Securing Task Allocation in Mobile Crowd Sensing: An Incentive Design Approach

Mingyan Xiao*, Ming Li*, Linke Guo†, Miao Pan‡, Zhu Han‡, Pan Li§
* The University of Texas at Arlington, Arlington, U.S.
† Binghamton University, Binghamton, U.S.
‡ University of Houston, U.S. & Kyung Hee University, South Korea
§ Case Western Reserve University, U.S.
Email: mingyan.xiao@mavs.uta.edu, ming.li@uta.edu, lguo@binghamton.edu, {mpan2,zhan2}@uh.edu, lipan@case.edu

***Abstract*—As a critical component of mobile crowd sensing (MCS), task allocation has been extensively investigated. In general, it addresses how to wisely distribute sensing tasks among sensing workers. Yet, the security threat involved therein has hardly been studied. In an ideal scenario, workers are trusted to report their accurate parameters to the platform, so that task allocation optimization problems can be correctly formulated and calculated. Nonetheless, malicious workers can explore illegal benefit gain by simply uploading falsified parameters. Even worse, such an attack is difficult to detect. In this paper, we start from a simplified case in which workers report erroneous objective functions to gain extra utility. To defend this attack, we novelly leverage incentive mechanism design. Workers are motivated to report desirable "indicators", based on which the platform can still obtain the accurate task allocation profile even without workers' genuine parameters. The effectiveness and efficiency of our mechanism is validated through both formal analysis and extensive simulation results.**

***Index Terms*—Mobile crowd sensing, task allocation, parameter manipulation attack, incentive design**

I. Introduction

Mobile crowd sensing (MCS) arises as a new sensing paradigm by exploring a plethora of embedded multi-modal sensors in today's ubiquitous mobile devices. By fusing and analyzing their sensing data, MCS has potential to accelerate the maturity of smart health caring, environment monitoring, traffic surveillance, social event observation, etc. An MCS system mainly consists of three types of entities, task requestors, sensing workers, and the platform. A typical MCS workflow can be divided into three stages: *task allocation*, *task sensing* and *data aggregation/analysis*.

For the stage of task allocation, its main objective is to distribute sensing tasks among workers such that sensing resources of the entire system are efficiently utilized. For this purpose, prevalent approaches are to formulate and solve the task allocation optimization problems at the platform, taking into account various optimization factors from all entities, e.g., [1]–[3]. Typically, workers are required to explicitly specify their parameters, including task preferences, computation capacities, affordable travel distances, cost functions. The accuracy of task allocation thus directly relies on the quality of these reported parameters. As pointed out later in this paper, malicious workers can easily manipulate task allocation of MCS by simply uploading falsified parameters. In this work, we name this type of attack as the *parameter manipulation attack*. Since these parameters are personal data owned by each worker, there is lack of evidence at the platform to decide their accuracy. As a result, parameter manipulation attacks will be difficult to detect.

While there have been some existing works tackling security issues in MCS, most of them focus on the stage of data analysis. Since MCS allows any voluntary participant to contribute data, it is vulnerable to erroneous or even malicious data injection. Great efforts have been devoted to the development of a so-called "reputation system" [4]–[9]; they initially establish reputation scores of workers based on the quality of their contributions, and later on use these scores to eliminate reports from less reputable workers.

Instead of tackling data trustworthy issues in the stage of data analysis as the above works [4]–[9], in this paper, for the first time, we target at the parameter manipulation attack in the stage of task allocation in MCS. To defend against it, *our design goal* is to enable the platform to find accurate and optimal solutions to task allocation formulations, even under the existence of malicious workers. As an initial work on this topic, we plan to start from a simplified case: workers report falsified objective function (parameters) while the rest parameters, i.e., the ones in constraints, are unaltered. Since the objective function is critical to task allocation and of complex form, it can easily become the adversary's target. However, even under this simplified case, the scheme design is not an easy task.

Since solving task allocation optimization problems alone is already complicated in general, it is infeasible to refer to computationally intensive cryptographic techniques for the defense scheme design. Instead, we leverage the *incentive mechanism*. Our scheme is built on top of an assumption that all workers are *rational* and *self-interest* in a sense that they launch an attack only to maximize their benefits achieved during task sensing. Since workers are skeptical to report genuine parameters for task allocation formulation, instead of collecting them, our scheme asks each worker to submit its tunable "indicator" of willingness for executing tasks. Then a new task allocation formulation, based on collected indicators,

978-1-5386-7117-7/19/$31.00 ©2019 IEEE

is constructed, which is slightly different from the original one. On the other hand, if the new problem's optimal solution is the same with that of the original one, the platform can still derive the accurate task allocation profile via the new problem. However, the challenge is how to make the optimal solutions of these two distinguishing problems identical. As pointed out later, it depends on indicator values chosen by workers. Incentive mechanism design is thus applied to elicit workers to offer "proper indicators" so as to achieve the goal.

We summarize the contributions of this work as follows.

- We address the parameter manipulation attack in the stage of task allocation in MCS, where workers report falsified parameters to for illegal beneficial gain. It is a critical security issue in MCS, yet receives rare attention so far.
- Instead of crypto primitives, we novelly apply the incentive mechanism in our defense scheme development. It guarantees that the platform can still find the accurate task allocation solution, even under the existence of malicious workers.
- We formally prove the security and convergence property of the proposed scheme. A real-world dataset is applied to evaluate the scheme performance.

II. Related Work

Security issues in MCS. Among the existing works the one that is closest to ours is [25]. It aims to thwart malicious behaviors in worker recruitment of crowdsourcing via the reverse game theory. However, it assumes that the worker's attack statistics are available at the platform. Besides, it formulates worker recruitment into a simple optimization problem, where no constraint is involved. For the other existing works that address security issues in MCS, such as [4], [5], [9], they mainly target at the data analysis stage. As sensing data are reported by workers, they can possibly be altered by malicious ones. This raises the issue of data trustworthiness. Effective schemes are proposed to either detect untrustworthy workers or avoid their reports during data analysis. The research focuses on how to evaluate trustworthiness of the shared data and how to maintain the reputation of various workers. Realizing that the collected data may include sensitive information regarding their reporters, such as locations, daily commute, behavior patterns and habits, the works [6]–[8] further guarantee data privacy and/or worker anonymity in their reputation systems design. As we aim to tackle the security issue in the stage of task allocation, the corresponding approach will be quite different.

Incentive mechanism design for MCS. To motivate mobile users to participate in MCS, the incentive mechanism is an effective approach [10]–[14]. Workers get paid by the platform to compensate their cost in task sensing. To model the interaction between the platform and workers as well as among workers themselves, auction theory and game theory are widely adopted. Different objectives are discussed, such as maximizing social welfare [13], maximizing the platform's profit [10], minimizing social cost [14], minimizing privacy loss [11], or minimizing the platform's payment [12]. Although these works also resort to incentive mechanisms, they have nothing to do with resisting parameter manipulation attacks.

III. Problem Statement

A. Background

The discussion of this work pertains to a standard MCS system, which mainly consists of a platform, a set of task requesters, and a set of participating sensing workers $\mathcal{U} = \{u_1, \cdots, u_i, \cdots, u_M\}$, who communicate with the platform via wireless connections, such as cellular networks or Wi-Fi. The platform hosts a set of sensing tasks $\mathcal{T} = \{\tau_1, \cdots, \tau_j, \cdots, \tau_N\}$ that are collected from their requesters. Conducting sensing tasks are resource-consuming for workers. Hence, to wisely utilize their resources, a task allocation framework is needed to coordinate among workers until task completion. Typically, optimization problems are formulated, taking into account of constraints from sensing capabilities and travel budget at workers, and sensing quality requirement from sensing tasks. Without loss of generality, in this work we consider a *cost minimization problem*[1]

$$
\begin{aligned}
\mathrm{P}_1: \quad \min_{\boldsymbol{x}_i} \quad & \sum_{i\in[1,M]} C_i(\boldsymbol{x_i}) \\
s.t. \quad & \sum_{j\in[1,N]} x_{ij} \le t_i, \quad \forall i \in [1,M] && (1)\\
& \sum_{i\in[1,M]} \theta_{ij} x_{ij} \ge T_j, \quad \forall j \in [1,N] && (2)\\
& \sum_{j\in[1,N]} \frac{d_{ij}}{w_{ij}(x_{ij})} \le D_i, \quad \forall i \in [1,M] && (3)\\
& x_{ij} \ge 0, \quad \forall i \in [1,M], \forall j \in [1,N]. && (4)
\end{aligned}
$$

The decision variable x_{ij} ($i \in [1,N]$, $j \in [1,M]$) stands for the sensing time that worker u_i is assigned for task τ_j and $\boldsymbol{x}_i$ is a vector $\boldsymbol{x}_i = \{x_{i1}, x_{i2}, \cdots, x_{iN}\}$. $C_i(\cdot)$ is u_i's component in the cost function and can be of different meanings. For example, in the case of minimizing the payments to workers [1], [2], it is u_i's pricing function; in the case of minimizing the overall travel distance [3], it stands for u_i's travel distance given a specific set of allocated tasks $\boldsymbol{x}_i$. In a word, $C_i(\cdot)$ is used to represent certain "burden" task sensing causes to worker u_i, and we aim to minimize the overall "burden" from all workers. Following the prevalent setting, we let $C_i(\cdot)$ be a convex function.

Denote by t_i the maximum sensing time that u_i can contribute. Constraint (1) states that each worker's total sensing time for all tasks cannot surpass this limit. For a task τ_j, we denote by T_j its minimum sensing time requirement. Constraint (2) says that the accumulated weighted sensing time from all workers regarding τ_j should not be less than

[1] Our scheme also works for the maximization case with mild modification.

T_j. Moreover, to carry out a task τ_j, u_i has to travel for a distance d_{ij} from its current location. Let D_i be u_i's maximal distance it is willing to travel. (3) says that each worker's total weighted travel distance should be no larger than D_i. $w_{ij}(x_{ij})$ is the weight and calculated by $w_{ij}(x_{ij}) = \frac{\theta_{ij}}{x_{ij}}$. A larger weight $w_{ij}(\cdot)$ implies that a worker w_i is more willing to conduct task τ_j since it can contribute with higher quality but cost shorter sensing time. The idea of (3) is taken from [15]. Note that the formulation of P_1 is not the contribution of this work. Once the platform formulates P_1, it can apply its favorite algorithms to solve it. The solution of x_{ij}'s is the final task allocation policy.

B. Adversary Model

In this work, we target at the parameter manipulation attack in the stage of task allocation in MCS. The malicious workers submit the falsified parameters, i.e., the cost functions in this paper, to the platform to manipulate the task allocation outcome, and thus receive illegal beneficial gain. As a result, those from others, including benign workers and the platform, may get harmed. Following a conventional approach, here we leverage "utility" to measure a worker's benefit received in MCS. It is defined as $U_i = P_i(\boldsymbol{x_i}) - C_i(\boldsymbol{x_i})$, where $\boldsymbol{x_i}$ is u_i's allocated task set and $P_i(\boldsymbol{x_i})$ is its corresponding payment.

TABLE I
A TOY EXAMPLE.

Parameters				
$T_1 = 3$	$t_1 = 2$ $t_2 = 3$	$d_{11} = 150$ $d_{21} = 100$	$\theta_{11} = 0.7$ $\theta_{21} = 0.8$	$D_1 = 3000$ $D_2 = 2000$
Without attack				
$C_1(x) = 0.1 \cdot e^{0.7x_{ij}}$ $C_2(x) = 0.1 \cdot e^{0.8x_{ij}}$	$x^*_{11} = 3.8$ $x^*_{21} = 0.4$	$U_{11} = 0.1$ $U_{21} = 0.1$		
With attack				
$C'_1(x) = 0.1 \cdot e^{5x_{ij}}$ $C_2(x) = 0.1 \cdot e^{0.8x_{ij}}$	$x'_{11} = 0.9$ $x'_{21} = 3.0$	$U'_{11} = 8.9$ $U'_{21} = 0.1$		

We now use a toy example to better illustrate the parameter manipulation attack. All the system parameters are provided in Table I. Under the attack-free scenario, the allocated sensing time to u_1 and u_2 is $x^*_{11} = 3.8$ and $x^*_{12} = 0.4$, respectively, through optimally solving P_1. Without loss of generality, we set the payment $P_i(\boldsymbol{x_i}) = C_i(\boldsymbol{x_i}) + 0.1$, i.e., each worker gets paid by 0.1 more than its actual cost. Under this setting, we calculate the utility of u_1 and u_2 as $U_{11} = U_{21} = 0.1$. Now, if worker u_1 is malicious and changes ρ_{11} in its cost function to 5, then x'_{11} and x'_{21} become 0.9 and 3.0, respectively. Accordingly, U'_{11} turns to 8.9, which is significantly larger than U_{11} (U'_{21} stays at 0.1 unchanged). Thus, not only does the parameter manipulation attack benefit malicious workers with illegal gain, but also compromises interest of other entities, i.e., the platform has to pay much more than it should.

In this work, malicious workers are modeled as rational and self-interest. When launching a parameter manipulation attack, a malicious worker chooses the strategy that brings itself the greatest benefit. In another word, its objective is solely to maximize its own utility. Therefore, it distinguishes from the attacker who aims to sabotage system operations. These more aggressive attackers are not the focus of this work. It is worth noting that the rational and self-interest attacker is a widely adopted attack model in game theoretical approaches to tackling security problems, such as attack-defense analysis [16] and security/dependability measurement [17].

IV. Our Proposed Scheme

A. KKT Conditions of P_1

Before digging into details of the scheme, we first briefly go through the KKT conditions of P_1, which play a critical role in the scheme design.

In P_1 introduced in Section III-A, its objective and constraint functions are convex. Hence, it admits a unique optimal solution that can be characterized using the necessary and sufficient Karush-Kuhn-Tucker (KKT) conditions [18].

We first derive the *Lagrangian* of P_1 as follows

$$\boldsymbol{L(\lambda, \mu, \nu, x)} = \sum_{i\in[1,M]} C_i(\boldsymbol{x_i}) + \sum_{i\in[1,M]} \lambda_i \cdot (\sum_{j\in[1,N]} x_{ij} - t_i)$$
$$+ \sum_{j\in[1,N]} \mu_j (T_j - \sum_{i\in[1,M]} \theta_{ij} x_{ij}) + \sum_{i\in[1,M]} \nu_i (\sum_{j\in[1,N]} d_{ij} \frac{x_{ij}}{\theta_{ij}} - D_i)$$

where $\boldsymbol{\lambda} \triangleq \{\lambda_i \geq 0 : \forall i \in [1, M]\}$, $\boldsymbol{\mu} \triangleq \{\mu_j \geq 0 : \forall j \in [1, N]\}$, and $\boldsymbol{\nu} \triangleq \{\nu_i \geq 0 : \forall i \in [1, M]\}$ are the vectors of Lagrange multipliers corresponding to constraints (1), (2), and (3), respectively. The KKT conditions that produce the optimal dual solution $\boldsymbol{\lambda}^\circ$, $\boldsymbol{\mu}^\circ$ and $\boldsymbol{\nu}^\circ$, and the optimal primal solution $\boldsymbol{x}^\circ$ for P_1 are given by the following set of equations $\forall i \in [1, M], \forall j \in [1, N]$,

$$\frac{\partial C_i(\boldsymbol{x_i}^\circ)}{\partial x_{ij}} = \mu^\circ_j \theta_{ij} - \frac{\nu^\circ_i d_{ij}}{\theta_{ij}} - \lambda^\circ_i, \tag{5}$$

$$\lambda^\circ_i \cdot (\sum_{j\in[1,N]} x^\circ_{ij} - t_i) = 0, \tag{6}$$

$$\mu^\circ_j \cdot (T_j - \sum_{i\in[1,M]} \theta_{ij} x^\circ_{ij}) = 0, \tag{7}$$

$$\nu^\circ_i \cdot (\sum_{j\in[1,N]} \frac{d_{ij} x^\circ_{ij}}{\theta_{ij}} - D_i) = 0, \tag{8}$$

$$x^\circ_{ij}, \lambda^\circ_i, \mu^\circ_j, \nu^\circ_i \geq 0. \tag{9}$$

The KKT conditions work well when all workers honestly report their genuine cost functions. However, the platform fails to yield $\boldsymbol{x}^\circ$ by solving (5)-(9) with any falsified $C'_i(\cdot) \neq C_i(\cdot)$.

B. Scheme Overview

Our objective is to develop a defense scheme to enable the platform to correctly find out $\boldsymbol{x}^\circ$ even without the correct information of $C_i(\cdot)$. Since workers are suspicious to report manipulated cost functions, rather than having each of them report its cost function $C_i(\cdot)$, it is asked to submit an "indicator" parameter, the willingness of this worker to participate in a specific task. Then we construct a new task allocation optimization problem P_2 (to be presented soon) that takes "indicator" parameters as the coefficients of its objective function, but shares identical constraints with P$_1$. If, with a carefully designed scheme, the optimal solution of P$_2$ is the same with that of P$_1$, $\boldsymbol{x}^\circ$, then we can bypass P$_1$ to find $\boldsymbol{x}^\circ$. It means we no longer need to worry about falsified $C_i'(\cdot)$'s. However, the challenge is how to have the optimal solutions of P$_1$ and P$_2$ identical. Apparently, it relies on the indicators submitted by workers. Then the problem becomes how to design a mechanism to elicit workers to offer proper indicators so as to fulfill this goal. Note that malicious workers may still report falsified indicators to manipulate the new problem P$_2$'s task allocation solution.

Inspired by the work [19], we adopt the incentive mechanism to stimulate workers to submit proper indicators. The platform pays workers according to their reported indicators. Since (malicious) workers are rational, they strategically report indicators to maximize their utility. Then, if indicators which produce $\boldsymbol{x}^\circ$, coincide with the ones which bring worker's maximal utility, then our objective achieves. To summarize, the scheme needs to address two questions. First, what are the values of "proper indicators"? Second, how to elicit all workers to report their "proper indicators"? Next, we are going to answer these two questions in the following two subsections, respectively.

C. Determination of "Proper Indicators"

Instead of $C_i(\cdot)$, we have each worker u_i submit an indicator vector $\boldsymbol{b}_i$ to the platform, where $\boldsymbol{b}_i \triangleq \{b_{i1}, b_{i2}, \cdots, b_{iN}\}$ and b_{ij} is u_i's indicator value for task $\tau_j \in \mathcal{T}$. A lower value of b_{ij} indicates that u_i is more willing to execute τ_j, while a larger value indicates the less willingness u_i has. As mentioned above, malicious workers may still strategically report their $\boldsymbol{b}_i$'s for illegal utility gain.

Upon receiving all $\boldsymbol{b}_i$'s, the platform formulates an alternative task allocation optimization problem (P$_2$)

$$\min_{\boldsymbol{x}} \quad \sum_{i\in[1,M]}\sum_{j\in[1,N]} \frac{b_{ij}}{2}x_{ij}^2, \quad s.t. \quad (1),(2),(3) \text{ and } (4).$$

P$_2$ shares the same set of constraints with P$_1$, but differs in its objective function; P$_2$ minimizes the overall unwillingness (or maximize the overall willingness), while P$_1$ minimizes an overall cost. Besides, P$_2$'s objective function has a much simpler form compared with that of P$_1$.

Since P$_2$ has a convex objective and constraint functions, similarly, it admits a unique optimal solution as well. We write the *Lagrange* of P$_2$ as

$$\tilde{\boldsymbol{L}}(\boldsymbol{\lambda},\boldsymbol{\mu},\boldsymbol{\nu},\boldsymbol{x}) = \sum_{i\in[1,M]}\sum_{j\in[1,N]}\frac{b_{ij}}{2}x_{ij}^2 + \sum_{i\in[1,M]}\lambda_i(\sum_{j\in[1,N]}x_{ij} - t_i) + \sum_{j\in[1,N]}\mu_j(T_j - \sum_{i\in[1,M]}\theta_{ij}x_{ij}) + \sum_{i\in[1,M]}\nu_i(\sum_{j\in[1,N]}d_{ij}\frac{x_{ij}}{\theta_{ij}} - D_i)$$

and denote the optimal primal and dual solutions of P$_2$ as $\boldsymbol{x}^*$ and $\boldsymbol{\lambda}^*$, $\boldsymbol{\mu}^*$ and $\boldsymbol{\nu}^*$, respectively. Its corresponding KKT conditions yield a set of equations,

$$x_{ij}^* = \frac{\mu_j^*\theta_{ij} - \frac{\nu_i^* d_{ij}}{\theta_{ij}} - \lambda_i^*}{b_{ij}}, \tag{10}$$

$$\lambda_i^* \cdot (\sum_{j\in[1,N]} x_{ij}^* - t_i) = 0, \tag{11}$$

$$\mu_j^* \cdot (T_j - \sum_{i\in[1,M]} \theta_{ij}x_{ij}^*) = 0, \tag{12}$$

$$\nu_i^* \cdot (\sum_{j\in[1,N]} \frac{d_{ij}x_{ij}^*}{\theta_{ij}} - D_i) = 0, \tag{13}$$

$$x_{ij}^*, \lambda_i^*, \mu_j^*, \nu_i^* \geq 0, \tag{14}$$

where (11)-(14) are identical to (6)-(9) of P$_1$, while (10) differs from (5).

Our goal is to have $\boldsymbol{x}^\circ \triangleq \boldsymbol{x}^*$, i.e., the optimal solution of P$_1$ is identical to that of P$_2$. Comparing equations (10)-(14) and (5)-(9), we observe that if $b_{ij}x_{ij}^* = \frac{\partial C_i(\boldsymbol{x}_i^*)}{\partial x_{ij}}$ then the goal is achieved. Or, equivalently, u_i submits the following indicator for each task τ_j

$$b_{ij} = \frac{1}{x_{ij}^*} \cdot \frac{\partial C_i(\boldsymbol{x}_i^*)}{\partial x_{ij}}. \tag{15}$$

Up to now, we have determined exact values of proper indicators b_{ij}'s. When workers submit their indicators following (15), the platform can still correctly find out $\boldsymbol{x}^\circ$ by formulating and solving P$_2$, even without the knowledge of workers' original cost functions.

D. Elicitation of "Proper Indicators"

Now the remaining issue is how to induce workers to offer indicators strictly following (15). This is where the incentive mechanism plays; the platform carefully chooses its payment to workers, so as to elicit them to offer the desirable indicators.

Since each worker is rational and self-interest in a sense to always submit its indicator to maximize its own utility, and thus u_i determines its optimal indicator $\boldsymbol{b}_i^*$ by solving the following utility maximization problem (UMP$_i$)

$$\begin{aligned} \max_{\boldsymbol{b}_i} \quad & P_i(\boldsymbol{x}_i) - C_i(\boldsymbol{x}_i) \\ s.t. \quad & b_{ij} \geq 0, \qquad \forall j \in [1,N]. \end{aligned}$$

The unique optimal solution of the UMP$_i$ meets the following optimality condition

$$\frac{\partial C_i(\boldsymbol{x}_i)}{\partial x_{ij}} = \frac{b_{ij}^2}{\lambda_i + \frac{\nu_i d_{ij}}{\theta_{ij}} - \mu_j\theta_{ij}} \frac{\partial P_i(\boldsymbol{x}_i)}{\partial b_{ij}}, \ \forall j \in [1,N], \tag{16}$$

where we utilize derivative $\frac{\partial x_{ij}}{\partial b_{ij}} = \frac{\lambda_i + \frac{\nu_i d_{ij}}{\theta_{ij}} - \mu_j\theta_{ij}}{b_{ij}^2}$ derived from (10).

Jointly consider (5) and (16). In order to elicit workers to submit desirable indicator (15), a feasible incentive mechanism is to pay u_i with

$$P_i(\boldsymbol{b}_i) = \sum_{j\in[1,N]} \frac{(\lambda_i + \frac{\nu_i d_{ij}}{\theta_{ij}} - \mu_j\theta_{ij})^2}{b_{ij}}, \tag{17}$$

where we express $P_i\left(\boldsymbol{b}_i\right)$ as the function of u_i's indicator. Alternatively, we can rewrite $P_i\left(\boldsymbol{b}_i\right)$ as the function of u_i's sensing time with the relation (10)

$$P_i(\boldsymbol{x}_i) = \sum_{j\in[1,N]} x_{ij}(\mu_j\theta_{ij} - \lambda_i - \frac{\nu_i d_{ij}}{\theta_{ij}}), \tag{18}$$

i.e., the payment to u_i is proportional to its devoted sensing time. (18) is intuitive; the more sensing time u_i contributes, the higher payment it receives. Till now, the second question has been answered as well.

Remark I. One may ask rather than eliciting workers to submit proper indicators, why not directly elicit them to submit accurate cost functions? This is because the original cost function (of P$_1$) can be in an arbitrary form. It is extremely difficult to develop an effective incentive mechanism to motivate workers to report the genuine cost functions when their forms are unknown. With the introduction of indicators, a new problem P$_2$ is formulated. Due to its fixed and simplified expression of the objective function, it makes the design feasible.

Remark II. It is worth mentioning two relevant but totally different research topics. The first one is to achieve truthfulness (or called incentive compatibility) in auctions. Its objective is to decide winners and their payment such that their best strategy is to submit their true values/costs as bids. The second one is to find out an employee's true *type* in a monopoly market, when such information is unknown to the employer. *Contract theory* has been widely adopted; incentives are provided to employees such that their utilities are maximized when reporting true types. Differently, we do not care about whether workers honestly submit their true cost functions/types or not; instead, we aim to enable the platform to derive accurate task allocation profile even without these information.

E. Scheme Implementation

With the payment rule (17) or (18), each worker u_i can compute its optimal indicator by solving UMP$_i$. Based on collected indicators, the platform then optimally solves P$_2$ for task allocation. However, UMP$_i$ and P$_2$ are intertwined with each other. On one hand, u_i has to be aware of the task allocation result $\boldsymbol{x}_i$ to solve UMP$_i$ and derive its indicator. On the other hand, $\boldsymbol{x}_i$ is obtained by the platform via solving P$_2$, which takes workers' indicators as inputs. Hence, there is a need for an iterative operation that gradually adjusts results of both P$_2$ and UMP$_i$ to reach the optimum point.

With this in mind, Algorithm 1 outlines our final scheme. The platform first initializes the primal variable $\boldsymbol{x}$ and dual variables $\boldsymbol{\lambda}$, $\boldsymbol{\mu}$ and $\boldsymbol{\nu}$ with their values satisfying KKT conditions (11)-(14). For example, we can choose $\lambda_i^{(0)} = \mu_j^{(0)} = \nu_i^{(0)} = 0$, with any positive value of $x_{ij}^{(0)}$.

Then the algorithm iteratively computes the primal and dual solutions of P$_2$ (at the platform) and indicators via UMP$_i$ (at u_i) until convergence. Specifically, the platform first announces dual solutions $\lambda_i^{(t)}$, $\mu_j^{(t)}$, $\nu_i^{(t)}$ of P$_2$ (line 5). With these values and the knowledge of $C_i(\cdot)$, u_i calculates its optimal indicator $\boldsymbol{b}_i^{(t)}$ by solving UMP$_i$ and submits it to the platform (line 6-7). Then the platform obtains a new allocation rule $\boldsymbol{x}^{(t)}$ by (10) (line 8). It also updates dual solutions $\lambda_i^{(t)}$, $\mu_j^{(t)}$ and $\nu_i^{(t)}$ ($\forall i \in [1,M], j \in [1,N]$) by using a gradient descent method (line 9).

Algorithm 1 Our proposed scheme

```
Output: $\boldsymbol{x}^*$ ($\boldsymbol{x}^\circ$), $P_i(\boldsymbol{x}_i^*)$, $\forall i \in [1,M]$
 1: $t \leftarrow 0$, $conv\_flag \leftarrow 0$;
 2: Initialize $x_{ij}^{(0)}$, $\lambda_i^{(0)}$, $\mu_j^{(0)}$, $\nu_i^{(0)}$, $\forall i \in [1,M], j \in [1,N]$;
 3: while $conv\_flag = 0$ do
 4:     $t \leftarrow t+1$;
 5:     The platform announces $\lambda_i^{(t)}$, $\mu_j^{(t)}$, $\nu_i^{(t)}$, $\forall i \in [1,M], j \in [1,N]$;
 6:     Each worker $u_i$ computes its optimal indicator $\boldsymbol{b}_i^{(t)}$ by solving UMP$_i$;
 7:     Each worker $u_i$ submits its indicator $\boldsymbol{b}_i^{(t)}$ to the platform;
 8:     The platform computes the new $\boldsymbol{x}^{(t)}$ by (10);
 9:     The platform uses gradient updates for dual variables:
            $\lambda_i^{(t)} = \left(\lambda_i^{(t-1)} + s^{(t)} \cdot (\sum_{j\in[1,N]} x_{ij}^{(t-1)} - t_i)\right)^+$
            $\mu_j^{(t)} = \left(\mu_j^{(t-1)} + s^{(t)} \cdot (T_j - \sum_{i\in[1,M]} \theta_{ij} x_{ij}^{(t-1)})\right)^+$
            $\nu_i^{(t)} = \left(\nu_i^{(t-1)} + s^{(t)} \cdot (\sum_{j\in[1,N]} \frac{d_{ij} x_{ij}^{(t-1)}}{\theta_{ij}} - D_i)\right)^+$
        $\forall i \in [1,M], j \in [1,N]$;
10:     if $\left|\frac{b_{ij}^{(t)} - b_{ij}^{(t-1)}}{b_{ij}^{(t-1)}}\right| < \epsilon$, $\forall i \in [1,M], j \in [1,N]$ then
11:         $conv\_flag \leftarrow 1$;
12:     end if
13: end while
14: $\boldsymbol{x}_i^*$ ($\boldsymbol{x}_i^\circ$) $\leftarrow \boldsymbol{x}_i^{(t)}$, $\forall i \in [1,M]$;
15: The platform computes $P_i\left(\boldsymbol{x}_i^*\right), \forall i \in [1,M]$, by (18).
```

Note that $(x)^+$ represents $\max\{x,0\}$. In the end, the platform checks the termination criterion (line 10). When changes of indicators for two consecutive iterations are sufficiently small with $\epsilon \geq 0$, the iteration terminates. Otherwise, another round of iteration is performed. Once convergence is reached, the solution of $\boldsymbol{x}^{(t)}$ is exactly the optimal solution $\boldsymbol{x}^*$ and thus $\boldsymbol{x}^\circ$, i.e., the optimal solution to P$_1$. Finally, the platform determines the final payment $P_i\left(\boldsymbol{x}_i^*\right)$ for each worker with (18) (line 15).

Theorem 1. ***(Convergence.)*** *Algorithm 1 converges to the optimal solution of P$_2$ globally.*

Proof: The formal proof is provided in Appendix A. ■

Besides, it is also critical to show that our scheme pays workers properly, without causing them negative utilities, as otherwise workers will be discouraged from participating.

Proposition 1. *(Non-negative Utility.) Each worker $u_i \in \mathcal{U}$ receives a nonnegative utility via our scheme, i.e.,*

$$P_i\left(\boldsymbol{x}_i^*\right) - C_i\left(\boldsymbol{x}_i^*\right) \geq 0. \tag{19}$$

Proof: The formal proof is provided in the technical report [26]. ∎

V. Enhancing the Convergence Speed

The effectiveness of our defense scheme is in trade of extra interactions between workers and the platform (until Algorithm 1 converges). As a result, calculation delay will be caused in deriving the final task allocation profile. To tackle this side effect, we plan to further accelerate the algorithm convergence speed.

The value of $s^{(t)}$ plays a critical role in the convergence speed of Algorithm 1. Basically, a large value implies a large step size in each iteration toward the optimal solution. However, it may cause oscillation in the algorithm. On the other hand, a small $s^{(t)}$ may lead to extra iterations, and thus a longer running time. Hence, in this section we propose to enhance the convergence speed of Algorithm 1 via adaptive selection of $s^{(t)}$ in each iteration. We adopt the backtracking line search [21], an efficient online search method used in unconstrained convex optimization. Its idea is to determine the maximum step size to move along a given search direction to find the optimal result. Since it operates over unconstrained convex optimization problems, we first transfer P_2 into an unconstrained form. The Lagrange dual function of P_2 is

$$\begin{aligned} g(\boldsymbol{\lambda}, \boldsymbol{\mu}, \boldsymbol{\nu}) &= \inf_{\boldsymbol{x}} \tilde{\boldsymbol{L}}\left(\boldsymbol{\lambda}, \boldsymbol{\mu}, \boldsymbol{\nu}, \boldsymbol{x}\right) \\ &= \sum_{i \in [1,M]} \sum_{j \in [1,N]} \left[\frac{\theta_{ij} f^2 + 2\theta_{ij}\lambda_i f - 2\theta_{ij}^2 \mu_j f + 2\nu_i d_{ij} f}{2 b_{ij} \theta_{ij}}\right] \\ &- \sum_{i \in [1,M]} \left(\lambda_i t_i + \nu_i D_i\right) + \sum_{j \in [1,N]} \mu_j T_j, \end{aligned}$$

where $f = \mu_j \theta_{ij} - \frac{\nu_i d_{ij}}{\theta_{ij}} - \lambda_i$. According to the backtracking line search, our objective is to find the optimal length to maximize the above Lagrange dual function.

Algorithm 2 The selection of suitable s

```
Input: $\boldsymbol{b}_i$, $\Delta(\boldsymbol{\lambda}, \boldsymbol{\mu}, \boldsymbol{\nu})$, $\alpha$, $\beta$
Output: $s$
1: $s \leftarrow 1$
2: while $g\left((\boldsymbol{\lambda}, \boldsymbol{\mu}, \boldsymbol{\nu}) + s\Delta(\boldsymbol{\lambda}, \boldsymbol{\mu}, \boldsymbol{\nu})\right) < g\left(\boldsymbol{\lambda}, \boldsymbol{\mu}, \boldsymbol{\nu}\right) + \alpha \cdot s \cdot \nabla g\left(\lambda_i, \mu_j, \nu_i\right)^T \Delta(\boldsymbol{\lambda}, \boldsymbol{\mu}, \boldsymbol{\nu})$ do
3:     $s \leftarrow \beta s$.
4: end while
5: return $s$;
```

The inputs of Algorithm 2 include $\boldsymbol{b}_i$, the descending direction $\Delta(\boldsymbol{\lambda}, \boldsymbol{\mu}, \boldsymbol{\nu}) = (\frac{d\lambda_i}{dt}, \frac{d\mu_j}{dt}, \frac{d\nu_i}{dt})$ for $g(\boldsymbol{\lambda}, \boldsymbol{\mu}, \boldsymbol{\nu})$, where $\lambda_i, \mu_j, \nu_i \geq 0$ ($i \in [1, M], j \in [1, N]$), and two predefined constants α and β. The output is the optimal line length s. It starts with a rough estimate, i.e., $s = 1$ (line 1). Then we iteratively adapt the step length (line 3) as long as the criterion in line 2 holds, where $\nabla g\left(\lambda_i, \mu_j, \nu_i\right)$ represents the local gradient of function g. Note that $g\left((\boldsymbol{\lambda}, \boldsymbol{\mu}, \boldsymbol{\nu}) + s\Delta(\boldsymbol{\lambda}, \boldsymbol{\mu}, \boldsymbol{\nu})\right) = g\left(\boldsymbol{\lambda}, \boldsymbol{\mu}, \boldsymbol{\nu}\right) + \alpha \cdot s \cdot \nabla g\left(\lambda_i, \mu_j, \nu_i\right)^T \Delta(\boldsymbol{\lambda}, \boldsymbol{\mu}, \boldsymbol{\nu})$ takes place at the optimal point of g. Following the standard approach in the backtracking line search, we set $\alpha \in (0, 0.5)$ and $\beta \in (0, 1)$.

Once the suitable s is identified via Algorithm 2, it will be used to update $s^{(t)}$ in each iteration of Algorithm 1.

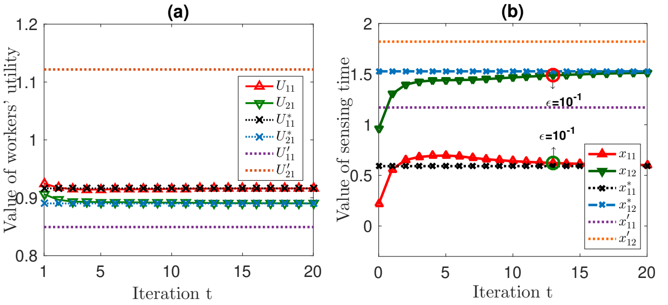

Fig. 1. Task allocation outcome comparison under different scenarios, i.e., without attack, with attack, with attack but protected by our scheme.

VI. Performance Evaluation

In this section, we conduct extensive simulations to evaluate the performance of our scheme. The study is based on the Yelp Dataset Challenge [22]. The data is sampled by Yelp from the greater Phoenix, AZ metropolitan area from March 2005 to January 2013. This dataset includes 11537 businesses, 229907 reviews by 229907 users, and 8282 check-in sets in the form of separate JSON or SQL files. Specifically, our evaluation selects a set of Yelp users as workers in the MCS system. We then take the distance between two consecutive check-in locations from the same user as the worker's travel distance for one task and the corresponding time interval in between as the worker's maximum sensing time it can contribute. Note that this dataset has been widely used in many other crowd/social sensing related research, such as [23], [24].

We assume that each worker u_i holds its convex objective function $C_i\left(\boldsymbol{x}_i\right) = 0.1 \cdot \sum_{j \in [1,N]} e^{\rho_{ij} x_{ij}}$ where ρ_{ij} is randomly chosen from $[0.5, 1]$. For the sensing quality θ_{ij}, it is a random value from $[0, 1]$. Besides, we set the termination condition $\epsilon = 10^{-5}$, i.e., the algorithm terminates if the gap between two consecutive iterations is less than 10^{-5}. Each simulation result is the average over 20 trials.

Effectiveness of Our Scheme. We start from a small-scale MCS with $M = 2$ workers and $N = 2$ tasks. Besides, u_2 is assumed as the malicious worker to manipulate the task allocation outcome. Performances of our scheme, in terms of security and convergence property, have been examined.

Fig. 1(a) shows utility of u_1 and u_2 under three scenarios, i.e., without attack (U_{11}^*, U_{21}^*), with attack (U_{11}', U_{21}'), with attack but protected by our scheme (U_{11}, U_{21}). Once the algorithm converges, we observe that $U_{21}' > U_{21}^*$. It means

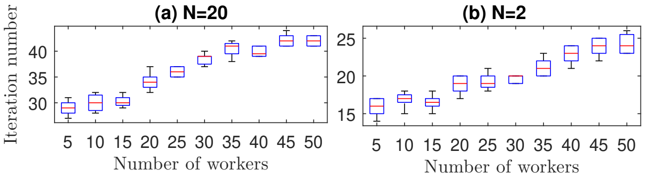

Fig. 2. Iteration number needed for our algorithm to converge under different MCS sizes.

the attacker u_2 gains extra utility by launching the parameter manipulation attack. However, such a utility gain diminishes once our scheme is implemented. Specifically, U_{21} is equal to U_{21}^*. Meanwhile, $U_{11}' < U_{11}^*$, i.e., the benign worker u_1's utility is harmed by u_2's parameter manipulation attack. However, it can be fully defended by our scheme, as $U_{11}^* = U_{11}$.

We further depict in Fig. 1(b) worker's total allocated sensing time under the three scenarios same as above. With the implementation of our scheme, we observe $x_{11} = x_{11}^* = 0.6$ and $x_{12} = x_{12}^* = 1.5$ after 20 iterations. It means that the platform is capable of finding the optimal task allocation policy even with the existence of malicious worker u_2. Besides, if setting $\epsilon = 10^{-1}$ (the gap between two consecutive iterations is less than 10^{-1}), our scheme will stop after only 13 iterations. Hence, if we want to achieve a shorter running time of the scheme, a larger ϵ is in need.

Impact of MCS Size. We further evaluate in Fig. 2 the impact of MCS size, in terms of worker and task numbers, to the scheme performance. For example, in Fig. 2(a), when $N = 20$ and $M = 5$, the average iteration number is about 29. It is slightly increased to 42 when $M = 50$. The similar trend is observed in Fig. 2(b); when $N = 2$ and M ranges from 5 to 50. We conclude that the algorithm converges pretty fast with moderate system sizes. Moreover, as discussed in the previous section, the platform can further choose a larger ϵ to accelerate the convergence speed.

Impact of Step Size s. Recall that s stands for the step size of our iterative Algorithm 1. Fig. 3 shows the convergence property of Algorithm 1 under different values of s. Still, we consider a system with $M = 50$ workers and $N = 20$ tasks. Note that the dashed line represents the optimal result of P_1 when all workers honestly report their genuine cost functions. We find that the convergence speed is dependent on the step size. For example, when $s = 3.5 \times 10^{-5}$, it takes 61 iterations to reach the optimal result. When choosing a larger $s = 4.1 \times 10^{-5}$, i.e., a larger step size, the iteration number decreases to 43. However, a larger step size does not necessarily lead to a faster convergence speed. For example, when $s = 4.7 \times 10^{-5}$, the iteration number becomes 48. Thus, a suitable s is critical to the convergence speed of our scheme.

Impact of Malicious Worker Ratio. We now examine the impact of malicious worker ratio to the effectiveness of our scheme. It is defined as the percentage of malicious workers to the entire worker set. We consider an MCS consisting of $M = 50$ workers and $N = 2$ sensing tasks, where u_1 is set as a benign worker. Fig. 4 shows u_1's utility U_1. We first examine the case without our scheme. When all workers honestly report their cost functions, i.e., the ratio is 0, we obtain $U_1 = 0.92$. As the ratio increases, U_1 drops fast. In particular, when the ratio is equal to 26%, i.e., there are 13 malicious workers, U_1 becomes 0.64. Moreover, the red dashed line (with our scheme) in this figure clearly demonstrates that U_1 keeps at 0.92, under different malicious worker ratios. Thus, we conclude that the malicious worker ratio does not impact the effectiveness of our scheme.

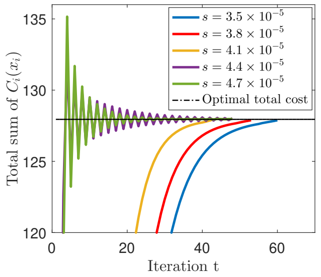

Fig. 3. Comparison of convergence speed under different step size s.

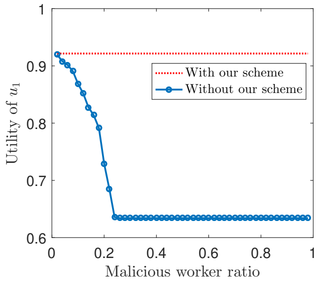

Fig. 4. u_1's utility with or without our scheme under different malicious worker ratios.

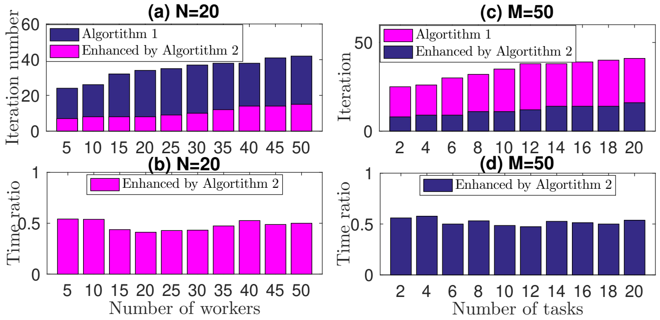

Fig. 5. Iteration numbers and time needed.

Performance Enhancement by Integrating Algorithm 2. We now validate the effectiveness of Algorithm 2. Fig. 5(a) compares the iteration number needed for Algorithm 1 to converge with and without the integration of Algorithm 2. Apparently, the former is significantly smaller than the latter in all cases when the number of workers is from 5 to 50. For example, when there are 30 workers, the enhanced Algorithm 1 requires 11 iterations to converge, while the other one requires 36 iterations; the former is about 1/3 of the latter. Under the same parameter setting, we further show in Fig. 5(b) the running time ratio between these two algorithms. We observe that the enhanced Algorithm 1 only needs about half running time than the other one. We have a similar observation in Fig. 5(c) and Fig. 5(d) under $M = 50$. Thus, we conclude that Algorithm 2 can significantly improve the performances of Algorithm 1, in terms of both iteration number and running time.

VII. Conclusion

In order to resist parameter manipulation attacks in the stage of task allocation in MCS, we leverage the incentive mechanism to develop an effective and novel defense scheme. Our idea is to stimulate workers to report desirable indicators, such that the new formulated problem shares the identical optimal solution with the original one. To implement the defense scheme, we further develop an iterative algorithm. The backtracking based approach is adopted to accelerate the convergence speed. We then formally prove its convergence property. Extensive simulation results show that the algorithm only takes a few iterations to converge, and thus to find the accurate solution under moderate system sizes.

Acknowledgment

The work of Ming Li was supported by the U.S. National Science Foundation under grants CNS-1566634 and ECCS-1711991. The work of Linke Guo was partially supported by the U.S. National Science Foundation under grants IIS-1722731 and ECCS-1710996. The work of Miao Pan was partially supported by the U.S. National Science Foundation under grants US CNS-1350230 (CAREER), CNS-1646607, CNS-1702850 and CNS-1801925. The work of Zhu Han was partially supported by the US MURI AFOSR MURI 18RT0073, the U.S. National Science Foundation under grants CNS-1717454, CNS-1731424, CNS-1702850 and CNS-1646607.

References

[1] M. Karaliopoulos, et al., "User recruitment for mobile crowdsensing over opportunistic networks," in *Proceedings of IEEE INFOCOM*, 2015.

[2] Z. Duan, et al., "Distributed auctions for task assignment and scheduling in mobile crowdsensing systems," in *Proceedings of IEEE ICDCS*, 2017.

[3] L. Pournajaf, et al., "Spatial task assignment for crowd sensing with cloaked locations," in *Proceedings of IEEE MDM*, July 2014.

[4] K. L. Huang, et al., "Are you contributing trustworthy data?: The case for a reputation system in participatory sensing," in *Proceedings of the ACM MSWiM*, October 2010.

[5] P. Gilbert, et al., "Toward trustworthy mobile sensing," in *Proceedings of the HotMobile*, February 2010.

[6] K. L. Huang, et al., "A privacy-preserving reputation system for participatory sensing," in *Proceedings of IEEE LCN*, October 2012.

[7] D. Christin, et al., "Incognisense: An anonymity-preserving reputation framework for participatory sensing applications," in *Proceedings of IEEE PerCom*, March 2012.

[8] D. He, et al., "User privacy and data trustworthiness in mobile crowd sensing," *IEEE Wireless Communications*, vol. 22, no. 1, pp. 28–34, February 2015.

[9] A. Dua, et al., "Towards trustworthy participatory sensing," in *Proceedings of the USENIX HotSec*, 2009.

[10] W. Jin, et al., "DPDA: A Differentially Private Double Auction Scheme for Mobile Crowd Sensing," in *Proceedings of IEEE CNS*, 2018.

[11] W. Jin, et al., "If You Do Not Care About It, Sell It: Trading Location Privacy in Mobile Crowd Sensing," in *Proceedings of IEEE INFOCOM*, 2019.

[12] H. Xie, et al., "Incentive mechanism and protocol design for crowdsourcing systems," in *Proceedings of Allerton*, September 2014.

[13] H. Jin, et al., "Quality of information aware incentive mechanisms for mobile crowd sensing systems," in *Proceedings of the ACM MobiHoc*, June 2015.

[14] Z. Feng, et al., "Trac: Truthful auction for location-aware collaborative sensing in mobile crowdsourcing," in *Proceedings of IEEE INFOCOM*, April 2014.

[15] P. Micholia, et al., "Mobile crowdsensing incentives under participation uncertainty," in *Proceedings of the ACM Workshop on MSCC*, July 2016.

[16] J. Jormakka and J. Molsa, "Modelling information warfare as a game," *Journal of information warfare*, vol. 4, no. 2, pp. 12–25, 2005.

[17] L. Jiang, et al., "How bad are selfish investments in network security?" *IEEE/ACM Transactions on Networking*, vol. 19, no. 2, pp. 549–560, 2011.

[18] D. P. Bertsekas, *Nonlinear programming*. Belmont, MA, USA: Athena Scientific, 1999.

[19] G. Iosifidis, et al., "A double-auction mechanism for mobile data-offloading markets," *IEEE/ACM Transactions on Networking*, vol. 23, no. 5, pp. 1634–1647, October 2015.

[20] C. Liu, et al., "Network optimization and control," *Foundations and Trends in Networking*, vol. 2, no. 3, pp. 271–379, 2007.

[21] S. Boyd and L. Vandenberghe, *Convex optimization*. Stanford, CA, USA: Cambridge university press, 2004.

[22] Yelp dataset challenge. [Online]. Available: https://www.yelp.com/dataset/challenge

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

[24] B. Liu, et al., "Protecting location privacy in spatial crowdsourcing using encrypted data," in *Proceedings of EDBT*, 2017.

[25] C. Liu, et al., "Mechanism design games for thwarting malicious behavior in crowdsourcing application," in *Proceedings of IEEE INFOCOM*, 2017.

[26] [Online]. Available: https://www.dropbox.com/s/mqnsutw7q6e47m2/TechnicalReport.pdf?dl=0

Appendix A
Proof of Theorem 1

We first cite a notation that has been used in [20]. It will be extensively used in our analysis. Given $g(x)$ an arbitrary function, and x and y arbitrary real values, $(g(x))_y^+$ is defined as

$$(g(x))_y^+ = \begin{cases} g(x), & y > 0, \\ \max(g(x), 0), & y = 0. \end{cases} \tag{20}$$

To prove Theorem 1, it is critical to derive the following lemma first.

Lemma 1. *Given $\boldsymbol{\lambda}$, $\boldsymbol{\mu}$ and $\boldsymbol{\nu}$*[2] *as dual solutions obtained in an arbitrary iteration of Algorithm 1, and $\boldsymbol{\lambda}^*$, $\boldsymbol{\mu}^*$ and $\boldsymbol{\nu}^*$ the optimal dual solutions, we have*

$$(\lambda_i - \lambda_i^*)(\sum_{j\in[1,N]} x_{ij} - t_i)_{\lambda_i}^+ \leq (\lambda_i - \lambda_i^*)(\sum_{j\in[1,N]} x_{ij} - t_i)$$

$$(\mu_j - \mu_j^*)(T_j - \sum_{i\in[1,M]} \theta_{ij}x_{ij})_{\mu_j}^+ \leq (\mu_j - \mu_j^*)(T_j - \sum_{i\in[1,M]} \theta_{ij}x_{ij})$$

$$(\nu_i - \nu_i^*)(\sum_{j\in[1,N]} \frac{d_{ij}x_{ij}}{\theta_{ij}} - D_i)_{\nu_i}^+ \leq (\nu_i - \nu_i^*)(\sum_{j\in[1,N]} \frac{d_{ij}x_{ij}}{\theta_{ij}} - D_i).$$

Proof: We focus on the first inequality. According to the update rule to $\boldsymbol{\lambda}$ in Algorithm 1, we have $\boldsymbol{\lambda} \succeq \mathbf{0}$. Then for $\lambda_i \in \boldsymbol{\lambda}$, we discuss under two cases: $\lambda_i = 0$ and $\lambda_i > 0$.

Case I: $\lambda_i = 0$. We have

$$\begin{aligned} & (\lambda_i - \lambda_i^*)(\sum_{j\in[1,N]} x_{ij} - t_i)_{\lambda_i}^+ \leq (\lambda_i - \lambda_i^*)(\sum_{j\in[1,N]} x_{ij} - t_i) \\ \Longleftrightarrow \quad & (\sum_{j\in[1,N]} x_{ij} - t_i)_{\lambda_i}^+ \geq \sum_{j\in[1,N]} x_{ij} - t_i \\ \Longleftrightarrow \quad & \max\{\sum_{j\in[1,N]} x_{ij} - t_i, 0\} \geq \sum_{j\in[1,N]} x_{ij} - t_i. \end{aligned} \tag{21}$$

Therefore, the first inequality holds.

[2] In the following, we use $\boldsymbol{\lambda}$, $\boldsymbol{\mu}$ and $\boldsymbol{\nu}$ to represent $\boldsymbol{\lambda}^{(t)}$, $\boldsymbol{\mu}^{(t)}$ and $\boldsymbol{\nu}^{(t)}$, respectively, without causing confusion.

Case II: $\lambda_i > 0$. We have

$$(\lambda_i - \lambda_i^*)(\sum_{j\in[1,N]} x_{ij} - t_i)^+_{\lambda_i} \le (\lambda_i - \lambda_i^*)(\sum_{j\in[1,N]} x_{ij} - t_i)$$

$$\iff \quad (\lambda_i - \lambda_i^*)(\sum_{j\in[1,N]} x_{ij} - t_i) \le (\lambda_i - \lambda_i^*)(\sum_{j\in[1,N]} x_{ij} - t_i).$$

Thus, the first inequality also holds.

In all, the first inequality holds in both cases. The proof for the rest two inequalities similarly follows. ■

With Lemma 1, we are now ready to prove Theorem 1. We first rewrite it here again.

Theorem 1. *Algorithm 1 converges to the optimal solution of* P_2 *globally.*

Proof: We consider a very small time slot, and hence assume that dual solutions are updated according to the differential equations

$$\frac{d\lambda_i}{dt} = (\sum_{j\in[1,N]} x_{ij} - t_i)^+_{\lambda_i}, \tag{22}$$

$$\frac{d\mu_j}{dt} = (T_j - \sum_{i\in[1,M]} \theta_{ij} x_{ij})^+_{\mu_j}, \tag{23}$$

$$\frac{d\nu_i}{dt} = (\sum_{j\in[1,N]} \frac{d_{ij} x_{ij}}{\theta_{ij}} - D_i)^+_{\nu_i}. \tag{24}$$

We first define the Lyapunov function

$$Z(\boldsymbol{\lambda}, \boldsymbol{\mu}, \boldsymbol{\nu}) = \sum_{i=1}^{M} \frac{(\lambda_i - \lambda_i^*)^2}{2} + \sum_{j=1}^{N} \frac{(\mu_j - \mu_j^*)^2}{2} + \sum_{i=1}^{M} \frac{(\nu_i - \nu_i^*)^2}{2}.$$

If we can prove that $\frac{dZ(\boldsymbol{\lambda}, \boldsymbol{\mu}, \boldsymbol{\nu})}{dt} \le 0$, it indicates that $Z(\boldsymbol{\lambda}, \boldsymbol{\mu}, \boldsymbol{\nu})$ is stable and thus our algorithm converges to the optimal solution of P_2. By applying the chain rule and taking the derivative with respect to t, we obtain

$$\frac{dZ(\boldsymbol{\lambda}, \boldsymbol{\mu}, \boldsymbol{\nu})}{dt} = \sum_{i\in[1,M]} (\lambda_i - \lambda_i^*) \frac{d\lambda_i}{dt} + \sum_{j\in[1,N]} (\mu_j - \mu_j^*) \frac{d\mu_j}{dt} + \sum_{i\in[1,M]} (\nu_i - \nu_i^*) \frac{d\nu_i}{dt},$$

which can be rewritten below with (22)-(24)

$$\begin{aligned} \frac{dZ(\boldsymbol{\lambda}, \boldsymbol{\mu}, \boldsymbol{\nu})}{dt} = & \sum_{i\in[1,M]} (\lambda_i - \lambda_i^*) \cdot (\sum_{j\in[1,N]} x_{ij} - t_i)^+_{\lambda_i} \\ & + \sum_{j\in[1,N]} (\mu_j - \mu_j^*) \cdot (T_j - \sum_{i\in[1,M]} \theta_{ij} x_{ij})^+_{\mu_j} \\ & + \sum_{i\in[1,M]} (\nu_i - \nu_i^*) \cdot (\sum_{j\in[1,N]} \frac{d_{ij} x_{ij}}{\theta_{ij}} - D_i)^+_{\nu_i}. \end{aligned}$$

From Lemma 1, we directly derive

$$\begin{aligned} \frac{dZ(\boldsymbol{\lambda}, \boldsymbol{\mu}, \boldsymbol{\nu})}{dt} \le & \sum_{i\in[1,M]} (\lambda_i - \lambda_i^*) \cdot (\sum_{j\in[1,N]} x_{ij} - t_i) \\ & + \sum_{j\in[1,N]} (\mu_j - \mu_j^*) \cdot (T_j - \sum_{i\in[1,M]} \theta_{ij} x_{ij}) \\ & + \sum_{i\in[1,M]} (\nu_i - \nu_i^*) \cdot (\sum_{j\in[1,N]} \frac{d_{ij} x_{ij}}{\theta_{ij}} - D_i). \end{aligned}$$

Next, we make some modification in the right-hand-side of the above inequality. Hence, we get

$$\begin{aligned} \dot{Z} \le & \sum_{i\in[1,M]} (\lambda_i - \lambda_i^*) \cdot (\sum_{j\in[1,N]} x_{ij} - \sum_{j=1}^{N} x_{ij}^*) \\ & + \sum_{i\in[1,M]} (\lambda_i - \lambda_i^*) \cdot (\sum_{j\in[1,N]} x_{ij}^* - t_i) \\ & + \sum_{j\in[1,N]} (\mu_j - \mu_j^*) \cdot (\sum_{i\in[1,M]} \theta_{ij} x_{ij}^* - \sum_{i\in[1,M]} \theta_{ij} x_{ij}) \\ & + \sum_{j\in[1,N]} (\mu_j - \mu_j^*) \cdot (T_j - \sum_{i\in[1,M]} \theta_{ij} x_{ij}^*) \\ & + \sum_{i\in[1,M]} (\nu_i - \nu_i^*) \cdot (\sum_{j\in[1,N]} \frac{d_{ij} x_{ij}}{\theta_{ij}} - \sum_{j\in[1,N]} \frac{d_{ij} x_{ij}^*}{\theta_{ij}}) \\ & + \sum_{i\in[1,M]} (\nu_i - \nu_i^*) \cdot (\sum_{j\in[1,N]} \frac{d_{ij} x_{ij}^*}{\theta_{ij}} - D_i). \end{aligned}$$

The second, forth and sixth terms in RHS of the above inequality are nonpositive due to (1)-(3) and (11)-(13). Thus,

$$\begin{aligned} \dot{Z} \le & \sum_{i\in[1,M]} (\lambda_i - \lambda_i^*) \cdot (\sum_{j\in[1,N]} x_{ij} - \sum_{j\in[1,N]} x_{ij}^*) \\ & + \sum_{j\in[1,N]} (\mu_j - \mu_j^*) \cdot (\sum_{i\in[1,M]} \theta_{ij} x_{ij}^* - \sum_{i\in[1,M]} \theta_{ij} x_{ij}) \\ & + \sum_{i\in[1,M]} (\nu_i - \nu_i^*) \cdot (\sum_{j\in[1,N]} \frac{d_{ij} x_{ij}}{\theta_{ij}} - \sum_{j\in[1,N]} \frac{d_{ij} x_{ij}^*}{\theta_{ij}}) \\ = & \sum_{i\in[1,M]} \sum_{j\in[1,N]} (x_{ij} - x_{ij}^*)(\frac{\partial C_i(\boldsymbol{x}_i^*)}{\partial x_{ij}} - \frac{\partial C_i(\boldsymbol{x}_i)}{\partial x_{ij}}). \end{aligned}$$

The last equation comes from (5). The RHS of the above inequality is nonpositive because of the following property that holds for any convex function $f(\cdot)$ [18]

$$f(u) \ge f(v) + \nabla f(v)^T (u - v). \tag{25}$$

To be specific, for any $\boldsymbol{x}_i$ and the optimal solution $\boldsymbol{x}_i^*$, we have

$$\left[\frac{\partial C_i(\boldsymbol{x}_i^*)}{\partial x_{i1}}, \cdots, \frac{\partial C_i(\boldsymbol{x}_i^*)}{\partial x_{iN}}\right]^T \cdot (\boldsymbol{x}_i - \boldsymbol{x}_i^*) \le C_i(\boldsymbol{x}_i) - C_i(\boldsymbol{x}_i^*)$$

$$\left[\frac{\partial C_i(\boldsymbol{x}_i)}{\partial x_{i1}}, \cdots, \frac{\partial C_i(\boldsymbol{x}_i)}{\partial x_{iN}}\right]^T \cdot (\boldsymbol{x}_i^* - \boldsymbol{x}_i) \le C_i(\boldsymbol{x}_i^*) - C_i(\boldsymbol{x}_i).$$

Then, add the above two inequalities,

$$\begin{aligned} & \left[\frac{\partial C_i(\boldsymbol{x}_i^*)}{\partial x_{i1}} - \frac{\partial C_i(\boldsymbol{x}_i)}{\partial x_{i1}}, \cdots, \frac{\partial C_i(\boldsymbol{x}_i^*)}{\partial x_{iN}} - \frac{\partial C_i(\boldsymbol{x}_i)}{\partial x_{iN}}\right]^T (\boldsymbol{x}_i - \boldsymbol{x}_i^*) \\ & = \sum_{j\in[1,N]} (x_{ij} - x_{ij}^*)(\frac{\partial C_i(\boldsymbol{x}_i^*)}{\partial x_{ij}} - \frac{\partial C_i(\boldsymbol{x}_i)}{\partial x_{ij}}) \le 0. \end{aligned}$$

By adding together the above inequalities for all $\boldsymbol{x}_i$'s ($i \in [1, M]$), we have

$$\sum_{i\in[1,M]} \sum_{j\in[1,N]} (x_{ij} - x_{ij}^*)(\frac{\partial C_i(\boldsymbol{x}_i^*)}{\partial x_{ij}} - \frac{\partial C_i(\boldsymbol{x}_i)}{\partial x_{ij}}) \le 0.$$

Thus, $\dot{Z} \le 0$ holds, which ends the proof. ■