][21]. For example, He *et al.* considered the time traveling to the location at which the tasks wait to be sensed [7]. Cheung *et al.* studies the task selection problem for heterogeneous users with different initial locations, movement costs, movement speeds, and reputation levels [8]. [6] considers the direction of location-dependent task assignment, in which each user has multiple skills. Users seek spatial tasks under the constraints of arrival deadline and cost budget. However, they did not consider that each task has multiple ST. Gao *et al.* presented the scenario that a composite event has several atomic event, and studied the composite event coverage in wireless sensor network [5]. They studied that there are several types of sensors and each type of sensors sample one kind of data and each location has several types of data to sample. Their goal is to design the coverage scheme. In their works, the coverage quality of each location is the probabilistic combination of the several kinds of sampled data so it's not necessary to sample all kinds of data. The problem under their scenario is quite different from ours.

VII. CONCLUSION

This paper studies the location-dependent composite task assignment problem for the commercial crowdsourcing applications. Unlike previous works, the users only receive reward when all STs of the CT are completed. We formulate this problem as an optimization problem and analyze it from the perspectives of both local cooperative game and centralized optimization. We proved that finding the optimal solution is NP-hard, and then proposed the localized composite task assignment algorithm (LCTS) to help users determine their ST selections cooperatively. We prove that LCTS can converge to NE in a finite number of iterations, and that the local cooperative game is an exact potential game theoretically. So the algorithm LCTS achieves the sub-optimality by finding NE. The convergence and sub-optimality of LCTS are further verified numerically. Numerical evaluation demonstrate that the performance of LCTS can be quite close to CTS.

ACKNOWLEDGE

This work is under the support of the General and Younger Program of the National Natural Science Foundation of China under Grants No.61473109, 61572164 and 61602141, the Public Welfare Technology Application Research Program of Zhejiang Province under Grant No.2015C33067, the Graduate Scientific Research Foundation and the Excellent Dissertation Fostering Foundation of Hangzhou Dianzi University.

REFERENCES

[1] Raghu K Ganti, Fan Ye, and Hui Lei. Mobile crowdsensing: current state and future challenges. *IEEE Communications Magazine*, 49(11):32–39, 2011.

[2] Dong Zhao, Xiang-Yang Li, and Huadong Ma. How to crowdsource tasks truthfully without sacrificing utility: Online incentive mechanisms with budget constraint. In *In Proc. of IEEE INFOCOM*, pages 1213–1221, 2014.

[3] Xinglin Zhang, Zheng Yang, Wei Sun, Yunhao Liu, Shaohua Tang, Kai Xing, and Xufei Mao. Incentives for mobile crowd sensing: A survey. *IEEE Communications Surveys & Tutorials*, 18(1):54–67, 2016.

[4] Yi Wang, Wenjie Hu, Yibo Wu, and Guohong Cao. SmartPhoto: a resource-aware crowdsourcing approach for image sensing with smartphones. In *Proc. of ACM MobiHoc*, pages 113–122. ACM, 2014.

[5] Jing Gao, Jianzhong Li, Zhipeng Cai, and Hong Gao. Composite event coverage in wireless sensor networks with heterogeneous sensors. In *In Proc. of IEEE INFOCOM*, pages 217–225, 2015.

[6] 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*, 28(8):2201–2215, 2016.

[7] Shibo He, Dong-Hoon Shin, Junshan Zhang, and Jiming Chen. Toward optimal allocation of location dependent tasks in crowdsensing. In *In Proc. of IEEE INFOCOM*, pages 745–753, 2014.

[8] Man Hon Cheung, Richard Southwell, Fen Hou, and Jianwei Huang. Distributed time-sensitive task selection in mobile crowdsensing. In *Proc. of ACM MobiHoc*, pages 157–166. ACM, 2015.

[9] Mingjun Xiao, Jie Wu, Liusheng Huang, Yunsheng Wang, and Cong Liu. Multi-task assignment for crowdsensing in mobile social networks. In *In Proc. of IEEE INFOCOM*, pages 2227–2235, 2015.

[10] Stephen A Cook. The complexity of theorem-proving procedures. In *Proceedings of the third annual ACM symposium on Theory of computing*, pages 151–158, 1971.

[11] Zoë Abrams, Ashish Goel, and Serge Plotkin. Set *k*-cover algorithms for energy efficient monitoring in wireless sensor networks. In *Proceedings of ACM IPSN*, pages 424–432, 2004.

[12] Dov Monderer and Lloyd S Shapley. Potential games. *Games and economic behavior*, 14(1):124–143, 1996.

[13] Georgios Chatzimilioudis, Andreas Konstantinidis, Christos Laoudias, and Demetrios Zeinalipour-Yazti. Crowdsourcing with smartphones. *IEEE Internet Computing*, 16(5):36–44, 2012.

[14] Arvind Thiagarajan, Lenin Ravindranath, Katrina LaCurts, Samuel Madden, Hari Balakrishnan, Sivan Toledo, and Jakob Eriksson. Vtrack: accurate, energy-aware road traffic delay estimation using mobile phones. In *ACM Proceedings of the 7th Conference on Embedded Networked Sensor Systems*, pages 85–98, 2009.

[15] Emmanouil Koukoumidis, Li-Shiuan Peh, and Margaret Rose Martonosi. Signalguru: leveraging mobile phones for collaborative traffic signal schedule advisory. In *Proceedings of the 9th international conference on Mobile systems, applications, and services*, pages 127–140. ACM, 2011.

[16] R. Honicky, E. A. Brewer, E. Paulos, and R. White. N-smarts: networked suite of mobile atmospheric real-time sensors. In *ACM SIGCOMM Workshop on Networked Systems for Developing Regions*, pages 25–30, 2008.

[17] The Air Sensor Toolbox: Citizen Scientists Measure Air Quality. https://crowdsourcing-toolkit.sites.usa.gov/air-sensor-toolbox/.

[18] Rajib Kumar Rana, Chun Tung Chou, Salil S Kanhere, Nirupama Bulusu, and Wen Hu. Ear-phone: an end-to-end participatory urban noise mapping system. In *In Proceedings of ACM/IEEE IPSN*, pages 105–116, 2010.

[19] Matthias Stevens and Ellie DHondt. Crowdsourcing of pollution data using smartphones. In *Workshop on Ubiquitous Crowdsourcing*, 2010.

[20] Zhenni Feng, Yanmin Zhu, Qian Zhang, L. M. Ni, and A. V. Vasilakos. Trac: Truthful auction for location-aware collaborative sensing in mobile crowdsourcing. In *In Proceedings of IEEE INFOCOM*, pages 1231–1239, 2014.

[21] J Zhang, Z Li, and S Tang. Value of information aware opportunistic duty cycling in solar harvesting sensor networks. *IEEE Transactions on Industrial Informatics*, 12(1):348–360, 2016.