Dynamic Task Pricing in Multi-Requester Mobile Crowd Sensing with Markov Correlated Equilibrium

Haiming Jin*†, Hongpeng Guo‡, Lu Su§, Klara Nahrstedt‡, Xinbing Wang†
*John Hopcroft Center for Computer Science, Shanghai Jiao Tong University, Shanghai, China
†Department of Electronic Engineering, Shanghai Jiao Tong University, Shanghai, China
‡Department of Computer Science, University of Illinois at Urbana-Champaign, IL, USA
§Department of Computer Science and Engineering, State University of New York at Buffalo, NY, USA
Email: jinhaiming@sjtu.edu, hg5@illinois.edu, lusu@buffalo.edu, klara@illinois.edu, xwang8@sjtu.edu.cn

***Abstract*—The recent proliferation of human-carried mobile devices has given rise to mobile crowd sensing (MCS) systems, where a myriad of data requesters outsource their sensing tasks to a crowd of workers via a cloud-based platform. In order to incentivize participation, requesters typically compensate workers with specific amount of payments. Clearly, setting an appropriate task price is critical for a requester to attract enough worker participation without unnecessary expenses. Therefore, we *investigate the problem of task pricing in MCS systems with multi-requester price competition*, and also *dynamically arriving workers*. Task pricing in such scenario is challenging, because of each requester's incomplete information about the others, uncertainty of future information, etc. So as to address these challenges, we use *Markov game* to model requesters' competitive task pricing, and *Markov correlated equilibrium (MCE)* as the solution concept. We propose that the platform uses the social cost-minimizing MCE to coordinate requesters' prices, which is *self-enforcing*, and optimizes the system-wide objective of social cost. Technically, we propose a *computationally efficient* algorithm to compute an approximately optimal MCE. Furthermore, through extensive performance evaluation, we show numerically that our algorithm yields close-to-minimum social cost in very short running time.**

I. Introduction

The recent proliferation of increasingly capable mobile devices (e.g., smartphones, smartwristbands, smartwatches) with a wide variety of on-board sensors (e.g., accelerometer, gyroscope, compass, camera, GPS) has given rise to mobile crowd sensing (MCS), a novel sensing paradigm which outsources the collection of large scale sensory data to a crowd of participants, namely (crowd) workers. Thus far, applications of crowd sensing [1, 2] have pervaded almost every aspect of our everyday life, including smart transportation, environment monitoring, urban sensing, healthcare, and many others.

In a typical MCS system, multiple data requesters who want to collect sensory data from the public crowd post their sensing tasks (e.g., estimating the traffic speeds of certain road segments, monitoring the noise levels of specific geographic areas) via a platform, which is typically a cloud-based central server. The platform usually collects and may process to a certain degree workers' sensory data. Together with the tasks to be executed, in order to incentivize worker participation, each requester also posts a price which specifies the amount of payment a worker will get by carrying out her tasks.

Needless to say, setting an appropriate price is critical to each requester, as an excessively high price will incur unnecessary expenses, whereas a price that is too low will possibly cause a requester to lose the price competition against others, and eventually fail to attract participants. Therefore, in this paper, we investigate the problem of *task pricing in MCS systems, where multiple data requesters compete against each other to attract worker participation*. More specifically, we study realistic *dynamic MCS systems*, where *workers arrive dynamically in an online manner*. We next elaborate upon the challenges of task pricing in such scenario.

The first challenge comes from the fact that a requester typically has *incomplete information* about other requesters. When a rational requester makes her pricing decision, not only does she have to consider factors regarding herself, such as how many sensing tasks she holds, and how time-sensitve her tasks are, but also will she take into account how other requesters set their prices, so that she could win the price competition without setting her price too high. However, in practice, a requester only knows the factors that affect her own price, but rarely has enough information about others'. Therefore, under such incomplete information, selecting a reasonable task price would be rather challenging for a requester.

Besides, *uncertainty of the future* also adds to a requester's challenges in deciding her task price. Different from a static MCS system studied in a plethora of existing literature [3–12] where requesters make one-shot pricing decisions, in a dynamic MCS system, a requester will have to select prices that not simply maximize her immediate utility, but more importantly maximize her long-term utility considering steps into the future. However, what will happen in the future is usually rather stochastic. On one hand, it is hard to predict the exact number of workers arriving in future time instances. On the other hand, as aforementioned, it is already very challenging to consider other requesters' pricing behaviors at the current time instance, let alone those in the future. Therefore, unpredictability of the future makes it even more

We sincerely thank Professor R. Srikant for his valuable contribution. We gratefully acknowledge the support of National Science Foundation grant CNS-1652503, National Key R&D Program of China 2018YFB1004702, and NSF China grants 61829201 and 61532012. The views and conclusions contained in this document are those of the authors and should not be interpreted as necessarily representing the official policies, either expressed or implied, of the sponsors.

challenging for requesters to set appropriate task prices.

Considering the challenge brought by incomplete information, we take the role of the platform, which typically has global information of all the requesters that register on it, including the numbers of tasks held by them, the time-sensitiveness of their tasks, and many others. As an entity that has access to system-wide information, the platform is thus much more suitable than the requesters themselves to decide the task prices. Therefore, more accurately put, the objective of this paper is to investigate *how the platform could decide reasonable task prices for the data requesters in MCS systems with multi-requester competition and dynamic worker arrival.*

In order to fully capture the influence of stochastic future information on how requesters set their prices, we model the dynamic task pricing among them as a *Markov game* [13], where at any time instance each requester aims to propose the price that maximizes her long-term cumulative utility. Below, we would like to shed some light on how we address the various challenges on formulating and solving the Markov game among requesters.

To begin with, we meticulously design the states of the Markov game and requesters' utility functions so as to capture, to the greatest extent, requesters' pricing behaviors in practice. Besides, choosing a proper solution concept for the formulated Markov game is also highly critical and non-trivial. In line with our objective of deciding reasonable prices for requesters, we adopt *Markov correlated equilibrium (MCE)* as the targeted solution. Simply put, an MCE is a probability distribution over the space of requesters' possible price choices which satisfies the *self-enforcing* property [14]. Such property ensures that if the platform suggests requesters to set their prices as the ones sampled from an MCE, only by setting her price as suggested, could each requester maximize her utility. Finally, although there might be multiple MCE's for a Markov game, we are specifically interested in computing the one that *minimizes the social cost*, which is computationally intractable in general. Such issue on computational complexity is also addressed by us through carefully splitting the overall Markov game into several smaller games with much fewer requesters, while ensuring an approximately minimum social cost.

In summary, we primarily make the following contributions in this paper.

- We solve the problem of task pricing in *dynamic MCS systems* with both *multi-requester price competition* and *dynamic worker arrival*.
- We address the various arising challenges (e.g., incomplete information, future uncertainties) by *realistically* and *meticulously modeling* the problem as a *Markov game*, and utilizing *Markov correlated equilibrium*, which is *self-enforcing*, to decide requesters' prices.
- We show that computing the social cost-minimizing MCE is computationally intractable in general, and we design a *computationally efficient* algorithm to compute an MCE that *approximately minimizes the social cost*.

II. Related Work

The set of literature most relevant to this paper are the suite of incentive mechanisms designed to stimulate worker participation in MCS systems [3–12, 15–30]. Among them, [3–12] focus on static MCS systems, where workers and requesters arrive all at once, and there is only a single data requester (usually represented by the central platform). Specifically, these works investigate contest design [3], quality-aware mechanisms [4–8], cooperation among service providers [16], distributed task selection [17], network effect [11], as well as many other issues related to incentivizing worker participation. Although MCS systems with dynamic worker arrivals have been studied in [21–28], these works consider the scenario with only one data requester, as well. Furthermore, [16–19] study incentive mechanism design in multi-requester, but static MCS systems.

Different from most of the existing work, this paper tackles the problem of task pricing in dynamic MCS systems with both multi-requester price competition and dynamic worker arrivals. Note that similar problem settings have been considered in [15, 20, 29]. However, in contrast to this paper, [15] applies specifically in proximity-based MCS systems, where workers have to be physically in the close vicinity of requesters to execute their tasks, and task prices in [20, 29] are decided without considering requesters' long-term cumulative utilities.

III. Preliminaries

A. System Overview

The MCS system studied in this paper consists of a cloud-based platform, a set of K requesters, denoted as $\mathcal{R} = \{r_1, \cdots, r_K\}$, and a crowd of participating workers. Each requester r_i holds N_i sensing tasks, which in practice could potentially belong to different types that require different knowledge, expertise, or even sensing devices from workers. For example, road sensing (e.g., traffic speed estimation, pothole detection) could only be executed by drivers that carry mobile devices in their vehicles, and air quality monitoring by pedestrians with specific kinds of air quality sensors.

Based on such observation, we assume that requesters hold a total number of W types (i.e., type 1, 2, $\cdots$, W) of tasks, and a type-w task can only be executed by a type-w worker. Furthermore, we use $N_{i,w}$ to denote the number of type-w tasks held by requester r_i, $\mathcal{R}_w$ the set of requesters that hold type-w tasks, $\mathbf{N}_i$ the vector containing all $N_{i,w}$'s such that $r_i \in \mathcal{R}_w$. Obviously, $\sum_{w:r_i\in\mathcal{R}_w} N_{i,w} = N_i$ and $\cup_{w=1}^{W}\mathcal{R}_w = \mathcal{R}$, and we let $N = \sum_{i:r_i\in\mathcal{R}} N_i$ denote the total number of tasks held by requesters in an MCS system.

In this paper, we consider both *static* MCS systems where workers arrive all at once, and *dynamic* MCS systems where workers arrive dynamically in an online manner. The primary goal of this paper is to study *how the platform, as an entity who has access to system-wide information, could set reasonable prices for requesters in dynamic MCS systems.* However, as models, solutions, and analyses for static MCS systems serve as preliminaries for and shed lights upon their counterparts in dynamic MCS systems, they are also discussed in this paper.

B. Static MCS System

1) System Model: In our model of a static MCS system, there are overall M workers arriving at the system all at once, out of which the number of each type-w workers is M_w. Each requester r_i posts a price p_i, which is the amount of payment that a worker gets for executing one of the requester's tasks. We adopt the following discrete choice model given in [31] to characterize workers' task selection behaviors.

Definition 1 (Task Selection Probability)**.** *Given $\mathbf{p}_w$, which denotes the vector of prices proposed by requesters in the set $\mathcal{R}_w$, the probability that a type-w worker chooses a task from requester $r_i \in \mathcal{R}_w$ is defined as*

$$\theta_{i,w}(\mathbf{p}_w) = \frac{\exp(a_{i,w}p_i - b_{i,w})}{\sum_{j:r_j \in \mathcal{R}_w} \exp(a_{j,w}p_j - b_{j,w})}, \tag{1}$$

where $a_{i,w}$ and $b_{i,w}$ for each $r_i \in \mathcal{R}_w$ are positive parameters.

As indicated by [31] that parameters $a_{i,w}$'s and $b_{i,w}$'s could be estimated using statistics of workers' historical choices, we thus assume them to be known by the platform. Such discrete choice model perfectly captures workers' task selection behavior in various aspects. On one hand, the probability $\theta_{i,w}(\mathbf{p}_w)$ monotonically increases with p_i, which conforms to the intuition that the higher the price a requester proposes, the more likely that her tasks will be chosen by workers. On the other hand, Equation (1) incorporates the randomness in workers' task selection caused by factors (e.g., age, gender) other than prices which could also affect their choices.

In this paper, we consider that requesters hold delay-sensitive tasks (e.g., traffic speed estimation, air quality monitoring), and thus a penalty will be incurred to a requester if any of her tasks is not chosen by participating workers. We let $\alpha_i \geq 0$ be the penalty to requester r_i for one unallocated task. Next, we define a requester's cost in Definition 2.

Definition 2 (Requester's Cost in Static MCS System)**.** *Given $\mathbf{p}$, the vector of prices proposed by all requesters, requester r_i's overall expected cost can be defined as*

$$C_i(\mathbf{p}) = \sum_{\mathbf{m}\in\mathcal{M}_i} \phi(\mathbf{m},\mathbf{p}) \Bigg(\sum_{w:r_i\in\mathcal{R}_w} \min\{m_w, N_{i,w}\}p_i + \alpha_i \Bigg(N_{i,w} - \sum_{w:i\in\mathcal{G}_w} \min\{m_w, N_{i,w}\} \Bigg) \Bigg)$$

where $\mathcal{M}_i = \prod_{w:r_i\in\mathcal{R}_w}\{0,1,\cdots,M_w\}$ is the space comprised by the product of each set $\{0,1,\cdots,M_w\}$ such that $r_i \in \mathcal{R}_w$, and $\phi(\mathbf{m},\mathbf{p}) = \prod_{w:r_i\in\mathcal{R}_w} \binom{M_w}{m_w} \big(\theta_{i,w}(\mathbf{p}_w)\big)^{m_w} \big(1-\theta_{i,w}(\mathbf{p}_w)\big)^{M_w-m_w}$ denotes the probability that task selection profile $\mathbf{m} \in \mathcal{M}_i$ happens under price vector $\mathbf{p}$.

By Definition 2, a requester's cost consists of the payments to workers who execute her tasks, as well as the penalty incurred by the unallocated tasks.

2) Game Theoretic Model: As given in Definition 2, the cost of a requester depends on all requesters' prices, and thus, her pricing decision will inevitably be affected by others'. Therefore, we use a *static (one-shot) non-cooperative game* with requesters as the players and their prices as the actions to characterize requesters' pricing behaviors. In this game, each requester r_i aims to choose a price p_i that minimizes her own cost $C_i(p_i, \mathbf{p}_{-i})$, while considering other requesters' prices $\mathbf{p}_{-i} = (p_1,\cdots,p_{i-1},p_{i+1},\cdots,p_K)$. We refer to such game as *static pricing (SP) game* in the rest of this paper.

An individual requester r_i usually knows her own parameters $\mathbf{N}_i$ and α_i, but rarely those of the others. It is thus hard for her to decide her price due to such incomplete information. However, the platform typically has global information about all requesters' parameters, as it can require them to provide such information at the time of registration. Thus, we take the role of the platform, and study how the platform could set reasonable prices for the requesters. To achieve this end, we adopt *correlated equilibrium (CE)* defined in Definition 3.

Definition 3 (Correlated Equilibrium)**.** *Let $\mathcal{P}$ be the set of prices that requesters could propose, which is assumed to be discrete and finite with size P, and $\pi(\cdot)$ be a probability distribution over space $\mathcal{P}^K$. Then, $\pi(\cdot)$ is a correlated equilibrium (CE) of the SP game, iff for each $r_i \in \mathcal{R}$ and price $p_i,\ p_i' \in \mathcal{P}$,*

$$\sum_{\mathbf{p}_{-i}\in\mathcal{P}^{K-1}} \pi(p_i,\mathbf{p}_{-i})\Big(C_i(p_i,\mathbf{p}_{-i}) - C_i(p_i',\mathbf{p}_{-i})\Big) \leq 0. \tag{2}$$

By Definition 3, a CE $\pi(\cdot)$ is a probability distribution over the space of requesters' price combinations. It satisfies that if the platform samples a price vector $\mathbf{p} = (p_i, \mathbf{p}_{-i})$ from $\pi(\cdot)$, and suggests requesters to set their prices according to $\mathbf{p}$, only by setting her price as p_i could each requester r_i minimizes her cost given that other requesters take $\mathbf{p}_{-i}$ as their prices. Due to such *self-enforcing* property, we *propose to set requesters' prices using the price vector sampled from a CE.* Apparently, there could be multiple CE's that satisfy the polytope defined by Inequality (2). Considering the system-wide performance, we are specifically interested in the CE that minimizes the expected social cost.

Definition 4 (Social Cost)**.** *Given a CE $\pi(\cdot)$, an SP game's expected social cost is $\sum_{\mathbf{p}\in\mathcal{P}^K} \pi(\mathbf{p})\big(\sum_{i:r_i\in\mathcal{R}} C_i(\mathbf{p})\big)$.*

To summarize, in MCS systems, we aim to *compute the CE of the SP game which minimizes the expected social cost.*

C. Dynamic MCS Systems

1) System Overview: As shown in Figure 1, in a dynamic MCS system, we discretize the time line into multiple time slots, denoted as $t = 0,1,\cdots,T$, and workers arrive dynamically in an online manner (workers' Step ②, ④, ⑥). In order to simplify presentation[1], we assume that worker arrival follows a Bernoulli process with parameter λ. That is, in each time slot, with probability λ there is one worker arrival, and with probability $1-\lambda$ there is no worker arrival. We also assume that a worker belongs to type w with probability μ_w.

[1]Our models and solutions can be generalized to cases with more complicated worker arrival processes, which however introduce unnecessary complications in presentation. Thus, we only consider Bernoulli arrival process.

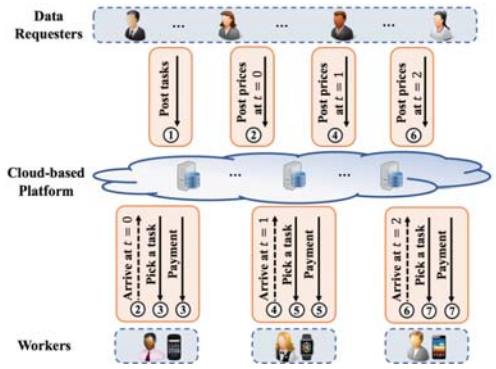

Fig. 1: Framework of a dynamic MCS system (where circled numbers represent the order of the events).

Instead of a one-shot price, each requester proposes a price in each time slot (requesters' Step ②, ④, ⑥). Her task is then selected according to the same probabilistic model as given in Definition 1. Requesters also hold delay-sensitive tasks in dynamic MCS systems, and any unallocated task at time slot T will incur a penalty α_i to requester r_i.

2) Game Theoretic Model: In such dynamic setting, we use the following *Markov game*, which we refer to as *dynamic pricing (DP) game* and define in Definition 5, to characterize requesters' pricing behaviors.

Definition 5 (Dynamic Pricing Game)**.** *A dynamic pricing (DP) game is a Markov game with the following components.*

- ***Player****: A DP game has the set of requesters $\mathcal{R}$ as players.*
- ***State****: A DP game has a series of states $(\mathbf{n}_1, \mathbf{n}_2, \cdots, \mathbf{n}_K, t)$, where t denotes the current time slot, and $\mathbf{n}_i$ denotes the vector containing the number of each type of requester r_i's unallocated tasks at time t. We let $\mathbf{n} = (\mathbf{n}_1, \mathbf{n}_2, \cdots, \mathbf{n}_K)$, and $\mathbf{N} = (\mathbf{N}_1, \mathbf{N}_2, \cdots, \mathbf{N}_K)$. Thus, a DP game starts at state $(\mathbf{N}, 1)$, and terminates at any state such that $t = T$ which is referred to as a terminal state.*
- ***Action****: At any state, each requester r_i's action is the task price p_i that she proposes.*
- ***State transition****: In a DP game, requesters' joint action profile $\mathbf{p}$ and workers' task selection jointly affect state transition. We let $\mathcal{I}_w$ denote the set of requesters that have type-w unallocated tasks at state $(\mathbf{n}, t)$. At any state such that $t \in \{0, 1, \cdots, T-1\}$, state transition follows the probabilistic model such that for each (i, w) with $r_i \in \mathcal{R}_w \cap \mathcal{I}_w$,*

$$\Pr\big[(n_{i,w} - 1, \mathbf{n}_{-(i,w)}, t+1)\big|(\mathbf{n}, t), \mathbf{p}\big] = \lambda \mu_w \theta_{i,w}(\mathbf{p}_w),$$

and otherwise we have

$$\Pr\big[(\mathbf{n}, t+1)\big|(\mathbf{n}, t), \mathbf{p}\big] = 1 - \sum_{w=1}^{W} \sum_{i: r_i \in \mathcal{R}_w \cap \mathcal{I}_w} \lambda \mu_w \theta_{i,w}(\mathbf{p}_w),$$

where $\mathbf{n}_{-(i,w)}$ denotes the vector obtained by excluding the element $n_{i,w}$ from $\mathbf{n}$.

- ***Immediate cost****: At every state, after requesters propose their prices $\mathbf{p}$, each requester r_i will experience an immediate cost such that for each $t \in \{0, 1, \cdots, T-2\}$,*

$$c_i\big((\mathbf{n}', t+1), (\mathbf{n}, t), p_i\big) = \sum_{w: r_i \in \mathcal{R}_w} (n_{i,w} - n'_{i,w}) p_i,$$

and for $t = T-1$,

$$c_i\big((\mathbf{n}', T), (\mathbf{n}, T-1), p_i\big) = \sum_{w: r_i \in \mathcal{R}_w} \Big((n_{i,w} - n'_{i,w}) p_i + n'_{i,w} \alpha_i\Big).$$

As given in Definition 5, the state of a DP game captures the various aspects that affect requesters' pricing decisions, including the number of remaining tasks $\mathbf{n}$, and the current time slot index t. Intuitively, a requester may want to increase her price, if she still has a large number of unallocated tasks near the end of the time line. Under our assumption that requesters are fully rational, at any state, a requester will propose the price that minimizes her cumulative expected cost from the time slot onwards. As in an SP game, we also *propose to use CE to coordinate requesters' prices*. However, the CE adopted in our DP game is the *Markov correlated equilibrium (MCE)* defined in Definition 6.

Definition 6 (Markov Correlated Equilibrium)**.** *Let $\pi(\cdot|\cdot)$ be a conditional probability distribution over space $\mathcal{P}^K$ conditioned on the state of a DP game. Then, $\pi(\cdot|\cdot)$ is a Markov correlated equilibrium (MCE), iff at each non-terminal state $(\mathbf{n}, t)$, and for each $r_i \in \mathcal{R}$, and price $p_i,\ p'_i \in \mathcal{P}$,*

$$\sum_{\mathbf{p}_{-i} \in \mathcal{P}^{K-1}} \pi\big(p_i, \mathbf{p}_{-i}|(\mathbf{n}, t)\big)\Big(C_i\big((\mathbf{n}, t), p_i, \mathbf{p}_{-i}\big) - C_i\big((\mathbf{n}, t), p'_i, \mathbf{p}_{-i}\big)\Big) \le 0, \tag{3}$$

where $C_i\big((\mathbf{n}, t), \mathbf{p}\big)$ denotes requester r_i's cumulative cost from state $(\mathbf{n}, t)$ to the terminal states (i.e., states with $t = T$). For $t \in \{0, 1, \cdots, T-2\}$, we define $C_i\big((\mathbf{n}, t), \mathbf{p}\big)$ as

$$C_i\big((\mathbf{n}, t), \mathbf{p}\big) = \sum_{\mathbf{n}'} \Pr\big[(\mathbf{n}', t+1)\big|(\mathbf{n}, t), \mathbf{p}\big]\Big(c_i\big((\mathbf{n}', t+1), (\mathbf{n}, t), p_i\big) + \sum_{\mathbf{p}' \in \mathcal{P}^K} \pi\big(\mathbf{p}'|(\mathbf{n}', t+1)\big) C_i\big((\mathbf{n}', t+1), \mathbf{p}'\big)\Big),$$

and for $t = T-1$, we have that

$$C_i\big((\mathbf{n}, T-1), \mathbf{p}\big) = \sum_{\mathbf{n}'} \Pr\big[(\mathbf{n}', T)\big|(\mathbf{n}, T-1), \mathbf{p}\big] c_i\big((\mathbf{n}', T), (\mathbf{n}, T-1), p_i\big).$$

Similar to the static case, at each state of a DP game, the platform samples a price vector $\mathbf{p}$ from the MCE $\pi(\cdot|\cdot)$, and suggests requesters to set their prices according to $\mathbf{p}$. We have inherently assumed in Definition 6 that, at each state, requesters will take the platform's pricing suggestions due to the CE condition. Again, similar to the SP game, the platform is interested in calculating the MCE that minimizes the social cost, which is defined in Definition 7, at every state.

Definition 7 (Social Cost in DP Game)**.** *Given an MCE $\pi(\cdot|\cdot)$ of a DP game, at any state $(\mathbf{n}, t)$, the expected social cost is represented as $\sum_{\mathbf{p} \in \mathcal{P}^K} \pi\big(\mathbf{p}|(\mathbf{n}, t)\big)\Big(\sum_{i: r_i \in \mathcal{R}} C_i\big((\mathbf{n}, t), \mathbf{p}\big)\Big)$.*

To summarize, in dynamic MCS systems, our goal is to *calculate the MCE of the DP game which minimizes the expected social cost at every state.*

IV. Task Pricing in Static MCS Systems

In this section, we present our formal mathematical formulation for computing the social cost-minimizing CE in SP games, as well as our solution methods and the corresponding analyses.

A. Problem Formulation

As discussed in Section III-B, we model requesters' task pricing in a static MCS system as an SP game. Next, we formulate the problem of calculating the CE that minimizes the social cost, which we refer to as the *SP-CE problem*, as the following linear optimization program.

SP-CE Problem:

$$\min \sum_{\mathbf{p}\in\mathcal{P}^K} \pi(\mathbf{p}) \Bigg(\sum_{i:r_i\in\mathcal{R}} C_i(\mathbf{p}) \Bigg) \tag{4}$$

$$\text{s.t.} \sum_{\mathbf{p}_{-i}\in\mathcal{P}^{K-1}} \pi(p_i,\mathbf{p}_{-i}) \Big(C_i(p_i,\mathbf{p}_{-i}) - C_i(p_i',\mathbf{p}_{-i}) \Big) \le 0,\ \forall r_i\in\mathcal{R},\ p_i,\ p_i'\in\mathcal{P}, \tag{5}$$

$$\sum_{\mathbf{p}\in\mathcal{P}^K} \pi(\mathbf{p}) = 1, \tag{6}$$

$$\pi(\mathbf{p}) \ge 0,\ \forall \mathbf{p}\in\mathcal{P}^K. \tag{7}$$

The SP-CE problem has continuous non-negative variables $\pi(\mathbf{p})$ for each $\mathbf{p}\in\mathcal{P}^K$, and all the other parameters of an SP game that are involved in this formulation are *a priori* known by the platform, and thus are treated as constants. As indicated by Objective Function (4), the goal of this optimization program is to minimize the expected social cost defined in Definition 4. In terms of the constraints, Constraint (5) is exactly the CE condition given in Definition 3, and Constraint (6) and (7) ensure that any feasible solution $\pi(\cdot)$ to the SP-CE problem is a probability distribution.

Obviously, SP-CE is a linear programming problem, which can be solved in time that is polynomial in the problem's input size using existing methods [32], including the interior-point, simplex, and ellipsoid algorithm[2], which we will collectively refer to as *ISE algorithms* in the rest of the paper. As we will show in the following Lemma 1, directly solving the SP-CE problem with current methods will incur excessively high computational complexity.

Lemma 1. *ISE algorithms have exponential computational complexity in the number of requesters K for the SP-CE problem.*

Proof. As aforementioned, the SP-CE problem has $\pi(\mathbf{p})$ for each $\mathbf{p}\in\mathcal{P}^K$ as variables. Thus, the number of variables of the problem is P^K. Furthermore, Inequality (5) corresponds to KP^2 constraints, and Inequality (7) is defined on each variable $\pi(\mathbf{p})$, and thus corresponds to P^K constraints. Therefore, the SP-CE problem has overall $O(P^K)$ constraints.

As indicated in [32], ISE algorithms have polynomial computational complexity with respect to the number of variables and constraints of a linear program. However, in the SP-CE problem, as we have discussed in the previous paragraph, the number of variables and constraints are both in the order of $O(P^K)$, which grow exponentially with the number of requesters. Therefore, existing solution methods have exponential computational complexity for the SP-CE problem in terms of the number of requesters K. □

In practice, there are typically a large number of requesters in an MCS system, and thus existing linear programming solution methods, which have exponential computational complexity in the number of requesters are not suitable for the SP-CE problem. Next, in Section IV-B, we propose our own method for solving the SP-CE problem in a computationally efficient manner.

B. Computationally Efficient Solution Method

Our intuition of removing the exponential relationship between the computational complexity and the number of requesters is to divide the SP game into several smaller games, each of which has much fewer requesters. Specifically, we break the SP game into W *sub-static pricing (SSP) games*[3], out of which SSP game w for each $w\in\{1,2,\cdots,W\}$ has the set of requesters $\mathcal{R}_w$ as the players, and each requester r_i in the game has $N_{i,w}$ tasks to be executed. Thus, based on such definition, requester r_i belongs to every SSP game w such that $r_i\in\mathcal{R}_w$, and in each SSP game that the requester belongs to, she proposes a separate price to compete with other requesters in the same SSP game.

In each SSP game w that requester r_i participates in, she has a cost

$$C_{i,w}(\mathbf{p}_w) = \sum_{m_w=0}^{M_w} \psi(m_w,\mathbf{p}_w) \Big(\min\{m_w,N_{i,w}\}p_{i,w} + \alpha_i \big(N_{i,w} - \min\{m_w,N_{i,w}\} \big) \Big),$$

where we use $\psi(m_w,\mathbf{p}_w) = \binom{M_w}{m_w} \big(\theta_{i,w}(\mathbf{p}_w)\big)^{m_w} \big(1-\theta_{i,w}(\mathbf{p}_w)\big)^{M_w-m_w}$ to denote the probability that m_w out of the M_w type-w workers choose requester r_i's tasks.

Then, in each SSP game w, the social cost minimization problem, referred to as the *SSP-CE*(w) *problem*, is defined as the following linear program, where $K_w=|\mathcal{R}_w|$ and $\mathbf{p}_{-(i,w)}$ denotes the price vector obtained by removing the element $p_{i,w}$ from $\mathbf{p}_w$.

SSP-CE(w) **Problem:**

$$\min \sum_{\mathbf{p}_w\in\mathcal{P}^{K_w}} \pi_w(\mathbf{p}_w) \Bigg(\sum_{i:r_i\in\mathcal{R}_w} C_{i,w}(\mathbf{p}_w) \Bigg) \tag{8}$$

$$\text{s.t.} \sum_{\mathbf{p}_{-(i,w)}\in\mathcal{P}^{K_w-1}} \pi_w\big(p_{i,w},\mathbf{p}_{-(i,w)}\big) \Big(C_{i,w}\big(p_{i,w},\mathbf{p}_{-(i,w)}\big) - C_{i,w}\big(p_{i,w}',\mathbf{p}_{-(i,w)}\big) \Big) \le 0,\ \forall r_i\in\mathcal{R}_w,\ p_{i,w},\ p_{i,w}'\in\mathcal{P}, \tag{9}$$

[2]Although there exist other solution methods for linear programming, we will treat the interior-point, simplex, and ellipsoid algorithm as state-of-the-art approaches, because they already have good enough performance.

[3]Note that the concept of sub-static pricing game is not exactly the same with that of a subgame defined in traditional game theory literatures [14].

$$\sum_{\mathbf{p}_w \in \mathcal{P}^{K_w}} \pi_w(\mathbf{p}_w) = 1, \tag{10}$$

$$\pi_w(\mathbf{p}_w) \geq 0, \ \forall \mathbf{p}_w \in \mathcal{P}^{K_w}. \tag{11}$$

The SSP-CE(w) problem is the exact counterpart of the SP-CE problem with a smaller number of requesters. Thus, we will skip the detailed explanation of this linear optimization program, but point out that the objective of the SSP-CE(w) problem is to solve the CE $\pi_w(\cdot)$ of SSP game w that minimizes the game's social cost. Next, we introduce in Algorithm 1 our *computationally efficient CE calculation (CE2C) algorithm*.

Algorithm 1 takes as inputs all constants that are needed to represent each SSP-CE(w) problem, and solves each problem using any of the ISE algorithms (line 2). Finally, the algorithm returns the CE $\pi_w(\cdot)$ for each SSP game w (line 3). Although the CE returned by the CE2C algorithm $\big(\pi_1(\cdot), \pi_2(\cdot), \cdots, \pi_W(\cdot)\big)$ will be different from the one obtained by solving the SP-CE problem directly with ISE algorithms, we will demonstrate numerically in Section VI that the two CE's yield approximately the same social cost in practice. Next, in Theorem 1, we analyze the computational complexity of the CE2C algorithm given in Algorithm 1.

Algorithm 1: CE2C Algorithm

Input: Instances of each SSP-CE(w) problem for each SSP game $w \in \{1, 2, \cdots, W\}$;
Output: $\big(\pi_1(\cdot), \pi_2(\cdot), \cdots, \pi_W(\cdot)\big)$;
1 **foreach** $w = 1, 2, \cdots, W$ **do**
2 solve the SSP-CE(w) problem for SSP game w using any of the ISE algorithms to obtain $\pi_w(\cdot)$;
3 **return** $\big(\pi_1(\cdot), \pi_2(\cdot), \cdots, \pi_W(\cdot)\big)$;

Theorem 1. *On the CE2C algorithm's computational complexity, we have the following two results:*

- *The CE2C algorithm has a polynomial computational complexity in the number of requesters K and the number of price choices P, under the assumption that the number of SSP games W is polynomial in K, and the maximum number of requesters in each SSP game, denoted as $K^* = \max_{w \in \{1,2,\cdots,W\}} K_w$ is much less than K.*
- *The CE2C algorithm's computational complexity does not depend on M_w's, $N_{i,w}$'s, N_i's, M, as well as N.*

Proof. In each SSP-CE(w) problem, there are P^{K_w} variables, and $O(P^{K_w})$ constraints. Thus, the computational complexity of using any ISE algorithm to solve an SSP-CE(w) problem is polynomial in P. Due to our assumption that the maximum number of requesters in each SSP game, denoted as K^* is much less than K, and by our analysis given in the proof of Lemma 1, any ISE algorithm could be treated as having a constant computational complexity with respect to K.

Furthermore, as the main loop of Algorithm 1 is executed for each $w \in \{1, 2, \cdots, W\}$, and W is assumed to be polynomial in K, the overall computational complexity of the CE2C algorithm is polynomial in the number of requesters K and the number of price choices P. Furthermore, it is fairly obvious that the number of iterations, and the size of the optimization problem to be solved in each iteration do not depend on the number of tasks and workers, and thus related parameters including the M_w's, $N_{i,w}$'s, N_i's, M, and N do not affect the algorithm's computational complexity. □

In many real-world scenarios, there are only a small number of requesters, usually no greater than 3, in each set $\mathcal{R}_w$. For example, in most countries, there are usually no more than 3 dominating map services (e.g., Google, Baidu, and Gaode maps), and thus the number of requesters interested in real-time traffic congestion information is typically less than 3. This justifies our assumption in Theorem 1 that $K^* \ll K$. Furthermore, although the computational complexity of the CE2C algorithm is exponential in K^*, it can still be regarded as computationally efficient, because K^* could typically be treated as a small constant integer less than or equal to 3.

V. Task Pricing in Dynamic MCS Systems

In this section, we present our formulation for computing the social cost-minimizing MCE, as well as our computationally efficient solution method.

A. Problem Formulation

As discussed in Section III-C, we model requesters' task pricing in a dynamic MCS system as a DP game. To further simplify our presentation, we use $s = (\mathbf{n}, t)$ to denote a state of a DP game. Next, we formulate the problem of calculating the MCE that minimizes the social cost at state s, which we refer to as the DP-MCE(s) problem, as the following linear optimization program.

DP-MCE(s) Problem:

$$\min \sum_{\mathbf{p} \in \mathcal{P}^K} \pi(\mathbf{p}|s) \left(\sum_{i: r_i \in \mathcal{R}} C_i\big(s, \mathbf{p}\big) \right) \tag{12}$$

$$\text{s.t.} \sum_{\mathbf{p}_{-i} \in \mathcal{P}^{K-1}} \pi\big(p_i, \mathbf{p}_{-i}|s\big) \Big(C_i\big(s, p_i, \mathbf{p}_{-i}\big) - C_i\big(s, p_i', \mathbf{p}_{-i}\big) \Big) \leq 0, \ \forall r_i \in \mathcal{R}, \ p_i, \ p_i' \in \mathcal{P}, \tag{13}$$

$$\sum_{\mathbf{p} \in \mathcal{P}^K} \pi(\mathbf{p}|s) = 1, \tag{14}$$

$$\pi(\mathbf{p}|s) \geq 0, \ \forall \mathbf{p} \in \mathcal{P}^K. \tag{15}$$

Similar to the SP-CE problem, the DP-MCE(s) problem has $\pi(\mathbf{p}|s)$ for each $\mathbf{p} \in \mathcal{P}^K$ as variables, and all the other parameters of a DP game used in this formulation are assumed to be known constants. The goal of this optimization program, as indicated by Objective Function (12), is to minimize the cumulative social cost from time slot t onwards, and Constraint (13), (14), and (15) ensure that any feasible solution to the DP-MCE(s) problem is an MCE at state s.

Next, in Lemma 2, we show that the social cost-minimizing MCE of a DP game is computationally intractable to calculate.

Lemma 2. *The computational complexity of calculating the social cost-minimizing MCE of a DP game is exponential in the number of requesters K.*

Proof. It is obvious that the above linear programming problem DP-MCE(s) for each state s has $O(P^K)$ variables and constraints, respectively. Therefore, computing a DP-MCE(s) problem using any of the ISE algorithms will take an exponential time in terms of the number of requesters K.

Furthermore, computing the social cost-minimizing MCE of a DP game will inevitably involve solving the DP-MCE(s) problem at each state s of the DP game. However, based on our definition, a DP game has $\Omega(2^K)$ states, and calculating the MCE that minimizes the social cost thus has an exponential computational complexity with respect to the number of requesters K. □

B. Computationally Efficient Solution Method

Because of Lemma 2, we design a computationally efficient method for solving the social cost-minimizing MCE. Enlightened by the method to reduce the computational complexity of deriving the social cost-minimizing CE of an SP game, we also take the approach of splitting the overall game into several smaller games with fewer numbers of requesters. Specifically, we break a DP game into W *sub-dynamic pricing (SDP) games*[4]. The set of players for each SDP game w is $\mathcal{R}_w$, and each requester r_i in SDP game w holds $N_{i,w}$ tasks to be executed at time slot $t = 0$. Based on such definition, the overall worker arrival process could be separated into W independent processes, such that each SDP game w has a Bernoulli worker arrival process with parameter $\lambda\mu_w$.

In an SDP game w, a state is defined as $(\mathbf{n}_w, t)$, where $\mathbf{n}_w$ denotes the vector containing the number of unallocated tasks that belong to all requesters in SDP game w at time slot t. Furthermore, an SDP game w has the following probabilistic state transition model such that at any state with $t \in \{0, 1, \cdots, T-1\}$, and for each (i, w) with $r_i \in \mathcal{R}_w \cap \mathcal{I}_w$,

$$\Pr\big[(n_{i,w}-1, \mathbf{n}_{-(i,w)}, t+1)\big|(\mathbf{n}_w, t), \mathbf{p}_w\big] = \lambda\mu_w\theta_{i,w}(\mathbf{p}_w),$$

and otherwise we have

$$\Pr\big[(\mathbf{n}_w, t+1)\big|(\mathbf{n}_w, t), \mathbf{p}_w\big] = 1 - \sum_{i:r_i\in\mathcal{R}_w\cap\mathcal{I}_w} \lambda\mu_w\theta_{i,w}(\mathbf{p}_w),$$

where we abuse the notation a little, and use $\mathbf{n}_{-(i,w)}$ to denote the vector obtained by excluding the element $n_{i,w}$ from $\mathbf{n}_w$. At every state of an SDP game w, for each $t \in \{0, 1, \cdots, T-2\}$, each requester r_i will experience an immediate cost such that

$$c_{i,w}\big((\mathbf{n}'_w, t+1), (\mathbf{n}_w, t), p_{i,w}\big) = (n_{i,w} - n'_{i,w})p_{i,w},$$

and for $t = T-1$,

$$c_i\big((\mathbf{n}'_w, T), (\mathbf{n}_w, T-1), p_{i,w}\big) = (n_{i,w} - n'_{i,w})p_{i,w} + n'_{i,w}\alpha_i.$$

Our definitions of each SDP game w's MCE, denoted as $\pi_w(\cdot|\cdot)$, and a requester r_i's cumulative cost from any state $(\mathbf{n}_w, t)$ onwards, denoted as $C_{i,w}\big((\mathbf{n}_w, t), \mathbf{p}_w\big)$, are the same as Definition 6 except that we substitute the state transition probability and the immediate cost function as those of SDP game w. Thus, we omit the detailed definitions of them, and directly provide our formal mathematical formulation of calculating the social cost-minimizing MCE of SDP game w at each state $s_w = (\mathbf{n}_w, t)$ as the following linear optimization program, which we refer to as the SDP-MCE(s_w) problem.

SDP-MCE(s_w) Problem:

$$\min \sum_{\mathbf{p}_w\in\mathcal{P}^{K_w}} \pi_w(\mathbf{p}_w|s_w)\left(\sum_{i:r_i\in\mathcal{R}} C_{i,w}\big(s_w, \mathbf{p}_w\big)\right) \tag{16}$$

$$\text{s.t.} \sum_{\mathbf{p}_{-(i,w)}\in\mathcal{P}^{K_w-1}} \pi_w\big(p_{i,w}, \mathbf{p}_{-(i,w)}|s_w\big)\Big(C_{i,w}\big(s_w, p_{i,w}, \mathbf{p}_{-(i,w)}\big) - C_{i,w}\big(s_w, p'_{i,w}, \mathbf{p}_{-(i,w)}\big)\Big) \le 0, \forall r_i \in \mathcal{R}_w, p_{i,w}, p'_{i,w} \in \mathcal{P}, \tag{17}$$

$$\sum_{\mathbf{p}_w\in\mathcal{P}^{K_w}} \pi_w(\mathbf{p}_w|s_w) = 1, \tag{18}$$

$$\pi_w(\mathbf{p}_w|s_w) \ge 0, \ \forall \mathbf{p}_w \in \mathcal{P}^{K_w}. \tag{19}$$

We skip again the detailed explanation of the SDP-MCE(s_w) problem, because it is a variant of the DP-MCE(s) problem defined specifically over SDP game w. Based on our definition of requester r_i's cumulative cost function $C_{i,w}\big((\mathbf{n}_w, t), \mathbf{p}_w\big)$ in SDP game w, the MCE $\pi\big(\cdot\,|(\mathbf{n}_w, t)\big)$ can be calculated by incorporating the MCE of related states at time slot $t+1$. Therefore, we adopt the technique of dynamic programming, and compute the MCE's backward from $t = T-1$. Next, we present in the following Algorithm 2 our *computationally efficient MCE calculation (MCE2C) algorithm*.

Algorithm 2: MCE2C Algorithm

Input: Instances of each SDP-MCE(s_w) problem for each state s_w with $t = T-1$ of each SDP game $w \in \{1, 2, \cdots, W\}$;
Output: $\big(\pi_1(\cdot|\cdot), \pi_2(\cdot|\cdot), \cdots, \pi_W(\cdot|\cdot)\big)$;

```
1 foreach w = 1, 2, ..., W do
2     foreach t = T-1, T-2, ..., 0 do
3         foreach state s_w at time slot t do
4             solve the SDP-MCE(s_w) problem using any of
              the ISE algorithms to obtain π_w(·|s_w);
              // store the constants for
                 SDP-MCE(s_w) at time slot t-1
5             store C_{i,w}(s_w, p_w) for each r_i ∈ R_w;
              // store the outputs
6             store π_w(·|s_w);
7 return (π_1(·|·), π_2(·|·), ..., π_W(·|·));
```

The MCE2C Algorithm given in Algorithm 2 takes as inputs the instances of each SDP-MCE(s_w) problem for each state s_w with $t = T-1$ of each SDP game $w \in \{1, 2, \cdots, W\}$. In each iteration, it solves each SDP-MCE(s_w) problem using any of the ISE algorithms (line 4), and it then stores the value $C_{i,w}(s_w, \mathbf{p}_w)$ for each $r_i \in \mathcal{R}_w$ (line 5) and $\pi_w(\cdot|s_w)$ (line 6) for the computation in the next iteration. Note that the computation starts from states with $t = T-1$, because at the very beginning only instances for those states are *a priori* known. As the iterations proceed, the problem instances for other states become available, and the corresponding SDP-MCE(s_w) problems will then be computed accordingly.

As shown in Algorithm 2, our approach of splitting a DP game into several SDP games could significantly reduce the

[4]Again, the concept of sub-dynamic pricing game is also different from the traditional definition of a subgame [14].

number of states in each game, as well as the size of the optimization problem corresponding to each state, which is why our method is fairly computationally efficient. Next, in Theorem 2, we analyze the MCE2C Algorithm's computational complexity. We let $N^* = \max_{i:r_i \in \mathcal{R}} N_i$ denote the maximum number of tasks held by requesters.

Theorem 2. *If the number of SDP games W is polynomial in K, and the maximum number of requesters in each SDP game K^* is much less than K, the MCE2C algorithm has polynomial computational complexity in K, T, P, and N^*.*

Proof. In each iteration of Algorithm 2, the computational complexity is dominated by solving the SDP-MCE(s_w) problem, which has $O(P^{K_w})$ variables and constraints. Due to the assumption that K^* is much less than K, we can treat the computational complexity of computing the SDP-MCE(s_w) problem with any of the ISE algorithms in each iteration as polynomial in P and constant with respect to K.

Furthermore, due to the three levels of loops in Algorithm 2, the total number of iterations is $O\big(WT(N^*)^{K^*}\big)$. Together with our assumption that the number of task and worker types W is polynomial in the number of requesters K, we can arrive at the conclusion that the MCE2C algorithm has polynomial computational complexity in K, T, P, and N^*. □

By Theorem 2, the MCE2C algorithm has a polynomial computational complexity in almost all of the primary parameters of an SDP game, including K, T, P, and N^*, except that it has an exponential computational complexity in K^*. As discussed in Section IV-B, K^* is usually practically a small constant, and thus the MCE2C algorithm could be regarded as computationally efficient in practice. Next, in Section VI, we will numerically evaluate the expected cumulative social cost guaranteed by the MCE's returned by the MCE2C algorithm.

VI. Performance Evaluation

In this section, we introduce the baseline methods, as well as the settings and results of our numerical evaluation.

A. Baseline Methods

In the case of static MCS systems, we compare CE2C with using an ISE algorithm to solve the SP-CE problem. In our implementation, we use the interior-point algorithm [32], as it has the lowest theoretical computational complexity among the three ISE algorithms. Furthermore, we also consider another baseline method, which returns not the social cost-minimizing CE, but simply any CE of each SSP-CE(w) problem. We refer to such baseline method as the *ACE method*. Similarly, in the case of dynamic MCS systems, we compare our MCE2C algorithm with using the interior-point algorithm to solve the DP-MCE(s) problem at each state s, and the *AMCE method* that returns not necessarily the MCE that minimizes the social cost, but any MCE of each SDP-MCE(s_w) problem.

B. Evaluation Settings

Our evaluation settings are given in Table I and II, where the set of requesters' price choices is set as $\mathcal{P} = \{1, 20, 40\}$, and K_w is chosen uniformly at random from the set $\{1, 2, 3\}$. In Table I, we present our parameter settings for static MCS systems. We skip the descriptions of the self-explanatory parts, and primarily explain the following points. Setting I has relatively small problem sizes, whereas Setting II-V have larger problem instances. In Setting I and II, we fix the other parameters and vary the number of requesters, and in Setting III and IV, we vary respectively the number of workers M and the number of SSP games W. Different from Setting I-IV where each requester r_i's penalty α_i is sampled uniformly at random from $[40, 60]$, in Setting V, we let all requesters have the same penalty α, and vary α from 50 to 110. Similarly, in Table II, we present our parameter settings for dynamic MCS systems, the descriptions of which are omitted, as they are direct counterparts of the settings in Table I. Moreover, the optimizations are computed with the GUROBI solver [33].

Setting	K	$\mathcal{P}$	α_i	W	$N_{i,w}$	M	K_w
I	$[7, 10]$	$\{1, 20, 40\}$	$[40, 60]$	3	$[1, 5]$	18	$[1, 3]$
II	$[300, 1200]$	$\{1, 20, 40\}$	$[40, 60]$	$\lceil \frac{K}{2} \rceil$	$[10, 15]$	$15W$	$[1, 3]$
III	600	$\{1, 20, 40\}$	$[40, 60]$	$\lceil \frac{K}{2} \rceil$	$[10, 15]$	$[1000, 1600]$	$[1, 3]$
IV	600	$\{1, 20, 40\}$	$[40, 60]$	$[100, 250]$	$[10, 15]$	$15W$	$[1, 3]$
V	600	$\{1, 20, 40\}$	$[50, 110]$	$\lceil \frac{K}{2} \rceil$	$[10, 15]$	$15W$	$[1, 3]$

TABLE I: Parameter Setting I-V

Setting	K	$\mathcal{P}$	T	λ	α_i	W	$N_{i,w}$	K_w
VI	$[2, 5]$	$\{1, 20, 40\}$	20	0.8	$[40, 60]$	3	$[1, 3]$	$[1, 3]$
VII	$[30, 120]$	$\{1, 20, 40\}$	$2K$	0.8	$[40, 60]$	$\lceil \frac{K}{2} \rceil$	$[10, 15]$	$[1, 3]$
VIII	40	$\{1, 20, 40\}$	$[60, 120]$	0.8	$[40, 60]$	$\lceil \frac{K}{2} \rceil$	$[10, 15]$	$[1, 3]$
IX	100	$\{1, 20, 40\}$	200	$[0.4, 1.0]$	$[40, 60]$	$\lceil \frac{K}{2} \rceil$	$[10, 15]$	$[1, 3]$
X	100	$\{1, 20, 40\}$	200	0.8	$[50, 80]$	$\lceil \frac{K}{2} \rceil$	$[10, 15]$	$[1, 3]$

TABLE II: Parameter Setting VI-X

C. Evaluation Results

Our results are shown in Table III-IV, and Figure 2-7. In Table IV, the time unit is second, and the social costs in Figure 2-5 and Figure 6-7 should be obtained by scaling up the values in the figures by 1000 and 10000 times, respectively.

K	ISE	CE2C	ACE
7	1303.2	1303.2	1601.0
8	1536.2	1536.2	1845.4
9	1785.6	1785.6	2095.2
10	2032.2	2032.2	2344.1

TABLE III: Social cost (I)

K	ISE	CE2C	ACE
7	24.697	0.0957	0.0938
8	218.12	0.0851	0.0727
9	2048.2	0.1227	0.0886
10	24282	0.2836	0.2774

TABLE IV: Running time (I)

Table III and IV, as well as Figure 2-5 show our evaluation results for Setting I-V given in Table I. Specifically, the two tables correspond to Setting I's results, where ISE refers to using an ISE algorithm to solve the SP-CE problem. On one hand, from Table III, we could conclude that CE2C and ISE have exactly the same social cost, which is much less than that yielded by ACE. On the other hand, from Table IV, we can observe that the running time of CE2C is much less than that of ISE, and approximately the same with that of ACE. Thus, these two tables indicate that our CE2C algorithm guarantees low social cost with low computational complexity. In Figure 2-5,

we consider Setting II-IV that have larger problem instances. For these instances, ISE is no longer able to terminate within reasonable time, because, even when $K = 10$, it already takes 24282s to finish. We observe in the experiments that CE2C still terminates fairly fast, and, as shown in these 4 figures, it has much less social cost compared with ACE.

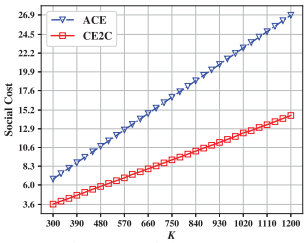

Fig. 2: Social cost (II)

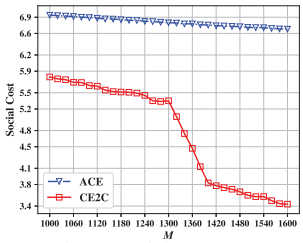

Fig. 3: Social cost (III)

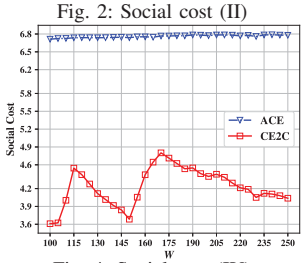

Fig. 4: Social cost (IV)

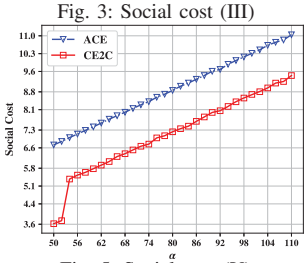

Fig. 5: Social cost (V)

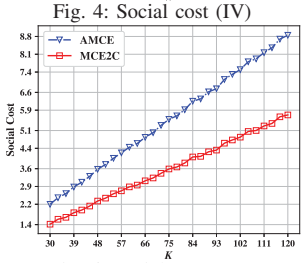

Fig. 6: Social cost (VII)

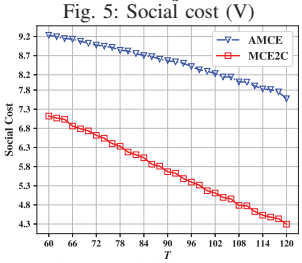

Fig. 7: Social cost (VIII)

Because of space limit, we omit the evaluation results for Setting IV, which show that MCE2C ensures a low social cost in a computationally efficient manner. In Figure 6 and 7, we consider DP games with larger problem instances. These figures show that MCE2C invariably ensures much less social cost than the baseline method AMCE. Again, because of space limit, figures with evaluation results for Setting IX and X are omitted, which show similar trends as Figure 6 and 7.

VII. Conclusion

In this paper, we study task pricing in multi-requester MCS systems with dynamically arriving workers. We use Markov game to model requesters' pricing behaviors, and Markov correlated equilibrium (MCE) as the solution concept. We propose that the platform uses the social cost-minimizing MCE to coordinate requesters' prices, which is self-enforcing, and optimizes the system-wide objective of social cost. Technically, we propose a computationally efficient algorithm to compute an approximately optimal MCE. Through extensive performance evaluation, we show numerically that our algorithm yields close-to-minimum social cost in very short running time.

References

[1] S. Hu, L. Su, H. Liu, H. Wang, and T. F. Abdelzaher, "Smartroad: Smartphone-based crowd sensing for traffic regulator detection and identification," in *TOSN*, 2015.

[2] Y. Cheng, X. Li, Z. Li, S. Jiang, Y. Li, J. Jia, and X. Jiang, "Aircloud: A cloud-based air-quality monitoring system for everyone," in *SenSys*, 2014.

[3] T. Luo, S. S. Kanhere, H. P. Tan, F. Wu, and H. Wu, "Crowdsourcing with tullock contests: A new perspective," in *INFOCOM*, 2015.

[4] D. Peng, F. Wu, and G. Chen, "Pay as how well you do: A quality based incentive mechanism for crowdsensing," in *MobiHoc*, 2015.

[5] H. Jin, L. Su, D. Chen, K. Nahrstedt, and J. Xu, "Quality of information aware incentive mechanisms for mobile crowd sensing systems," in *MobiHoc*, 2015.

[6] H. Jin, L. Su, D. Chen, H. Guo, K. Nahrstedt, and J. Xu, "Thanos: Incentive mechanism with quality awareness for mobile crowd sensing," in *TMC*, 2018.

[7] Y. Wen, J. Shi, Q. Zhang, X. Tian, Z. Huang, H. Yu, Y. Cheng, and X. Shen, "Quality-driven auction-based incentive mechanism for mobile crowd sensing," in *TVT*, 2015.

[8] X. Gong and N. Shroff, "Incentivizing truthful data quality for quality-aware mobile data crowdsourcing," in *Mobihoc*, 2018.

[9] D. Yang, G. Xue, X. Fang, and J. Tang, "Crowdsourcing to smartphones: Incentive mechanism design for mobile phone sensing," in *MobiCom*, 2012.

[10] S. He, D. H. Shin, J. Zhang, and J. Chen, "Toward optimal allocation of location dependent tasks in crowdsensing," in *INFOCOM*, 2014.

[11] Y. Chen, B. Li, and Q. Zhang, "Incentivizing crowdsourcing systems with network effects," in *INFOCOM*, 2016.

[12] Q. Zhang, Y. Wen, X. Tian, X. Gan, and X. Wang, "Incentivize crowd labeling under budget constraint," in *INFOCOM*, 2015.

[13] E. Maskin and J. Tirole, "Markov perfect equilibrium, i: Observable actions," *Journal of Economic Theory*, vol. 100, pp. 191–219, 2001.

[14] N. Nisan, T. Roughgarden, E. Tardos, and V. V. Vazirani, *Algorithmic game theory*. Cambridge University Press, 2007.

[15] H. Zhang, B. Liu, H. Susanto, G. Xue, and T. Sun, "Incentive mechanism for proximity-based mobile crowd service systems," in *INFOCOM*, 2016.

[16] X. Zhang, G. Xue, R. Yu, D. Yang, and J. Tang, "Truthful incentive mechanisms for crowdsourcing," in *INFOCOM*, 2015.

[17] M. H. Cheung, R. Southwell, F. Hou, and J. Huang, "Distributed time-sensitive task selection in mobile crowdsensing," in *MobiHoc*, 2015.

[18] J. Lin, M. Li, D. Yang, G. Xue, and J. Tang, "Sybil-proof incentive mechanisms for crowdsensing," in *INFOCOM*, 2017.

[19] H. Jin, L. Su, and K. Nahrstedt, "Centurion: Incentivizing multi-requester mobile crowd sensing," in *INFOCOM*, 2017.

[20] Y. Wei, Y. Zhu, H. Zhu, Q. Zhang, and G. Xue, "Truthful online double auctions for dynamic mobile crowdsourcing," in *INFOCOM*, 2015.

[21] J. Lin, M. Li, D. Yang, and G. Xue, "Sybil-proof online incentive mechanisms for crowdsensing," in *INFOCOM*, 2018.

[22] H. Wang, S. Guo, J. Cao, and M. Guo, "Melody: A long-term dynamic quality-aware incentive mechanism for crowdsourcing," in *ICDCS*, 2017.

[23] K. Han, Y. He, H. Tan, S. Tang, H. Huang, and J. Luo, "Online pricing for mobile crowdsourcing with multi-minded users," in *Mobihoc*, 2017.

[24] L. Gao, F. Hou, and J. Huang, "Providing long-term participation incentive in participatory sensing," in *INFOCOM*, 2015.

[25] D. Zhao, X. Y. Li, and H. Ma, "How to crowdsource tasks truthfully without sacrificing utility: Online incentive mechanisms with budget constraint," in *INFOCOM*, 2014.

[26] ——, "Budget-feasible online incentive mechanisms for crowdsourcing tasks truthfully," in *TON*, 2016.

[27] K. Han, H. Huang, and J. Luo, "Posted pricing for robust crowdsensing," in *MobiHoc*, 2016.

[28] X. Zhang, Z. Yang, Z. Zhou, H. Cai, L. Chen, and X. Li, "Free market of crowdsourcing: Incentive mechanism design for mobile sensing," in *TPDS*, 2014.

[29] Z. Zheng, Y. Peng, F. Wu, S. Tang, and G. Chen, "An online pricing mechanism for mobile crowdsensing data markets," in *Mobihoc*, 2017.

[30] H. Jin, L. Su, H. Xiao, and K. Nahrstedt, "Incentive mechanism for privacy-aware data aggregation in mobile crowd sensing systems," in *TON*, 2018.

[31] Y. Gao and A. Parameswaran, "Finish them!: Pricing algorithms for human computation," *Proc. VLDB Endow.*, vol. 7, no. 14, pp. 1965–1976, Oct. 2014.

[32] D. Bertsimas and J. N. Tsitsiklis, *Introduction to linear optimization*. Athena Scientific Belmont, MA, 1997, vol. 6.

[33] "Gurobi solver," http://www.gurobi.com/.