Distributed Time-Sensitive Task Selection in Mobile Crowdsensing

Man Hon Cheung[*#], Richard Southwell[#], Fen Hou[*], and Jianwei Huang[#]
[*]Department of Electrical and Computer Engineering, University of Macau, Macau
[#]Department of Information Engineering, The Chinese University of Hong Kong, Hong Kong, China
mhcheung@ie.cuhk.edu.hk, richardsouthwell254@gmail.com, fenhou@umac.mo, jwhuang@ie.cuhk.edu.hk

ABSTRACT

With the rich set of embedded sensors installed in smartphones and the large number of mobile users, we witness the emergence of many innovative commercial mobile crowdsensing applications that combine the power of mobile technology with crowdsourcing to deliver time-sensitive and location-dependent information to their customers. Motivated by these real-world applications, we consider the task selection problem for heterogeneous users with different initial locations, movement costs, movement speeds, and reputation levels. Computing the social surplus maximization task allocation turns out to be an NP-hard problem. Hence we focus on the distributed case, and propose an asynchronous and distributed task selection (ADTS) algorithm to help the users plan their task selections on their own. We prove the convergence of the algorithm, and further characterize the computation time for users' updates in the algorithm. Simulation results suggest that the ADTS scheme achieves the highest Jain's fairness index and coverage comparing with several benchmark algorithms, while yielding similar user payoff to a greedy centralized benchmark. Finally, we illustrate how mobile users coordinate under the ADTS scheme based on some practical movement time data derived from Google Maps.

Categories and Subject Descriptors

C.2.4 [**Computer-Communication Networks**]: Distributed Systems—*Distributed applications*

Keywords

Mobile crowdsensing; crowdsourcing; game theory

1. INTRODUCTION

Today's smartphones and wearable devices include a rich set of *embedded sensors*, such as cameras, microphones, global positioning systems (GPS), thermometers, and accelerometers [4, 10]. Thanks to the large number of mobile users and their inherent mobility, we are witnessing the rise of *mobile crowdsensing* (MCS), where individuals use their mobile devices to collectively extract and share information related to some phenomenon of interest. Applications of MCS include traffic jam alerts, wireless indoor localization, and small cell network monitoring.

Recently, commercial MCS platforms, such as Gigwalk [6] and Field Agent [3], combine the power of mobile technology with crowdsourcing to provide their customers with *time-sensitive* and *location-dependent* information related to their stores and products. For example, in Field Agent [3], there are two types of *tasks*, namely audit and research. An audit task mainly involves fact finding and data gathering (such as checking on-shelf availability and prices), while a research task mainly involves the collection of users' opinions and insights (such as surveys and shop-alongs). For a location-dependent and time-sensitive sensing task, such as checking the on-shelf availability of Coca-Cola in a convenience store on Main Street at 9am, the users need to collect the data at the *precise* time and location. With this information, a district manager can reduce the cost of taking inventories, while maintaining the proper stock levels at different stores. Currently, several well-known brands and retailers (such as Coca-Cola and Johnson & Johnson) are among the customers of Gigwalk and Field Agent. This suggests that the collection of location-dependent and time-sensitive data using MCS is a practice of growing importance.

A key question for MCS platforms is *how to find users to complete tasks, while accounting for the users' different initial locations, movement costs, movement speeds, and reputation levels?* A number of recent results have focused on the *centralized* task allocation in MCS, with the objectives of either improving the energy efficiency or maximizing the social surplus. For example, Sheng *et al.* in [14] considered opportunistic energy-efficient collaborative sensing. Given a set of roads, mobile devices, and their trajectories, the objective is to find a sensing schedule that minimizes the total energy consumption by reducing redundancy in sensing. Zhao *et al.* in [15] considered fair and energy-efficient task allocation in MCS, and solved a min-max aggregate sensing time problem. He *et al.* in [7] considered social surplus maximization for location-dependent task scheduling in MCS. They formulated a task scheduling problem, where the objective is to maximize the overall net reward of all the users, subject to the time budget of each user and the sensing redundancy constraint of each sensing task.

Different from the above literature, which focuses on the collection of location-dependent data *without* any time constraints in a *centralized* fashion, we consider the case where the service provider aims to collect *time-sensitive* and location-dependent information for its customers through *distributed* decisions of mobile users. The solution of such a problem needs to balance the rewards and movement costs of the users for completing tasks. The consideration along both space and time dimensions makes the design of a good solution very challenging. The distributed nature of the solution

that is required by the current commercial platform such as Gigwalk and Field Agent further complicates the analysis.

In this paper, we focus on solving the distributed time-sensitive and location-dependent task selection problem, where users are heterogeneous in their initial locations, movement costs, movement speeds, and reputation levels. We formulate the interactions among users as a non-cooperative task selection game (TSG), and propose an asynchronous and distributed task selection (ADTS) algorithm for each user to compute her task selection and mobility plan. Each user only requires limited information on the aggregate task choices of all users, which is publicly available in many of today's crowdsourcing platforms, such as Amazon Mechanical Turk (MTurk) [1], Gigwalk [6], and Field Agent [3].

As a *performance benchmark* on the social surplus, we formulate a centralized task allocation (CTA) problem, assuming that all users are controlled by the service provider, and show that it is an NP-hard problem. We propose a heuristic greedy centralized algorithm that computes the approximate centralized solution with a lower complexity.

To the best of our knowledge, this is the first paper that considers distributed time-sensitive and location-dependent data collection in MCS motivated by real-life commercial applications. To summarize, the contributions of our work are as follows:

- *Practical MCS modeling*: Motivated by commercial MCS applications, we consider the collection of time-sensitive and location-dependent sensing data by multiple users. We assume that the users are heterogeneous in their initial locations, movement costs, movement speeds, and reputation levels.
- *Asynchronous and distributed task selection algorithm*: We propose a distributed algorithm that helps the users determine their task selections and mobility plans. Each user only needs to know limited information on the aggregate task choices readily available in the MCS mobile apps.
- *Convergence guaranteed*: We show that the task selection game has the finite improvement property, which means that users' asynchronous best response updates globally converge to a Nash equilibrium. We also show that each best response update can be computed in polynomial time.
- *Balanced performance*: Simulation results suggest that the proposed asynchronous and distributed task selection scheme achieves the best performance in terms of fairness and coverage comparing with various benchmark algorithms.

2. SYSTEM MODEL

We consider a mobile crowdsensing (MCS) platform that involves the collection of *time-sensitive* and *location-dependent* information in the form of photos, video, audio, data, opinions, and feedback. In the platform, once a user has installed the mobile app on her smartphone, she will be able to use the built-in map to check the available tasks, together with the task attributes including locations, execution times, and rewards, as shown in Fig. 1.

More specifically, let $\mathcal{L} = \{1, \ldots, L\}$ be the set of *locations*, such as buildings, landmarks, or fields. Let $\mathcal{T} = \{1, \ldots, T\}$ be the set of *time slots*. Let $\mathcal{K} = \{1, \ldots, K\}$ be the set of *tasks*. We describe the attributes related to a task as follows.

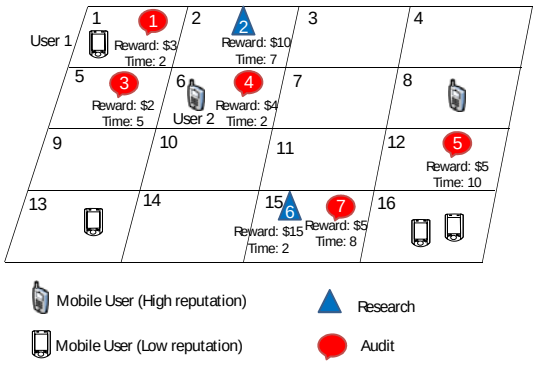

Figure 1: In a MCS platform, we assume that the service provider collects location-dependent time-sensitive information with the help of the users. Through the mobile app interface, the service provider would announce the rewards, locations, and execution times of the tasks to the users. Each user then decides how to move and which tasks to work on over the coming time slots.

DEFINITION 1. (Task Attributes) Each task $k \in \mathcal{K}$ is associated with:

- A location $l[k] \in \mathcal{L}$ *where* the task must be performed.
- A time $t[k] \in \mathcal{T}$ *when* the task must be executed[1].
- A *reward* amount $\rho[k] \geq 0$ for performing the task. In the case where multiple users perform the same task, the reward is shared evenly among them [2].

For example, for task 5 in Fig. 1, we have $l[5] = 12$, $t[5] = 10$, and $\rho[5] = \$5$. Notice that there can be multiple tasks at the same location, such as $l[6] = l[7] = 15$.

Let $\mathcal{I} = \{1, \ldots, I\}$ be the set of users, who can move through space and perform tasks. Given a user's initial location, the set of tasks that she chooses to work on is influenced by her *movement time*, *movement cost*, and *reputation*. Formally, we define a user's attributes as follows.

DEFINITION 2. (User Attributes) Each user $i \in \mathcal{I}$ is associated with:

- An *initial location* $l_i^{\text{init}} \in \mathcal{L}$. (For example, we have $l_2^{\text{init}} = 6$ in Fig. 1.)
- An (integer valued) *movement time* $\Delta_i^{l,l'} \geq 1$, which equals the number of time slots it takes user i to move from location l to location l'.
- A (real valued) *movement cost* $c_i^{l,l'} \geq 0$ for user i to move from location l to location l'.
- A non-empty set $\mathcal{K}_i \subseteq \mathcal{K}$ of tasks that user i is eligible for.

[1]Each task k only takes one time slot to execute, and it cannot be executed before the corresponding time $t[k]$.

If a user has specified her mode of transportation (e.g., walking, cycling, driving, or taking the bus), then her mobile device can retrieve data from Google Maps and local public transportation databases to compute her movement time and movement cost. For example, user i may have a zero cost (i.e., $c_i^{l,l'} = 0$) for walking, and a linear cost for driving[2].

The set of eligible tasks $\mathcal{K}_i$ often relates to the *reputation* of user i. Reputation mechanisms are commonly found in crowdsensing applications, such as Gigwalk [6] and MTurk [1], to allocate tasks based on users' past performance. A service provider can reserve the more complicated but higher paying tasks (e.g., research tasks) for users with higher reputations. For example, in Fig. 1, the lower reputation user 1 can only work on the audit tasks, so $\mathcal{K}_1 = \{1, 3, 4, 5, 7\}$; whereas the higher reputation user 2 can work on both the audit and research tasks, so $\mathcal{K}_2 = \{1, \ldots, 7\}$.

Given the rewards, locations, and execution times of different tasks, each user needs to decide how she should move and which tasks she should select in order to maximize her total profit (i.e., reward minus movement cost). In Section 3, we formulate the task selection of multiple users as a task selection game, and propose an ADTS algorithm. In Section 4, we establish a performance benchmark under the ideal centralized task allocation scenario. We use simulations to evaluate the performance of our proposed algorithms in Section 5, and draw conclusions in Section 6.

3. DISTRIBUTED TASK SELECTION GAMES

In this section, we formulate the users' task selection problem as a *task selection game (TSG)*, where each user aims to maximize her own payoff. We also study the equilibria and convergence properties of the game.

3.1 Task-Time Routing

A user's decision making includes two aspects. First, she must choose which locations to visit over the T time slots (i.e., selecting a route through space-time). Second, she must choose which tasks to do along that route. However, since each task k is only identified with a single location $l[k]$ and time $t[k]$, we can simplify the modeling by just considering how the users select different tasks at different times (i.e., choosing a route through *task-time*).

Since a user may choose not to work on any task, for each user i we define an initial virtual task $k_i^{\text{init}} \in \mathcal{K}_i$, which occurs at all time slots $t[k_i^{\text{init}}] \in \mathcal{T}$ and at the same location $l[k_i^{\text{init}}] = l_i^{\text{init}}$ (user i's initial location) with a zero reward (i.e., $\rho[k_i^{\text{init}}] = 0$).

To clearly describe the task-time routing decision problem, we use a sequences of task-time points, described in Definition 3, to represent a user's task-time route. We say that a task-time route is available to a user when it is physically possible for her to visit the corresponding locations over the time slots, and she is eligible to perform each of the tasks at those locations.

Definition 3. (Available Task-Time Route) An *available task-time route for a user i* is a sequence

$$r_i = \left(\left(k_i^1, t_i^1\right), \left(k_i^2, t_i^2\right), \ldots, \left(k_i^n, t_i^n\right)\right) \in (\mathcal{K} \times \mathcal{T})^n \qquad (1)$$

[2]We can set the movement cost $c_i^{l,l'} = \infty$ if a user i is unwilling or unable to travel from location l to location l'.

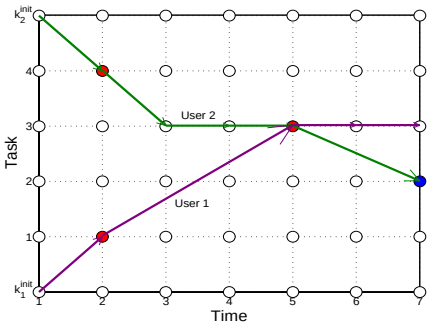

Figure 2: Task-time routes chosen by two users with audit (red circles) and research (blue circle) tasks.

of n task-time points (for some $n \geq 1$) which satisfies the following conditions:

1. *Time increases:* $1 = t_i^1 < t_i^2 < \ldots < t_i^n \leq T$.
2. *The user is eligible for the tasks:* $k_i^1, k_i^2, \ldots, k_i^n \in \mathcal{K}_i$.
3. *Sequence starts at the initial virtual task*: $k_i^1 = k_i^{\text{init}}$.
4. *Sequence accounts for movement time:* $t_i^{a+1} - t_i^a = \Delta_i^{l[k_i^a], l[k_i^{a+1}]}$, for each $a \in \{1, \ldots, n-1\}$.

Condition 1 accounts for the fact that time is always increasing. Condition 2 ensures that the users are eligible to perform each of their chosen tasks. Condition 3 ensures that the user begins at her initial location. Condition 4 ensures that the time difference between successive elements in the sequence of task-time elements is equal to the movement time between the locations of the corresponding tasks. (In other words, it ensures that the movement time is indeed the time they spend moving.) When a user does not move, we define $\Delta_i^{l,l} = 1, \forall\, i \in \mathcal{I}, l \in \mathcal{L}$ to represent the fact that the user stays at the same location after one time slot.

Based on the example in Fig. 1 by considering task 1 to task 4, we show an example of the task-time routes of two users in Fig. 2. Here the solid and empty circles represent task-time points with and without positive rewards, respectively. When a user's route passes through a solid circle, it means that the user works on the task (e.g., user 2 works on task 3 at time 5). On the other hand, when a user's route passes through an empty circle, it means that the user is only physically present at the location of a task, but she is not working on the task (e.g., user 2 is at location $l[3]$ at time 3). In this way, a user can move to the location of a task before the actual execution time of that task (e.g., user 2 arrives at location of task 3 at time 3, while the execution time of task 3 is at time 5).

In Fig. 2, we suppose that a low reputation user 1 has movement times $\Delta_1^{l_1^{\text{init}}, l[1]} = 1$ and $\Delta_1^{l[1], l[3]} = 3$, while a high reputation user 2 has movement times $\Delta_2^{l_2^{\text{init}}, l[4]} = 1$, $\Delta_2^{l[4], l[3]} = 1$, and $\Delta_2^{l[3], l[2]} = 2$. The purple and green curves represent the two task-time routes: $r_1 = \big((k_1^{\text{init}}, 1), (1, 2), (3, 5), (3, 6), (3, 7)\big)$ and $r_2 = \big((k_2^{\text{init}}, 1), (4, 2), (3, 3), (3, 4), (3, 5), (2, 7)\big)$.

3.2 Task Selection Game

Based on the discussion aforementioned, we can formulate the users' task selection problem as a TSG, where users act as players that choose available task-time routes.

In the task-time routing framework, changing tasks will often involve changing locations and hence will involve movement costs. To allow a user to move to the location of a task before its execution time, we define the *time-dependent reward* $\rho^*[k,t']$ for task $k \in \mathcal{K}$ and time $t' \in \mathcal{T}$ as

$$\rho^*[k,t'] = \begin{cases} \rho[k], & \text{if } t' = t[k], \\ 0, & \text{otherwise.} \end{cases}$$

For example, in Fig. 2, the rewards for task-time points $(3,3)$ and $(3,5)$ are $\rho^*[3,3] = 0$ and $\rho^*[3,5] = \rho[3] = \$2$, respectively. We assume that the reward $\rho^*[k,t']$ is evenly shared among all the users who have chosen to work on task k at time t' (i.e., the task-time point (k,t').)

DEFINITION 4. (Task-time points) Let us define the *task-time points* of a task-time route r_i in (1) to be the set $\mathbb{V}(r_i) = \{(k_i^1, t_i^1), (k_i^2, t_i^2), \ldots, (k_i^n, t_i^n)\}$, of all task-time points visited by the route $r_i \in \mathcal{R}_i$, where $\mathcal{R}_i$ denotes the set of all available task-time routes of user i.

The *payoff* $U_i(\boldsymbol{r})$ that user i gets for choosing route r_i in (1) in a strategy profile $\boldsymbol{r} = (r_1, \ldots, r_I) \in \mathcal{R}_1 \times \ldots \times \mathcal{R}_I$ is equal to

$$U_i(\boldsymbol{r}) = \left(\sum_{a=1}^{n} \frac{\rho^*[k_i^a, t_i^a]}{m^{(k_i^a, t_i^a)}(\boldsymbol{r})} \right) - \sum_{a=1}^{n-1} c_i^{l[k_i^a], l[k_i^{a+1}]}, \quad (2)$$

where $m^{(k,t)}(\boldsymbol{r}) = |\{j \in \mathcal{I} : (k,t) \in \mathbb{V}(r_j)\}|$ is the number of routes that pass through task-time point (k,t). The first term in (2) corresponds to the total reward that user i obtains (taking into account how the reward is shared evenly when multiple users perform the same task). The second term in (2) corresponds to the total movement cost, which user i spends in order to travel to the locations of the selected tasks.

DEFINITION 5. (Task selection game) A *task selection game* $\Omega = \left(\mathcal{I}, \mathcal{L}, \mathcal{K}, \mathcal{T}, (\rho^*[k,t])_{k \in \mathcal{K}, t \in \mathcal{T}}, (c_i^{l,l'}, \Delta_i^{l,l'})_{i \in \mathcal{I}, l, l' \in \mathcal{L}}\right)$, involves each user (player) i choosing an available task-time route (strategy) $r_i \in \mathcal{R}_i$ and receiving the payoff in (2).

DEFINITION 6. (Task-time point pairs) The *task-time point pairs* $\mathbb{E}(r_i)$ of a task-time route r_i in (1) is the set $\mathbb{E}(r_i) = \left\{ \left[(k_i^a, t_i^a), (k_i^{a+1}, t_i^{a+1}) \right] : a = 1, \ldots, n-1 \right\}$ of task-time point pairs subsequently visited by the route r_i.

Let $g_i\left(\left[(k_i^a, t_i^a), (k_i^{a+1}, t_i^{a+1}) \right]\right) = c_i^{l[k_i^a], l[k_i^{a+1}]}$ be the movement cost between two adjacent task-time points of user i. The payoff of user i in (2) can be rewritten as

$$U_i(\boldsymbol{r}) = \left(\sum_{(k,t) \in \mathbb{V}(r_i)} \frac{\rho^*[k,t]}{m^{(k,t)}(\boldsymbol{r})} \right) - \sum_{e \in \mathbb{E}(r_i)} g_i(e). \quad (3)$$

3.3 Equilibrium Existence and Convergence Analysis

Before analyzing the task selection games, let us recall some commonly used definitions from game theory. Let $\boldsymbol{r}_{-i} = (r_1, \ldots, r_{i-1}, r_{i+1}, \ldots, r_I)$ denote the strategies of all the users except user i. A strategy profile can be written as $\boldsymbol{r} = (r_i, \boldsymbol{r}_{-i})$.

DEFINITION 7. (Better and best response updates) A *better response update*, starting from some strategy $\boldsymbol{r} = (r_i, \boldsymbol{r}_{-i})$, is an event where a single user i changes to another strategy, $r_i' \in \mathcal{R}_i$, and increases her payoff as a result, i.e., $U_i(r_i', \boldsymbol{r}_{-i}) > U_i(r_i, \boldsymbol{r}_{-i})$.

A *best response update* is a special type of better response update, where the newly selected strategy r_i' maximizes user i's payoff among user i's all possible better response updates.

DEFINITION 8. (Pure Nash equilibrium) A *pure Nash equilibrium (NE)* is a strategy profile $\boldsymbol{r}^*$, where no user can perform a better response update unilaterally.

DEFINITION 9. (Finite improvement property) A game possesses the *finite improvement property* (FIP) when asynchronous[3] better response updates always converge to a pure NE within a finite number of steps, irrespective of the initial strategy profile or the users' updating order.

Existence of the finite improvement property implies that better response updating always leads to pure Nash equilibria, which implies the existence of pure Nash equilibria.[4]

THEOREM 1. *Every task selection game possesses the finite improvement property.*

The proof of Theorem 1 is given in Appendix A. We then proceed to study how long it takes for a strategy profile to converge to a pure NE. Theorem 2 ensures that each best response update can be computed in polynomial time.

THEOREM 2. *A best response update can be computed in* $\mathcal{O}(K^3T^3)$ *time.*

The proof of Theorem 2 is given in Appendix B. The proof is constructive, as it allows us to compute best response updates in polynomial time. Simulations suggest that the number of best response updates required to reach a pure Nash equilibrium grows linearly with the number of users in a wide variety of scenarios, although a theoretical proof is quite challenging to obtain.

3.4 Asynchronous and Distributed Task Selection Algorithm

Theorems 1 and 2 guarantee the convergence of our asynchronous and distributed task selection (ADTS) algorithm (Algorithm 1). To initialize the algorithm, a user inputs her private information on the movement cost (line 2), checks the task descriptions (line 3), and then computes her move-

[3]Asynchronous updates imply that there will be no two users updating their strategies at the same time.

[4]Note that the time steps involved in understanding the convergence of the best response updates is not the same time slot that we introduced in Section 2.

Algorithm 1 *Asynchronous and distributed task selection (ADTS) algorithm for user $i \in \mathcal{I}$.*

1: Initialization
2: User Input: Movement cost $c_i^{l,l'}\ \forall l, l' \in \mathcal{L}$.
3: Check the task description, location $l[k]$, time $t[k]$, and reward $\rho[k]$ for all task $k \in \mathcal{K}$ on the mobile app interface.
4: Calculate movement time $\Delta_i^{l,l'}, \forall l, l' \in \mathcal{L}$ using Google Maps data based on movement speed ν_i.
5: Planning Phase: Task Selection Game
6: **repeat**
7: Check clock timer τ on the mobile app.
8: **if** $\tau \in \Gamma_i$
9: Check the mobile app for the number of participants $q^{(k,t)}$ for all $(k,t) \in \mathcal{H}$.
10: Calculate the number of participants excluding user i itself: $q_{-i}^{(k,t)} := q^{(k,t)} - q_i^{(k,t)}, \forall (k,t) \in \mathcal{H}$.
11: Perform a best response update: Find a route $r_i \in \mathcal{R}_i$ that maximizes user i's payoff:

$$U_i(r_i, \boldsymbol{r}_{-i}) := \left(\sum_{(k,t) \in \mathbb{V}(r_i)} \frac{\rho^*[k,t]}{q_{-i}^{(k,t)} + 1} \right) - \sum_{e \in \mathbb{E}(r_i)} g_i(e).$$

12: Update the task selection decision:

$$q_i^{(k,t)} := \begin{cases} 1, & \text{if } (k,t) \in \mathbb{V}(r_i), \\ 0, & \text{otherwise,} \end{cases} \quad \forall (k,t) \in \mathcal{H}. \tag{4}$$

13: Report $\boldsymbol{q}_i := (q_i^{(k,t)}, \forall (k,t) \in \mathcal{H})$ to the service provider.
14: **end if**
15: **until** $\tau \geq \tau^{\max}$.
16: Data Collection and Sensing Phase
17: **for** each user $i \in \mathcal{I}$
18: Move and complete the sensing task in each time slot t based on the task selection plan r_i.
19: **end for**

ment time from Google Maps data[5] based on her speed[6] (line 4).

With the aggregate information[7] on the total number of users working on different tasks (line 9) provided by the service provider in Algorithm 2, each user performs a best response update (lines 11 and 12), and claims the tasks by sending her updated task selection $\boldsymbol{q}_i = (q_i^{(k,t)}, \forall (k,t) \in \mathcal{H})$ to the service provider. Here, $\mathcal{H} = \{(k,t) \in \mathcal{K} \times \mathcal{T} : \rho^*[k,t] > 0\}$ is the set of task-time points that provide a positive reward.

Notice that users only need to claim the tasks that they are interested in working on, and do not need to reveal their movement plans to the service provider. This will preserve the privacy of users. Let Γ_i be the set of iterations[8] during which user i updates her task selection strategy. We assume that each user updates her strategy distributively and asynchronously until a predefined iteration limit $\tau^{\max}$. We set $\tau^{\max}$ to be a large enough value such that the ADTS can converge. Simulations suggest it is enough to set $\tau^{\max} \geq 5$ when $I \leq 30$ and $K \leq 10$.

[5]Google Maps assume an average walking speed of about 3 miles/hour for pedestrians. For example, if the Google Maps show that movement time between locations l and l' is 2 mins, then user i with movement speed of 4 miles/hour would have a movement time $\Delta_i^{l,l'} = 1.5$ mins.

[6]Prototype systems, such as BreadCrumbs [12], can track the movement speed of the device's owner.

[7]Such information is often available for users in crowdsourcing platforms.

[8]Note that an iteration only takes a fraction of a single time slot.

Algorithm 2 *Information Update Algorithm for the Service Provider.*

1: Initialization
2: Announce the task description, location $l[k]$, time $t[k]$, and reward $\rho[k]$ for all task $k \in \mathcal{K}$ on the mobile app interface.
3: Allocate memory for $q_i^{(k,t)}, \forall i \in \mathcal{I}, (k,t) \in \mathcal{H}$.
4: Initialize clock timer $\tau := 1$ on the mobile app interface.
5: Information Update for Task Selection Game in Algorithm 1
6: **repeat**
7: **if** task selection update message $\boldsymbol{q}_i$ is received from user i
8: Calculate the number of participants for all $(k,t) \in \mathcal{H}$: $q^{(k,t)} := \sum_{i \in \mathcal{I}} q_i^{(k,t)}, \forall (k,t) \in \mathcal{H}$.
9: Update $q^{(k,t)}, \forall (k,t) \in \mathcal{H}$ on the mobile app interface.
10: **end if**
11: Set $\tau := \tau + 1$.
12: **until** $\tau \geq \tau^{\max}$.

In the data collection and sensing phase of Algorithm 1, each user moves around and completes her claimed sensing tasks. The service provider would only pay a user, if the user has claimed the task in the planning phase, and has completed the task with an acceptable quality. As in Field Agent [3], for quality control, the service provider can employ techniques such as GPS marking (for location verification), time stamping (for time verification), and photo/video confirmation. Except the user input (line 2) and sensing execution (lines 16-19), all the other computation, communication, and information checking from maps can be done by the mobile app on behalf of the users.

4. CENTRALIZED TASK ALLOCATION

In this section, we consider the benchmark centralized task allocation (CTA) problem, where the service provider seeks to maximize the social surplus in the TSG. We prove that the CTA problem is NP-hard. Due to the high complexity of finding the optimal solution, we propose a heuristic greedy centralized algorithm, which turns out to have close-to-optimal performance in our simulations.

4.1 Centralized Task Allocation Problem

In the CTA benchmark problem, the service provider allocates tasks to the users in order to maximize the social surplus (i.e., users' total rewards minus total movement costs). The social surplus represents the maximum total profit that the service provider can generate, under the ideal case that all the users are under its direct control[9].

For user i's given strategy (task-time route choice) r_i in (1), we use $y_i^k(r_i) = 1$ to denote that user i works on task k under strategy r_i, and $y_i^k(r_i) = 0$ otherwise. That is,

$$y_i^k(r_i) = \begin{cases} 1, & \text{if } (k, t[k]) \in r_i, \\ 0, & \text{otherwise.} \end{cases} \tag{5}$$

Given the users' strategy profile $\boldsymbol{r}$, the social surplus is

$$\text{surplus}(\boldsymbol{r}) = \text{reward}(\boldsymbol{r}) - \text{cost}(\boldsymbol{r}), \tag{6}$$

where $\text{reward}(\boldsymbol{r})$ is the total rewards received by all users, and $\text{cost}(\boldsymbol{r})$ is the total movement costs of all users. Since

[9]In this case, each user will get a zero payoff, i.e., the payment from the service provider equals to the user's movement costs involved in finishing the tasks.

the reward $\rho[k]$ is equally shared among all the users working on task k, reward($\boldsymbol{r}$) equals the total reward of those tasks that have at least one assigned user. So we have

$$\text{reward}(\boldsymbol{r}) = \sum_{k\in\mathcal{K}} \mathbf{1}_{\{\sum_{i\in\mathcal{I}} y_i^k(r_i)\geq 1\}}\rho[k], \tag{7}$$

where $\mathbf{1}_{\{.\}}$ is the indicator function. Moreover, we have

$$\text{cost}(\boldsymbol{r}) = \sum_{i\in\mathcal{I}} \text{cost}_i(r_i) = \sum_{i\in\mathcal{I}}\sum_{a=1}^{n-1} c_i^{l[k_i^a],l[k_i^{a+1}]}, \tag{8}$$

where $\text{cost}_i(r_i) = \sum_{a=1}^{n-1} c_i^{l[k_i^a],l[k_i^{a+1}]}$ is the movement cost of user i.

In practice, it is reasonable to assume that the total movement cost of a user is non-decreasing in the number of locations she visited. That is,

$$c_i^{l,l'} + c_i^{l',l''} \geq c_i^{l,l''}, \forall\, i\in\mathcal{I}, l, l', l''\in\mathcal{L}. \tag{9}$$

With this assumption, we can show that there always exists an optimal task allocation where each task is allocated to at most one user.

LEMMA 1. *There always exists a social surplus maximizing (optimal) task assignment* $\boldsymbol{r}^* = \arg\max_{\boldsymbol{r}\in\mathcal{R}_1\times\ldots\times\mathcal{R}_I} \textit{surplus}(\boldsymbol{r})$ *such that* $\sum_{i\in\mathcal{I}} y_i^k(r_i^*) \leq 1, \forall\, k\in\mathcal{K}$.

The proof of Lemma 1 is given in Appendix C. With this lemma[10], we can establish the NP-hardness of the CTA benchmark.

THEOREM 3. *The problem of finding the social surplus maximization solution of the TSG is NP-hard.*

The proof of Theorem 3 is given in Appendix D. Theorem 3 motivates us to consider a greedy heuristic algorithm to solve the CTA problem.

4.2 Greedy Centralized Task Allocation Algorithm

We propose a low complexity greedy centralized (GC) algorithm (Algorithm 3), which computes an approximate solution to the social surplus maximization problem. The key idea is that the service provider sorts the tasks in ascending order of the execution time, and then allocates one user to each task sequentially in a greedy manner. Specifically, in line 6, the service provider first serves tasks with the earliest execution time. In line 9, the service provider sorts users in the order of an increasing movement costs (from the users' current locations to the location of the current task k). Line 12 indicates that a user i will be chosen if it is eligible to perform the task ($i : k\in\mathcal{K}_i$), can reach the task location on time ($t[k] - \phi_i \geq \Delta_i^{l_i,l[k]}$), and has the incentive to perform the task ($\rho[k] \geq c_i^{l_i,l[k]}$) (line 12). Here ϕ_i is the first time slot of user i after finishing any existing allocating task. It is possible that some task k may not be allocated to any users (i.e., $y_i^k = 0, \forall\, i\in\mathcal{I}$).

[10]Note that there can be other optimal solutions that do not satisfy Lemma 1. However, these solutions can be obtained readily from the optimal solution in Lemma 1 as we will discuss in Appendix C.1.

Algorithm 3 *Greedy Centralized (GC) Algorithm for Task Allocation.*

1: Initialization
2: Obtain movement costs, speeds, and initial locations from all the users: $c_i^{l,l'}, \forall\, l, l'\in\mathcal{L}, i\in\mathcal{I}$, ν_i and $l_i^{\text{init}}, \forall\, i\in\mathcal{I}$.
3: Initialize the next available time and current location of the users: $\phi_i := 1$ and $l_i := l_i^{\text{init}}, \forall\, i\in\mathcal{I}$.
4: Initialize the optimization variables $y_i^k := 0, \forall\, i\in\mathcal{I}, k\in\mathcal{K}$.
5: Greedy Centralized Task Allocation
6: Sort the tasks in set $\mathcal{K}$ in the ascending order of the execution time $t[k]$.
7: **for** $k = 1$ to K
8: Set $flag := 0$.
9: Sort the users in set $\mathcal{I}$ in the ascending order of $c_i^{l_i,l[k]}$.
10: Set $i := 1$.
11: **while** $i \leq I$ **and** $flag = 0$
12: **if** $k\in\mathcal{K}_i$ **and** $t[k] - \phi_i \geq \Delta_i^{l_i,l[k]}$ **and** $\rho[k] \geq c_i^{l_i,l[k]}$
13: Update user i's next available time $\phi_i := t[k]$ and current location $l_i := l[k]$.
14: Indicate the task allocation $y_i^k := 1$ and set $flag := 1$.
15: **end if**
16: Set $i := i + 1$.
17: **end while**
18: **end for**
19: **for** $i\in\mathcal{I}$
20: Inform user i the task allocation ($y_i^k, \forall\, k\in\mathcal{K}$).
21: **end for**
22: Data Collection and Sensing Phase
23: **for** each user $i\in\mathcal{I}$
24: Move and complete the sensing tasks based on the task allocation ($y_i^k, \forall\, k\in\mathcal{K}$).
25: **end for**

Table 1: Simulation Parameters

Parameters	Values
Number of tasks K	10
Number of time slots T	15
Rewards for three levels of tasks	\$10, \$15, \$20
Duration of a time slot δ	1 minute
Movement speed ν	0.1 km/min

LEMMA 2. *Algorithm 3 has a computational complexity of* $\mathcal{O}(KI\log I)$.

The proof of Lemma 2 is given in Appendix E.

5. PERFORMANCE EVALUATIONS

In this section, we evaluate the performance of our proposed ADTS scheme by comparing with the CTA benchmark, the GC scheme, and a greedy distributed (GD) scheme. We study the impact of various system parameters on the average user payoff, fairness, coverage, and average reward per measurement on different schemes. We also present an example using real movement time data from Google Maps. Our simulations involve the following schemes:

- ADTS scheme: Algorithms 1 and 2.
- CTA benchmark: Global optimal solution of the social surplus maximization problem.
- GC scheme: Algorithm 3 that approximately solves the social surplus maximization problem.
- GD scheme: Each user chooses to work on the earliest feasible task without coordinating with other users.

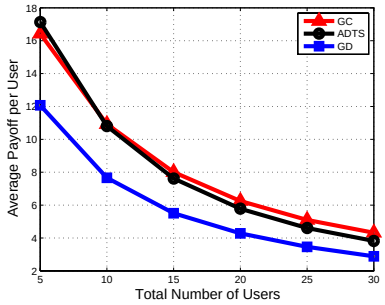

Figure 3: The average user payoff versus the total number of users I for $K = 10$ and $c^{\text{move}} = 0.1$.

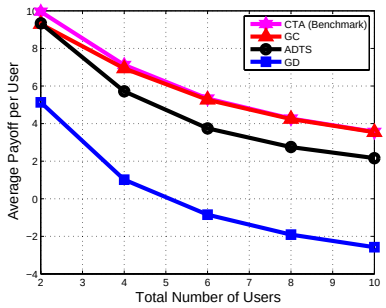

Figure 4: The average user payoff versus the total number of users I for $K = 5$ and $c^{\text{move}} = 1$.

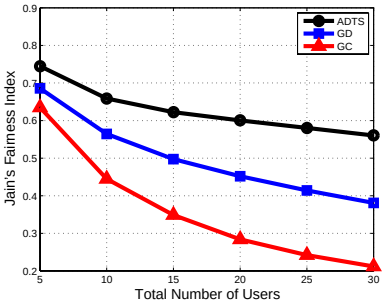

Figure 5: Jain's fairness index versus the total number of users I for $K = 10$ and $c^{\text{move}} = 0.1$.

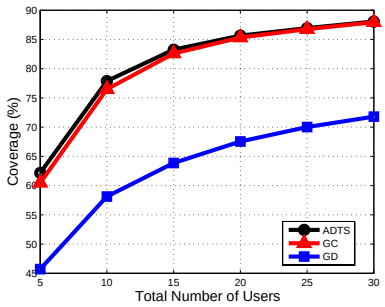

Figure 6: The coverage versus the total number of users I for $K = 10$ and $c^{\text{move}} = 0.1$.

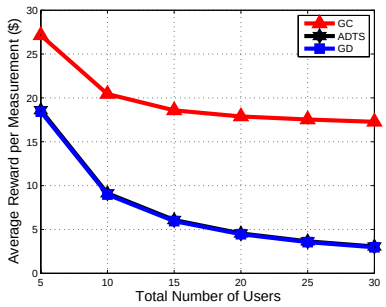

Figure 7: The average reward per measurement versus the total number of users I for $K = 10$ and $c^{\text{move}} = 0.1$.

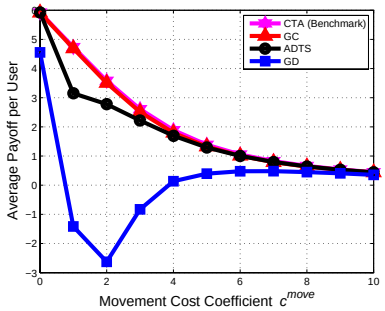

Figure 8: The average user payoff versus the movement cost coefficient c^{move} for $I = 7$ and $K = 5$.

Like the GC scheme, user i should be eligible for the task (i.e., $i : k \in \mathcal{K}_i$), can arrive on time to perform the task (i.e., when $t[k] - \phi_i \geq \Delta_i^{l_i, l[k]}$ given that user i is at location l_i), and has an incentive to perform sensing assuming no other users share the reward of that task (i.e., $\rho[k] \geq c_i^{l_i, l[k]}$). If there is more than one feasible task with the same execution time, then the user will choose the task with the highest reward.

For each set of system parameter choices, we run the simulations 1000 times with randomized initial user locations, and randomized locations and times for the tasks, and show the average value. Unless specified otherwise, we assume that the locations of the K tasks and I users are randomly placed in a 1 km $\times$ 1 km region. Each user is randomly allocated one of three reputation levels, and each task is randomly assigned a threshold. A user is eligible to perform a task when her reputation exceeds the threshold of the task. Each user i moves at constant speed ν_i, and the movement time from location $l \in \mathcal{L}$ to location $l' \in \mathcal{L}$ is $\Delta_i^{l,l'} = \left\lceil \frac{\text{dist}(l,l')}{\nu_i \delta} \right\rceil$, where $\text{dist}(l, l')$ is the traveling distance on the road between locations l and l', δ is the length of a time slot, and $\lceil \cdot \rceil$ is the ceiling function. We assume that the movement cost of user i is linearly proportional to the distance between two locations in the form $c_i^{l,l'} = c_i^{\text{move}} \text{dist}(l, l')$. For simplicity, we assume that all the users have the same movement cost coefficient $c_i^{\text{move}} = c^{\text{move}}$ and the same movement speed $\nu_i = \nu$. Other simulation parameters are listed in Table 1.

5.1 Average User Payoff and Fairness

Impact of user number on user payoff: In Fig. 3, we plot the average user payoff against the number of users I, with $K = 10$ tasks and movement cost coefficient $c^{\text{move}} = 0.1$. The average user payoff decreases with I under all three schemes, due to the increased competition with the increasing number of users. The ADTS scheme achieves a similar average user payoff as the GC scheme.

Comparison with centralized optimal solution: In Fig. 4, we plot the average user payoff against I in a smaller scale, with $K = 5$ and $c^{\text{move}} = 1$, so that we can evaluate the CTA benchmark as well. Note that GC achieves a similar payoff to the benchmark, especially when I is large.

Fairness: In Fig. 5, we study how the payoffs are distributed among the users by plotting Jain's fairness index [8] (defined as $\left(\sum_{i \in \mathcal{I}} U_i(\boldsymbol{r})\right)^2 / \left(I \sum_{i \in \mathcal{I}} U_i(\boldsymbol{r})^2\right)$) against I, with $K = 10$ and $c^{\text{move}} = 0.1$. The ADTS scheme has the highest fairness index, as each user has an equal chance to update her strategy profile in ADTS in Algorithm 1. On the other hand, the GC scheme has the lowest fairness index, as it allocates K tasks to at most K users. As a result, when I increases, the number of users not allocated any task increases, so the fairness decreases.

5.2 Coverage and Reward Per Measurement

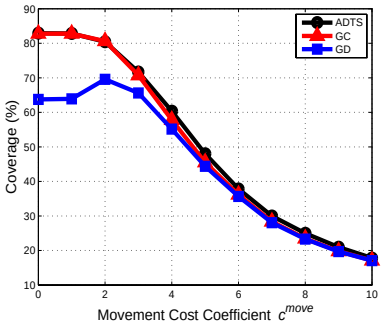

Figure 9: The coverage versus the movement cost coefficient c^{move} for $I = 10$ and $K = 10$.

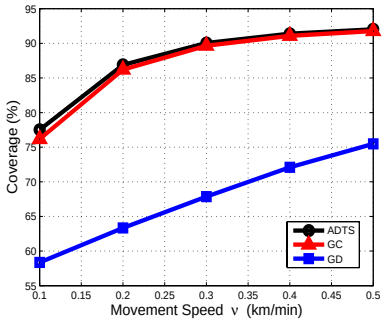

Figure 10: The coverage versus the movement speed ν for $I = 10$ and $K = 10$.

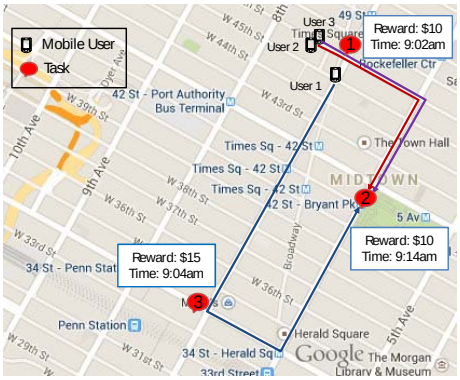

Figure 11: An illustration on the task selection and mobility plans of the users on the map under the ADTS scheme.

Table 2: Movement Time Data from Google Maps

Drive/Walk (min)	Location 1	Location 2	Location 3
Location 1	0 / 0	3 / 11	4 / 15
Location 2	3 / 11	0 / 0	2 / 6
Location 3	4 / 15	2 / 6	0 / 0

Table 3: Summary of Performance in the Real-world Example

	ADTS	GC and CTA benchmark	GD
Average user payoff	$10.33	$11	$7
Jain's fairness index	0.93	0.64	0.93
Coverage	100%	100%	66.67%

We examine the coverage and average reward per measurement under the three schemes, with $K = 10$ tasks and $c^{\text{move}} = 0.1$. The coverage is defined as the percentage of tasks with at least one measurement received.

Impact of user number on coverage: In Fig. 6, we plot the coverage against I. When I increases, we see the coverage under all the schemes increases, as there are more users to work on the tasks. The ADTS scheme has a slightly higher coverage than the GC scheme, as users try to avoid too much overlapping with each other in ADTS. The large difference between the coverage of the ADTS and GD schemes is due to the lack of coordination in the GD scheme.

Average reward per measurement: Fig. 7 shows the average reward per measurement against I. We can see that the ADTS and GD schemes have the same smaller average reward per measurement than the centralized scheme GC. From a service provider's point of view, with the same amount of total reward, lower average reward per measurement means that more sensing data can be collected, which is beneficial for the service provider.

5.3 Impact of Moving Cost and Speed

Impact of movement cost on user payoff: In Fig. 8, we plot the average user payoff against c^{move} with $I = 7$ and $K = 5$. As c^{move} increases, the payoffs for collecting measurements under both ADTS and GC decrease. Moreover, since users under both ADTS and GC are aware of the actual reward that they can receive (after sharing with other users), the users would not receive a negative payoff for collecting data. However, for the GD scheme without any user coordination, a user makes decisions with the belief that she will gain the full reward $\rho[k]$ for task k, rather than the actual reward she can achieve (after sharing with other users). When $c^{\text{move}} \leq 4$, users become overly aggressive in GD, and it becomes common that multiple users work on the same task. As a result, the actual reward per user is much less than the full reward of the task, which results in the decrease of the average payoff (even to a negative value) in GD. However, when c^{move} increases beyond 4, users are less likely to work on the same task in GD, so the evaluation of the reward is more accurate, and the average user payoff in GD gradually converges to a non-negative value, similar to those under the ADTS and GC schemes.

Impact of movement cost on coverage: In Fig. 9, we plot the coverage against the movement cost coefficient c^{move} with $I = 10$ and $K = 10$. As c^{move} increases, the coverage decreases in general, because the users are less willing to perform sensing. The slight increase in coverage in GD for $c^{\text{move}} \leq 2$ is due to the inaccurate reward estimation mentioned in the previous paragraph.

Impact of movement speed on coverage: In Fig. 10, we plot the coverage against the movement speed ν for $I = 10$ and $K = 10$. As ν increases, the coverages under all three schemes increases. The ADTS scheme has a slightly larger coverage than the GC scheme.

5.4 A Real-World Example Using Google Maps

To better illustrate the schemes, we next introduce a more concrete example based on the neighborhood map in New York City with real data from Google Maps. We consider $I = 3$ users and $K = 3$ tasks. The locations of the users, tasks, the execution times, and rewards are shown in Fig. 11. We assume that user 1 prefers driving, while users 2 and 3 prefer walking. We compute the movement time of driving and walking between different locations through Google Maps, as shown in Table 2. For the movement cost, we assume that driving between any two different locations costs $2 and walking is free.

Fig. 11 shows the task selection plans of three users (i.e., the blue, red, and purple broken lines with arrows) under the ADTS scheme. As we can see, user 1 takes the task selection plan of task 3 $\rightarrow$ task 2, while users 2 and 3 take the task selection plan of task 1 $\rightarrow$ task 2. The intuition is that as users 2 and 3 are pedestrians, who do not have enough time to go to the far away location 3 from their initial locations. As a result, user 1, who is driving, prefers to take task 3 with the minimum competition, and enjoys a higher reward. After finishing their first chosen task, all three users have enough time to work on task 2, and share the reward of $10. In contrast, under both the GC scheme and CTA benchmark (not shown in the figure), user 1 works on task 3, user 2 works on both tasks 1 and 2, and user 3 does not have any task to work on. For the GD scheme (not shown in the figure), since the initial locations of all the users are nearby, their task selection plans are identical: task 1 $\rightarrow$ task 2, as everyone chooses to pursue the earliest task without any coordination. The performance of the three schemes are summarized in Table 3. We can see that the ADTS scheme achieves a high level of fairness, coverage, and user payoff.

6. CONCLUSION

Motivated by commercial mobile crowdsensing applications, such as Gigwalk and Field Agent, we studied the distributed time-sensitive and location-dependent task selection problem in this paper. We examined the problem from the perspectives of both noncooperative game and centralized optimization. We proposed an asynchronous and distributed task selection (ADTS) algorithm to help users determine their task selection and mobility plans in a distributed fashion, based on limited information on users' aggregate task choices. We proved that finding the social surplus maximization solution is NP-hard. Simulation results showed that the ADTS scheme achieves the highest Jain's fairness index and coverage as compared with two heuristic schemes. In this paper, we focus on the distributed task selections of a fixed set of users. In the future work, we will consider the case a changing user population, where users can dynamically join and leave the system.

7. ACKNOWLEDGMENT

This work is supported by University of Macau Grant MYRG2014-00140-FST. This work is also supported by the General Research Funds (Project Number CUHK 412713 and 14202814) established under the University Grant Committee of the Hong Kong Special Administrative Region, China.

APPENDIX

A. PROOF OF THEOREM 1

The key idea is to show that the mapping

$$\Phi(\boldsymbol{r}) = \left(\sum_{(k,t)\in\mathcal{K}\times\mathcal{T}} \sum_{q=1}^{m^{(k,t)}(\boldsymbol{r})} \frac{\rho^*[k,t]}{q} \right) - \sum_{i\in\mathcal{I}} \sum_{e\in\mathbb{E}(r_i)} g_i(e)$$

is a *potential function* [11] of the TSG (where $\mathbb{E}(r_i)$ is defined in Definition 6). This means that for each strategy profile $\boldsymbol{r} = (r_1, \ldots, r_I) \in \mathcal{R}_1 \times \ldots \times \mathcal{R}_I$, each user $i \in \mathcal{I}$, and each strategy $r_i' \in \mathcal{R}_i$ available to user i, we have $\Phi(r_i', \boldsymbol{r}_{-i}) - \Phi(\boldsymbol{r}) = U_i(r_i', \boldsymbol{r}_{-i}) - U_i(\boldsymbol{r})$. In [11], the authors prove that every finite game with a potential function has the FIP. Since the TSG is a finite game, we prove the statement in Theorem 1.

To show that $\Phi(\boldsymbol{r})$ is a potential function, we shall separate our sums into two parts.

$$\Phi^V(\boldsymbol{r}) = \sum_{(k,t)\in\mathcal{K}\times\mathcal{T}} \sum_{q=1}^{m^{(k,t)}(\boldsymbol{r})} \frac{\rho^*[k,t]}{q} \tag{10}$$

and

$$\Phi^E(\boldsymbol{r}) = \sum_{i\in\mathcal{I}} \sum_{e\in\mathbb{E}(r_i)} -g_i(e). \tag{11}$$

Adding equations (10) and (11) gives

$$\Phi(\boldsymbol{r}) = \Phi^V(\boldsymbol{r}) + \Phi^E(\boldsymbol{r}). \tag{12}$$

Also, for a subset $S \subseteq \mathcal{K}\times\mathcal{T}$ of task-time points, let us define

$$\Phi_S^V(\boldsymbol{r}) = \sum_{(k,t)\in S} \sum_{q=1}^{m^{(k,t)}(\boldsymbol{r})} \frac{\rho^*[k,t]}{q}. \tag{13}$$

Clearly $\Phi_{\mathcal{K}\times\mathcal{T}}^V(\boldsymbol{r}) = \Phi^V(\boldsymbol{r})$. Let us define

$$U_i^V(\boldsymbol{r}) = \sum_{(k,t)\in\mathbb{V}(r_i)} \frac{\rho^*[k,t]}{m^{(k,t)}(\boldsymbol{r})} \tag{14}$$

and

$$U_i^E(\boldsymbol{r}) = \sum_{e\in\mathbb{E}(r_i)} -g_i(e). \tag{15}$$

By comparing this with equation (11) one can see that

$$\Phi^E(\boldsymbol{r}) = \sum_{i\in\mathcal{I}} U_i^E(\boldsymbol{r}). \tag{16}$$

Adding equations (14) and (15) gives

$$U_i(\boldsymbol{r}) = U_i^V(\boldsymbol{r}) + U_i^E(\boldsymbol{r}). \tag{17}$$

Suppose that our game is in strategy profile $\boldsymbol{r} = (r_1, .., r_I)$ and then user $j \in \mathcal{I}$ changes their strategy to $r_j' \in \mathcal{R}_j$, and so we have a new strategy profile $\boldsymbol{r}' = (r_j', \boldsymbol{r}_{-j})$. In order to show that Φ is an exact potential function we will show

$$\Phi(r_j', \boldsymbol{r}_{-j}) - \Phi(\boldsymbol{r}) = U_j(r_j', \boldsymbol{r}_{-j}) - U_j(\boldsymbol{r}). \tag{18}$$

To begin, note that we can expand out the left hand side of equation (18) using equation (12) to obtain:

$$\begin{aligned} &\Phi(r_j', \boldsymbol{r}_{-j}) - \Phi(\boldsymbol{r}) \\ &= \left[\Phi^V(r_j', \boldsymbol{r}_{-j}) - \Phi^V(\boldsymbol{r})\right] + \left[\Phi^E(r_j', \boldsymbol{r}_{-j}) - \Phi^E(\boldsymbol{r})\right]. \end{aligned} \tag{19}$$

Equation (16) implies that

$$\begin{aligned} \Phi^E(r_j', \boldsymbol{r}_{-j}) - \Phi^E(\boldsymbol{r}) &= \sum_{i\in\mathcal{I}} \left[U_i^E(r_j', \boldsymbol{r}_{-j}) - U_i^E(\boldsymbol{r})\right] \\ &= U_j^E(r_j', \boldsymbol{r}_{-j}) - U_j^E(\boldsymbol{r}). \end{aligned} \tag{20}$$

Using equation (13), we can obtain that

$$\begin{aligned}
&\Phi^V\left(r'_j,\boldsymbol{r}_{-j}\right)-\Phi^V\left(\boldsymbol{r}\right)\\
=&\left[\Phi^V_{\mathcal{K}\times\mathcal{T}-\left(\mathbb{V}\left(r_j\right)\cup\mathbb{V}\left(r'_j\right)\right)}\left(r'_j,\boldsymbol{r}_{-j}\right)-\Phi^V_{\mathcal{K}\times\mathcal{T}-\left(\mathbb{V}\left(r_j\right)\cup\mathbb{V}\left(r'_j\right)\right)}\left(\boldsymbol{r}\right)\right]\\
&+\left[\Phi^V_{\mathbb{V}\left(r_j\right)\cap\mathbb{V}\left(r'_j\right)}\left(r'_j,\boldsymbol{r}_{-j}\right)-\Phi^V_{\mathbb{V}\left(r_j\right)\cap\mathbb{V}\left(r'_j\right)}\left(\boldsymbol{r}\right)\right]\\
&+\left[\Phi^V_{\mathbb{V}\left(r'_j\right)-\mathbb{V}\left(r_j\right)}\left(r'_j,\boldsymbol{r}_{-j}\right)-\Phi^V_{\mathbb{V}\left(r'_j\right)-\mathbb{V}\left(r_j\right)}\left(\boldsymbol{r}\right)\right]\\
&+\left[\Phi^V_{\mathbb{V}\left(r_j\right)-\mathbb{V}\left(r'_j\right)}\left(r'_j,\boldsymbol{r}_{-j}\right)-\Phi^V_{\mathbb{V}\left(r_j\right)-\mathbb{V}\left(r'_j\right)}\left(\boldsymbol{r}\right)\right].
\end{aligned}\tag{21}$$

Now since $m^{(k,t)}\left(\boldsymbol{r}\right)=m^{(k,t)}\left(r'_j,\boldsymbol{r}_{-j}\right)$ holds true for every $(k,t)\in\mathcal{K}\times\mathcal{T}-\left(\mathbb{V}\left(r_j\right)\cup\mathbb{V}\left(r'_j\right)\right)$ or $(k,t)\in\mathbb{V}\left(r_j\right)\cap\mathbb{V}\left(r'_j\right)$ we have that

$$\begin{aligned}
&\left[\Phi^V_{\mathcal{K}\times\mathcal{T}-\left(\mathbb{V}\left(r_j\right)\cup\mathbb{V}\left(r'_j\right)\right)}\left(r'_j,\boldsymbol{r}_{-j}\right)-\Phi^V_{\mathcal{K}\times\mathcal{T}-\left(\mathbb{V}\left(r_j\right)\cup\mathbb{V}\left(r'_j\right)\right)}\left(\boldsymbol{r}\right)\right]\\
&+\left[\Phi^V_{\mathbb{V}\left(r_j\right)\cap\mathbb{V}\left(r'_j\right)}\left(r'_j,\boldsymbol{r}_{-j}\right)-\Phi^V_{\mathbb{V}\left(r_j\right)\cap\mathbb{V}\left(r'_j\right)}\left(\boldsymbol{r}\right)\right]=0,
\end{aligned}$$

and so equation (21) can be simplified to give

$$\begin{aligned}
&\Phi^V\left(r'_j,\boldsymbol{r}_{-j}\right)-\Phi^V\left(\boldsymbol{r}\right)\\
=&\left[\Phi^V_{\mathbb{V}\left(r'_j\right)-\mathbb{V}\left(r_j\right)}\left(r'_j,\boldsymbol{r}_{-j}\right)-\Phi^V_{\mathbb{V}\left(r'_j\right)-\mathbb{V}\left(r_j\right)}\left(\boldsymbol{r}\right)\right]\\
&+\left[\Phi^V_{\mathbb{V}\left(r_j\right)-\mathbb{V}\left(r'_j\right)}\left(r'_j,\boldsymbol{r}_{-j}\right)-\Phi^V_{\mathbb{V}\left(r_j\right)-\mathbb{V}\left(r'_j\right)}\left(\boldsymbol{r}\right)\right].
\end{aligned}\tag{22}$$

Using equation (13), together with the fact that $m^{(k,t)}\left(r'_j,\boldsymbol{r}_{-j}\right)=m^{(k,t)}\left(\boldsymbol{r}\right)+1$, $\forall(k,t)\in\mathbb{V}\left(r'_j\right)-\mathbb{V}\left(r_j\right)$ gives us that

$$\begin{aligned}
&\Phi^V_{\mathbb{V}\left(r'_j\right)-\mathbb{V}\left(r_j\right)}\left(r'_j,\boldsymbol{r}_{-j}\right)-\Phi^V_{\mathbb{V}\left(r'_j\right)-\mathbb{V}\left(r_j\right)}\left(\boldsymbol{r}\right)\\
=&\sum_{(k,t)\in\mathbb{V}\left(r'_j\right)-\mathbb{V}\left(r_j\right)}\left[\left[\sum_{q=1}^{m^{(k,t)}\left(r'_j,\boldsymbol{r}_{-j}\right)}\frac{\rho^*\left[k,t\right]}{q}\right]-\left[\sum_{q=1}^{m^{(k,t)}\left(\boldsymbol{r}\right)}\frac{\rho^*\left[k,t\right]}{q}\right]\right]\\
=&\sum_{(k,t)\in\mathbb{V}\left(r'_j\right)-\mathbb{V}\left(r_j\right)}\frac{\rho^*\left[k,t\right]}{m^{(k,t)}\left(\boldsymbol{r}\right)+1}.
\end{aligned}\tag{23}$$

Similarly, using equation (13), together with the fact that $m^{(k,t)}\left(r'_j,\boldsymbol{r}_{-j}\right)=m^{(k,t)}\left(\boldsymbol{r}\right)-1$, $\forall(k,t)\in\mathbb{V}\left(r_j\right)-\mathbb{V}\left(r'_j\right)$ gives us that

$$\begin{aligned}
&\Phi^V_{\mathbb{V}\left(r'_j\right)-\mathbb{V}\left(r_j\right)}\left(r'_j,\boldsymbol{r}_{-j}\right)-\Phi^V_{\mathbb{V}\left(r'_j\right)-\mathbb{V}\left(r_j\right)}\left(\boldsymbol{r}\right)\\
=&\sum_{(k,t)\in\mathbb{V}\left(r_j\right)-\mathbb{V}\left(r'_j\right)}\left[\left[\sum_{q=1}^{m^{(k,t)}\left(r'_j,\boldsymbol{r}_{-j}\right)}\frac{\rho^*\left[k,t\right]}{q}\right]-\left[\sum_{q=1}^{m^{(k,t)}\left(\boldsymbol{r}\right)}\frac{\rho^*\left[k,t\right]}{q}\right]\right]\\
=&\sum_{(k,t)\in\mathbb{V}\left(r_j\right)-\mathbb{V}\left(r'_j\right)}-\frac{\rho^*\left[k,t\right]}{m^{(k,t)}\left(\boldsymbol{r}\right)}.
\end{aligned}\tag{24}$$

Substituting equations (23) and (24) into equation (22) gives us that

$$\begin{aligned}
&\Phi^V\left(r'_j,\boldsymbol{r}_{-j}\right)-\Phi^V\left(\boldsymbol{r}\right)\\
=&\left[\sum_{(k,t)\in\mathbb{V}\left(r'_j\right)-\mathbb{V}\left(r_j\right)}\frac{\rho^*\left[k,t\right]}{m^{(k,t)}\left(\boldsymbol{r}\right)+1}\right]-\sum_{(k,t)\in\mathbb{V}\left(r_j\right)-\mathbb{V}\left(r'_j\right)}\frac{\rho^*\left[k,t\right]}{m^{(k,t)}\left(\boldsymbol{r}\right)}\\
=&U_j^V\left(r'_j,\boldsymbol{r}_{-j}\right)-U_j^V\left(\boldsymbol{r}\right).
\end{aligned}\tag{25}$$

Substituting equations (20) and (25) into equation (19), gives us that

$$\begin{aligned}
&\Phi\left(r'_j,\boldsymbol{r}_{-j}\right)-\Phi\left(\boldsymbol{r}\right)\\
=&\left[U_j^V\left(r'_j,\boldsymbol{r}_{-j}\right)-U_j^V\left(\boldsymbol{r}\right)\right]+\left[U_j^E\left(r'_j,\boldsymbol{r}_{-j}\right)-U_j^E\left(\boldsymbol{r}\right)\right].
\end{aligned}\tag{26}$$

Now equation (17) can be used to simplify the right hand side of equation (26), and this yields the right hand side of equation (18), as required. ■

B. PROOF OF THEOREM 2

The key proof idea is to show that the problem of a generic user j finding a best response strategy in a strategy profile $\boldsymbol{r}$ can be reformulated as the problem of finding a longest path in a weighted directed acyclic graph $\mathcal{G}^*$, which can be solved in polynomial time.

To construct such a graph $\mathcal{G}^*$, we construct a weighted directed graph $\mathcal{G}=(\mathcal{V},\mathcal{E})$. Recall that $\mathbb{V}$ is defined in Definition 4 and $\mathbb{E}$ is defined in Definition 6. The vertex set $\mathcal{V}=\bigcup_{s\in\mathcal{R}_j}\mathbb{V}\left(s\right)$ corresponds to the set of task-time points (k,t), which lie on routes which are feasible for user j. The directed edge set $\mathcal{E}=\bigcup_{s\in\mathcal{R}_j}\mathbb{E}\left(s\right)$ corresponds to the set of all pairs $\left[\left(k,t\right),\left(k',t'\right)\right]\in\mathbb{E}\left(s\right)$ such that user j can move between. Each directed edge $e=\left[\left(k,t\right),\left(k',t'\right)\right]$ of the graph $\mathcal{G}$ is associated with an edge weight $w\left(e\right)=-g_j(e)=-c_j^{l[k],l[k']}$. Each vertex (k,t) is associated with a value

$$\theta\left(\left(k,t\right)\right)=\frac{\rho^*\left[k,t\right]}{\left|\left\{i\in\mathcal{I}:i\neq j,\left(k,t\right)\in\mathbb{V}\left(r_i\right)\right\}\right|+1},\tag{27}$$

where $\left|\left\{i\in\mathcal{I}:i\neq j,\left(k,t\right)\in\mathbb{V}\left(r_i\right)\right\}\right|$ equals the number of users other than j which have selected task-time routes that pass through (k,t) in the strategy profile $\boldsymbol{r}$. Notice that $\theta\left(\left(k,t\right)\right)$ equals the reward that user j would get for visiting task-time point (k,t).

A path in a directed graph is a sequence of vertices, where each vertex is linked to its successor in the sequence. A best response for user j corresponds to a path in the graph $\mathcal{G}$, which starts at the task-time point $(k_j^{\text{init}},1)$, and maximizes the sum of the weights of the edges, and the values of the vertices along the path.

Starting from graph $\mathcal{G}$ defined above, we form a new graph $\mathcal{G}^*=(\mathcal{V}^*,\mathcal{E}^*)$, by replacing each vertex (k,t) in $\mathcal{G}$ with a pair of new vertices $(k,t,0)$ and $(k,t,1)$. These two new vertices are linked by a new directed edge $\left[\left(k,t,0\right),\left(k,t,1\right)\right]$, which has an edge weight $w^*\left(\left[\left(k,t,0\right),\left(k,t,1\right)\right]\right)=\theta((k,t))$ as in (27), equal to the value of the original vertex (k,t).

As a result, graph $\mathcal{G}^*$ has a vertex set

$$\mathcal{V}^*=\left\{\left(k,t,\gamma\right):\left(k,t\right)\in\mathcal{V},\gamma\in\left\{0,1\right\}\right\},$$

and an edge set $\mathcal{E}^* = \mathcal{E}_1^* \cup \mathcal{E}_2^*$. Here set

$$\mathcal{E}_1^* = \left\{ \left[(k,t,1),(k',t',0)\right] : \left[(k,t),(k',t')\right] \in \mathcal{E} \right\},$$

and each edge $[(k,t,1),(k',t',0)] \in \mathcal{E}_1^*$ has a weight

$$w^*\left(\left[(k,t,1),(k',t',0)\right]\right) = w\left(\left[(k,t),(k',t')\right]\right),$$

with the right hand side being the weight of the edge originally in graph $\mathcal{G}$. Also set $\mathcal{E}_2^* = \{[(k,t,0),(k,t,1)] : (k,t) \in \mathcal{V}\}$, and each edge $[(k,t,0),(k,t,1)] \in \mathcal{E}_2^*$ has a weight

$$w^*\left(\left[(k,t,1),(k',t',0)\right]\right) = \theta\left((k,t)\right),$$

where the right hand side follows (27).

By considering the weights of all edges in $\mathcal{G}^*$ as lengths, one can see that a best response for user j in strategy profile $\boldsymbol{r}$ corresponds to a longest path in $\mathcal{G}^*$ that starts at the vertex $(k_j^{init}, t, 0)$. Such a longest path can be computed in $\mathcal{O}\left(|\mathcal{V}^*| \times |\mathcal{E}^*|\right) = \mathcal{O}\left(K^3T^3\right)$ time using the algorithm described in [13], because $\mathcal{G}^*$ is a directed acyclic graph. Since $\mathcal{G}^*$ itself can be constructed in $\mathcal{O}\left(K^2T^2\right)$ time based on the previous discussions, the entire process for a user to compute the best response has a complexity of $\mathcal{O}\left(K^3T^3\right)$. ■

C. PROOF OF LEMMA 1

We prove the lemma by contradiction. Assume that for any optimal solution $\boldsymbol{r}^*$, we can always find a task $q \in \mathcal{K}$ such that $\sum_{i\in\mathcal{I}} y_i^q(r_i^*) > 1$. Let us focus on a particular user $j \in \mathcal{I}$ that is assigned to work on task q (i.e., $y_j^q(r_i^*) = 1$). We can define another strategy profile $\boldsymbol{r}$, which is the same as $\boldsymbol{r}^*$ for the task allocation of all users, except that task q is not allocated to user j. That is,

$$y_i^k(r_i) = \begin{cases} 0, & \text{if } k = q \text{ and } i = j, \\ y_i^k(r_i^*), & \text{otherwise.} \end{cases} \tag{28}$$

Since the coverage of the tasks under $\boldsymbol{r}$ is the same as $\boldsymbol{r}^*$, where $\sum_{i\in\mathcal{I}} y_i^k(r_i) \geq 1$ and $\sum_{i\in\mathcal{I}} y_i^k(r_i^*) \geq 1$ for each task $k \in \mathcal{K}$, we have from (7) that

$$\text{reward}(\boldsymbol{r}) = \text{reward}(\boldsymbol{r}^*). \tag{29}$$

Since user j works on one task less in r_j than in r_j^*, from (9), we have

$$\text{cost}_j(r_j) \leq \text{cost}_j(r_j^*). \tag{30}$$

However, the movement costs of other users remain the same such that

$$\text{cost}_i(r_i) = \text{cost}_i(r_i^*),\ \forall\, i \in \mathcal{I}\backslash\{j\}. \tag{31}$$

Overall, substituting (29), (30), and (31) into (6), we have

$$\text{surplus}(\boldsymbol{r}^*) \leq \text{surplus}(\boldsymbol{r}), \tag{32}$$

which means that strategy profile $\boldsymbol{r}$ is also an optimal solution. This leads to a contradiction, and hence proves the theorem. ■

C.1 Discussion of multiple solutions of the social surplus maximization problem

We note that there can be other optimal solutions that do not satisfy Lemma 1. However, these solutions can be obtained from the optimal solution in Lemma 1, when there are no *extra* movement costs for users to work on some tasks, so that there can be more than one user working on a particular task in the optimal solution.

As an example, consider the case with two users (i.e., $I = 2$) and three tasks (i.e., $K = 3$). For the execution time of the tasks, we assume that $t[1] = 2$, $t[2] = 3$, and $t[3] = 4$. For the locations of tasks, we assume that the three tasks are located on a straight line, where the movement cost of user 1 satisfy the condition

$$c_1^{1,2} + c_1^{2,3} = c_1^{1,3}. \tag{33}$$

That is, the location $l[2]$ of task 2 is between the locations $l[1]$ and $l[3]$ of tasks 1 and 3.

Assuming that the strategy profile with

$$r_1 = \left((k_1^{\text{init}}, 1), (1,2), (3,4)\right)$$

and

$$r_2 = \left((k_2^{\text{init}}, 1), (2,2), (2,3), (2,4)\right)$$

is an optimal solution of the social surplus maximization problem, which satisfies the condition in Lemma 1. From this strategy profile, we can define another strategy profile

$$\bar{r}_1 = \left((k_1^{\text{init}}, 1), (1,2), (2,3), (3,4)\right)$$

and

$$\bar{r}_2 = \left((k_2^{\text{init}}, 1), (2,2), (2,3), (2,4)\right),$$

where both users work on task 2 at time 3. Notice that the social surplus achieved under this strategy profile $\bar{\boldsymbol{r}}$ is the same as the strategy profile $\boldsymbol{r}$, since user 1 does not need to incur any extra movement cost in completing task 2.

D. PROOF OF THEOREM 3

We prove the NP-hardness by *restriction* [5]: We show that finding the social surplus maximization solution in a special case of a TSG can be transformed into a 3-dimensional matching decision problem, which is NP-complete [5,9].

DEFINITION 10. (3-dimensional matching) Let $\mathcal{X}$, $\mathcal{Y}$, and $\mathcal{Z}$ be three finite disjoint sets. Let $\mathcal{R} \subseteq \mathcal{X} \times \mathcal{Y} \times \mathcal{Z}$ be a set of ordered triples, i.e., $\mathcal{R} = \{(x,y,z) : x \in \mathcal{X}, y \in \mathcal{Y}, z \in \mathcal{Z}\}$. $\mathcal{R}' \subseteq \mathcal{R}$ is a *3-dimensional matching* if for any two different triples $(x_1,y_1,z_1) \in \mathcal{R}'$ and $(x_2,y_2,z_2) \in \mathcal{R}'$, we have $x_1 \neq x_2$, $y_1 \neq y_2$, and $z_1 \neq z_2$.

DEFINITION 11. (3-dimensional matching decision problem) Suppose $|\mathcal{X}| = |\mathcal{Y}| = |\mathcal{Z}| = M$. Given an input $\mathcal{R}$ with $|\mathcal{R}| \geq M$, decide whether there exists a 3-dimensional matching $\mathcal{R}' \subseteq \mathcal{R}$ with the maximum size $|\mathcal{R}'| = M$.

Consider a restricted TSG, which is a special case of a TSG, which we place the following restrictions:

(a) Tasks and time slots: There are $T = 3$ time slots, and there are M tasks in each time slot (hence a total of $3M$ tasks in 3 time slots). Sets $\mathcal{X}$, $\mathcal{Y}$, and $\mathcal{Z}$ represent the sets of available tasks in the three time slots, where $|\mathcal{X}| = |\mathcal{Y}| = |\mathcal{Z}| = M$.

(b) Task-time route: Set $\mathcal{R}$ represents the set of available task-time routes of all the users. Assume that each user i can only choose one particular available task-time route $r_i \in \mathcal{R}$. We assume that the number of users I is large enough, such that the task-time routes of all the users cover all the tasks in $\mathcal{X} \cup \mathcal{Y} \cup \mathcal{Z}$, so $|\mathcal{R}| = I \geq M$. $\mathcal{R}' \subseteq \mathcal{R}$ in Definition 11 represents a feasible task allocation. We assume that a user, whose route is not chosen in $\mathcal{R}'$, will not work on any real task.

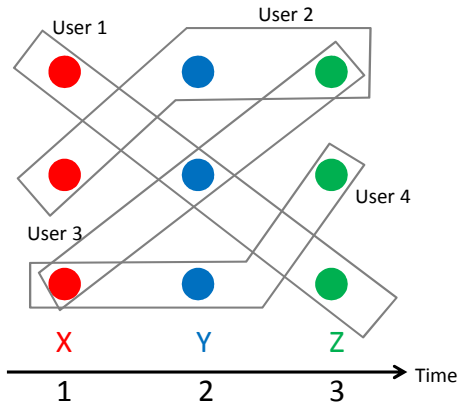

Figure 12: An example of 3-dimensional matching. Here, the set $\mathcal{R}$ consists of the four grey areas, which represent the routes of users 1 to 4. The solution of the 3-dimensional matching problem is the set $\mathcal{R}'$ that consists of the routes of users 1, 2, and 4. It is also the solution of the social surplus maximization problem.

(c) Large reward: The reward of a task is larger than the movement cost to work on the task, i.e., $\rho[k] > c_i^{l[k'],l[k]}$ for each user $i \in \mathcal{I}$ and all tasks $k, k' \in \mathcal{K}$.

(d) User-independent movement cost: The movement cost only depends on its destination location and is independent of its initial location, such that $c_i^{l,l'} = c[l']$ for all $i \in \mathcal{I}$ and $l, l' \in \mathcal{L}$, where $c[l']$ is the movement cost of destination $l' \in \mathcal{L}$.

In the restricted TSG, first, restrictions (c) and (d) imply that we can maximize the social surplus by covering *all* the tasks with *any* available users. Second, Lemma 1 (and the discussion in Appendix C.1) implies that we can focus on an optimal solution, where each task should be allocated to *at most one* user. So the optimal task allocation should not contain any overlapping components (i.e., multiple users working on the same task) as defined in Definition 10. Putting the above discussions together, we know that in the social surplus maximization solution, *every* element of $\mathcal{X} \times \mathcal{Y} \times \mathcal{Z}$ (i.e., every task) should be contained in *exactly one* of the triples (i.e., the task-time routes) in $\mathcal{R}'$. In other words, $\mathcal{R}' \subseteq \mathcal{R}$ is the optimal task allocation. So the surplus maximization problem can be transformed to a 3-dimensional matching decision problem, which is NP-complete [5, 9]. By restriction, we establish that the problem of finding the social surplus maximization solution of the TSG is NP-hard. ■

E. PROOF OF LEMMA 2

The sorting of execution time in line 6 takes $\mathcal{O}(K \log K)$ time [9]. For each task, besides some simple computations, most of the time is consumed for sorting the movement costs of the users in line 9, and it takes a running time of $\mathcal{O}(I \log I)$. Since we generally have more users than tasks, i.e., $I \log I > \log K$, the total run time with K tasks is $\mathcal{O}(KI \log I)$. ■

F. REFERENCES

[1] Amazon Mechanical Turk. https://www.mturk.com/mturk/.

[2] L. Duan, T. Kubo, K. Sugiyama, J. Huang, T. Hasegawa, and J. Walrand. Motivating smartphone collaboration in data acquisition and distributed computing. *IEEE Trans. on Mobile Computing*, 13(10):2320–2333, Oct. 2014.

[3] Field Agent. http://www.fieldagent.net/.

[4] R. K. Ganti, F. Ye, and H. Lei. Mobile crowdsensing: Current state and future challenges. *IEEE Communications Magazine*, 49(11):32–39, Nov. 2011.

[5] M. R. Garey and D. S. Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness*. Freeman, San Francisco, CA, first edition, 1979.

[6] Gigwalk. http://gigwalk.com/.

[7] S. He, D. Shin, J. Zhang, and J. Chen. Towards optimal allocation of location dependent tasks in crowdsensing. In *Proc. of IEEE INFOCOM*, Toronto, Canada, Apr. 2014.

[8] R. K. Jain, D. Chiu, and W. R. Hawe. A quantitative measure of fairness and discrimination for resource allocation in shared computer systems. Tech. Report DEC-TR-301, Eastern Research Lab, Sept. 1984.

[9] J. Kleinberg and E. Tardos. *Algorithm Design*. Addison-Wesley, Boston, MA, first edition, 2005.

[10] N. D. Lane, E. Miluzzo, H. Lu, D. Peebles, T. Choudhury, and A. T. Campbell. A survey of mobile phone sensing. *IEEE Communications Magazine*, 48(9):140–150, Sept. 2010.

[11] D. Monderer and L. S. Shapley. Potential games. *Games and Economic Behavior*, 14(1):124–143, May 1996.

[12] A. J. Nicholson and B. D. Noble. BreadCrumbs: Forecasting mobile connectivity. In *Proc. of ACM MobiCom*, San Francisco, CA, Sept. 2008.

[13] R. Sedgewick and K. Wayne. *Algorithms*. Pearson Education, 2011.

[14] X. Sheng, J. Tang, and W. Zhang. Energy-efficient collaborative sensing with mobile phones. In *Proc. of IEEE INFOCOM*, Orlando, FL, Mar. 2012.

[15] Q. Zhao, Y. Zhu, H. Zhu, J. Cao, G. Xue, and B. Li. Fair energy-efficient sensing task allocation in participatory sensing with smartphones. In *Proc. of IEEE INFOCOM*, Toronto, Canada, Apr. 2014.