Sumedh Dongare, Bernd Simon, Andrea Ortiz, and Anja Klein ”Two-Sided Learning: A Techno-Economic View of Mobile Crowdsensing under Incomplete Information”, in *IEEE International Conference on Communications (ICC), Denver, USA*, June 2024.

Two-Sided Learning: A Techno-Economic View of Mobile Crowdsensing under Incomplete Information

Sumedh Dongare, Bernd Simon, Andrea Ortiz, Anja Klein
Communications Engineering Lab, Technical University of Darmstadt, Germany.
{s.dongare, b.simon, a.ortiz, a.klein}@nt.tu-darmstadt.de

***Abstract*—In Mobile Crowdsensing (MCS) a mobile crowdsensing platform (MCSP) collects sensing data from mobile units (MUs) in exchange for payment. The MCSP broadcasts a list of available sensing tasks. Based on this list, each MU solves a task proposal problem to decide which task it is willing to perform and sends a proposal to the MCSP. Based on the MUs' proposals, the MCSP solves a task assignment problem. There are two challenges when finding efficient task proposal strategies for the MUs and an efficient task assignment strategy for the MCSP (i) The techno-economic perspective of MCS: From the technical perspective, MCS should maximize the data quality while minimizing time and energy consumption. From the economic perspective, there are two sides, the MUs and the MCSP which act as selfish decision-makers, who aim at maximizing their own income. (ii) Incomplete information at two sides: Initially, the MCSP does not know the expected data quality and the MUs do not know the expected effort required for task completion. To overcome these challenges, we propose a novel Two-Sided Learning (TSL) approach. At the MU side, TSL is based on an innovative gradient-based multi-armed bandit solution to maximize the MUs' utility under incomplete information about the strategies of other MUs. At the MCSP side, a learning strategy is used to find the task assignment strategy that maximizes its utility. Simulation results show that TSL achieves near-optimal social welfare, which is the sum of MUs' and MCSP's utilities, and a near-optimal energy efficiency.**

I. Introduction

Mobile Crowdsensing (MCS) refers to the collection of sensing data from a group of mobile units (MUs) [1]. MUs, such as smartphones or smartwatches are equipped with heterogeneous sensors which enable the completion of a variety of sensing tasks. These tasks may include e.g. sensing temperature, noise level, taking pictures, or recording videos [2]. Advantages of MCS compared to wireless sensor networks are reduced infrastructure and operation cost, higher coverage and a wider range of applications [3]. MCS allows the MUs to leverage their sensing capabilities in exchange for payments [4].

A typical MCS system is composed of a data requester (DR), a mobile crowdsensing platform (MCSP) and MUs [1]. The DR offers the MCSP a payment in exchange for sensing data [1]. This payment is agreed upon by a contract and depends on the quality of the sensing result. The MCSP converts the sensing requests from the DR into sensing tasks which can be performed by the MUs [4]. The MUs signal their willingness to the MCSP to participate in a task by submitting a task proposal which includes the required payment. The *task proposal* strategy of the MUs depends on their preferences and the effort required to complete the task. The MCSP performs the *task assignment* based on the proposals of the MUs, the expected quality of their results and their payment proposals.

The success of MCS depends on finding efficient task proposal strategies for the MUs as well as an efficient task assignment strategy for the MCSP. This poses two major challenges: 1) The *technical and economic perspectives* of the MCSP and the MUs have to be considered [5]. From the technical perspective, the goal of MCS is to maximize the quality of the sensing results while minimizing the effort in terms of time and energy consumption [6]. From the economic perspective, the MCSP and the MUs act as individual rational decision makers which aim to maximize their income in a selfish manner [5]. 2) *Incomplete information* has to be considered. Initially, the MUs do not know the effort required to perform each task. Therefore, it may occur that more MUs than required send task proposals for the same task. Such events are termed *collisions* and they significantly reduce the overall performance, as the sensing capabilities of the MUs in the collision cannot be used for a different task. Note that the MUs can only obtain information about their efforts by actually performing the tasks. Moreover, the MCSP has incomplete information about the expected quality of the task results of the MUs. This information can only be obtained by assigning tasks to the MUs.

A summary of how the aforementioned challenges have been addressed in the related works [4] and [7]–[13] is presented in Table I. Although [7] and [8] consider the technical and economic perspective of MCS, it is assumed that the MCSP and the MUs have complete information regarding the quality of the task results and the effort required to perform the task, which is arguably not realistic in real scenarios. In [9], the authors only focus on a technical perspective of MCS, neglecting the individual utilities of the MUs and the MCSP. Although in [10]–[12] learning approaches for the MCSP are proposed to handle the uncertainty about the MUs' expected quality, it is still assumed that the MUs have perfect information. This is, the MUs perfectly know the

This work has been funded by the German Research Foundation (DFG) as a part of the projects C1, B3, and T2 within the Collaborative Research Center (CRC) 1053 - MAKI (Nr. 210487104) and has been supported by the BMBF project Open6GHub (Nr. 16KISK014) and the LOEWE Center EmergenCITY. This work is also supported by DAAD with funds from the Federal Ministry of Education and Research (BMBF).

TABLE I
SUMMARY OF LITERATURE

Available Information	Economic perspective	Technical perspective
Complete Information at the MCSP and at the MUs	[4], [7]–[9]	[4], [7]–[9]
Complete Information at the MCSP	[13]	[13]
Complete Information at the MUs	[10]–[12]	[10]–[12]
Incomplete Information at the MCSP and at the MUs	**This work**	

efforts required to perform each task in advance. The authors of [13] propose an online learning approach to learn the MUs' efforts, but assume that the MCSP has complete information regarding the MUs. Although all these works have greatly contributed to the understanding of the task proposal and task assignment problems in MCS systems, they failed to address the two previously mentioned challenges simultaneously, i.e., finding a solution for MCS which overcomes the challenge of incomplete information at the MUs and MCSP while considering the technical and economic perspectives.

Having in mind that in real MCS systems both, the MUs and the MCSP, have incomplete information, we propose a novel learning approach, termed Two-Sided Learning (TSL). Our proposed TSL approach considers the technical and economic perspectives of MCS and solves the task proposal problem locally at the MUs and the task assignment problem locally at the MCSP, thus utilizing the learning capabilities of the MUs as well as of the MCSP. Specifically, TSL consists of two parts, i) Two-Sided Learning Task Proposal (TSLTP) which learns the task proposal strategies of the MUs to maximize their utilities and ii) Two-Sided Learning Task Assignment (TSLTA) which allows the MCSP to learn the task assignment strategy to maximize its utility. Furthermore, to reduce the number of collisions, TSLTP is based on an innovative gradient-based multi-armed bandit solution. This allows the MUs to adapt their preferences according to tasks which are popular and thus prone to collisions. To the best of our knowledge, this is the first work in MCS that considers decentralized learning on the MUs and the MCSP simultaneously. Simulation results show a close to optimal performance in terms of social welfare, i.e., the sum of MUs' and MCSP's utilities, and a close to optimal energy efficiency of our proposed approach.

The rest of this paper is organized as follows. In Section II, we introduce the MCS system model. The problem formulation is explained in Section III and in Section IV, we present our novel TSL algorithm. The numerical evaluation of the proposed LT algorithm is presented in Section V and finally, Section VI concludes the paper.

II. SYSTEM MODEL

We consider an MCS system formed by a DR, an MCSP, and a crowd of K MUs, denoted by the set $\mathcal{K} = \{\mathrm{MU}_k\}_{k=1,\dots,K}$. Time is divided into discrete time steps of equal duration indexed by $t = \{1, \dots, T\}$.

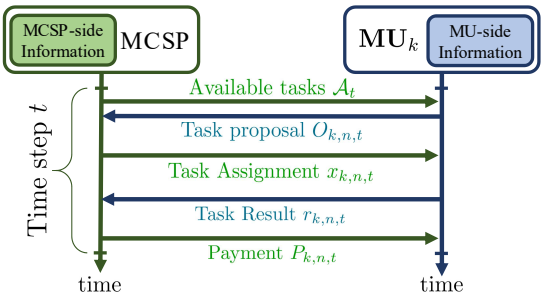

Fig. 1. System Model

Fig. 1 illustrates the sequence of events in every time step t. Note that these events take place within the time step duration. At the beginning of t, the MCSP publishes N available tasks. These tasks are collected in the set $\mathcal{A}_t = \{a_{n,t}\}_{n=\{1,\dots,N\}}$. Each task $a_{n,t}$ is classified into a task type z, e.g., sensing temperature or taking a picture. The different task types are collected in the set $\mathcal{Z} = \{z\}_{z=1,\dots,Z}$. We collect all tasks of the same type z in the set $\mathcal{A}_{z,t} \subseteq \mathcal{A}_t$. Each task type z is characterized by the average size s_z of the sensing result, measured in bits. All the tasks of type z have identical characteristics. We assume that each published task $a_{n,t}$ requires only one MU to complete. The MCSP may publish multiple tasks of the same task type z if the DR requires multiple sensing samples from that task type.

A. Mobile Units

After the MCSP publishes the tasks, each MU_k identifies its preferred task of type $z \in \mathcal{A}_{z,t}$, and sends a task proposal $O_{k,t}$ to the MCSP. To make this decision, the MU_k considers its task preference and the effort it requires to perform the task. The efforts are measured in terms of the time and energy consumed during the task execution. We assume all MUs are capable of performing every task irrespective of its type z. However, each MU_k can perform only one task per time step.

MU_k needs time $\tau^{\text{sense}}_{k,n,t}$ to sense and produce valid sensing data of size d_z measured in bits. $\tau^{\text{sense}}_{k,n,t}$ is drawn from a stationary random distribution with probability density function (PDF) $f^z_{\tau^{\text{sense}}_{k,n,t}}(\tau^{\text{sense}}_{k,n,t})$. The expected value $\bar{\tau}^{\text{sense}}_{k,z} = \mathbb{E}(\tau^{\text{sense}}_{k,n,t})$ of the sensing time depends on the task type z and the capabilities of MU_k. Each MU is required to process the sensing data such that a sensing result can be generated [14]. The computation time MU_k requires to process the sensing data of size d_z for the task $a_{n,t}$ of type z is calculated as, $\tau^{\text{comp}}_{k,n,t} = \frac{c_z d_z}{f^{\text{local}}_k}$, where c_z represents the processing complexity for task type z and f^{local}_k is the central processing unit frequency of MU_k measured in Hz. After the processing, the final sensing result $r_{k,n,t}$ has a size $s_z < d_z$ measured in bits. This sensing result $r_{k,n,t}$ is transmitted to the MCSP as result of the task execution. The time required by MU_k to transmit the sensing result of task $a_{n,t}$ back to the MCSP is termed $\tau^{\text{comm}}_{k,n,t}$. This time depends on the communication channel between MU_k and the MCSP. The total time required by MU_k to perform task $a_{n,t}$ is given by $\tau_{k,n,t} = \tau^{\text{sense}}_{k,n,t} + \tau^{\text{comm}}_{k,n,t} + \tau^{\text{comp}}_{k,n,t}$.

To perform a task, MU_k also spends energy $E_{k,n,t} = p^{\text{comm}}_k \tau^{\text{comm}}_{k,n,t} + p^{\text{comp}}_k \tau^{\text{comp}}_{k,n,t}$, where p^{comm}_k is MU_k's transmit

power and p_k^{comp} is the power required to process task $a_{n,t}$. We neglect the energy required for sensing because it is assumed to be small when compared with the communication and computation energy.

The time and energy efforts every MU requires to perform task $a_{n,t} \in \mathcal{A}_{z,t}$ are denoted by $J_{k,t}^z$. Without loss of generality, we assume a MU-specific linear cost function to evaluate the efforts, $J_{k,t}^z = c_k^{\text{effort}}(\tau_{k,n,t}, E_{k,n,t}) = \alpha\tau_{k,n,t} + \beta E_{k,n,t}$, measured in monetary units. The variables α and β are importance factors for the time and energy efforts, respectively. Note that other cost functions can be easily integrated in our approach.

The MCSP pays $P_{k,n,t} = P^{\text{effort}}(\tau_{k,n,t}, E_{k,n,t})$ monetary units to MU_k for the execution of task $a_{n,t}$. Like c_k^{effort}, the payment function P^{effort} considers the time and energy spent on performing task $a_{n,t}$. The utility of MU_k in time step t is

$$U_{k,n,t}^{\text{MU}} = P_{k,n,t} - c_k^{\text{effort}}(\tau_{k,n,t}, E_{k,n,t}). \tag{1}$$

To calculate $U_{k,n,t}^{\text{MU}}$, MU_k should know the exact time and energy efforts it requires to perform task $a_{n,t}$. However, given the random nature of the time and energy required, the MUs cannot know the exact efforts required before performing the task. Thus, MU_k estimates the utility as

$$\begin{aligned}\bar{U}_{k,z}^{\text{MU}} &= \mathbb{E}\{U_{k,n,t}^{\text{MU}} | a_{n,t} \in \mathcal{A}_{z,t}\} \\ &= \mathbb{E}\{P_{k,n,t}\} - \mathbb{E}\{c_k^{\text{effort}}(\tau_{k,n,t}, E_{k,n,t})\},\end{aligned} \tag{2}$$

to make its task proposal decision and to select $P_{k,n,t}$. We define $\mathcal{I}_k^{\text{MU}} = \{\bar{U}_{k,z}^{\text{MU}}, \forall z\}$ as the *MU-side* information. Note that $\mathcal{I}_k^{\text{MU}}$ is not available in advance at MU_k and has to be learned over time by performing tasks. Based on its own $\mathcal{I}_k^{\text{MU}}$, every MU_k sends a proposal $O_{k,t}$ to the MCSP containing the task type index z of the task it wants to perform and its payment $P_{k,n,t}$.

B. Mobile Crowdsensing Platform

The MCSP receives the task proposals $O_{k,t}$ from all MUs for the available tasks in $\mathcal{A}_t \in \mathcal{Z}$ and decides which MU_k executes each task $a_{n,t}$. The task assignment decision is denoted by $x_{k,n,t} \in \{0, 1\}$. $x_{k,n,t} = 1$ means $a_{n,t}$ is assigned to MU_k and $x_{k,n,t} = 0$ means otherwise.

The DR pays the MCSP for every executed task. The earning $w_{z,t}$ which the MCSP gets when task $a_{n,t}$ is performed by MU_k depends on the task type z and the quality factor $q_{k,n,t} \in [0, 1]$ of the sensing result $r_{k,n,t}$. The quality factor $q_{k,n,t}$ is calculated using a quality function Q_z as

$$q_{k,n,t} = Q_z(r_{k,n,t}), \forall a_{n,t} \in \mathcal{A}_{z,t}. \tag{3}$$

The MCSP defines a quality function Q_z for every task type z. These quality functions could be, e.g., Peak Signal-to-Noise Ratio (PSNR) for images or accuracy of the temperature sensing. We assume that the MCSP and the DR make a contractual agreement on the calculation of $w_{z,t}$ as

$$w_{z,t} = (1 + q_{k,n,t})w_z. \tag{4}$$

w_z is the minimum payment in monetary units the MCSP charges the DR for performing a task of type z. As the quality $q_{k,n,t}$ of the sensing result $r_{k,n,t}$ is not known to the MCSP in advance, $w_{z,t}$ is also not known. The utility $U_{k,n,t}^{\text{MCSP}}$ of the MCSP when assigning MU_k to task $a_{n,t} \in \mathcal{A}_{z,t}$ is defined as

$$U_{k,n,t}^{\text{MCSP}} = (w_{z,t} - P_{k,n,t}). \tag{5}$$

The MCSP can maximize its utility $U_{k,n,t}^{\text{MCSP}}$ by balancing the quality of the sensing result of MU_k and its payment. However, as the MCSP does not know the quality factor $q_{k,n,t}$ of the MU performing task $a_{n,t}$, it estimates its utility when assigning MU_k to a task of type z, as

$$\begin{aligned}\bar{U}_{k,z}^{\text{MCSP}} &= \mathbb{E}\{U_{k,n,t}^{\text{MCSP}} | a_{n,t} \in \mathcal{A}_{z,t}\} \\ &= \mathbb{E}\{w_{z,t}\} - \mathbb{E}\{P_{k,n,t}\}.\end{aligned} \tag{6}$$

We define $\mathcal{I}_z^{\text{MCSP}} = \{\bar{U}_{k,z}^{\text{MCSP}}, \forall k\}$ as the *MCSP-side* information about the MUs. $\mathcal{I}_z^{\text{MCSP}}$ contains information about the earnings and the required payment for all MUs. $\mathcal{I}_z^{\text{MCSP}}$ is not available at the MCSP in advance and has to be learned over time from experience gained from selecting MUs.

The combination of MU-side and MCSP-side information, denoted by $\mathcal{I} = \{\mathcal{I}_k^{\text{MU}}, \mathcal{I}_z^{\text{MCSP}}, \forall k, z\}$, is called the *complete* information and is unknown to the MUs and the MCSP. Our goal is to optimize the task proposals and the task assignments without knowledge of $\mathcal{I}$. For this purpose, the MUs learn the efforts of each task type and find their most preferred tasks. The MCSP learns the MUs' characteristics and selects a suitable MU for each task type z.

III. Task Proposal and Task Assignment Game

In this work, we assume that the MCSP and the MUs are rational and independent entities which make their own decisions based on their preferences. Since their preferences influence their respective utilities, we use game theory, specifically matching theory [15], to analyze and solve the joint task proposal and task assignment problem. Matching theory aims to obtain a *stable matching*, i.e., to find task assignments where the MUs and the MCSP cannot improve their individual utilities by changing the assignment. This corresponds to MUs and MCSP that selfishly and individually try to obtain their best task proposals and task assignments respectively. A stable matching is relevant for the MCS system because it allows the maximization of both, the MUs' and MCSP's utilities, with regard to their individual preferences.

The considered MCS scenario is modelled as a matching game $\mathcal{G}$, i.e., a two-sided market in which the MCSP requires sensing resources to execute tasks and the MUs offer their sensing resources in exchange for a payment [16]. The MUs' payment function P^{effort}, the cost function c_k^{effort}, and the MCSP's quality function Q_z are given functions which depend on the task assignment [17]. The MUs' preference ordering $\succeq_k^{\text{MU}}$ ranks the task types $z \in \mathcal{Z}$ w.r.t. the expected utility associated with them, i.e.,

$$z \succeq_k^{\text{MU}} z' \iff \bar{U}_{k,z}^{\text{MU}} \geq \bar{U}_{k,z'}^{\text{MU}}. \tag{7}$$

This means, the MU_k prefers task type z over z' if the expected utility $\bar{U}_{k,z}^{\text{MU}}$ of performing tasks of type z is higher

than of tasks of type z'. Similarly, the MCSP prefers MUs which yield the highest expected utility $\bar{U}_{k,z}^{\text{MCSP}}$ for each task type z, i.e.,

$$\text{MU}_k \succeq_z^{\text{MCSP}} \text{MU}_l \iff \bar{U}_{k,z}^{\text{MCSP}} \geq \bar{U}_{l,z}^{\text{MCSP}}. \quad (8)$$

This means that for the assignment of a task of type z, the MCSP prefers MU_k over MU_l if MU_k provides higher utility compared to MU_l. This preference ranking can only be correctly determined with the MCSP-side information $\mathcal{I}^{\text{MCSP}}$.

The task proposal and task assignment game $\mathcal{G}_t$ in time slot t is a tuple $\mathcal{G}_t = (\mathcal{K}, \mathcal{A}_t, \succeq_k^{\text{MU}}, \succeq_z^{\text{MCSP}})$. MU_k signals its willingness to participate in any task of type z by sending a task proposal $O_{k,t}$ to the MCSP. Based on the proposals, the MCSP performs the task assignment according to $\succeq_z^{\text{MCSP}}$. The task assignment decisions $x_{k,n,t}$ for all MUs and tasks in $\mathcal{A}_t$ in time step t are collected in the matrix $\mathbf{X}_t$.

Definition 1. *A task assignment* $\mathbf{X}_t$ *is* unstable *if there are two MUs,* MU_k *and* MU_l*, and two tasks,* $a_{n,t}$ *and* $a_{m,t}$*, such that: (i)* $x_{k,n,t} = 1$*, i.e.* MU_k *is assigned to task* $a_{n,t} \in \mathcal{A}_{z,t}$*. (ii)* $x_{l,m,t} = 1$*, i.e.* MU_l *is assigned to task* $a_{m,t} \in \mathcal{A}_{z',t}$*. (iii)* $z' \succ_k^{\text{MU}} z$ *and* $\text{MU}_k \succeq_{z'}^{\text{MCSP}} \text{MU}_l$*, i.e.,* MU_k *strictly prefers the task with type* z' *over its current matched task of type* z*, and the MCSP would profit more if the task of type* z' *is performed by* MU_k *instead of its current matched* MU_l*.*

The pair (MU_k, z') is called a blocking pair [18]. Here, both, the MU_k and the MCSP, are unsatisfied with the current assignment. The existence of the blocking pair (MU_k, z') causes the matching $\mathbf{X}_t$ to be unstable because MU_k could switch to $a_{m,t} \in \mathcal{A}_{z',t}$ and both, the MU_k and the task $a_{m,t}$ would obtain a more efficient matching and therefore a higher expected utility. The assignment $\mathbf{X}_t$ is said to be stable if no blocking pairs exist [18]. In such cases, no MU or task could change the assignment and improve their expected utilities. In MCS, this means that each MU is assigned to its most preferred task while the MCSP selects its most preferred MU for each task. Note that the stable matching may not be unique. There are, in fact, potentially multiple solutions.

IV. The Two-Sided Learning (TSL) Algorithm

To optimally solve the game $\mathcal{G}_t$ formulated in Sec. III, complete information $\mathcal{I}$ is required, which is unrealistic to assume. Considering every MU_k can only learn its own MU-side information $\mathcal{I}_k^{\text{MU}}$ and the MCSP only learns the MCSP-side information $\mathcal{I}_z^{\text{MCSP}}$, in this section, we present the Two-Sided Learning (TSL) algorithm which aims to maximize the MUs' and MCSP's utilities. TSL consists of two parts, 1) Two-Sided Learning: Task Proposal (TSLTP), executed by each MU, and 2) Two-Sided Learning: Task Assignment (TSLTA), run by the MCSP.

A. Two-Sided Learning: Task Proposal (TSLTP)

TSLTP is a gradient-based multi-armed bandit solution implemented at each MU_k to find the task proposal strategy that reduces collisions. Using TSLTP, each MU_k independently learns the effort $J_{k,t}^z$ required to perform task $a_{n,t}$ of type z and builds a preference $H_t(z)$ for each task type $z \in \mathcal{Z}$. Using $H_t(z)$, MU_k monitors the likelihood of getting assigned to any task of type z.

Algorithm 1 Two-Sided Learning: Task Proposal (TSLTP)

1: **Initialize**: $\hat{U}_{k,0}(z), \hat{J}_{k,0}^z, H_t(z)\ \forall k \in \mathcal{K}, z \in \mathcal{Z}$
2: **for** $t = 1, \ldots, T$ **do**
3: Select task $a_{n,t} \in \mathcal{A}_{z,t}$ out of published tasks $\mathcal{A}_t$ based on *Softmax* distribution $\pi_t(z)$ over task preferences $H_t(z)$.
4: Select payment $\hat{P}_{k,z} \leftarrow P^{\text{effort}}(\hat{J}_{k,t-1}^z)$
5: Send task proposal $O_{k,t} = [z, \hat{P}_{k,z}]$.
6: Wait for the MCSP's decision $x_{k,n,t}$ from Algorithm 2.
7: **if** $x_{k,n,t} = 1$, i.e., $\bar{O}_{k,t} = a_{n,t}$ **then**
8: Perform the task $a_{n,t}$ and transmit the result $r_{k,n,t}$ to MCSP.
9: Receive payment $\hat{P}_{k,z}$ and observe $U_{k,n,t}^{\text{MU}}$, $\tau_{k,n,t}$ and $E_{k,n,t}$.
10: Update estimates $\hat{U}_{k,t}(z)$ and $\hat{J}_{k,t}^z$.
11: **else**
12: $\hat{U}_{k,t}(z) \leftarrow \hat{U}_{k,t-1}(z)$, $\hat{J}_{k,t}^z \leftarrow \hat{J}_{k,t-1}^z$.
13: **end if**
14: Update the task preference $H_t(z), \forall z \in \mathcal{Z}$
15: Update the task proposal probabilities $\pi_t(z) \leftarrow H_t(z), \forall z \in \mathcal{Z}$.
16: **end for**

TSLTP is summarized in Alg. 1. Every MU_k initializes $\bar{U}_{k,0}^{\text{MU}}, \hat{J}_{k,0}(z), H_t(z)$ for all task types (line 1). In every time step, MU_k observes the tasks $a_{n,t} \in \mathcal{A}_t$ published by the MSCP. Based on its task type preferences $H_t(z)$, $\forall z \in \mathcal{Z}$, MU_k uses a Softmax distribution $\pi_t(z) = \frac{e^{H_t(z)}}{\sum_{z=1}^{Z} e^{H_t(z)}}$ to determine the probability of selecting a task type for task proposal (line 3). A task of type z is selected for task proposal according to this probability distribution. After this, MU_k selects the payment $\hat{P}_{k,z}$ based on its payment function and estimated efforts $\hat{J}_{k,t}^z$ (line 4). After sending the task proposal, MU_k waits for the MCSP's response (line 5-6). If the task is assigned to MU_k, i.e., $x_{k,n,t} = 1$, MU_k performs the sensing task $a_{n,t}$, generates the sensing result $r_{k,n,t}$, and transmits it back to the MCSP (line 8). Afterwards, it receives the payment $\hat{P}_{k,z}$ from the platform and observes the exact efforts $J_{k,t}^z$ it required to complete the task. $J_{k,t}^z$ is used to evaluate the exact task utility $U_{k,n,t}^{\text{MU}}$ and update $\bar{U}_{k,n,t}^{\text{MU}}$ as $\bar{U}_{k,n,t}^{\text{MU}} = \bar{U}_{k,n,t-1}^{\text{MU}} + \frac{(U_{k,n,t}^{\text{MU}} - \bar{U}_{k,n,t-1}^{\text{MU}}(z))}{N_k^z}$, where N_k^z represents the number of times MU_k was assigned to a task of type z (line 9). The effort estimate $\hat{J}_{k,t}^z$ is updated similarly (line 10). If MU_k is rejected, it maintains its old effort and utility estimates since it received no new information (line 12). At the end of time step t, MU_k updates its preferences for all the task types including the proposed task type z as,

$$H_{t+1}(z) = H_t(z) + \alpha^{\text{lr}}(R_t - \bar{R}_t)(1 - \pi_t(z)),$$
$$H_{t+1}(z') = H_t(z') - \alpha^{\text{lr}}(R_t - \bar{R}_t)\pi_t(z'), \forall z' \neq z, \quad (9)$$

where $\alpha^{\text{lr}} > 0$ is a step-size parameter, and $\bar{R}_t$ is the average of the MU_k payments up to $t-1$ (line 14). Using the updated preferences H_{t+1}, MU_k updates the probability distribution $\pi_t(z)$ for all task types $z \in \mathcal{Z}$ (line 15). Using (9), MU_k learns to estimate its likelihood of getting assigned to a task of certain task type.

B. Two-Sided Learning: Task Assignment (TSLTA)

At the MCSP, the task assignment decisions $\mathbf{X}_t$ are made in every time step t based on the proposals $O_{k,t}$ received from

Algorithm 2 Two-Sided Learning: Task Assignment (TSLTA)

Require: $O_{k,t}, Q_z, w_z, \epsilon_t \in [0,1), \eta \quad \forall k \in \mathcal{K}, z \in \mathcal{Z}, t \in T$

1: **Initialize:** $\bar{U}^{\mathrm{MCSP}}_{k,z} \quad \forall k \in \mathcal{K}, z \in \mathcal{Z}$.
2: **for** $t = 1, \ldots, T$ **do**
3: Publish available sensing tasks $\mathcal{A}_t$.
4: Wait for all task proposals $O_{k,t}$ from Algorithm 1.
5: **for** $z = 1, \ldots, Z$ **do**
6: **for** $n = 1, \ldots, N$ **do**
7: From $O_{k,t}, \quad \forall k \in \mathcal{K}$:
8: **if** $\eta < \epsilon_t$ **then**
9: Randomly assign MU_k to task $a_{n,t}$.
10: **else**
11: Assign MU_k to task $a_{n,t}$ which maximizes $\bar{U}^{\mathrm{MCSP}}_{k,z}$.
12: **end if**
13: **end for**
14: Send acceptance $x_{k,n,t} = 1$ to the selected MUs, i.e., $\bar{O}_{k,t} = a_{n,t} \; \forall a_{n,t} \in \mathcal{A}_{z,t}$
15: Send rejection $x_{k,n,t} = 0$ to all other MUs, i.e., $\bar{O}_{l,t} = \varnothing$.
16: Receive the task result $r_{k,n,t}$ from the selected MUs and observe $q_{k,n,t}$ and $U^{\mathrm{MCSP}}_{k,n,t}$.
17: Update estimate $\bar{U}^{\mathrm{MCSP}}_{k,z}$
18: **end for**
19: **end for**

the MUs. For this purpose, we present Two-Sided Learning: Task Assignment (TSLTA) which is based on a multi-armed bandit formulation of the task assignment problem. TSLTA is summarized in Alg. 2. The platform initializes the $\bar{U}^{\mathrm{MCSP}}_{k,z}$ for all MUs and all task types (line 1). In every time step, the MCSP publishes the available tasks $\mathcal{A}_t$ and waits for the proposals from the MUs (line 3-4). The MCSP may assign the task to a random MU proposing for a task $a_{n,t} \in \mathcal{A}_{z,t}$ to improve the estimate $\bar{U}^{\mathrm{MCSP}}_{k,z}$ (line 9). Otherwise, it may exploit the available knowledge to assign the task to MUs which maximize its utility $\bar{U}^{\mathrm{MCSP}}_{k,z}$ (line 11). To balance the exploration-exploitation, the MCSP uses ϵ-greedy action selection. The assignment decisions are sent back individually to each MU (line 14-15). After this, the assigned MUs perform the task and transmit the sensing result $r_{k,n,t}$ back to the MCSP. Using $r_{k,n,t}$, the MCSP evaluates the quality of the result $q_{k,n,t}$ as in (3) and observes the $U^{\mathrm{MCSP}}_{k,n,t}$ (line 16). Finally, the MCSP updates the estimate $\bar{U}^{\mathrm{MCSP}}_{k,z,t} = \bar{U}^{\mathrm{MCSP}}_{k,z,t-1} + \frac{(U^{\mathrm{MCSP}}_{k,n,t} - \bar{U}^{\mathrm{MCSP}}_{k,n,t-1})}{N^z_k}$, where N^z_k is the number of times MU_k is assigned to task type z (line 17).

With the help of TSLTP and TSLTA, we jointly optimize the MUs' task proposals to maximize their utilities as well as MCSP's task assignments to maximize the MCSP's utility.

V. Numerical evaluation

In this section we present and discuss the numerical evaluation of the performance of our proposed TSL algorithm in comparison with the reference schemes. The results are generated by averaging over $I = 500$ independent Monte Carlo iterations. In each iteration we consider $T = 20000$ time steps. The total number K of MU is set to 100. We consider $Z = 10$ different types of tasks. For each task type $z \in \mathcal{Z}$, the number of published tasks in time step t is randomly chosen from the interval $[0, 10]$ with equal probability. The number of tasks published N is selected from the range $[0, 100]$. The size of the sensing data is drawn from a uniform random distribution in the interval $[50, 100]\,\mathrm{Mbit}$. The size of the task result after processing the sensing data lies in the range $[10, 20]\,\mathrm{Mbit}$. The sensing time varies for every MU_k in every time step t with a mean value of $\bar{\tau}^{\mathrm{sense}}_{k,z} \in [60, 180]\,\mathrm{s}$. The communication rate between the MU and the MCSP is also randomly drawn from the interval $[40, 80]\,\mathrm{Mbit/s}$ for every MU_k. Due to this, $\bar{\tau}^{\mathrm{comm}}_{k,z}$ falls in the range $[0.125, 0.5]\,\mathrm{s}$. The computation time $\tau^{\mathrm{comp}}_{k,n,t}$ depends on the local CPU frequency $f^{\mathrm{local}}_k \in [1, 2]\,\mathrm{GHz}$ and the task processing complexity $c_z \in [200, 300]$.

For comparison, we consider following reference schemes:
Learning MCSP: In this approach, only the MCSP learns about the quality of task results sent by the MUs to obtain a task assignment strategy. The MUs propose randomly to any available task from the set $\mathcal{A}_t$.
Learning MUs: In this approach, the MUs learn about the task efforts to obtain a task proposal strategy. The MCSP randomly assigns available tasks to the MUs which proposed.
Epsilon Greedy algorithm: Here, the MUs and the MCSP both independently learn their own utilities based on a simple online ϵ-greedy algorithm. To ensure balance between exploration and exploitation, a decaying ϵ parameter is used.
Gale-Shapley algorithm: This approach requires the complete information $\mathcal{I}$. It provides stable solution to the task proposal and task assignment problem formulated in Sec. III to maximize the utilities of the MUs and the MCSP.
Optimal algorithm: This approach aims to maximize the achieved social welfare, i.e., the sum of MUs' and MCSP's utilities, without considering the MU preferences. The task proposal and task assignment decisions are centrally optimized at the MCSP by utilizing the complete information $\mathcal{I}$.
Note that Gale-Shapley and the optimal algorithm cannot be implemented in real-world applications due to their strict and unrealistic requirement of complete information $\mathcal{I}$. Nevertheless, they are used as theoretical baselines for comparison.

In Fig. 2 we show the achieved social welfare of the considered approaches. The performances of the Optimal and the Gale-Shapley algorithm define the upper bound by exploiting the complete information $\mathcal{I}$. Our TSL algorithm outperforms the reference schemes and achieves 99.7% social welfare as compared to the Gale-Shapley algorithm. To achieve this, TSL algorithm maximizes the individual utilities of the MUs and the MCSP. The MUs also take into account the task type preferences to propose to different task types. This greatly reduces the collisions resulting in a near-optimal performance. The results show that learning only at the MU-side or at the MCSP-side is not sufficient and results in a sub-optimal social welfare. The ϵ-greedy algorithm learns the task proposal strategies for every MU and a task assignment strategy for the MCSP. However, it falls short as it does not take collisions into account. This deteriorates its performance as multiple MUs are deferred from task assignment as a result of collision.

The TSL algorithm maximizes the social welfare and it is also energy efficient. To show this, in Fig. 3, we compare the energy consumption of the MUs, measured in Joule per bit. To retain the fairness in the comparison, we weight the energy consumed by the MUs with their corresponding task completion ratios. If an algorithm performs a less tasks, only a small

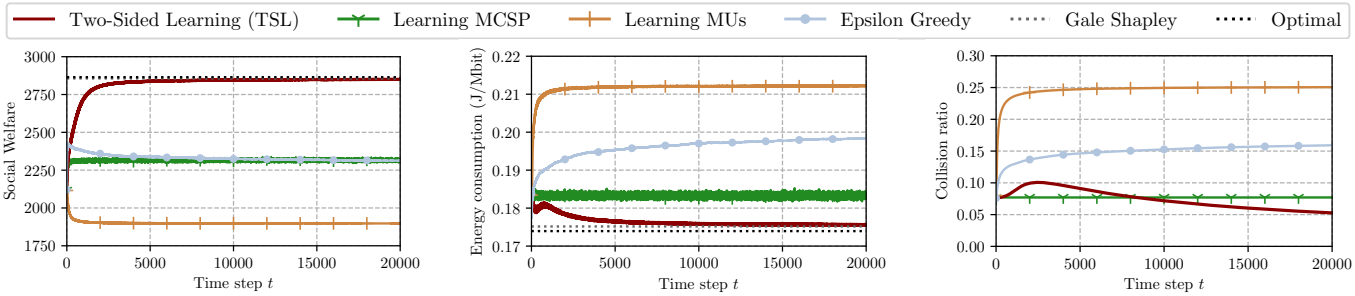

Fig. 2. Social welfare vs. time step t

Fig. 3. Energy consumption (J/bit) vs. time step t

Fig. 4. Collision ratio vs. time step t

amount of energy will be consumed. However, by normalizing with the task completion ratio, we maintain fairness in this comparison. Results show that TSL outperforms the Learning-MU algorithm by approx. 20.4%, the ϵ-greedy algorithm by approx. 12.5%, and the Learning-MCSP algorithm by approx. 3.8% and achieves 99.7% close-to optimal performance.

In Fig. 4, we compare the number of collisions per total number of task proposals. Note that the Gale-Shapley and optimal algorithms are excluded from this comparison since collisions do not occur in them. They both exploit the complete information $\mathcal{I}$ to identify the optimal task proposal strategy which avoids collisions altogether. For this analysis, we change the simulation setup slightly. The total number of tasks available N in every time step t is always equal to the total number of MUs $K = 100$. With this setup, every MU can propose to at least one task without any collision. We observe that TSL outperforms the Learning-MU algorithm by 20%, the ϵ-greedy algorithm by 10%, and the Learning-MCSP algorithm by 3%. This is because our algorithm iteratively updates the task type preferences based on the acceptance mechanism of the MCSP. This helps the MUs to defer from proposing to some task types if the MCSP always prefers some other MU. Thus reducing the collisions and improving the system performance.

VI. Conclusion

In this work, we investigated an MCS scenario from a techno-economic perspective. From the MUs side, we solved the task proposal problem and from the MCSP side, we solved the task assignment problem. The MUs and the MCSP are selfish and rational decision makers aiming to maximize their own individual utility. The challenge in finding the task proposal and task assignment strategies resided in the lack of complete information on both sides, which demanded a two-sided learning solution. To solve this problem, we proposed the novel TSL algorithm, which consists of a task proposal algorithm, TSLTP, at every MU and a task assignment algorithm, TSLTA, at the MCSP. Using TSLTP, every MU learns the task preferences and the expected task efforts to develop its own task proposal strategy, which improves its utility. TSLTA allows the MCSP to learn the expected data quality of MUs to design a task assignment strategy, which improves its own utility. Simulation results showed that our TSL algorithm achieved a social welfare of 99.7% of the stable optimal solution, which requires complete information of both sides. Moreover, the TSL algorithm outperformed all the reference schemes by at least 18.9%.

References

[1] Y. Liu, L. Kong, and G. Chen, "Data-Oriented Mobile Crowdsensing: A Comprehensive Survey," *IEEE Commun. Surveys & Tutorials*, vol. 21, no. 3, pp. 2849–2885, 2019.

[2] Y. Liu, Z. Yu, H. Cui *et al.*, "SafeCity: A Heterogeneous Mobile Crowd Sensing System for Urban Public Safety," *IEEE Internet of Things J.*, vol. 10, no. 20, pp. 18 330–18 345, May 2023.

[3] W. Guo, W. Zhu, Z. Yu *et al.*, "A Survey of Task Allocation: Contrastive Perspectives From Wireless Sensor Networks and Mobile Crowdsensing," *IEEE Access*, vol. 7, pp. 78 406–78 420, 2019.

[4] C.-L. Hu, K.-Y. Lin, and C. K. Chang, "Incentive Mechanism for Mobile Crowdsensing With Two-Stage Stackelberg Game," *IEEE Trans. on Services Computing*, vol. 16, no. 3, pp. 1904–1918, 2023.

[5] B. Simon, K. Keller, A. Sterz *et al.*, "A Multi-Stakeholder Modeling Framework for the Techno-Economic Analysis of Telecommunication Networks," *IEEE Commun. Mag.*, vol. 61, no. 2, pp. 52–56, 2023.

[6] L. Wang, D. Zhang, Y. Wang *et al.*, "Sparse mobile crowdsensing: challenges and opportunities," *IEEE Commun. Mag.*, vol. 54, no. 7, pp. 161–167, 2016.

[7] B. Simon, S. Dongare, T. Mahn *et al.*, "Delay- and Incentive-Aware Crowdsensing: A Stable Matching Approach for Coverage Maximization," in *Proc. of the IEEE Int. Conf. on Commun. (ICC)*, 2022, pp. 2984–2989.

[8] F. Yucel and E. Bulut, "Online Stable Task Assignment in Opportunistic Mobile Crowdsensing With Uncertain Trajectories," *IEEE Internet of Things J.*, vol. 9, no. 11, pp. 9086–9101, Oct. 2021.

[9] M. Zhang, P. Yang, C. Tian *et al.*, "Quality-Aware Sensing Coverage in Budget-Constrained Mobile Crowdsensing Networks," *IEEE Trans. on Veh. Technol.*, vol. 65, no. 9, pp. 7698–7707, 2016.

[10] F. Wu, S. Yang, Z. Zheng *et al.*, "Fine-Grained User Profiling for Personalized Task Matching in Mobile Crowdsensing," *IEEE Trans. on Mobile Computing*, vol. 20, no. 10, pp. 2961–2976, 2021.

[11] Y. Li, F. Li, L. Zhu *et al.*, "Fair Incentive Mechanism With Imperfect Quality in Privacy-Preserving Crowdsensing," *IEEE Internet of Things J.*, vol. 9, no. 19, pp. 19 188–19 200, 2022.

[12] M. Xiao, B. An, J. Wang *et al.*, "CMAB-Based Reverse Auction for Unknown Worker Recruitment in Mobile Crowdsensing," *IEEE Trans. on Mobile Computing*, vol. 21, no. 10, pp. 3502–3518, 2022.

[13] B. Simon, A. Ortiz, W. Saad *et al.*, "Decentralized online learning in task assignment games for mobile crowdsensing," 2023.

[14] Y. Huang, H. Chen, G. Ma *et al.*, "OPAT: Optimized Allocation of Time-Dependent Tasks for Mobile Crowdsensing," *IEEE Trans. on Ind. Informatics*, vol. 18, no. 4, pp. 2476–2485, Jul. 2022.

[15] Y. Gu, W. Saad, M. Bennis *et al.*, "Matching theory for future wireless networks: fundamentals and applications," *IEEE Commun. Mag.*, vol. 53, no. 5, pp. 52–59, May 2015.

[16] L. S. Shapley and M. Shubik, "The Assignment Game I: The Core," *Int. J. Game Theory*, vol. 1, no. 1, p. 111–130, Dec. 1971.

[17] S. H. Cen and D. Shah, "Regret, stability & fairness in matching markets with bandit learners," in *Proc. of the Int. Conf. on Artificial Intell. and Stat. (AISTATS)*, Valencia, Spain, Mar. 2022, pp. 8938–8968.

[18] N. Nisan, T. Roughgarden, É. Tardos *et al.*, *Algorithmic Game Theory*. New York, NY, USA: Cambridge University Press, 2007.