

Online Context-Aware Task Assignment in Mobile Crowdsourcing via Adaptive Discretization

Sepehr Elahi¹, Student Member, IEEE, Andi Nika², and Cem Tekin¹, Senior Member, IEEE

Abstract—Mobile crowdsourcing is rapidly boosting the Internet of Things revolution. Its natural development leads to an adaptation to various real-world scenarios, thus imposing a need for wide generality on data-processing and task-assigning methods. We consider the task assignment problem in mobile crowdsourcing while taking into consideration the following: (i) we assume that additional information is available for both tasks and workers, such as location, device parameters, or task parameters, and make use of such information; (ii) as an important consequence of the worker-location factor, we assume that some workers may not be available for selection at given times; (iii) the workers' characteristics may change over time. To solve the task assignment problem in this setting, we propose *Adaptive Optimistic Matching for Mobile Crowdsourcing (AOM-MC)*, an online learning algorithm that incurs $\tilde{O}(T^{(\bar{D}+1)/(\bar{D}+2)+\epsilon})$ regret in T rounds, for any $\epsilon > 0$, under mild continuity assumptions. Here, \bar{D} is a notion of dimensionality which captures the structure of the problem. We also present extensive simulations that illustrate the advantage of adaptive discretization when compared with uniform discretization, and a time- and location-dependent crowdsourcing simulation using a real-world dataset, clearly demonstrating our algorithm's superiority to the current state-of-the-art and baseline algorithms.

Index Terms—Crowdsourcing, online learning, task assignment, contextual multi-armed bandits, adaptive discretization.

I. INTRODUCTION

GENERALLY, *crowdsourcing* (CS) refers to the practice of outsourcing a task to crowds, each individual of which is referred to as a worker. In recent years, due to tremendous growth in mobile devices and the average time spent on them, a new trend of this practice has emerged, called *mobile crowdsourcing* (MCS). Here, the tasks are distributed to workers' devices, and they usually require gathering sensory data (mobile crowdsensing) or user-contributed data from social networking services. This sensory data is becoming extremely

Manuscript received 22 June 2021; revised 29 July 2022; accepted 10 September 2022. Date of publication 16 September 2022; date of current version 6 January 2023. Recommended for acceptance by Dr. Jiangchuan Liu. (Sepehr Elahi and Andi Nika contributed equally to this work.) (Corresponding author: Sepehr Elahi.)

Sepehr Elahi and Cem Tekin are with the Department of Electrical and Electronics Engineering, Bilkent University, 06800 Bilkent, Ankara, Turkey (e-mail: sepehr.elahi@ug.bilkent.edu.tr; cemtekin@ee.bilkent.edu.tr).

Andi Nika was with the Department of Electrical and Electronics Engineering, Bilkent University, Ankara 06800, Turkey. He is now with Max Planck Institute for Software Systems, 66123 Saarbrücken, Germany (e-mail: andi.nika@bilkent.edu.tr).

This article has supplementary downloadable material available at <https://doi.org/10.1109/TNSE.2022.3207418>, provided by the authors.

Digital Object Identifier 10.1109/TNSE.2022.3207418

useful in the Internet of Things paradigm. Thus, the thorough study of mobile crowdsensing has become paramount in this direction, and incentive mechanisms are being researched to attract users to tasks [1], [2]. Additionally, methods that can preserve user privacy are naturally required [3], [4].

There are different key points to consider when optimizing the completed tasks' overall quality. First, how should the task assignment process be controlled? There are generally two ways of task assignment used in the literature: the *server assigned tasks* (SAT) mode and the *worker selected tasks* (WST) mode [5]. In SAT mode, the assignment process is controlled by a *mobile crowdsourcing platform* (MCSP) that assigns tasks to workers, taking into consideration the previous history of task completions and the workers' potential traits. Then, the workers decide whether or not to accept the task and complete it. In the WST mode, the assignment process is primarily controlled by the workers, where a list of different tasks is presented to them, and they can select the task they wish to complete. Note that a key advantage of the SAT mode is the access to previous history and worker information, both of which are essential in guiding the selection process. On the other hand, the SAT mode may cause communication overhead and violate worker privacy. However, even though WST may preserve worker privacy, there is no guarantee of an optimal assignment due to the large number of tasks that the workers have to skim through before they make their choice. In this case, the MCSP may intervene by suggesting some tasks to the workers, thereby narrowing down the set of potential tasks. This additional *task recommendation* (TR) is called the WST/TR mode.

In this work, we consider the SAT mode, where each new task that arrives is assigned to multiple workers by the MCSP. Then, the workers decide whether or not to accept and perform the given task. We choose SAT to fully utilize all the available information about the workers to optimize task completion, something that cannot be done by the workers individually.

Furthermore, another critical factor affecting the overall performance is the side-information about both the task and the workers. Side-information may include the task and worker locations [6], [7], the task type and time required to be completed, and worker device type and quality (e.g., in the camera quality of a worker's device in image crowdsensing).

When it comes to assigning workers to tasks, one can restrict each task to be assigned to one worker, called homogeneous CS. Alternatively, multiple tasks can be assigned to multiple workers in heterogeneous CS. In our work, we consider a heterogeneous CS setup where one task is assigned to multiple workers.

Aside from these, it is realistic to assume a time-varying set of workers, especially for a location-dependent crowdsensing task, where the worker's location is crucial. For instance, workers may be on the move and thus be available for a particular task only a handful of times. Furthermore, certain types of tasks can be assigned to multiple workers within some budget constraints to maximize their expected performance on the task. In this case, it is essential to note that even when assuming that the MCSP has perfect knowledge about workers' performance in advance, the problem of selecting multiple of them to complete the task to maximize the overall reward is combinatorial and may turn out to be NP-hard.

We consider an MCS task assignment problem that considers all the features above. In order to solve this problem, we use an online learning framework known as multi-armed bandits (MAB) that can be used to learn a given environment's features assuming they come from unknown probability distributions [8], [9]. In its classical version, the problem is formulated as a game that occurs in rounds between a learning agent and a stochastic environment. A finite set of actions characterizes the environment, also called arms, each one of which corresponds to a probability distribution with finite moments. The learner selects arms sequentially over rounds, one at a time, to maximize its cumulative reward, oblivious of the parameters of the arms' distributions. The main two challenges of the learner are i) to avoid exploiting suboptimal arms, based on some "good" observations about them, meanwhile forgetting about other potentially better arms, and ii) to incentivize the exploration of arms that have not been explored yet. All this must be done cleverly so as not to waste useful resources. The metric used to evaluate the learner's performance is called the (*expected*) *regret*, which is defined as the cumulative loss of the learner with respect to an oracle that acts optimally based on arms' reward distributions. It is known that maximizing the cumulative reward is equivalent to minimizing the regret [10]. Many algorithms have been proposed to minimize regret by balancing exploration and exploitation. Two notable examples are upper confidence bound (UCB) based index policies [9], [10], [11] and Thompson sampling [8], [12], [13].

An essential extension of the standard MAB is the contextual MAB [14], [15], [16], [17], where at the beginning of each round, the learner observes side-information, also called context, about the arm rewards in that particular round. As the learner tries to maximize its cumulative reward by taking this information into account, its regret is typically measured with respect to an oracle that selects the best arm in each round, given its context. Contextual MAB algorithms have been used in various applications ranging from personalized news article recommendation [18] to sequential decision-making in mobile healthcare [19].

Another important extension of the standard MAB is the combinatorial MAB (CMAB), where in each round, the learner chooses a subset of base arms, also called the super arm, and obtains a reward that depends on the outcomes of the base arms that are in the chosen super arm [20], [21], [22], [23]. This problem is mainly investigated under the semi-

bandit feedback setting, where the learner also observes the outcomes of the selected base arms. It is also extended to handle the cases when some base arms can only get probabilistically triggered [24], [25], [26]. Combinatorial MAB have found applications in slate recommendation [27], crowdsourcing [28] and online influence maximization [25].

Deviating from the aforementioned works, another strand of literature considers MAB with time-varying arm sets under the names *sleeping* MAB [29], [30] or *volatile* MAB [31], both of which are remnants of *mortal* MAB [32]. In this setting, the learner tries to select the best available arm in each round to maximize its cumulative reward. The concept of volatility is quite common in applications that involve sequential decision-making. For instance, in online advertising, ads become unavailable after they expire [32]. Similarly, in crowdsourcing, the set of available tasks and workers may change over time [33].

In this paper, we focus on solving the MCS task assignment problem by exploiting both task and worker context, assuming that workers' availability is time-variant and that the MCSP may select multiple workers to solve a task. Therefore, our model, namely *contextual combinatorial volatile multi-armed bandit* (CCV-MAB), encapsulates a wide range of crowdsourcing problems. To solve these problems, we propose an online learning algorithm called *Adaptive Optimistic Matching for Mobile Crowdsourcing* (AOM-MC), which is an extension to ACC-UCB of [34], a UCB-based algorithm designed for the CCV-MAB setting. It simultaneously tackles volatility and high-cardinality problems by partitioning the context space associated with the set of task-worker pairs into continuously refined regions while becoming more and more confident of the outcomes yielded from that region. Under mild continuity assumptions, it achieves sublinear in time regret and outperforms the previous state-of-the-art.

Compared with [34], we consider new crowdsourcing-related aspects, including task budgets, worker costs, and worker acceptance. Thus, our setup not only uses crowdsourcing terminology but, more importantly, it has more facets and is more general than that of [34]. Moreover, our algorithm, AOM-MC, allows for feasible worker assignment sets with varying cardinalities, contrasting ACC-UCB, which assumed a fixed cardinality for every worker assignment (super arm). These changes are, of course, associated with theoretical modifications. First, we relax the Lipschitz and monotonicity assumptions on the expected reward function to accommodate any feasible super arm. Then, based on these assumptions, we analyze the expected regret incurred by AOM-MC. We also made some changes to the implementation of the algorithm, the most important of which involves using a binary search-like algorithm to traverse the context tree to find leaf nodes. Lastly, we performed a more realistic higher-dimensional simulation (9D in this work versus 3D in [34]), comprising a complex and non-greedy-oracle-maximizing reward function. We also visually illustrate the advantage of using adaptive versus uniform discretization, something missing from [34].

To sum up, the main contribution of this paper is to propose AOM-MC, an online learning algorithm that:

- adaptively discretizes the context space once confident enough and thus increases the accuracy of its estimates about the performance of the workers;
- solves both volatility and high-cardinality problems under mild continuity assumptions;
- achieves sublinear in time regret and converges to the optimal assignment with respect to any available set of workers.

Lastly, we first illustrate the difference between adaptive and uniform discretization in our detailed experiments, demonstrating why and how adaptive discretization is superior, especially in crowdsourcing problems. Then, we present the results of a time- and location-dependent crowdsourcing problem that uses a real-world dataset, showing our algorithm's superiority compared with state-of-the-art algorithms in bandit literature and baseline algorithms in the crowdsourcing literature.

The rest of the paper is organized as follows: Related work is given in Section II followed by problem formulation in Section III. Description of the algorithm and regret bounds are given in Section IV. Numerical results are presented in Section VI, and concluding remarks are given in Section VII. Finally, proofs of the lemmas that are used in the regret analysis are given in the supplemental document.

II. RELATED WORK

A. Related Work in Crowdsourcing

The multi-armed bandit framework has been extensively used for modeling crowdsourcing and crowdsensing problems [35], [36], [37], [38], [39], [40]. In [35], the authors consider a heterogeneous MCS model, where a requester aims to collect traffic data from workers in the form of visual images under some budget constraints B . Their model is heterogeneous in that it allows a task to be completed by several workers and a worker to complete several tasks. The requester first distributes a list of M tasks to N workers, each of which (potentially) completes multiple tasks. This model corresponds to the WST assignment mode in that the workers are responsible for selecting which tasks to complete. Moreover, each task j is associated with weight ω_j that represents the task's level of importance for the requester. They propose the Unknown Worker Recruitment (UWR) algorithm, a CMAB algorithm that first presents the list of tasks to the workers and then, for each worker, picks only one completed task. It does this for K workers in each round and selects the best quality data among them. The algorithm stops when the cost exceeds the budget. Note that their algorithm does not consider other unknown factors, such as worker device type or quality, on which the worker performance also depends. They also provide extensive simulations and prove theoretical guarantees for UWR, which achieves $\tilde{O}(NLK^3)$ -regret¹, where L is the number of completed tasks from a single worker in a given round.

In [36], the authors consider a homogeneous spatial crowdsourcing problem, that is, the data to be collected depends on the locations of both tasks and workers, and the worker-task

assignment is one-to-one, so a worker cannot complete more than one task, and a task is assigned only to one worker. The dynamic nature of the problem is taken into consideration by framing the problem as a bi-objective optimization problem, where both worker reliability and travel cost are simultaneously optimized under uncertainty by using a combinatorial semi-bandit approach. Worker reliability represents the probability that the worker will complete a particular task, depending on the worker and task context. Since the assignment process is done in SAT mode, the central platform will consider the worker's availability and distance from the task, consequently introducing the need to optimize the second objective, travel cost. They propose the Distance-reliability ratio (DRR) algorithm, which reduces the travel costs by 80% while maximizing worker reliability. They also use an interval estimation heuristic to approximate worker reliabilities when they are unknown beforehand.

The authors of [37] consider a general crowdsourcing problem and employ a modified Thompson sampling technique for worker selection while simultaneously maintaining an exploration-exploitation balance via reinforcement learning. They consider the workers' extrinsic and intrinsic abilities when selecting them and thus provide a novel worker-ability model. Their worker selection algorithm incurs $\tilde{O}(mn)$ -regret, where m and n represent the number of tasks and workers, respectively.

On the other hand, a learning algorithm that uses the Bayesian framework and models the reward process in time as a Gaussian process is used in [41], which sequentially solves the MCS problem. The MCS model considered in their work is a spatial one, and the inherent assumption is the similarity of information about the environmental conditions (e.g., noise in a particular park) of two nearby places. Therefore, the overall aim is to maximize information gain about the environmental conditions by selecting a subset of workers in every round and observing the information they give. Using the submodularity of information gain, they sequentially select the workers that maximize the marginal information gain with respect to the previous selections.

In [42], a theoretical framework for general crowdsourcing scenarios is introduced, and then a bandit algorithm, called BLISS, is proposed to make learning possible under uncertainty. Their method is UCB-based and uses a frequentist approach, incurring sublinear in time regret. Their model accommodates many scenarios, although it does not consider the potential task/worker contexts that essentially affect the performance. They also provide extensive simulations that demonstrate the robustness of their method.

In [38], the authors propose a differentially private MAB algorithm under budget constraints B to maximize the workers' expected cumulative performance. Given a parameter $\epsilon \in (0, 1)$, they use an ϵ -fraction of the budget for exploration and the rest for exploitation. They show that their algorithm incurs sublinear in time regret and is δ -differentially private.

In [39], the authors also use a MAB-based model to solve a spatial crowdsensing problem under budget constraints. [40] proposed a context-aware hierarchical online learning algorithm for mobile crowdsourcing that uses uniform discretization and control function-based exploration-exploitation strategies.

¹ By using the $\tilde{O}(\cdot)$ notation we omit all polylog factors.

TABLE I
COMPARISON WITH RELATED WORK IN CROWDSOURCING

Properties	This work	[35]	[36]	[37]	[41]	[42]	[38]	[39]	[40]
Crowdsourcing type	General	Spatial	Spatial	General	Spatial	General	General	Spatial	Loc.-Ind.
Assignment type	Heterogen.	Heterogen.	Homogen.	Heterogen.	Heterogen.	Heterogen.	Homogen.	Heterogen.	Heterogen
Context-aware worker	Yes	No	No	Yes	No	No	No	No	Yes
Context-dependent task	Yes	Yes	Yes	Yes	Yes	Not mentioned	Not mentioned	Yes	Yes
Regret bounds	Yes	Yes	No	Yes	Yes	Yes	Yes	Yes	Yes

TABLE II
COMPARISON WITH RELATED WORK IN BANDITS

Properties	This work	[47]	[43]	[22]	[46]	[17]	[29]
Contextual	Yes	Yes	Yes	No	No	Yes	No
Combinatorial	Yes	Yes	Yes	Yes	No	No	No
Volatile (base) arms	Yes	No	Yes	No	No	Yes	Yes
Arm and/or context space	Infinite	Finite/infinite	Infinite	Finite	Infinite	Infinite	Finite
Adaptive discretization	Yes	No	No	No	Yes	Yes	No
Reward function	General	General	Submodular	General	General	General	General
Feedback type	Semi-bandit	Cascading	Semi-bandit	Semi-bandit	Full-bandit	Full-bandit	Full-bandit
Computation oracle	α	α	$(1 - 1/e)$	(α, β)	Exact	Exact	Exact

We model the MCS problem using a CCV-MAB framework, thereby accommodating a wide range of MCS scenarios: (i) workers are volatile, and their characteristics may change over time, (ii) workers' expected performances depend on the joint task-worker contexts, (iii) the task type is general, and the MCS type is general; and (iv) the assignment process is heterogeneous. We propose an algorithm called AOM-MC that achieves $\tilde{O}(T^{(\bar{D}+1)/(\bar{D}+2)+\epsilon})$ regret for any $\epsilon > 0$ with respect to an α -approximation oracle under the assumptions that the expected performances and the expected rewards are Lipschitz continuous in the contexts and the expected workers' performances respectively. Here, dimension \bar{D} , which can usually be much smaller than the dimension D of the context space, captures the benignness of the worker arrivals and the structure of the expected reward.

Table I compares our work with the works discussed above.

B. Related Work in MAB

The most closely related work to ours from the bandit literature is [43], which also investigates a variant of the CCV-MAB setting. Using the MAB terminology, this work assumes that the reward function is submodular and the expected base arm outcomes are Hölder continuous in contexts with exponent $\beta > 0$, and uses a greedy algorithm as the approximation oracle. Their proposed learning algorithm, CC-MAB, uses the similarity information in the space of contexts to learn the expected base arm outcomes. For this, it uniformly discretizes the context space \mathcal{X} into hypercubes whose sizes are set according to the time horizon T , resulting in a regret of $\tilde{O}(T^{(2\beta+D)/(3\beta+D)})$. As opposed to that work, our algorithm adaptively discretizes the context space to leverage worker arrivals' benignness and the expected performance structure. Thereby, the regret bounds proven for AOM-MC do not directly depend on the context space dimension D , and it achieves a strictly smaller regret than CC-MAB under Lipschitz continuity ($\beta = 1$). We also provide a recipe for obtaining more optimistic regret bounds while

considering the volatility of the workers in the supplemental document.

Adaptive discretization was first introduced to address the problem of the *continuum-armed* bandit [44], [45], where there are infinitely many arms to choose from, and thus, learning their expected rewards becomes intractable. This problem was generalized in [46] to generic measurable spaces of arms, which also introduced the Hierarchical Optimistic Optimization (HOO) algorithm. Under a set of weak continuity assumptions on the mean reward function around its maxima, HOO was shown to achieve $\tilde{O}(T^{(D_n+1)/(D_n+2)+\epsilon})$ regret, for any $\epsilon > 0$, where D_n is the near optimality dimension related to the arm space. AOM-MC significantly differs from HOO because it selects multiple arms in each round, and the set of arms that it can select from changes in every round.

We compare our work with other MAB-based algorithms in Table II.

III. SYSTEM MODEL

In this section, we describe the components of the MCSP.

A. The Tasks

We assume that the tasks arrive sequentially over time at the MCSP. Tasks are ordered based on their arrival time and are indexed by $t \in \{1, 2, \dots\}$. Task owners can place location-dependent or -independent tasks into the MCSP. A task t is defined by a tuple (b_t, l_t, c_t) , where $b_t > 0$ denotes the budget that the task owner is willing to pay for the task, l_t denotes the task location (we write $l_t = *$ when the task is location-independent) and c_t denotes the task context. For instance, the task context can represent the task difficulty or the amount of skill required to perform the task. We assume that the task location is an element of the location set \mathcal{L} and the task context is an element of the task context set \mathcal{C} .

The task owner has to pay the MCSP a fixed price $e_t \in [e_{\min}, e_{\max}]$ for each worker that completes the task, where $0 < e_{\min} \leq e_{\max}$. This price depends on the task context and

is determined by the MCSP's fixed context-specific price list. We assume that $b_t \geq e_t, \forall t$, i.e., the task owners can afford at least one worker. Based on the task budget, MCSP assigns at most $m_t := \lfloor b_t/e_t \rfloor$ of the available workers to the task.

B. The Workers

Let $\mathcal{W} := [W]$ denote the set of workers in the MCSP. Each worker corresponds to a mobile user that runs the MCS application. A worker is defined by his location and context. For instance, in an MCS where the workers need to use their phones, the worker's context can give information about the phone's processing capacity or the worker's skill level. We assume that a worker's location and context can change over time. Thus, at the arrival time of task t worker $w \in \mathcal{W}$ is defined by the location-context pair $(l_{t,w}, z_{t,w})$. Here, $l_{t,w} \in \mathcal{L}$ and $z_{t,w} \in \mathcal{Z}$, where \mathcal{Z} denotes the worker context set. We assume that the workers periodically send their location information to the MCSP and that the MCSP has an up-to-date knowledge of $l_{t,w}, \forall w \in [W]$.²

We denote by $\mathcal{W}_t \subseteq \mathcal{W}$ the set of available workers for task t . For location-independent tasks, this can be the set of workers who have their mobile devices turned on and are willing to accept new tasks. For location-dependent tasks, this can be the set of workers within the maximum travel distance of task t and are willing to accept new tasks.

C. Worker Performance

Let $\mathcal{X} := \mathcal{L} \times \mathcal{C} \times \mathcal{L} \times \mathcal{Z}$ represent the *joint context set* that includes task and worker locations as well as task and worker contexts, and D represent its dimension. We assume that \mathcal{X} is compact. Let $a(x) \in \{0, 1\}$ represent a worker's decision given joint context $x \in \mathcal{X}$, where $a(x) = 0$ ($a(x) = 1$) represents the event that the worker rejects (accepts) the task. Similarly, let $q(x) \in [0, 1]$ represent the (random) quality of a worker when it accepts the task given joint context $x \in \mathcal{X}$. Both $q(x)$ and $a(x)$ are random variables whose distributions depend on x . Based on these, we define the random variable $p(x)$ that represents the performance of a worker given joint context $x \in \mathcal{X}$ as

$$p(x) := \begin{cases} q(x) & \text{if } a(x) = 1, \\ 0 & \text{otherwise.} \end{cases}$$

We define $\mu : \mathcal{X} \rightarrow [0, 1]$ such that $\mu(x) := \mathbb{E}[p(x)], \forall x \in \mathcal{X}$ as the expected performance function. Notice that the worker's decision to accept the task is not influenced by the task assignment decision, meaning that given the same x in different rounds, $\mathbb{E}[a(x)]$ will be identical in these rounds.

Let $x_{t,w} := (l_t, c_t, l_{t,w}, z_{t,w})$ and $\mathcal{X}_t := \{x_{t,w}\}_{w \in \mathcal{W}_t}$. Then, the performance, interchangeably called the outcome, of worker $w \in \mathcal{W}_t$ for task t is given as $p_{t,w} := p(x_{t,w})$. Given a task with location-context pair $(l, c) \in \mathcal{L} \times \mathcal{C}$ and K workers with location-context pairs $(l_w, z_w) \in \mathcal{L} \times \mathcal{Z}, \forall w \in [K]$, let $x_w := (l, c, l_w, z_w)$ represent the *joint context* for worker $w \in$

² Some of the workers in \mathcal{W} might be unavailable, or they may not share their location information with the MCSP. If this is the case, then the MCSP only assigns the task to a subset of the available workers.

$[K]$ and $\mathbf{x} := (x_1, \dots, x_K)$. We represent the performance and expected performance vectors associated with \mathbf{x} as $p(\mathbf{x}) := [p(x_1), \dots, p(x_K)]$ and $\mu(\mathbf{x}) := [\mu(x_1), \dots, \mu(x_K)]$. Based on the notation above, we define the random reward of a context vector associated with a given task.

Definition 1: Let t be a given task and \mathcal{W}_t be the set of available workers for it, together with the set of available contexts \mathcal{X}_t . Let $\mathbf{x} := [x_1, \dots, x_K]$, where $x_i \in \mathcal{X}_t$, for all $1 \leq i \leq K$. Then, the random reward of \mathbf{x} for task t is

$$U(p(\mathbf{x})) = U([p_{t,1}, \dots, p_{t,K}]),$$

where U is a problem-specific non-negative function of the vector \mathbf{x} .

Remark 1: The function U can be a linear function of the individual (random) outcomes for each worker or a more complicated non-linear function. We only make the assumption (A4) that the expected value of this reward can be expressed as a function of individual expected outcomes of workers. This assumption is not too restrictive and is satisfied by a wide range of functions if we know the distributions of the outcomes associated with individual workers without knowing their expectations [22].

Remark 2: In our setup, the reward of task assignment U only depends on the worker performance $p(x)$, which is the product of worker quality $q(x)$ and worker decision $a(x)$. Simply, the performance is 0 if the worker rejects the task and $q(x)$ if the worker accepts the task. A similar performance model is also used in the prior literature [40].

D. The Optimization Problem

We first define the task assignment problem as a combinatorial optimization problem.

Definition 2: Given task t with budget b_t and per worker price e_t , task context (l_t, c_t) , available worker set \mathcal{W}_t , and workers' contexts $(l_{t,w}, z_{t,w}), \forall w \in \mathcal{W}_t$, the optimal worker assignment for task t is

$$\begin{aligned} S_t^* &\in \arg \max_S \mathbb{E}[U(p(\mathbf{x}_{t,S}))] \\ &\text{subject to } S \in \mathcal{S}_t \\ &|S| \leq \min\{\lfloor b_t/e_t \rfloor, |\mathcal{W}_t|\}, \end{aligned} \quad (1)$$

where $\mathcal{S}_t \subseteq 2^{\mathcal{W}_t}$ is the set of *feasible* assignments in round t and $\mathbf{x}_{t,S} := (x_{t,w})_{w \in S}$.

Henceforth, we will denote by $\underline{m}_t := \min\{m_t, |\mathcal{W}_t|\}$ the maximum number of available workers that can be assigned to task t . We denote by $\mathcal{S} = \cup_{t \geq 1} \mathcal{S}_t$ the set of all possible feasible worker assignments. Moreover, we assume $\max_{t \geq 1} \underline{m}_t := K < \infty$.

The general formulation of the combinatorial optimization problem can assume different instances of applications, many of which are known to be NP-hard. Two notable examples are the Social Influence Maximization problem, shown to be NP-hard [48], and the Maximum Coverage problem with its derivatives [49]. Given an instance of the function U , we can comment on the NP-hardness of the problem. Let us take an example that

satisfies the assumptions we make on U , and is intuitive from the practical point of view. Given task $t \geq 1$ together with its associated available worker set \mathcal{W}_t , context set \mathcal{X}_t and maximum number of assignment \underline{m}_t , we let

$$U(p(\mathbf{x})) = \sum_{x_{t,w} \in \mathbf{x}: w \in \mathcal{W}_t} p(x_{t,w}).$$

Note that in this case we have $\mathbb{E}[U(p(\mathbf{x}))] = \sum_{x \in \mathcal{X}} \mu(x)$.³ Then, we can formulate the optimization problem (1) using the integer linear programming formulation of the Maximum Weighted Coverage problem, as follows.

$$\begin{aligned} & \max \sum_{x \in \mathcal{X}_t} \mu(x) y_x \\ \text{subject to } & y_x \leq \sum_{j=1}^{|\mathcal{X}_t|} \alpha_{xj} z_j, \quad \forall x \in \mathcal{X}_t \\ & \sum_{j=1}^{|\mathcal{X}_t|} z_j \leq \underline{m}_t \\ & y_x \in \{0, 1\}, \quad \forall x \in \mathcal{X}_t \\ & z_j \in \{0, 1\}, \quad \forall j \in [|\mathcal{X}_t|]. \end{aligned}$$

Here the sets are considered as singletons $x \in \mathcal{X}_t$, and the problem is to find the set of such singletons with maximum weighted coverage, that is, the set that maximizes the sum of weights of its elements. Above, y_x is 1 if x is covered and 0 otherwise, z_j is 1 if set j (in our case, singleton $x_j \in \mathcal{X}_t$) is selected and 0 otherwise, the variable α_{xj} is 1 if x is in set j (or $x = x_j$) and 0 otherwise. Note that the second constraint can be omitted (and therefore the last one) since we are considering the special case of singletons. The full formulation is given to show that this instance of the combinatorial bandit problem, in its offline version, is NP-hard, as an example of the Maximum Weighted Coverage problem.

Nevertheless, computationally efficient approximation oracles exist for a wide set of reward functions u . In this context, we say that a computation oracle is an α -approximation oracle, if given inputs \mathcal{W}_t and $\boldsymbol{\mu}_t := (\mu(x_{t,w}))_{w \in \mathcal{W}_t}$ it outputs $S_t^\alpha := \text{Oracle}(\boldsymbol{\mu}_t, \mathcal{S}_t, \underline{m}_t)$ such that $S_t^\alpha \in \mathcal{S}_t$ and

$$\mathbb{E}[U(p(\mathbf{x}_{t,S_t^\alpha}))] \geq \alpha \mathbb{E}[U(p(\mathbf{x}_{t,S_t^*}))].$$

Throughout the paper, we assume the MCSP has black-box access to an α -approximation oracle.

E. Task and Worker Properties

For the MCSP to learn the approximately optimal worker assignments for each new task, we make some mild assumptions about the structure of the worker qualities and the reward function.

Assumptions.

³ We abuse notation and write $x \in \mathbf{x}$ to mean that x is a component of vector \mathbf{x} .

- A1 The MCSP can assess the performance of a single worker's reply, i.e., if the MCSP assigns workers S_t to task t , then it observes $a(x_{t,w}), \forall w \in S_t$ and $q(x_{t,w})$ for $w \in S_t$ such that $a(x_{t,w}) = 1$.
- A2 Performances of different workers are independent, i.e., $p(x_{t,w}), w \in \mathcal{W}_t$ are independent random variables.
- A3 Expected performance is Lipschitz continuous in the joint context: $\forall x, x' \in \mathcal{X}$,

$$|\mu(x) - \mu(x')| \leq d(x, x')$$

where $d(x, x')$ denotes the distance between joint contexts x and x' .

- A4 Let $\Gamma_K([0, 1])$ denote the set of K -element subsets of $[0, 1]$ and $\Gamma([0, 1]) = \cup_{K=1}^W \Gamma_K([0, 1])$. Then, $\forall K \in [W]$, $\forall \mathbf{x} = [x_1, \dots, x_K]$ such that $x_w \in [0, 1], \forall w \in [K]$, we have $\mathbb{E}[U(p(\mathbf{x}))] = u(\boldsymbol{\mu}(\mathbf{x}))$, where $u: \Gamma([0, 1]) \rightarrow \mathbb{R}_+$.
- A5 $\forall S \in \mathcal{S}$, $\boldsymbol{\mu} = [\mu_1, \dots, \mu_{|S|}] \in [0, 1]^{|S|}$ and $\boldsymbol{\mu}' = [\mu'_1, \dots, \mu'_{|S|}] \in [0, 1]^{|S|}$, if $\mu_m \leq \mu'_m, \forall m \leq |S|$, then $u(\boldsymbol{\mu}) \leq u(\boldsymbol{\mu}')$, i.e., the expected reward is non-decreasing in the expected performances.
- A6 Expected task reward is Lipschitz continuous in the expected performances: $\exists B > 0$ such that for all $S \in \mathcal{S}$, $\boldsymbol{\mu} = [\mu_1, \dots, \mu_{|S|}] \in [0, 1]^{|S|}$ and $\boldsymbol{\mu}' = [\mu'_1, \dots, \mu'_{|S|}] \in [0, 1]^{|S|}$, we have

$$|u(\boldsymbol{\mu}) - u(\boldsymbol{\mu}')| \leq B \sum_{i=1}^{|S|} |\mu_i - \mu'_i|.$$

A1 and A2 are standard in the crowdsourcing literature [40], [42], [50], [51]. A3 merely says that worker-task pairs with similar joint contexts have similar expected performances. A4 states that the expected reward only depends on the expected performances of the workers. This assumption holds when the reward is a linear function of the performances of the workers or when the type of distributions of the performances are known, only the expectations of the performances are unknown, and the performances of different workers are independent [22]. A5 states that assigning workers with higher expected performances will never make the expected reward worse than what it was previously. Note that this does not reduce the problem into greedily selecting the workers with the highest observed performance because the oracle also considers the problem structure and the nature of the expected reward function. For example, note that in the simulation setup of Section VI-C, the expected reward function is the mutual information between the task and workers' locations. There, the oracle sequentially selects the workers that maximize the marginal reward with respect to the workers' performances already selected. Thus, in this scenario, greedy maximization is a suboptimal strategy. A6 holds for many different types of crowdsourcing tasks, including crowd solving tasks such as translation and retrieval tasks [50], [52], where the performances of the workers are additive, and tasks like cryptocurrency mining [53], where the task is completed if at least one of the assigned workers can complete the task.

F. The Learning Problem

The MCSP faces a sequential decision-making problem that proceeds over rounds indexed by t , where the following events happen in each round $t \in \{1, 2, \dots\}$:

- Task t arrives and the MCSP observes (b_t, l_t, c_t) .
- The MCSP identifies the set of available workers \mathcal{W}_t for task t .
- The MCSP selects an \underline{m}_t -element subset S_t of the workers in \mathcal{W}_t .
- At the end of round t , the MCSP observes $\forall w \in S_t$, $p(x_{t,w}) = a(x_{t,w})q(x_{t,w})$ and $u(p(\mathbf{x}_{t,S_t}))$.

Given $T \in \mathbb{N}$, the goal of the MCSP is to maximize its expected cumulative reward over T rounds by learning to match with tasks the right workers. We assume that the MCSP knows the form of u , which is directly related to the properties of the crowdsourcing task, but that it does not know μ . Since computing S_t^* that achieves an expected reward $\text{opt}_t := \max_{S \in \mathcal{S}_t} u(\mu(\mathbf{x}_{t,S}))$ is computationally intractable even when μ is perfectly known, we assume that the MCSP has access to an α -approximation oracle, which when given as inputs \mathcal{W}_t and $\boldsymbol{\mu}_t := (\mu(x_{t,w}))_{w \in \mathcal{W}_t}$ returns an α -optimal solution. Since the MCSP does not know $\boldsymbol{\mu}_t$ a priori, instead of $\boldsymbol{\mu}_t$ it gives some parameters $\boldsymbol{\theta}_t := (\theta_{t,w})_{w \in \mathcal{W}_t}$ as input to the α -approximation oracle to get $S_t = \text{Oracle}(\boldsymbol{\theta}_t, \mathcal{S}_t, \underline{m}_t)$, which is an approximately optimal solution under $\boldsymbol{\theta}_t$ but not necessarily under $\boldsymbol{\mu}_t$. As we will explain in Section IV, the learning algorithm of the MCSP chooses $\boldsymbol{\theta}_t$ judiciously in each round in order to balance exploration and exploitation.

We measure the performance of the MCSP by using the notion of α -approximation regret [22] (referred to as the regret hereafter). The regret for the first T tasks given $\{l_t, c_t, b_t, e_t, \{l_{t,w}, z_{t,w}\}_{w \in \mathcal{W}}, \mathcal{W}_t\}_{t=1}^T$ is defined as

$$R_\alpha(T) := \alpha \sum_{t=1}^T \text{opt}_t - \sum_{t=1}^T u(\mu(\mathbf{x}_{t,S_t})).$$

Note that $R_\alpha(T)$ is a random variable whose distribution depends on the randomness of the performances of the workers and the choices of the MCSP given $\{l_t, c_t, b_t, e_t, \{l_{t,w}, z_{t,w}\}_{w \in \mathcal{W}}, \mathcal{W}_t\}_{t=1}^T$.

Our goal is to construct a learning algorithm for the MCSP that minimizes the growth rate of the regret. In particular, if we can show that $R_\alpha(T) = O(T^\gamma)$ for some $\gamma < 1$, that will imply that $R_\alpha(T)/T \rightarrow 0$, and hence, in the limit as $T \rightarrow \infty$, the MCSP will accumulate an average that is at least α fraction of the highest possible average reward. Achieving sublinear regret for any given $\{l_t, c_t, b_t, e_t, \{l_{t,w}, z_{t,w}\}_{w \in \mathcal{W}}, \mathcal{W}_t\}_{t=1}^T$ is an extremely challenging problem since μ is not known by the MCSP a priori and the set of available workers and the contextual information can change arbitrarily in every round. The assumptions made in Section III-E are essential in building a robust learning algorithm that will work under this adversarial environment. In particular, they will allow the MCSP to adaptively partition the joint context space (\mathcal{X}, d) into regions, which are assumed to contain joint contexts with similar expected performances, and create partition-based estimates

of expected performances. In the next section, we list the properties of the context space and the reward function that follow from the assumptions in Section III-E.

G. Properties of the Context Space

We first define a well-behaved metric space.

Definition 3: (Well-behaved metric space [46]) A compact metric space (\mathcal{X}, d) is said to be well-behaved if there exists a sequence of subsets $(\mathcal{X}_h)_{h \geq 0}$ of \mathcal{X} satisfying the following properties:

- 1) Given $N \in \mathbb{N}$, each subset \mathcal{X}_h has N^h elements, i.e. $\mathcal{X}_h = \{\chi_{h,i}, 1 \leq i \leq N^h\}$ and to each element $\chi_{h,i}$ is associated a cell $X_{h,i} := \{x \in \mathcal{X} : d(x, \chi_{h,i}) \leq d(x, \chi_{h,j}), \forall j \neq i\}$.
- 2) For all $h \geq 0$ and $1 \leq i \leq N^h$, we have: $X_{h,i} = \bigcup_{j=N(i-1)+1}^{Ni} X_{h+1,j}$. The nodes $\chi_{h+1,j}$ for $N(i-1) + 1 \leq j \leq Ni$ are called the children of $\chi_{h,i}$, which in turn is referred to as the parent.
- 3) We assume that the cells have geometrically decaying radii, i.e. there exists $0 < \rho < 1$ and $0 < v_2 \leq 1 \leq v_1$ such that we have $B(\chi_{h,i}, v_2 \rho^h / 2) \subseteq X_{h,i} \subseteq B(\chi_{h,i}, v_1 \rho^h / 2)$, where $B(x, r)$ denotes a closed ball centered at x with radius r . Note that we have $v_2 \rho^h \leq \text{diam}(X_{h,i}) \leq v_1 \rho^h$, where $\text{diam}(X_{h,i}) := \sup_{x,y \in X_{h,i}} d(x, y)$.

The first property implies that for every $h \geq 0$ the cells $X_{h,i}, 1 \leq i \leq N^h$ partition \mathcal{X} . This can be observed trivially by reductio ad absurdum. The second property intuitively means that as h grows, we get a more refined sequence of partitions. The third property implies that the nodes $\chi_{h,i}$ are evenly spread out in the space. We assume the metric space of joint contexts (\mathcal{X}, d) is well-behaved. For instance, it was proven before that when d is the Euclidean norm, (\mathcal{X}, d) is well-behaved [46], [54].

Our regret bounds depend on the notion of approximate optimality dimension, which relates to the dimensions of sets of optimistic joint contexts that yield approximately optimal expected rewards. First, we define the approximate optimality dimension of the joint context space, tailored to our combinatorial setting, which is inspired by definitions of the near optimality dimension given in [46], [54], [55].

Definition 4: (The approximate optimality dimension)

- A subset \mathcal{X}_2 of \mathcal{X} is called r -separated if for any $x_1, x_2 \in \mathcal{X}_2$ we have $d(x_1, x_2) \geq r$. The cardinality of the largest such set is called the r -packing number of \mathcal{X} with respect to d , and is denoted by $M(\mathcal{X}, d, r)$. Equivalently, the r -packing number of \mathcal{X} is the maximum number of disjoint d -balls of radius r that are contained in \mathcal{X} .
- Let $\mathcal{Z} = \{\mathbf{x} \in \mathcal{X}^{|\mathcal{S}|} : S \in \mathcal{S}\}$. For any $\kappa > 0, r > 0, t > 0$ and $f : \mathbb{R}^+ \rightarrow \mathbb{R}^+$, we define the set

$$\begin{aligned} \mathcal{X}_{f(r)}^\kappa &:= \{x \in \mathcal{X} : \kappa - u(\mu(\mathbf{x})) \\ &\leq f(r), \text{ for some } \mathbf{x} \in \mathcal{Z} \text{ such that } x \in \mathbf{x}\} \end{aligned}$$

to be an $(f(r), u, \kappa)$ -optimal set. Let $M(\mathcal{X}_{f(r)}^\kappa, d, r)$ be its r -packing number. We define the (f, u, κ) -optimality

Algorithm 1: AOM-MC.

Input: \mathcal{X} , $(\mathcal{X}_h)_{h \geq 0}$, v_1, v_2, ρ, K, N, T .
Initialize: $C_0(\chi_{h,i}) = 0$, $\hat{\mu}_0(\chi_{h,i}) = 0$, $\forall \chi_{h,i} \in \mathcal{X}$; $\mathcal{X}_0 = \mathcal{X}$,
 $\mathcal{L}_1 = \{\chi_{0,1}\}$.
for arriving task $t = 1, 2, \dots$ **do**:
 Observe available workers in \mathcal{W}_t and their joint contexts \mathcal{X}_t .
 Identify available active leaf nodes $\mathcal{N}_t \subseteq \mathcal{L}_t$ and compute the indices of the workers in \mathcal{W}_t .
 $S_t \leftarrow \text{Oracle}((g_t(x_{t,w}))_{w \in \mathcal{W}_t}, \mathcal{S}_t, \underline{m}_t)$
 Observe performances of workers in S_t and collect the reward.
 Identify the set of selected nodes \mathcal{P}_t .
 for $\chi_{h,i} \in \mathcal{P}_t$ **do**:
 Update $\hat{\mu}_t(\chi_{h,i})$ as in (2) and $C_t(\chi_{h,i})$ as in (3).
 if $c_t(\chi_{h,i}) \leq v_1 \rho^h$ **then:** \triangleright Refine
 $\mathcal{L}_{t+1} \leftarrow \mathcal{L}_t \cup \{\chi_{h+1,j} : N(i-1) + 1 \leq j \leq Ni\} \setminus \{\chi_{h,i}\}$.
 end if
 end for
end for

dimension $D^u(f, \kappa)$ associated with $\mathcal{X}_{f(r)}^\kappa$ and u as follows:

$$D^u(f, \kappa) = \max \left\{ 0, \limsup_{r \rightarrow 0} \frac{\log(M(\mathcal{X}_{f(r)}^\kappa, d, r))}{\log(r^{-1})} \right\}$$

Note that if $\mathbf{x} \in \mathcal{Z}$ is such that $\kappa - u(\mu(\mathbf{x})) \leq f(r)$, then all elements of \mathbf{x} will be in $\mathcal{X}_{f(r)}^\kappa$. The use of the κ -optimality dimension allows us to bound the regret in a way that the time order of the regret depends on $D^u(f, \kappa)$, which can be strictly smaller than dimension D of the joint context space \mathcal{X} , as opposed to the prior work [43] that has bounds that depend on D . For instance, we can let $u_{\min}^* = \min_{t \in [T]} u(\mu(\mathbf{x}_{t, S_t^*}))$ and $\kappa = \alpha u_{\min}^*$ to obtain a worst-case approximate optimality dimension $\bar{D} = D^u(f, \alpha u_{\min}^*)$.

Remark 3: If we remove the volatility assumption from the problem setting and assume that the MCSP can choose only one worker in a given round, \bar{D} will be the near optimality dimension of the joint context space (where we let κ be the optimal expected reward), thus recovering the notion as introduced in previous works. In Appendix B we explain ways to construct more optimistic regret bounds using the approximate optimality dimension.

IV. A CONTEXT-AWARE TASK ASSIGNMENT ALGORITHM

Our algorithm is called *Adaptive Optimistic Matching for Mobile Crowdsourcing* (AOM-MC) and is motivated by several tree-based methods that have been used for function optimization under continuity assumptions [46], [54], [55] (pseudocode given in Algorithm 1).

Two main intuitive components characterize AOM-MC. First, note that to solve problem (1), we need to know the expected rewards associated with different assignments. As this is impossible, the algorithm uses its past random observations to construct good estimates of them based on the law of large numbers. To avoid suboptimal exploitation, it also adds an inflation term to the estimates, which incentivizes exploration of underexplored assignments. However, the CCV-MAB setting introduces additional problems that make the abovementioned

rationale futile. Note that the amount of exploration of a particular assignment is conditioned on its availability over time. Thus, some assignments may be hopelessly underexplored.

On the other hand, using the side information about assignments is beneficial since information never hurts. However, the continuous nature of these contexts makes the optimization over them hard in a discrete-time fashion. In worker-task assignment in crowdsourcing, the interaction between the worker and the task affects the worker's performance on the task. An arm is a task-worker pair whose context is the concatenation of the worker location, task location, and other relevant side information. Then, if the context space is only uniformly discretized, high-outcome context regions will not be correctly identified. This is because if the uniformly discretized regions are too large, then they will fail to capture the difference between high-outcome and low-outcome regions. Therefore, we employ adaptive discretization, which is the second main component of AOM-MC that enables differentiation of high-outcome and low-outcome regions. Based on the smoothness of the expected reward and the well-behavedness of the context space, we utilize the similarity information between nearby contexts and therefore do not need to learn all of them individually. That is the idea behind adaptively discretizing, only when we are confident that the region yields a high expected return. To that end, the estimates are kept with respect to regions (i.e., groups of similar assignments) and are updated based on the observations made on their elements. Below, we formally define these components and their interrelation.

By Definition 3, there exists a sequence $(\mathcal{X}_h)_{h \geq 0}$, each element of which containing N^h nodes whose associated cells form a tree of partitions of \mathcal{X} . The procedure is described as follows: for each arriving task t , the MCSP observes the available workers and the arrived joint contexts. The MCSP maintains an active set of leaf nodes denoted by \mathcal{L}_t . For the available workers, we identify the set of available active leaf nodes, whose regions contain the available contexts, and denote it by \mathcal{N}_t . By $\text{par}(\chi_{h,i})$ we denote the parent of node $\chi_{h,i}$. Fig. 1 illustrates the nodes and cells of a 1D adaptive discretization. For each active leaf node, we maintain an index which is an upper confidence bound on the maximum expected performance of the workers with contexts in the region associated with the node. The index is defined as

$$g_t(\chi_{h,i}) := \vartheta_t(\chi_{h,i}) + v_1 \rho^h$$

where the term $\vartheta_t(\chi_{h,i})$ is a high probability upper bound on $\mu(\chi_{h,i})$ defined as

$$\vartheta_t(\chi_{h,i}) := \min \{ \hat{\mu}_{t-1}(\chi_{h,i}) + c_{t-1}(\chi_{h,i}), \\ \hat{\mu}_{t-1}(\text{par}(\chi_{h,i})) + c_{t-1}(\text{par}(\chi_{h,i})) + v_1 \rho^{(h-1)} \}$$

and $c_t(\chi_{h,i}) := \sqrt{2 \log(\sqrt{K \sqrt{2} N T}) / C_t(\chi_{h,i})}$ is the confidence radius, tailored to give high probability upper bounds on μ .⁴ Here, $C_t(\chi_{h,i})$ is the number of times a worker associated

⁴ Note that here we assume the learner knows K beforehand. In practice, this is usually the case. The crowdsourcing experiments that we perform allow for such an assumption. We use K instead of W because, in reality, W can be in the orders of millions, while K may be less than 50.

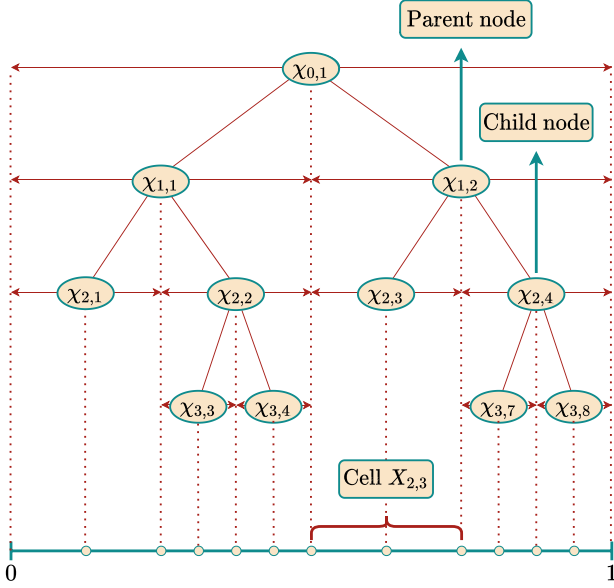


Fig. 1. Illustration of adaptive discretization in the case when $\mathcal{X} = [0, 1]$.

with a joint context from the cell $X_{h,i}$ was selected by the MCSP, formally defined as

$$C_t(\chi_{h,i}) := \sum_{t'=1}^t \sum_{k=1}^{|S_{t'}|} \mathbb{I}\{(H_{t',k}, I_{t',k}) = (h, i)\}$$

where we denote by $(H_{t',k}, I_{t',k})$ level and index of the active leaf node associated with the cell containing the context of the k th selected worker at time t' , and let $\phi_{t,k} := \chi_{H_{t,k}, I_{t,k}}$. We define the total performance accumulated by the algorithm until round t from selecting workers with contexts associated with the node $\chi_{h,i}$ as follows:

$$v_t(\chi_{h,i}) := \sum_{t'=1}^t \sum_{k=1}^{|S_{t'}|} r(x_{t',s_{t',k}}) \mathbb{I}\{(H_{t',k}, I_{t',k}) = (h, i)\}$$

where we denote by $s_{t,k}$ the k th worker selected by the algorithm at round t and $r(x_{t',s_{t',k}})$ represents the random performance of that worker. Consequently, we define the empirical mean used in the index as

$$\hat{\mu}_t(\chi_{h,i}) := \begin{cases} v_t(\chi_{h,i})/C_t(\chi_{h,i}) & \text{for } C_t(\chi_{h,i}) > 0 \\ 0 & \text{otherwise.} \end{cases}$$

Note that $C_t(\chi_{h,i})$ can be larger than t since the MCSP may choose workers with joint contexts belonging to the same cell in a certain round. However, it always holds that $C_t(\chi_{h,i}) \leq \sum_{t'=1}^t m_{t'} \leq Kt$. The constants v_1 and ρ are parameters as described in Definition 3 that are given as input to the algorithm.

Next, we define the index of a worker $w \in \mathcal{W}_t$ at round t . For this, let $(\tilde{H}_{t,w}, \tilde{I}_{t,w})$ represent level and index of the active leaf node associated with the cell containing the context $x_{t,w}$, and let $\tilde{\phi}_{t,w} := \chi_{\tilde{H}_{t,w}, \tilde{I}_{t,w}}$. We define the index of a worker $w \in \mathcal{W}_t$ as

$$g_t(x_{t,w}) := g_t(\tilde{\phi}_{t,w}) + N(v_1/v_2)v_1\rho^{\tilde{H}_{t,w}}$$

where the second term guarantees (with high probability) that $g_t(x_{t,w})$ upper bounds $\mu(x_{t,w})$.

Remark 4: Since at any round t , a cell associated with an active leaf node $\chi_{h,i}$ may contain several contexts of the available workers, we have $g_t(x_{t,w}) = g_t(x_{t,y})$ when w and y are two available workers, both of which have contexts that live in the cell $X_{h,i}$. As a consequence, indices of all worker contexts inside one cell are equal.

After the indices of the available workers, i.e., $\{g_t(x_{t,w})\}_{w \in \mathcal{W}_t}$ are computed, they are given as input θ_t to the approximation oracle in round t to obtain the super arm $S_t \subset \mathcal{W}_t$ that will be played in round t .⁵ At this point, we identify the active leaf nodes that are ‘‘selected,’’ denote their collection by \mathcal{P}_t , and update their statistics (after these are played and their outcomes are observed) according to the following rules. For each $\chi_{h,i} \in \mathcal{P}_t$:

$$\hat{\mu}_t(\chi_{h,i}) = \frac{C_{t-1}(\chi_{h,i})\hat{\mu}_{t-1}(\chi_{h,i}) + \text{rew}_t(\chi_{h,i})}{C_{t-1}(\chi_{h,i}) + \text{num}_t(\chi_{h,i})} \quad (2)$$

$$C_t(\chi_{h,i}) = C_{t-1}(\chi_{h,i}) + \text{num}_t(\chi_{h,i}). \quad (3)$$

where

$$\begin{aligned} \text{rew}_t(\chi_{h,i}) &:= \sum_{k=1}^{|S_t|} r(x_{t',s_{t',k}}) \mathbb{I}\{(H_{t',k}, I_{t',k}) = (h, i)\} \\ \text{num}_t(\chi_{h,i}) &:= \sum_{k=1}^{|S_t|} \mathbb{I}\{(H_{t',k}, I_{t',k}) = (h, i)\}. \end{aligned}$$

Statistics of the other active leaf nodes do not change. Subsequently, for each node $\chi_{h,i} \in \mathcal{P}_t$, we decide whether or not to expand it into N children nodes, according to the following condition:

- *Refine.* If $c_t(\chi_{h,i}) \leq v_1\rho^h$, then the node $\chi_{h,i}$ is expanded into N children nodes $\{\chi_{h+1,j} : N(i-1) + 1 \leq j \leq Ni\}$ which are added to the set of active leaves, whereas $\chi_{h,i}$ is removed from it.

If the above condition is not satisfied, we continue. Basically, we refine the partitions when we are confident enough about the μ values inside that cell. Fig. 2 visualizes the steps of our algorithm. Also, see the table of notation in Appendix C for a summarized terminology.

V. THEORETICAL PERFORMANCE ANALYSIS

A. Context Dependent Finite Time Upper Bounds on the Regret

Below, we state the main theoretical result of the paper, which shows that for a fixed time horizon T and a sequence of worker arrivals, the regret incurred by AOM-MC is sublinear in T and depends on the approximate optimality dimension of

⁵ The workers are randomly chosen in the first round.

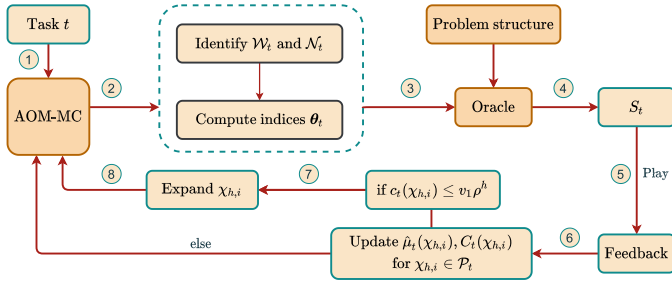


Fig. 2. Illustration of the AOM-MC algorithm. Once task t arrives, the learner identifies \mathcal{W}_t and \mathcal{N}_t and thereby computes the indices θ_t , which he then feeds to the oracle. The oracle then returns the α -optimal worker assignment, which the learner plays and from which he consequently observes semi-bandit feedback. Using this feedback, the indices of the nodes in \mathcal{P}_t corresponding to the played assignments are updated. Finally, the learner decides whether any nodes in \mathcal{P}_t need to be expanded and then proceeds to the next round.

the context space. The proof depends on several auxiliary results, which we state and prove in the Appendix A.

Theorem 1: Fix $T > 0$. Given the parameters of the problem $0 < \alpha \leq 1$, $N \in \mathbb{N}$, $K \in \mathbb{N}$, $B > 0$ and $0 < v_2 \leq 1 \leq v_1$, define $\bar{D} = D^\alpha(f, \alpha u_{\min}^*)$ and $f(r) = cr$, where $c = BK(6Nv_1/v_2 + 2)v_1/v_2$. Then, for any $D_1 > \bar{D}$, there exists $Q = Q(\mathcal{X}, u, \mu, \alpha u_{\min}^*, c) > 0$ (independent of T), for which the α -regret incurred by AOM-MC is at most

$$(C_1 + C_2) \cdot T^{1 - \frac{1}{D_1 + 2}} \cdot (\log(Tv_3))^{\frac{1}{D_1 + 2}},$$

with probability at least $1 - 1/T$, where $C_1 := 2QBK(6Nv_1/v_2 + 2) \frac{v_2 - D_1}{v_1(\rho - 1)}$ and $C_2 := KB(6Nv_1/v_2 + 2)v_1$.

Sketch of proof. The high-level intuition of the proof is as follows. First, we prove that the expected outcome of any given node is bounded above by its index. Then, using this result, we proceed by computing upper bounds on the actual deviation of the index of any given node from its true mean (expected value), and we give upper bounds on the maximum number of times a node can be selected before its expansion. We then show that the index of any given worker is an upper bound on the true mean associated with that worker and use this to obtain an upper bound on the simple regret. After this point, we use a technical result that relates the simple regret expressed in terms of the tree levels with the approximate optimality dimension. So far, the idea of the proof is clear, and the natural subsequent step would be to give an upper bound on the expected regret. However, to do this, we need to express the summation over rounds in terms of something else, lest we end up with linear bounds. For this, we split the summation over levels of the context tree and define a specific level of the tree which, using the aforementioned relation with the approximate optimality dimension, yields sublinear regret bounds. \square

Theorem 1 shows that AOM-MC achieves $\tilde{O}(T^{(D+1)/(D+2)+\epsilon})$ regret for any $\epsilon > 0$. This implies that the time-averaged regret converges to 0 at a rate that depends on the optimality dimension of the context space, making AOM-MC's average reward optimal in the long-run. Moreover, we always have $\bar{D} \leq D$, where the latter represents the context dimension. Indeed, based on the context arrival and reward structure \bar{D} can be smaller than D .

On the other hand, in our setup, the CC-MAB algorithm in [43] will incur $\tilde{O}(T^{(2+D)/(3+D)})$ regret, which is strictly larger than that of AOM-MC. In Section VI, we provide an extensive set of experimental evidence that corroborate our theoretical findings.

B. Computation and Memory Overhead

The required memory to run AOM-MC can be estimated by inspecting the number of statistics the algorithm updates. Note that in round t , the algorithm updates and stores $C_t(\chi_{h,i})$ and $\hat{\mu}_t(\chi_{h,i})$, for at most NK nodes from the set \mathcal{N}_t . This follows because we cannot expand more than K nodes in any given round, and each node will have N children. So in T rounds, the required storage would be linear in NKT , i.e., $O(NKT)$. Note that this is a loose upper bound because the algorithm only needs the leaf nodes and their parents' statistics. Thus, other nodes' statistics can be discarded from memory.

As for the time complexity, note that in each round t , the algorithm has to observe and go through every available worker in \mathcal{W}_t . Then, using linear search and using the fact that the total number of nodes by round t is upper bounded by $O(NKt)$, we would have a time complexity of $O(|\mathcal{W}_t|NKt)$ to identify the node to which each worker's joint task-worker context belongs. However, using a binary search-like algorithm that finds the appropriate leaf node for each context by starting from the root node and traversing the tree, we get a time complexity of $O(|\mathcal{W}_t| \log(NKt))$. Furthermore, in the case where the oracle greedily selects the α -optimal assignments according to the given indices, it would have to sort these indices out in every round, which would take $O(|\mathcal{W}_t| \log |\mathcal{W}_t|)$ time using the quick sort algorithm. At the end of the round, the algorithm goes through the selected nodes in \mathcal{P}_t to update their statistics and whether to refine them. This takes no more than $O(\underline{m}_t)$ time. Therefore, the time complexity of AOM-MC in T rounds is

$$O\left(\sum_{t \leq T} (|\mathcal{W}_t| \log(NKt) + |\mathcal{W}_t| \log |\mathcal{W}_t| + \underline{m}_t)\right) \\ = O(W_{\max} T \log T + W_{\max} \log W_{\max} T),$$

where $W_{\max} := \max_{t \leq T} |\mathcal{W}_t|$. As we can see, the time complexity is log-linear in the total number of rounds T and the maximum number of available workers in each round.

VI. NUMERICAL RESULTS

We perform three simulations to visually illustrate adaptive discretization, its importance and also to show that AOM-MC beats the current state-of-the-art algorithm for the CCV-MAB setting, CC-MAB of [43]. In the first simulation, we visualize the discretizations AOM-MC makes over a 2D context space, clearly showing that AOM-MC discretizes regions with high outcomes more than those with low outcomes. In the second simulation, we demonstrate the advantage of adaptive discretization in the NP-hard dynamic probabilistic maximum coverage problem by comparing AOM-MC with a version of CUCB [22] running on a pre-defined uniform discretization of the context space. Lastly, in the third simulation, we use location data

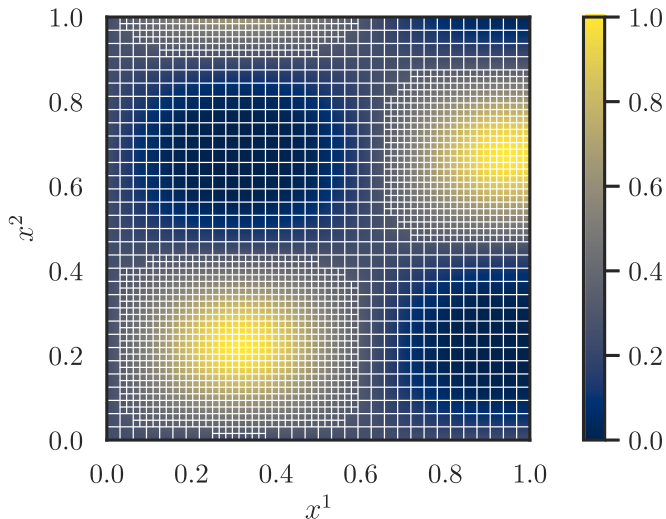


Fig. 3. Visualization of the regions of the leaves of AOM-MC by the end of Simulation I with uniform arrivals plotted over the heatmap of the expected outcome function.

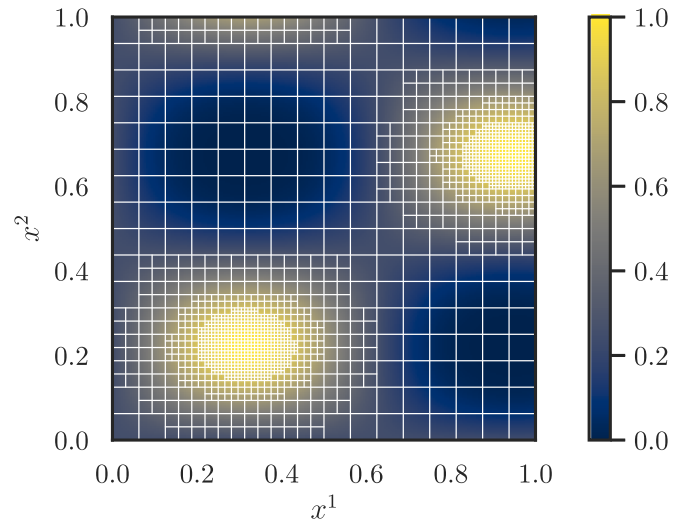


Fig. 4. Visualization of the regions of the leaves of AOM-MC by the end of Simulation I with non-uniform arrivals plotted over the heatmap of the expected outcome function.

from the Foursquare dataset [56] to run a time and location-dependent crowdsourcing simulation with realistic high-dimensional contexts and show that AOM-MC vastly outperforms CC-MAB. We ran all our simulations on a desktop computer equipped with an i7-6800 K, a GTX-1080Ti, and 32 GB of RAM. We provide the code for our algorithm, the implementation of all other used algorithms, as well as the code to generate the plots in our GitHub repository at github.com/Bilkent-CYBORG/AOM-MC.

A. Simulation I: Visualizing Adaptive Discretization

Setup: In this simulation, we consider a simple CCV-MAB problem where the number of arriving workers is Poisson distributed with mean 350 and each task-worker pair has a joint 2D context (i.e., $\mathcal{X} = [0, 1]^2$). Moreover, in each round, the MCSP must pick 100 workers (i.e., the task budget is $b_t = 100$ and each worker costs $e_t = 1$) to assign to the arrived task. Given a task-worker pair with joint context x whose components are x^1 and x^2 , respectively, the random quality of x is a Bernoulli random variable with probability given by $q(x) = (1 + \sin(5x^1) \sin(7x^2))^2/4$. Furthermore, the acceptance probability, $a(x)$, is always 1. Finally, the reward is a linear sum of all base arm performances.

Algorithms: We only run AOM-MC as it is the only algorithm with adaptive discretization. We set $v_1 = \sqrt{2}$, $v_2 = 1$, $\rho = 0.5$, and $N = 2^2 = 4$. The initial (root) context cell, $X_{0,1}$, is a two dimensional unit hypercube centered at $(0.5, 0.5)$.

Results: We first run the simulation with uniform context arrivals, where the context of each task-worker pair, x , is picked uniformly from $[0, 1]^2$. Next, we run the simulation 5 times using AOM-MC for 300000 rounds and average over the 5 runs. In Fig. 3, we plot the regions (squares) of the leaves by the end of the simulation over a heatmap of the expected outcome function. First, notice that each white square corresponds

to the region of a leaf, and the smaller the square, the deeper the node is in the tree, and thus the more the algorithm focused on and discretized the node's surrounding context region. Second, notice that the three regions of the context space corresponding to high-outcome, shown in yellow, have the smallest squares and are thus discretized the most. More specifically, AOM-MC discretized context regions corresponding to high outcomes one level deeper than those corresponding to low outcomes.

We then run the same simulation with non-uniform context arrivals, where x is sampled from a weighted mixture of two Gaussian distributions, one centered at the first major high-outcome region, $(0.314, 0.224)$, and the other centered at the second major high-outcome region, $(0.942, 0.673)$, with weights 0.6 and 0.4, respectively. Both distributions have a covariance matrix of $\begin{pmatrix} 0.02 & 0 \\ 0 & 0.01 \end{pmatrix}$. Note that such a non-uniform arrival is equivalent to many high-quality worker arrivals and, consequently, a small number of low-quality worker arrivals. Fig. 4 shows the plot of the leaf regions at the end of the simulation over the heatmap of the outcome function. Compared with uniform arrivals, low-outcome (blue) regions are discretized less, and high-outcome regions are discretized more, evident by the larger and smaller node regions (white squares), respectively. Furthermore, the nodes corresponding to high-outcome regions are three levels deeper than those corresponding to low-outcome regions. This shows that in a setting where there is a larger proportion of high-outcome to low-performance workers (e.g., skilled workers in crowdsourcing), AOM-MC takes advantage of the large number of high-performance worker arrivals and can very finely discretize the regions corresponding to them.

Since in many crowdsourcing problems contexts (including location) are continuous, adaptive discretization would help AOM-MC accurately identify high-quality context regions and thus pair up optimal workers with each arriving task.

B. Simulation II: Adaptive Versus Uniform Discretization

Setup: In this simulation, we illustrate the advantages of adaptive discretization over uniform discretization by considering the dynamic probabilistic maximum coverage problem (DPMC). In the DPMC problem, the goal in each round is to select a subset of the left nodes of a bipartite graph to trigger as many right nodes, where a left node triggers a right node with a probability given by the edge connecting them. In the context of crowdsourcing, left nodes can be thought of as sub-tasks and right nodes as workers. Thus, the goal is to assign sub-tasks to workers optimally.

More specifically, in each round, the number of left and right nodes are sampled from a Poisson distribution with mean 50 and 300, respectively. Then, each left node is connected to a random number of right nodes, where the number of nodes is picked uniformly from $[1,4]$. Each edge has a 3D context picked uniformly from $[0, 1]^3$, and given edge context x with components x^1 , x^2 , and x^3 , the probability of the right node connected to the edge being triggered given that the left node is picked is given by $q(x) = Af((1 + \sin(7x^1) \cos(8x^2) \sin(9x^3))/2)$, where f is the PDF of a zero-mean Gaussian distribution with standard deviation 0.15; and $A = 3\sqrt{2\pi}/20$ is the normalizing constant.

In each round, the learner observes the arriving left nodes, right nodes, and edges and must choose a subset of K many left nodes, where K is randomly sampled from $[4,10]$. After the learner picks a subset of left nodes, Bernoulli random variables with probabilities given by the outgoing edges of the picked left nodes are sampled, and the triggered right nodes are determined. Then, the final reward is the number of triggered right nodes. Note that even if a right node is triggered by multiple left nodes (i.e., the Bernoulli samples of multiple edges connected to the right node are 1), the right node is only counted once in the final reward.

Algorithms: We run this simulation using our algorithm and a modified version of CUCB adapted to the volatile and contextual nature of this simulation setting. The details of each algorithm are given below.

AOM-MC: We set $v_1 = \sqrt{3}$, $v_2 = 1$, $\rho = 0.5$, and $N = 2^3 = 8$. The initial (root) context cell, $X_{0,1}$, is a three dimensional unit hypercube centered at $(0.5, 0.5, 0.5)$. We use TIM+ as the oracle with $\epsilon = 0.1$ and $l = 1$ [57]. The TIM+ algorithm is an approximate algorithm that solves the DPMC problem. Notice that the DPMC problem is NP-hard in general, and thus even with all edge probabilities known, no polynomial-time algorithm exists for determining which left nodes to select. Instead, the TIM+ algorithm is an (α, β) -approximate polynomial-time algorithm, with $\alpha = 1 - 1/e - \epsilon$ and $\beta = 1 - 3n^{-l}$, meaning that its solution is guaranteed to be α times the true solution β portion of the time, with $0 < \alpha, \beta < 1$. TIM+ takes as input the graph and also the weight (i.e., probability) of each edge, and outputs the set of left nodes to select. In the context of our algorithm, the TIM+ algorithm acts as our oracle. In other words, the probabilities of the edges come from the indices computed by our algorithm and act as estimate probabilities. Notice that even though TIM+ is an (α, β) -oracle and not an α -oracle, AOM-MC has no problems using it in practice, as we will see in the results section. Lastly,

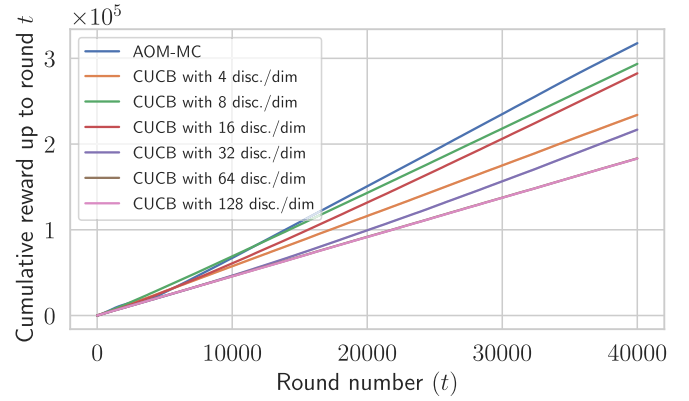


Fig. 5. The cumulative rewards of AOM-MC and different instances of CUCB for Simulation II.

since AOM-MC indices are in general not smaller than one, we normalize them to $[0,1]$ before passing them to TIM+. This normalization is monotone and thus the context with the highest index will still have the highest index after normalization.

CUCB: CUCB is a non-contextual combinatorial MAB algorithm and thus cannot directly be used for the DPMC problem [22]. Instead, we discretize the context space into small regions (cubes) a priori and treat each region as an arm. For example, if we discretize each dimension into two parts $[0,0.5]$ and $[0.5,1.0]$, then in total the 3D context space will be discretized into eight cubes of length 0.5. Then, each one of the eight cubes will be treated as an arm by CUCB. More specifically, for each sub-task-worker pair (i.e., edge), we feed to TIM+ the index (i.e., $\bar{\mu}$ in [22]) of the region to which the sub-task-worker's joint context belongs. In our simulation, we run different instances of CUCB with the following discretization per dimension (disc./dim) factors: $\{4, 8, 16, 32, 64, 128\}$.

Results: We run the simulation for 40000 rounds and average the results over 5 independent runs. We track the reward of each round for each algorithm and cumulatively add them up to get Fig. 5. AOM-MC outperforms all variations of CUCB starting from $t = 14000$ until the end of the simulation. Its slow start can be attributed to the exploration needed at the beginning of the simulation due to AOM-MC's tree starting with just one node whose region is $[0, 1]^3$. As more nodes are created, and AOM-MC gains a better idea about the estimated outcome of each node, it explores less and begins to dominate over CUCB in terms of reward.

Looking at the different CUCB instances, we notice that at first, there is an increase in reward when going from 4 to 8 disc./dim, but then a drop in reward when going to 16 disc./dim, with further increases in discretization decreasing the reward even more. As discretizations are increased, CUCB tends to explore more, thus hindering its performance. For example, with 128 disc./dim, there are $128^3 = 2097152$ many regions (i.e., arms from the point of view of CUCB), and given that there are on average $(1 + 4)/2 \times 50 = 125$ edges in each round, we would expect CUCB to have played a base arm from each region once after $2097152/125 \approx 16777$ rounds! In short, CUCB with 128 disc./dim is likely exploring for the entirety of the simulation. Furthermore, although AOM-MC's

deepest nodes reached a height of $h = 7$, thus having the same length as the cubes of CUCB with 128 disc./dim, AOM-MC performs much better thanks to its adaptive discretization, because of which it only discretizes high-outcome regions.

In conclusion, even with a pre-defined discretization tuned a priori, something not possible in real-life online settings, CUCB is still beaten by AOM-MC's adaptive discretization.

C. Simulation III: Time and Location-Dependent Crowdsourcing Simulation Using Real-Life Data

Setup: In our last simulation, we consider a time and location-dependent crowdsourcing setting where the goal is to assign the best workers to arriving tasks. In each round, a task arrives with a 2D location (longitude and latitude), a preference vector $[0, 1]^2$, and the time when the task was posted in the unit of seconds since the start of the current day, scaled to $[0, 1]$. Thus, midnight is mapped to 0, and 23:59:59 is mapped to 1. Furthermore, a group of T_0 many tasks arrives within the same hour. In other words, tasks $1, 2, \dots, T_0$ arrive between 00:00:00 and 00:59:59. Then, the next group of T_0 many tasks ($T_0 + 1, \dots, 2T_0$) arrive between 01:00:00 and 01:59:59. In our simulation, we set $T_0 = 3$ and also randomly sample the arrival time from a uniform distribution.

Each task has a different budget of workers assigned to it, and after a task arrives, available workers are revealed to the learner. Each worker also has a 2D location as well as a 2D preference vector. The learner then picks as many workers as the task budget allows, after which the workers either reject or accept the task. We assume that workers are most likely to accept a task when it is daytime and least likely to do so when it is nighttime. Mathematically, given that τ is the scaled arriving time of an arrived task, we define a worker accepting a task as a Bernoulli random variable with probability

$$a(\tau) := \frac{1 + \sin(2\pi\tau - \frac{\pi}{2})}{2}. \quad (4)$$

Notice that the acceptance probability is small when τ is close to 0 or 1, which both correspond to nighttime, and it is large when τ is close to 0.5, which corresponds to noon. Then, the workers who accepted the task start working on the task, each either completing it or not. Then, given a joint task-worker context (including time of day) $x \in \mathcal{X} = [0, 1]^9$, we define the performance of a worker who has accepted the task as Bernoulli random variable with probability given by the cosine similarity between the task and worker preference vectors, scaled to $[0, 1]$. Formally, given that $p(x)$ is the expected performance (i.e., probability of successfully completing the task) of any worker, we have

$$p(x) = \begin{cases} \left(1 + \frac{\langle x_{\text{task pref.}}, x_{\text{worker pref.}} \rangle}{\|x_{\text{task pref.}}\| \|x_{\text{worker pref.}}\|}\right) / 2 & \text{if worker accepts task,} \\ 0 & \text{otherwise.} \end{cases} \quad (5)$$

We sample task t 's budget, b_t , from $\{3, 4, 5, 6, 7, 8\}$, assume each worker has a fixed cost of $e_t = 1$, and also sample the task and worker preferences as well as the worker location uniformly from $[0, 1]^2$, but we use the Foursquare dataset for the task locations. The Foursquare dataset contains check-in data from New York City (227,428 check-ins) and Tokyo (573,703 check-ins) for a period of 10 months from April 2012 to February 2013 [56]. Each check-in comes with a location tag as well as the time of the check-in. In our simulation, we sample the task locations from the Tokyo check-in locations and scale both the longitude and latitude to $[0, 1]$.

Finally, to model a realistic crowdsourcing reward function that considers the distance between the worker and task in a complex and non-uniform manner, we define our reward function as the mutual information between the task location and successful worker locations. More specifically, we first discretize the location space by randomly sampling 14450 points from $[0, 1]^2$ to get a list of discretized scaled location points. We chose 14450 because it was the largest number of discretizations we could perform before our PC ran out of memory. Of the 14450 discretized (scaled) location points, we reserve 80% of them for worker locations and the remaining 20% for task locations. Then, we perform the following step to generate the data for each arriving task and its workers:

- 1) Sample the task's location from the Foursquare dataset and sample the task's preference vector uniformly from $[0, 1]^2$.
- 2) Uniformly sample K many worker locations from the worker-reserved discretized location points that are in the disk centered at the task location with radius $\sqrt{2}/4$, where K is sampled from a Poisson distribution with mean 50. To do this, we first sample 5 K worker locations from the worker-reserved discretized location points and then pick K that are inside the disk. If the number of workers inside the disk is less than K , we resample. We limit worker locations to the disk to simulate a realistic location-dependent crowdsourcing setting where only workers close enough to the task are available to be selected. Then, we sample the available worker preferences from $[0, 1]^2$.
- 3) Save the task-reserved discretized location points inside the square of length $\sqrt{2}/8$ centered at the task location. These points will be later used to compute the reward.

Fig. 6 shows a visualization of the relevant task and worker locations for a random task.

Then, after the learner has observed and selected workers for a task, the reward is determined as follows: First, the worker acceptances are sampled from Bernoulli distributions with probabilities given in (4). Then, the worker performances are sampled from Bernoulli distributions with probabilities given in (5). Finally, assuming that a function from a Gaussian Process (GP) dictates the relationship between different locations in the location space, the mutual information between the task square discretization points and the successful workers' discretized location points is computed as the reward. More specifically, in our simulations, we assume that the location relationship is explained by a function sampled from

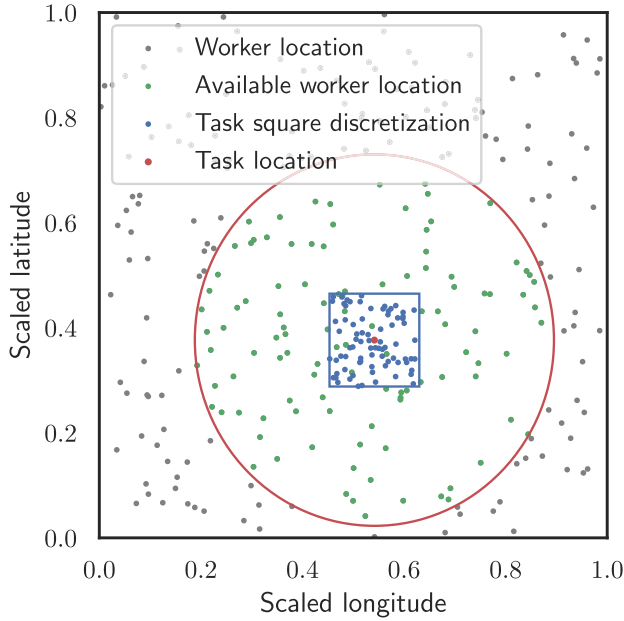


Fig. 6. A visualization of the relevant locations for a random task and its workers in Simulation III. Worker locations are the location of all workers that were sampled for the task, whereas available worker locations are the location of available workers (i.e., workers whose distances to the task are less than $\sqrt{2}/4$). Task square discretization indicates the task-reserved discretized location points inside the blue square that is centered at the task location with length $\sqrt{2}/8$.

$\mathcal{GP}(0, k(x, x'))$, where k is the Matern kernel with lengthscale and variance 0.1. Note that this assumption is intuitive because the Matern kernel is decreasing in the distance between x and x' , similar to how the farther the workers are from a task in a location-dependent crowdsourcing setting, the lower the final performance of them. To find the mutual information between the location of a set of successful workers and task square discretizations, we compute the covariance matrix of their locations. Notice that our discretized location points follow a multi-variate Gaussian distribution with zero mean and covariance matrix computed using the Matern kernel because of our GP assumption. Therefore, mathematically, the mutual information between the successful workers' locations and the task square discretizations is $\frac{1}{2} \log \left(\frac{\det \Sigma_X \det \Sigma_Y}{\det \Sigma_{X,Y}} \right)$, where Σ_X is the covariance matrix of the successful workers' discretized location points, Σ_Y is the covariance matrix of the task square discretizations, and $\Sigma_{X,Y}$ is the joint covariance matrix of the successful workers' discretized location points and the task square discretizations.

Algorithms: We run this simulation using our algorithm, CC-MAB, modified versions of CUCB and ϵ_n -greedy, and a random algorithm.

AOM-MC: We set $v_1 = \sqrt{9}$, $v_2 = 1$, $\rho = 0.5$, and $N = 2^9 = 512$. The initial (root) context cell, $X_{0,1}$, is a nine dimensional unit hypercube. Furthermore, we use a marginal reward maximizing greedy oracle that takes as input the indices of each worker, picks the worker whose index would maximize the reward function (i.e., maximizes the index times the information gain between the worker's location and the task square discretizations), and then picks the next worker and so on.

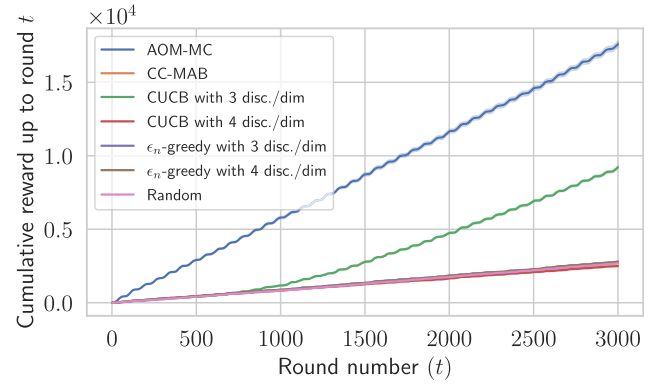


Fig. 7. The cumulative rewards of AOM-MC, CC-MAB, CUCB, ϵ_n -greedy, and Random in Simulation III.

CC-MAB: CC-MAB is another combinatorial contextual volatile bandit algorithm, but unlike AOM-MC, it starts with a pre-discretized context space and never refines its discretizations. Also, unlike AOM-MC, CC-MAB does not use a general α -oracle and instead has its own marginal reward maximizing greedy oracle that in our setup is an $(1 - 1/e)$ -oracle. Then, since we run our simulations for 3000 rounds, we set $h_T = \left\lceil 3000^{\frac{1}{3(1-1/e)+9}} \right\rceil = 3$. Hence, each hypercube has a length of $1/h_t = 1/3$.

CUCB: We also use the modified version of CUCB as described in Simulation II. CUCB uses the same marginal reward maximizing oracle as AOM-MC. We run different instances of CUCB with the following discretization per dimension (disc./dim) factors: $\{3, 4\}$.

ϵ_n -greedy: ϵ_n -greedy is a simple single-armed bandit algorithm that randomly explores and exploits, where the probability of random exploration decreases as the round number increases [10]. Since our crowdsourcing setting is combinatorial and volatile, the original ϵ_n -greedy as described in [10] cannot be used. Instead, we modify it similar to how we modified CUCB: we discretize the context space into hypercubes and treat each hypercube as an arm. We set $c = 0.1$ and $d = 0.2$ and we also run different instances of ϵ_n -greedy with the following discretization per dimension (disc./dim) factors: $\{3, 4\}$.

Random: The random algorithm picks random workers to assign to each task.

Results: We run the simulation for 5 tries and report the cumulative reward in Fig. 7. Unsurprisingly, Random and both ϵ_n -greedy algorithms perform the worst. Note that ϵ_n -greedy does not use any exact or approximate oracle and simply picks workers from hypercubes with the largest indices or randomly, depending on whether it is exploiting or exploring, respectively. Therefore, it is oblivious to any information about the problem instance. For example, it may pick two very close workers, which would be suboptimal in our crowdsourcing setup. This contrasts to CUCB, which uses an $(1 - 1/e)$ -approximate oracle, helping it perform much better than ϵ_n -greedy, but only with 3 disc./dim. The reason for the performance discrepancy between CUCB with 3 disc./dim and CUCB with 4 disc./dim is that when each dimension is discretized into 4 segments, then in total, there are $4^9 = 262144$ hypercubes. Thus, given that the expected number of workers in each

round is 50, CUCB would require $262144/50 \approx 5243$ rounds to explore a worker from every hypercube just once. In other words, it is exploring for the entire duration of the simulation. However, when there are 3 discretizations per dimension, CUCB only needs 394 rounds to finish its exploration.⁶

Surprisingly, CC-MAB also performs among the worst and just as badly as Random. This is surprising because CC-MAB discretizes each dimension of the context space into 3 segments, just like the favorable-performing CUCB with 3 disc./dim (green in Fig. 7), yet it performs much worse than CUCB. Moreover, both CC-MAB and CUCB use the same marginal reward maximizing oracle. In fact, the only difference between them is that CC-MAB performs exploration by assembling a set of under-explored arms and randomly selecting arms from them. On the other hand, CUCB and our algorithm use a UCB term to account for exploration. Thus, it appears that for this crowdsourcing setup, a UCB-based exploration is more apt.

Compared with the other algorithms, AOM-MC achieves a higher final cumulative reward and quickly outperforms them from the first few hundred rounds. AOM-MC performs better than CUCB with 3 disc./dim because of CUCB's coarse discretization. CUCB only discretizes each dimension into 3 parts; hence it cannot tell the difference between a context with its first element as 0.7 and another with its first element as 0.95. Moreover, its discretization is fixed and does not improve throughout the simulation. Thus, CUCB can never distinguish between the two contexts mentioned above and performs suboptimally compared with AOM-MC, which adaptively discretizes the context space. Even though at the beginning of the simulation, AOM-MC starts with a very coarse discretization of just one hypercube for the entire context space, it quickly discretizes the context regions where it believes high-performance workers to be. In fact, its tree reaches a height of $h = 2$ just after 42 rounds, after which the length of the hypercubes of the leaf nodes is $1/4$, which is smaller than those of CC-MAB and CUCB with 3 disc./dim. Lastly, even though when an AOM-MC node is refined, it is split into 512 children nodes, AOM-MC ended up with fewer nodes by the end of the simulation than what CC-MAB started (and ended) with (16560 versus $3^9 = 19683$). This is because AOM-MC only refines nodes whose region it believes to be a high-outcome region. In conclusion, not only does AOM-MC substantially outperform the other algorithms, but it also does so by storing information about fewer hypercubes and thus using less memory.

VII. CONCLUSION

In this paper, we proposed AOM-MC, an online learning algorithm that solves the CCV-MAB problem and incurs sublinear regret. We also proved the regret bounds for our algorithm and offered time- and space-complexity analyses. The CCV-MAB framework that we considered is very general in that it can accommodate many real-life scenarios and consider many features. Most importantly, we described in detail

how the CCV-MAB framework could be applied to the problem of mobile crowdsourcing and presented the results of our extensive simulations. Our simulations illustrated the advantage of adaptive discretization compared with uniform discretization and showcased how our algorithm outperforms the state-of-the-art MAB algorithm and baseline crowdsourcing algorithms.

REFERENCES

- [1] Y. Li, F. Li, S. Yang, P. Zhou, L. Zhu, and Y. Wang, "Three-stage stackelberg long-term incentive mechanism and monetization for mobile crowdsensing: An online learning approach," *IEEE Trans. Netw. Sci. Eng.*, vol. 8, no. 2, pp. 1385–1398, April–Jun. 2021.
- [2] J. Xiong, X. Chen, Q. Yang, L. Chen, and Z. Yao, "A task-oriented user selection incentive mechanism in edge-aided mobile crowdsensing," *IEEE Trans. Netw. Sci. Eng.*, vol. 7, no. 4, pp. 2347–2360, Oct.–Dec. 2020.
- [3] T. Li, T. Jung, Z. Qiu, H. Li, L. Cao, and Y. Wang, "Scalable privacy-preserving participant selection for mobile crowdsensing systems: Participant grouping and secure group bidding," *IEEE Trans. Netw. Sci. Eng.*, vol. 7, no. 2, pp. 855–868, Apr.–Jun. 2020.
- [4] H. Wu, L. Wang, and G. Xue, "Privacy-aware task allocation and data aggregation in fog-assisted spatial crowdsourcing," *IEEE Trans. Netw. Sci. Eng.*, vol. 7, no. 1, pp. 589–602, Jan.–Mar. 2020.
- [5] B. Zhao, Y. Wang, Y. Li, Y. Gao, and X. Tong, "Task allocation model based on worker friend relationship for mobile crowdsourcing," *Sensors*, vol. 19, no. 4, 2019, Art. no. 921.
- [6] F. Alt, A. S. Shirazi, A. Schmidt, U. Kramer, and Z. Nawaz, "Location-based crowdsourcing: Extending crowdsourcing to the real world," in *Proc. 6th Nordic Conf. Hum.-Comput. Interact.: Extending Boundaries*, 2010, pp. 13–22.
- [7] D. Wu, Y. Zhang, L. Bao, and A. C. Regan, "Location-based crowdsourcing for vehicular communication in hybrid networks," *IEEE Trans. Intell. Transp. Syst.*, vol. 14, no. 2, pp. 837–846, Jun. 2013.
- [8] W. R. Thompson, "On the likelihood that one unknown probability exceeds another in view of the evidence of two samples," *Biometrika*, vol. 25, no. 3/4, pp. 285–294, 1933.
- [9] T. L. Lai and H. Robbins, "Asymptotically efficient adaptive allocation rules," *Adv. Appl. Math.*, vol. 6, no. 1, pp. 4–22, 1985.
- [10] P. Auer, N. Cesa-Bianchi, and P. Fischer, "Finite-time analysis of the multi-armed bandit problem," *Mach. Learn.*, vol. 47, no. 2–3, pp. 235–256, 2002.
- [11] R. Agrawal, "Sample mean based index policies by o (log n) regret for the multi-armed bandit problem," *Adv. Appl. Probability*, vol. 27, no. 4, pp. 1054–1078, 1995.
- [12] S. Agrawal and N. Goyal, "Analysis of Thompson sampling for the multi-armed bandit problem," in *Proc. 25th Annu. Conf. Learn. Theory*, 2012, pp. 39.1–39.26.
- [13] D. Russo and B. Van Roy, "Learning to optimize via posterior sampling," *Math. Oper. Res.*, vol. 39, no. 4, pp. 1221–1243, 2014.
- [14] J. Langford and T. Zhang, "The epoch-greedy algorithm for contextual multi-armed bandits," in *Proc. Adv. Neural Inf. Process. Syst.*, 2007, pp. 817–824.
- [15] T. Lu, D. Pál, and M. Pál, "Contextual multi-armed bandits," in *Proc. 13th Int. Conf. Artif. Intell. and Statist.*, 2010, pp. 485–492.
- [16] W. Chu, L. Li, L. Reyzin, and R. Schapire, "Contextual bandits with linear payoff functions," in *Proc. 14th Int. Conf. Artif. Intell. Statist.*, 2011, pp. 208–214.
- [17] A. Slivkins, "Contextual bandits with similarity information," *J. Mach. Learn. Res.*, vol. 15, no. 1, pp. 2533–2568, 2014.
- [18] L. Li, W. Chu, J. Langford, and R. E. Schapire, "A contextual-bandit approach to personalized news article recommendation," in *Proc. 19th Int. Conf. World Wide Web*, 2010, pp. 661–670.
- [19] A. Tewari and S. A. Murphy, "From ads to interventions: Contextual bandits in mobile health," in *Mobile Health*. Cham, Switzerland: Springer, 2017, pp. 495–517.
- [20] N. Cesa-Bianchi and G. Lugosi, "Combinatorial bandits," *J. Comput. Syst. Sci.*, vol. 78, no. 5, pp. 1404–1422, 2012.
- [21] Y. Gai, B. Krishnamachari, and R. Jain, "Combinatorial network optimization with unknown variables: Multi-armed bandits with linear rewards and individual observations," *IEEE/ACM Trans. Netw.*, vol. 20, no. 5, pp. 1466–1478, Oct. 2012.
- [22] W. Chen, Y. Wang, and Y. Yuan, "Combinatorial multi-armed bandit: General framework and applications," in *Proc. 30th Int. Conf. Mach. Learn.*, 2013, pp. 151–159.

⁶ Note that since the context arrivals are not uniform, the actual number of rounds needed before CUCB explores every hypercube is likely much larger than 394.

- [23] B. Kveton, Z. Wen, A. Ashkan, and C. Szepesvari, "Tight regret bounds for stochastic combinatorial semi-bandits," in *Proc. 18th Int. Conf. Artif. Intell. Statist.*, 2015, pp. 535–543.
- [24] B. Kveton, C. Szepesvari, Z. Wen, and A. Ashkan, "Cascading bandits: Learning to rank in the cascade model," in *Proc. 32nd Int. Conf. Mach. Learn.*, 2015, pp. 767–776.
- [25] W. Chen, Y. Wang, Y. Yuan, and Q. Wang, "Combinatorial multi-armed bandit and its extension to probabilistically triggered arms," *J. Mach. Learn. Res.*, vol. 17, no. 1, pp. 1746–1778, 2016.
- [26] A. Huyuk and C. Tekin, "Analysis of thompson sampling for combinatorial multi-armed bandit with probabilistically triggered arms," in *Proc. 22nd Int. Conf. Artif. Intell. Statist.*, 2019, pp. 1322–1330.
- [27] F. Radlinski, R. Kleinberg, and T. Joachims, "Learning diverse rankings with multi-armed bandits," in *Proc. 25th Int. Conf. Mach. Learn.*, 2008, pp. 784–791.
- [28] P. Yang, N. Zhang, S. Zhang, K. Yang, L. Yu, and X. Shen, "Identifying the most valuable workers in fog-assisted spatial crowdsourcing," *IEEE Internet Things J.*, vol. 4, no. 5, pp. 1193–1203, Oct. 2017.
- [29] R. Kleinberg, A. Niculescu-Mizil, and Y. Sharma, "Regret bounds for sleeping experts and bandits," *Mach. Learn.*, vol. 80, no. 2-3, pp. 245–272, 2010.
- [30] F. Li, J. Liu, and B. Ji, "Combinatorial sleeping bandits with fairness constraints," *IEEE Trans. Netw. Sci. Eng.*, vol. 7, no. 3, pp. 1799–1813, Jul.–Sep. 2020.
- [31] Z. Bnaya, R. Puzis, R. Stern, and A. Felner, "Volatile multi-armed bandits for guaranteed targeted social crawling," in *Proc. Workshops 27th AAAI Conf. Artif. Intell.*, 2013, pp. 16–21.
- [32] D. Chakrabarti, R. Kumar, F. Radlinski, and E. Upfal, "Mortal multi-armed bandits," in *Proc. Adv. Neural Inf. Process. Syst.*, 2009, pp. 273–280.
- [33] A. Jain, A. D. Sarma, A. Parameswaran, and J. Widom, "Understanding workers, developing effective tasks, and enhancing marketplace dynamics: A study of a large crowdsourcing marketplace," *Proc. VLDB Endowment*, vol. 10, no. 7, pp. 829–840, 2017.
- [34] A. Nika, S. Elahi, and C. Tekin, "Contextual combinatorial volatile multi-armed bandit with adaptive discretization," in *Proc. 23rd Int. Conf. Artif. Intell. Statist.*, (ser. Proceedings of Machine Learning Research), S. Chiappa and R. Calandra, Eds., vol. 108, Aug. 2020, pp. 1486–1496. [Online]. Available: <http://proceedings.mlr.press/v108/nika20a.html>
- [35] G. Gao, H. Huang, M. Xiao, J. Wu, Y.-e. Sun, and Y. Du, "Budgeted unknown worker recruitment for heterogeneous crowdsensing using CMAB," *IEEE Trans. Mobile Comput.*, to be published, doi: [10.1109/TMC.2021.3064324](https://doi.org/10.1109/TMC.2021.3064324).
- [36] U. ul Hassan and E. Curry, "Efficient task assignment for spatial crowdsourcing: A combinatorial fractional optimization approach with semi-bandit learning," *Expert Syst. Appl.*, vol. 58, pp. 36–56, 2016.
- [37] Y. Wu, F. Li, L. Ma, Y. Xie, T. Li, and Y. Wang, "A context-aware multi-armed bandit incentive mechanism for mobile crowd sensing systems," *IEEE Internet Things J.*, vol. 6, no. 5, pp. 7648–7658, Oct. 2019.
- [38] H. Zhao, M. Xiao, J. Wu, Y. Xu, H. Huang, and S. Zhang, "Differentially private unknown worker recruitment for mobile crowdsensing using multi-armed bandits," *IEEE Trans. Mobile Comput.*, vol. 20, no. 9, pp. 2779–2794, Sep. 2021.
- [39] X. Gao, S. Chen, and G. Chen, "MAB-based reinforced worker selection framework for budgeted spatial crowdsensing," *IEEE Trans. Knowl. Data Eng.*, vol. 34, no. 3, pp. 1303–1316, Mar. 2022.
- [40] S. K. née Müller, C. Tekin, M. van der Schaar, and A. Klein, "Context-aware hierarchical online learning for performance maximization in mobile crowdsourcing," *IEEE/ACM Trans. Netw.*, vol. 26, no. 3, pp. 1334–1347, Jun. 2018.
- [41] S. Yang et al., "Selecting most informative contributors with unknown costs for budgeted crowdsensing," in *Proc. IEEE/ACM 24th Int. Symp. Qual. Serv.*, 2016, pp. 1–6.
- [42] K. Han, C. Zhang, and J. Luo, "Taming the uncertainty: Budget limited robust crowdsensing through online learning," *IEEE/ACM Trans. Netw.*, vol. 24, no. 3, pp. 1462–1475, Jun. 2016.
- [43] L. Chen, J. Xu, and Z. Lu, "Contextual combinatorial multi-armed bandits with volatile arms and submodular reward," in *Proc. Adv. Neural Inf. Process. Syst.*, 2018, pp. 3247–3256.
- [44] P. Auer, R. Ortner, and C. Szepesvári, "Improved rates for the stochastic continuum-armed bandit problem," in *Proc. 20th Annu. Conf. Learn. Theory*, 2007, pp. 454–468.
- [45] R. D. Kleinberg, "Nearly tight bounds for the continuum-armed bandit problem," in *Proc. Adv. Neural Inf. Process. Syst.*, 2005, pp. 697–704.
- [46] S. Bubeck, R. Munos, G. Stoltz, and C. Szepesvári, "X-armed bandits," *J. Mach. Learn. Res.*, vol. 12, no. 05, pp. 1655–1695, 2011.
- [47] S. Li, B. Wang, S. Zhang, and W. Chen, "Contextual combinatorial cascading bandits," in *Proc. 33rd Int. Conf. Mach. Learn.*, vol. 16, 2016, pp. 1245–1253.
- [48] D. Kempe, J. Kleinberg, and É. Tardos, "Maximizing the spread of influence through a social network," in *Proc. 9th ACM SIGKDD Int. Conf. Knowl. Discov. Data Mining*, 2003, pp. 137–146.
- [49] L. A. Wolsey and G. L. Nemhauser, *Integer and Combinatorial Optimization*. Hoboken, NJ, USA: Wiley, 2014.
- [50] C.-J. Ho and J. W. Vaughan, "Online task assignment in crowdsourcing markets," in *Proc. 26th AAAI Conf. Artif. Intell.*, 2012, vol. 26, no. 1, pp. 45–51.
- [51] L. Tran-Thanh, S. Stein, A. Rogers, and N. R. Jennings, "Efficient crowdsourcing of unknown experts using bounded multi-armed bandits," *Artif. Intell.*, vol. 214, pp. 89–111, 2014.
- [52] D. Geiger and M. Schader, "Personalized task recommendation in crowdsourcing information systems—Current state of the art," *Decis. Support Syst.*, vol. 65, pp. 3–16, 2014.
- [53] U. Mukhopadhyay, A. Skjellum, O. Hambolu, J. Oakley, L. Yu, and R. Brooks, "A brief survey of cryptocurrency systems," in *Proc. Int. Conf. Privacy, Secur. Trust*, 2016, pp. 745–752.
- [54] S. Shekhar et al., "Gaussian process bandits with adaptive discretization," *Electron. J. Stat.*, vol. 12, no. 2, pp. 3829–3874, 2018.
- [55] R. Munos, "Optimistic optimization of a deterministic function without the knowledge of its smoothness," in *Proc. Adv. Neural Inf. Process. Syst.*, 2011, pp. 783–791.
- [56] D. Yang, D. Zhang, V. W. Zheng, and Z. Yu, "Modeling user activity preference by leveraging user spatial temporal characteristics in LBSNs," *IEEE Trans. Syst., Man, Cybern. Syst.*, vol. 45, no. 1, pp. 129–142, Jan. 2015.
- [57] Y. Tang, X. Xiao, and Y. Shi, "Influence maximization: Near-optimal time complexity meets practical efficiency," in *Proc. ACM SIGMOD Int. Conf. Manage. Data*, 2014, pp. 75–86.



Sepehr Elahi (Student Member, IEEE) received two B.Sc. degrees in electrical and electronics engineering and mathematics from Bilkent University, Ankara, Turkey, in June 2022. He is currently working towards the Ph.D. degree in computer and communication sciences with EPFL, Lausanne, Switzerland. His research interests include bandit problems, Bayesian learning, optimization, statistical learning, and applications of machine learning to other fields.



Andi Nika received the B.Sc. degree in mathematics from the University of Tirana, Tirana, Albania, in 2015, M.S. degree in mathematics and M.S. degree in electrical and electronics engineering from Bilkent University, Ankara, Turkey, in 2018 and 2021, respectively. He is currently working towards the Ph.D. degree with the Max Planck Institute for Software Systems, Saarbrücken, Germany, working with the Machine Teaching group, under the supervision of Adish Singla. His research interests include multi-agent reinforcement learning, robustness in reinforcement learning, and multi-armed bandits.



Cem Tekin (Senior Member, IEEE) received the B.Sc. degree in electrical and electronics engineering from the Middle East Technical University, Ankara, Turkey, in 2008, and the M.S.E. degree in electrical engineering: systems, M.S. degree in mathematics, and Ph.D. degree in electrical engineering: systems from the University of Michigan, Ann Arbor, MI, USA, in 2010, 2011 and 2013, respectively. From February 2013 to January 2015, he was a Postdoctoral Scholar with the University of California, Los Angeles, CA, USA. He is currently an Associate Professor with the Department

of Electrical and Electronics Engineering, Bilkent University, Ankara, Turkey. His research interests include cognitive communications, reinforcement learning, multiarmed bandit problems, and multiagent systems. He was the recipient of the numerous awards including the Fred W. Ellersick Award for the Best Paper in MILCOM 2009 and the Distinguished Young Scientist (BAGEP) Award of the Science Academy Association of Turkey in 2019.