Task recommendation in crowdsourcing based on learning preferences and reliabilities

Kang, Qiyu; Tay, Wee Peng

2020

Kang, Q. & Tay, W. P. (2020). Task recommendation in crowdsourcing based on learning preferences and reliabilities. IEEE Transactions On Services Computing.
<https://dx.doi.org/10.1109/TSC.2020.3020338>

<https://hdl.handle.net/10356/153703>

<https://doi.org/10.1109/TSC.2020.3020338>

© 2020 IEEE. Personal use of this material is permitted. Permission from IEEE must be obtained for all other uses, in any current or future media, including reprinting/republishing this material for advertising or promotional purposes, creating new collective works, for resale or redistribution to servers or lists, or reuse of any copyrighted component of this work in other works. The published version is available at:
<https://doi.org/10.1109/TSC.2020.3020338>.

Downloaded on 26 Apr 2023 22:38:50 SGT

Task Recommendation in Crowdsourcing Based on Learning Preferences and Reliabilities

Qiyu Kang and Wee Peng Tay, *Senior Member, IEEE*

Abstract—Workers participating in a crowdsourcing platform can have a wide range of abilities and interests. An important problem in crowdsourcing is the task recommendation problem, in which tasks that best match a particular worker's preferences and reliabilities are recommended to that worker. A task recommendation scheme that assigns tasks more likely to be accepted by a worker who is more likely to complete it reliably results in better performance for the task requester. Without prior information about a worker, his preferences and reliabilities need to be learned over time. In this paper, we propose a multi-armed bandit (MAB) framework to learn a worker's preferences and his reliabilities for different categories of tasks. However, unlike the classical MAB problem, the reward from the worker's completion of a task is unobservable. We therefore include the use of gold tasks (i.e., tasks whose solutions are known *a priori* and which do not produce any rewards) in our task recommendation procedure. Our model could be viewed as a new variant of MAB, in which the random rewards can only be observed at those time steps where gold tasks are used, and the accuracy of estimating the expected reward of recommending a task to a worker depends on the number of gold tasks used. We show that the optimal regret is $O(\sqrt{n})$, where n is the number of tasks recommended to the worker. We develop three task recommendation strategies to determine the number of gold tasks for different task categories, and show that they are order optimal. Simulations verify the efficiency of our approaches.

Index Terms—Crowdsourcing, task recommendation, multi-armed bandit

1 INTRODUCTION

IN a typical crowdsourcing platform, a worker may be recommended a wide variety of tasks to choose from [1]–[3]. For example, on Amazon Mechanical Turk (MTurk) and CrowdFlower, tasks can include labeling the content of an image, determining whether or not sentences extracted from movie reviews are positive, or determining whether or not a website has adult contents. In these examples, the different tasks require different sets of skills from a worker. In the image labeling task, a worker who is good at visual recognition will perform reliably, whereas identifying whether a movie review is positive or not requires a worker with language skills and knowledge of the nuances of that particular language the review is written in. Workers on crowdsourcing platforms are globally distributed [4], hailing from different countries with different education backgrounds, wide-ranging levels of expertise and motivations. Workers may thus have different performance over different task categories. Workers also have different interests, and may choose not to accept tasks that they are qualified for. Therefore, the crowdsourcing platform needs to find a way to best match the available tasks to the most suitable workers to improve the likelihood of obtaining high-quality solutions to the tasks.

Crowdsourcing workers are typically not experts and errors are common in their answers [5]. Therefore, quality control to ensure the correctness of the final answers is

imperative. To this end, various inference algorithms and task allocation methods have been proposed. For example, in [6], the authors proposed to use Distributed Classification Fusion using Error-Correcting Codes (DCFECF), a strategy from coding theory, to divide a single multi-class labeling task into binary questions, which are then assigned to the workers. They have shown their algorithm performs well even with spammers who are unreliable workers that make decisions at random, as compared to the trivial majority voting strategy. In [7], a sequential questioning strategy is developed to accommodate and recover from errors made by the workers. In many other works [8]–[10], spammers are filtered out when assigning tasks. On MTurk, a task requester can place restrictions on targeted workers according to their *approval rate*, a metric provided by MTurk that measures the overall rating for a specific worker. For example, in [9], the authors have limited their tasks to be assigned to workers with at least 95% approval rating. However, setting a high threshold on approval rates decreases the number of available workers. Potentially qualified workers may be excluded as this metric does not distinguish between the qualification of workers on different task categories. To employ more qualified workers on specific tasks and shorten the completion time, task requesters tend to use qualification tests with *gold* tasks [11] whose solutions are known *a priori* to filter out incompetent workers. For example, in [8], [12], gold tasks are used for estimating workers' reliabilities and for filtering purposes. Including more gold tasks leads to better estimates of the worker's reliabilities for different task categories, but also results in less efficient task completion. It is also noted in [8] that workers who have passed a qualification test may choose to behave like spammers during the actual task completion period. This could

This work was supported in part by the Singapore Ministry of Education Academic Research Fund Tier 2 grant MOE2018-T2-2-019 and by A*STAR under its RIE2020 Advanced Manufacturing and Engineering (AME) Industry Alignment Fund – Pre Positioning (IAF-PP) (Grant No. A19D6a0053). The authors are with the School of Electrical and Electronic Engineering, Nanyang Technological University, Singapore. Email: KANG0080@e.ntu.edu.sg, wptay@ntu.edu.sg.

be related in the level of interest or preference of the worker for that particular task, i.e., being an expert in a task does not mean that the worker prefers to always perform that task. In our work, to mitigate these problems, we sparingly assign gold tasks to workers sequentially during the task completion process to optimize the trade-off between the number of gold and non-gold tasks. We show theoretically how this can be achieved. Using gold tasks allow us to obtain better estimates of the workers' reliabilities over time. This is a common industrial technique called *honeypot* [8], [13], which is adapted in some works in crowdsourcing like [14]. However, to the best of our knowledge, there has been no theoretical studies on when to assign gold tasks to workers and how to achieve an optimal trade-off.

In this paper, we assume that tasks are organized into several categories. Tasks belonging to the same category require similar domain knowledge to be completed. In some works like [15], [16], workers in crowdsourcing are characterized as different types to reflect their expertise, e.g., experts who have deep domain knowledge and answer questions with high accuracy, normal workers who have general domain knowledge and make mistakes occasionally, and sloppy workers or spammers who cannot provide answers with good accuracy. In our work, we assume workers may have different levels of domain knowledge and *preferences* over different task categories. A worker who is an expert in languages may very well perform terribly in image labeling tasks. At the same time, the same worker may prefer not to accept language related tasks due to her own interests and motivations (e.g., language related tasks may be more tedious). Our approach is to recommend more tasks from categories that a worker is more likely to complete with reliable results.

As the number of tasks uploaded by requesters daily can be large [17], the crowdsourcing platform aims to recommend tasks to each worker that he will likely accept in order to optimize the platform productivity. As of 2015, it was estimated that there was an average of 1278 active crowdsourcers per day in MTurk [18]. It was also reported in [19] that some workers, who are trying to make MTurk a core source of income, spend hours each day scrolling through pages upon pages of job posting to search for tasks that they are interested in and good at, which obviously limits the platform productivity. Including workers' preferences in task assignment can help workers to find their preferred tasks faster as they will have fewer recommendations to consider which will in turn increase the efficiency of task completion. In some works [9], [20], [21], researchers have studied the monetary payment effect on workers' preferences or quality of their answers. In [20], the authors concluded that increased financial incentives increase the quantity but not the quality of task completion, while [21] found that more financial incentives encourage higher task completion quality. Therefore, we do not explicitly include financial incentives into our problem formulation in this paper. However, we note that one way to include financial incentives into our problem formulation is to discretize monetary payments into different levels and integrate that implicitly into the task categories.

Based on the above discussion, we propose an online learning approach to estimate workers' preferences and

reliabilities jointly. We assume that tasks can be categorized, and that each worker can choose to accept or reject a recommended task at each time step. If he chooses to accept the task, he completes it within the current time step. For each time step, our goal is to recommend a task from a given number of task categories so that it best matches the worker's preferences and reliabilities according to a reward function. Note that this reward refers to the crowdsourcing platform's system reward, and is distinct from any monetary payment or incentives given to the worker. As the worker's preferences and reliabilities for different task categories are unknown *a priori*, we adopt a multi-armed bandit (MAB) formulation [22], [23] in which these are learned through exploration, while the cumulative sum reward is optimized through the exploitation of the empirical best match.

1.1 Related Work

In this section we review the literature related to our work, and discuss their differences to our contributions.

Within the task assignment context, a number of studies have focused on offline strategies, which assign the tasks to workers without any adaptation or feedback during the workers' task completion process. For example, in [24], [25], the authors proposed a task assignment scheme using a bipartite graph to model the affinity of workers for different binary tasks and an iterative algorithm based on belief propagation to infer the final decision from the workers' responses. Reference [26] studied a task allocation model that provides distributional fairness among workers, while maximizing the task allocation ratio. A privacy-preserving task recommendation scheme was proposed in [27], which only considers workers' preferences without taking workers' reliabilities into account. The paper [28] considered heterogeneous tasks with different difficulties using the generalized Dawid-Skene model [29], while [30] utilizes the expectation maximization approach to estimate the tasks' solutions and workers' reliabilities. The aforementioned approaches are all offline which is different from our online approach, i.e., the task assignments in their works are made without taking into account additional information about the workers' performances and behaviors that can be gleaned while the workers complete tasks sequentially over time.

Online learning for adaptive task assignments has also been considered in recent years. Our work belongs to this category. In [7], [31], as workers may be unreliable, the authors proposed to perform sequential questioning in which the questions posed to the workers are designed based on previous questions and answers. Reference [32] studied sequential user selection using a Bayes update aided online solution in the problem of news disseminating. The paper [33] investigated the problem of label inference for heterogeneous classification tasks by applying online primal-dual techniques. In [34], the authors studied sequential task assignment with budgets that specify how many times the requester would like each task to be completed. Expert-crowdsourcing in which workers are experts who have unknown reliabilities was studied in [35], which also assumed that the requester has a budget constraint and each worker has a maximum task limit. The authors proposed an algorithm, which uses an initial exploration phase to uniformly

sample the performance of a wide range of workers, and in the latter exploitation phase solves a bounded knapsack problem [36]. In the mobile micro-task allocation problem in spatial crowdsourcing scenarios, where the orders of arriving tasks and workers are dynamic, [37] proposed an online two-phase-based bipartite graph matching algorithm with good average performance. In [38], the authors proposed an adaptive crowdsourcing framework, which continuously estimates the reliability of a worker by evaluating her performance on the completed tasks, and predicts which tasks the worker is well acquainted with. In the above work, the authors assumed that the crowdsourcing platform is able to quickly and accurately evaluate the quality of all the tasks that have been completed. However, as is often the case in practice, the crowdsourcing platform cannot directly and immediately evaluate the quality of a completed task [39]. For example, in tasks where workers are asked to determine whether an image contains a particular object, the crowdsourcing platform has no way to determine if the task completed by a worker is successful or not [40]. In our work, we do not have this assumption. In [40], the authors proposed an online algorithm, in which workers' reliabilities are estimated by repetitively assigning each worker the same tasks occasionally.

The above-mentioned works do not take workers' interests or preferences for certain tasks into consideration when performing a task assignment. In [41], the authors only considered workers' preferences when dealing with demand responses. The paper [42] proposed to perform task assignment using workers' preferences based on many-to-one matching. However, the paper assumes that workers' preferences and reliabilities are known *a priori*, which is unrealistic in a practical system. In [17], the authors proposed two recommendation strategies that compute the top- k most suitable tasks for a given worker and the top- k best workers for a given task. However, they assume that the workers can be evaluated immediately. The references [43], [44] employed matrix factorization techniques to capture workers' preferences and reliability where, however, the evaluated scores in the matrix are also assumed to be gotten immediately.

Unlike the classical MAB problem in which the reward at each time step can be observed, the reward is *unobservable* in our formulation as the crowdsourcing platform does not know the true solution of each task. In some works like [21], the authors assumed that task completion can be measured, i.e., every task is essentially a gold task with a known correct response in our framework. This is clearly not realistic in practice.

Our MAB formulation is also related to, but different from, the risk-averse MAB problem considered in [45]–[47], which uses mean-variance [48], [49] as its risk objective. The differences between our MAB formulation and the risk-averse MAB are: (i) The worker in our problem may choose not to accept a task at each time step, leading to no reward at that time step, whereas in the risk-averse MAB, such an option is not available. (ii) The variance of the reward is used in the formulation of the risk-averse MAB, while we use the variance of the reward *estimator* in formulating our regret. This is because as explained above, in a crowdsourcing platform, the reward at each time step is unobservable,

and our regret is dependent on the estimation accuracy from using gold tasks to estimate the expected reward instead of the reward variance.

1.2 Summary of Our Contributions

We propose an online learning approach to estimate workers' preferences and reliabilities with the use of gold tasks whose solutions are known *a priori*. However, gold tasks do not generate any rewards other than the estimation of workers' preferences and reliabilities. Including more gold tasks leads to better estimates of the worker's preferences and reliabilities for different task categories, but also results in less reward for the crowdsourcing platform. In such a platform, a plausible compensation scheme is to pay a worker for completed tasks according to her reliability [50]–[52]. Furthermore, to accurately fuse the results from multiple workers, the knowledge of the workers' reliabilities are required [33], [53]. Therefore, the platform aims to estimate the worker's reliability as accurately as possible. To reflect this, we include both the reward and estimation accuracy in computing the regret. We show that the optimal regret under our formulation has order $O(\sqrt{n})$, where n is the number of time steps. In addition, we propose three strategies that achieve the order optimal regret. Our sequential task recommendation strategy for crowdsourcing can therefore guide platform designers to build more efficient recommendation systems that achieve a better matching in terms of workers' preferences and reliabilities, and subsequently a more efficient task completion rate. To the best of our knowledge, our work is the first to provide theoretical bounds for the trade-off in using gold tasks.

The rest of this paper is organized as follows. In Section 2, we present our system model and assumptions. In Section 3, we derive the optimal order of the regret and introduce three strategies that are order optimal in Section 4. In Section 5, we present simulations to compare the performance of our approaches. Section 6 concludes the paper.

Notations: We use $\text{Bern}(p)$ to denote the distribution of a Bernoulli random variable X with $\mathbb{P}(X = 1) = p$. The notation $\sim$ denotes equality in distribution. We use $\mathbb{N}$ to denote the set of positive integers and E^c to denote the complement of the event E . The indicator function $\mathbf{1}_A(\omega) = 1$ if and only if $\omega \in A$. We let $x^+ = \max\{x, 0\}$. For non-negative functions $f(n)$ and $g(n)$, we write $f(n) = O(g(n))$ if $\limsup_{n \rightarrow \infty} f(n)/g(n) < \infty$, $f(n) = o(g(n))$ if $\lim_{n \rightarrow \infty} f(n)/g(n) = 0$, and $f(n) = \Theta(g(n))$ if $0 < \liminf_{n \rightarrow \infty} f(n)/g(n) \leq \limsup_{n \rightarrow \infty} f(n)/g(n) < \infty$.

2 SYSTEM MODEL

We consider a crowdsourcing platform where each task belongs to one of K categories. The platform recommends a task from some category to a worker at each time $n \in \mathbb{N}$. The worker has different reliabilities and preferences for different categories. The worker's reliability for category k is the probability p_k of completing a task from category k correctly, and his preference refers to the probability q_k of accepting a task from category k . We let $A_{k,t} = 1$ if the worker accepts the t -th task recommended to him from category k ,

and $A_{k,t} = 0$ otherwise. We let $X_{k,t} = 1$ if the worker completes the task correctly, and $X_{k,t} = 0$ otherwise. For all k and t , we assume that $A_{k,t} \sim \text{Bern}(q_k)$ are independent and identically distributed. Similarly, $X_{k,t} \sim \text{Bern}(p_k)$ are independent and identically distributed. We assume that $A_{k,t}$ and $X_{k,t}$ are independent. Without loss of generality, our analysis focuses on a single worker as we do not utilize the fusion result from multiple workers to determine preferences or reliabilities. Instead, gold tasks (see below) are introduced for this purpose. However, we note that our formulation can also be applied to a recommendation platform that uses clustering techniques like k -means on workers' personality traits [15] or more advanced collaborative filtering [54] techniques to define worker clusters. Each worker cluster can then be regarded as a "macro worker" with its own macro preferences and reliabilities. Workers from the same cluster are assigned the same task and their results are then fused. This is a common technique that has been used in works like [55] to improve task accuracy.

We model the task recommendation problem as a MAB problem. However, since $X_{k,t}$ is typically unobservable since the system does not know the task solution *a priori*, we include the use of gold tasks in our recommendation. A gold task is a task whose solution is known *a priori* to the system. A reward is obtained only if the worker accepts and completes a non-gold task correctly.

Let $Y_{k,t} = A_{k,t}X_{k,t}$, whose distribution is $\text{Bern}(q_k p_k)$. We denote the task category that maximizes $q_k p_k$ as $* = \arg \max_{1 \leq k \leq K} q_k p_k$. For each time step t , let $G_{k,t}$ denote the event that the t -th task from category k recommended to the worker is a gold task. Let

$$g_k(t) = \sum_{i=1}^t A_{k,i} \mathbf{1}_{G_{k,i}} \quad (1)$$

be the number of completed gold tasks. If the worker completes a non-gold task, he is rewarded in proportion to p_k , the probability that he has completed the task correctly. However, since we do not know p_k *a priori*, the crowdsourcing platform estimates it using the empirical mean from the $g_k(t)$ completed gold tasks:

$$\bar{X}_k(t) = \frac{1}{g_k(t)} \sum_{i=1}^t A_{k,j_i} X_{k,j_i} \mathbf{1}_{G_{k,i}}. \quad (2)$$

Due to the uncertainty in the estimate (2), we assume that the reward for the t -th non-gold task from category k if the worker completes it is given by

$$\left(p_k - \beta \frac{\sigma_k^2}{g_k(t)} \right)^+, \quad (3)$$

where $\beta > 0$ is a predefined weight that quantifies the importance of the estimator $\bar{X}_k(t)$'s accuracy to the crowdsourcing platform, and $\sigma_k^2 = p_k(1 - p_k)$ is the variance of a task reward from category k . The quantity $\sigma_k^2/g_k(t)$ is the variance of $\bar{X}_k(t)$ conditioned on $g_k(t)$.

For each $k = 1, \dots, K$ and time n , let $N_k(n)$ denote the number of tasks recommended so far till time n from

category k . We define the cumulative reward function at time n as:

$$r(n) = \mathbb{E} \sum_{k=1}^K \sum_{t=1}^{N_k(n)} q_k \left(p_k - \beta \frac{\sigma_k^2}{g_k(t)} \right)^+ \mathbf{1}_{G_{k,t}}, \quad (4)$$

since $A_{k,t}$, $X_{k,t}$ and $(G_{k,t}, N_k(n))$ are independent. Note that the coefficient β can also be interpreted as the Lagrange multiplier for a constrained optimization problem in which the reward

$$\mathbb{E} \sum_{k=1}^K \sum_{t=1}^{N_k(n)} q_k p_k \mathbf{1}_{G_{k,t}}$$

is maximized subject to a constraint on the uncertainty in the estimator (2). A larger β corresponds to a more stringent requirement for the accuracy of the estimator $\bar{X}_k(t)$, which leads to more gold tasks being recommended to the worker. Traditional reward functions defined in the literature like [23] sets $\beta = 0$ in (3) and their sole objective is to maximize the cumulative reward over time. In summary, the cumulative reward function given in (4) has the following properties:

- (i) A non-zero reward at each time step t occurs only for non-gold tasks.
- (ii) A non-zero reward at each time step t occurs only if the worker accepts the task.
- (iii) The size of the expected reward at time step t for a non-gold task from category k the worker accepts is given by (3), where a higher reliability p_k gives a higher expected reward. This reward is in turn discounted by the variance of the estimator for the reward of task category k . This estimator is derived based on *gold* tasks (as we only know the correct responses for such tasks *a priori*). The motivation for including the variance in the reward function is discussed in Section 1.2.

To avoid dividing by zero, i.e., to ensure $g_k(t) > 0$, we assume that the worker completes one gold task from each category before the recommendation starts, which can be done through a calibration process in a crowdsourcing platform.

To maximize the above reward function (4) is equivalent to minimizing the regret function at time n , defined as:

$$R(n) = nq_*p_* - r(n). \quad (5)$$

For simplicity, we assume that every task category is non-empty at each time step. This is a reasonable assumption since the task pool in a crowdsourcing platform is updated constantly, and a single task may be assigned to more than one worker. In the following, we use the terms "task category" and "arm" interchangeably in our MAB formulation.

3 OPTIMAL REGRET ORDER

In this section, we first show that the optimal order of the regret function (5) is $O(\sqrt{n})$, where n is the number of time steps. We then propose three recommendation strategies, all of which achieve the order optimal regret.

Theorem 1. *The optimal order of the regret function (5) is $O(\sqrt{n})$, where n is the number of time steps.*

Proof: Let

$$f(n) = \sum_{k=1}^K \sum_{t=1}^{N_k(n)} \mathbf{1}_{G_{k,t}} \quad (6)$$

be the total number of gold tasks recommended till time n . Then, from (4), we have

$$\begin{aligned} r(n) &= \mathbb{E} \sum_{k=1}^K \sum_{t=1}^{N_k(n)} q_k \left(p_k - \beta \frac{\sigma_k^2}{g_k(t)} \right)^+ \mathbf{1}_{G_{k,t}^c} \\ &\leq \mathbb{E} \sum_{k=1}^K \sum_{t=1}^{N_k(n)} \left(q_* p_* - \beta \frac{q_k \sigma_k^2}{f(n)} \right)^+ \mathbf{1}_{G_{k,t}^c} \\ &\leq \mathbb{E} \left[(n - f(n)) \left(q_* p_* - \beta \frac{\min_k q_k \sigma_k^2}{f(n)} \right) \right]^+. \quad (7) \end{aligned}$$

From (5) and (7), with $a = \beta \min_k q_k \sigma_k^2$, when n is sufficiently large, we then have

$$\begin{aligned} R(n) &\geq n q_* p_* - \mathbb{E} \left[(n - f(n)) \left(q_* p_* - \beta \frac{\min_k q_k \sigma_k^2}{f(n)} \right) \right]^+ \\ &\geq \mathbb{E} \left[\min \left\{ a \frac{n}{f(n)} + q_* p_* f(n) - a, n q_* p_* \right\} \right] \quad (8) \end{aligned}$$

$$\begin{aligned} &\geq \min \{ 2\sqrt{a q_* p_* n} - a, n q_* p_* \} \quad (9) \\ &= 2\sqrt{a q_* p_* n} - a, \quad (10) \end{aligned}$$

where the inequality in (9) follows because $a \frac{n}{f(n)} + q_* p_* f(n) - a \geq 2\sqrt{a q_* p_* n} - a$ with probability 1. The theorem is now proved. $\square$

4 ORDER OPTIMAL STRATEGIES

In this section, we propose three strategies that achieve the optimal order regret of $O(\sqrt{n})$ in Theorem 1, and discuss the advantages of each.

4.1 Greedy Recommendation Strategy

In our greedy recommendation strategy (GR), we divide time into epochs. An epoch is a collection of consecutive time steps over which tasks from the same category are recommended to the worker. Specifically, in each epoch r , a single task category or arm c_r is chosen and all tasks in that epoch are drawn from that category. In the first K epochs, each of which consists of a single time step, the worker completes a gold task from each of the K categories. Subsequently, the r -th epoch, where $r \geq K + 1$, consists of $\tau(r) - \tau(r-1) + 1$ time steps, where $\tau(r) = \lceil \alpha r^2 \rceil$ (we call this the *epoch function*), and $\alpha > 0$. In each of these epochs, we set one task to be a gold task, which can be inserted randomly at any time step in this epoch to prevent intended cheating during the process, while for all other time steps in the epoch, the tasks recommended are non-gold tasks chosen from a particular task category. For each $r \geq K + 1$, the task category c_r is chosen based on a ϵ_r -greedy policy [23] as follows.

Let $\Delta_k = q_* p_* - q_k p_k$, and choose d so that $0 < d \leq \min_{k \neq *} \Delta_k$. For each $r = 1, 2, \dots$, let $\epsilon_r = \min\{1, cK/(d^2 r)\}$, where $c > 0$ is a fixed constant. At the beginning of each epoch r , where $r \geq K + 1$, we choose

$$z_r = \arg \max_k \bar{Y}_k(T_k(r-1)), \quad (11)$$

where $T_k(r)$ is the number of epochs within the first r epochs that category k is chosen,

$$\bar{Y}_k(g) = \frac{1}{g} \sum_{i=1}^g A_{k,j_i} X_{k,j_i}, \quad (12)$$

and $j_1, \dots, j_g$ are the indices of the first g gold tasks recommended from category k . Then, with probability $1 - \epsilon_r$, we let $c_r = z_r$ to be the task category chosen for epoch r , and with probability ϵ_r , we let c_r to be a randomly chosen category. The GR strategy is summarized in Algorithm 1.

Algorithm 1 GR

Inputs: $0 < d \leq \min_{k \neq *} \Delta_k$, $c > 0$, $\alpha > 0$.

Initialization:

- 1: Recommend one gold task from each category.
 - 2: Set $r = K + 1$, $T_k(r) = 1$ and $\bar{Y}_k(1) = Y_{k,1}$ for all $k = 1, \dots, K$.
 - 3: **loop**
 - 4: Find z_r in (11).
 - 5: Set $\epsilon_r = \min\{1, \frac{cK}{d^2 r}\}$. With probability $1 - \epsilon_r$, choose $c_r = z_r$, otherwise set c_r to be a randomly chosen arm.
 - 6: Choose a random time step within epoch r and recommend a gold task from arm c_r at that time step. Recommend a non-gold task drawn from arm c_r at each of the other $\tau(r) - \tau(r-1)$ time steps.
 - 7: Update $T_{c_r}(r) = T_{c_r}(r-1) + 1$, $n_{c_r}(r)$ and $\bar{Y}_{c_r}(T_{c_r}(r))$. Set $r = r + 1$.
 - 8: **end loop**
-

Theorem 2. Suppose that $c > \max\{5d^2, 2\}$. Then, GR has regret of order $O(\sqrt{n})$, where n is the number of time steps.

Proof: For each arm k in epoch r , we denote by $P_{k,r} = \mathbb{P}(c_r = k)$, the probability that arm k is chosen. From Theorem 3 in [23], if $c > \max\{5d^2, 2\}$ and $k \neq *$, we have

$$P_{k,r} \leq \frac{c}{d^2 r} + o(r^{-1}), \quad (13)$$

so that

$$P_{*,r} \geq 1 - \frac{c(K-1)}{d^2 r} - o(r^{-1}). \quad (14)$$

Let M be the number of epochs completed till the time step n , i.e., M is the largest integer satisfying $K + \sum_{r=K+1}^M (\tau(r) - \tau(r-1) + 1) = \lceil \alpha M^2 \rceil - \lceil \alpha K^2 \rceil + M \leq n$. We have $M = O(\sqrt{n})$. Let $n_k(r)$ be the number of category k gold tasks completed in the first r epochs. Since a single gold task is recommended in each epoch, we have $g_{c_r}(t) = n_{c_r}(r)$ for all time steps t in epoch r . From (4), we obtain

$$\begin{aligned} r(n) &\geq \mathbb{E} \sum_{r=K+1}^M q_{c_r} (\tau(r) - \tau(r-1)) \left(p_{c_r} - \frac{\beta \sigma_{c_r}^2}{n_{c_r}(r)} \right)^+ \\ &\geq \mathbb{E} \sum_{r=K+1}^M q_* (2\alpha r - \alpha - 1)^+ \left(p_* - \frac{\beta \sigma_*^2}{n_*(r)} \right) \mathbf{1}_{\{c_r = *\}} \\ &= \sum_{r=K+1}^M q_* (2\alpha r - \alpha - 1)^+ \left(p_* P_{*,r} - \mathbb{E} \left[\frac{\beta \sigma_*^2}{n_*(r)} \right] \right). \quad (15) \end{aligned}$$

Recall also that $T_k(r) = \sum_{i=1}^r \mathbf{1}_{\{c_i=k\}}$ is the number of epochs within the first r epochs in which arm k was chosen. Since GR assumes that the worker completes a gold task from arm $*$ before the recommendation starts, we have

$$n_*(r) - 1 \mid T_*(r) \sim \text{Bern}(T_*(r) - 1, q_*),$$

and

$$\begin{aligned} & \mathbb{E} \left[\frac{1}{n_*(r)} \right] \\ &= \mathbb{E} \left[\mathbb{E} \left[\frac{1}{n_*(r)} \mid T_*(r) \right] \right] \\ &= \mathbb{E} \sum_{j=0}^{T_*(r)-1} \frac{1}{1+j} \binom{T_*(r)-1}{j} q_*^j (1-q_*)^{T_*(r)-1-j} \\ &= \mathbb{E} \sum_{j=0}^{T_*(r)-1} \frac{1}{T_*(r)} \binom{T_*(r)}{j+1} q_*^j (1-q_*)^{T_*(r)-1-j} \\ &= \mathbb{E} \frac{1}{T_*(r)q_*} \sum_{j=0}^{T_*(r)-1} \binom{T_*(r)}{j+1} q_*^{j+1} (1-q_*)^{T_*(r)-1-j} \\ &= \mathbb{E} \frac{1}{T_*(r)q_*} \left(1 - (1-q_*)^{T_*(r)} \right) \\ &\leq \mathbb{E} \left[\frac{1}{T_*(r)q_*} \right]. \end{aligned} \quad (16)$$

From (14), (15) and (16), we obtain

$$\begin{aligned} & r(n) \\ &\geq \sum_{r=K+1}^M q_* p_* (2\alpha r - \alpha - 1)^+ (1 - O(r^{-1})) \\ &\quad - \sum_{r=K+1}^M (2\alpha r - \alpha - 1)^+ \mathbb{E} \left[\frac{\beta \sigma_*^2}{T_*(r)} \right] \\ &\geq q_* p_* \alpha M^2 - O(M) - \sum_{r=K+1}^M (2\alpha r - \alpha - 1)^+ \mathbb{E} \left[\frac{\beta \sigma_*^2}{T_*(r)} \right] \\ &\geq n q_* p_* - O(\sqrt{n}) - \sum_{r=K+1}^M (2\alpha r - \alpha - 1)^+ \mathbb{E} \left[\frac{\beta \sigma_*^2}{T_*(r)} \right]. \end{aligned} \quad (17)$$

We next prove the following lemma.

Lemma 1. $\mathbb{E} \left[\frac{1}{T_*(r)} \right] \leq O(r^{-1})$.

We have

$$\begin{aligned} & \mathbb{E} \left[\frac{1}{T_*(r)} \right] \\ &= \mathbb{E} \left[\frac{1}{T_*(r)} \mid T_*(r) \leq \frac{r}{K} \right] \mathbb{P} \left(T_*(r) \leq \frac{r}{K} \right) \\ &\quad + \mathbb{E} \left[\frac{1}{T_*(r)} \mid T_*(r) > \frac{r}{K} \right] \mathbb{P} \left(T_*(r) > \frac{r}{K} \right) \\ &\leq \mathbb{P} \left(T_*(r) \leq \frac{r}{K} \right) + O(r^{-1}). \end{aligned} \quad (18)$$

Therefore, to show the lemma, it suffices to show that the tail probability $\mathbb{P} \left(T_*(r) \leq \frac{r}{K} \right)$ is of order $O(r^{-1})$.

Consider $k \neq *$. Let $T_k^B(r)$ be the number of times $z_j = k$ for $j \leq r$, and $T_k^R(r)$ be the number of times arm k was randomly chosen for $j \leq r$. From the union bound, we have

$$\mathbb{P} \left(T_k(r) \geq \frac{r}{K} \right) \leq \mathbb{P} \left(T_k^R(r) \geq \frac{r}{2K} \right) + \mathbb{P} \left(T_k^B(r) \geq \frac{r}{2K} \right). \quad (19)$$

From Hoeffding's inequality, we obtain

$$\begin{aligned} \mathbb{P} \left(T_k^R(r) \geq \frac{r}{2K} \right) &\leq e^{-2(\frac{1}{2K} - \mathbb{E} T_k^R(r)/r)^2 r} \\ &\leq e^{-2(\frac{1}{2K} - O(\frac{c \ln r}{d^2 r}))^2 r} \\ &= e^{-O(r)}, \end{aligned} \quad (20)$$

where the second inequality follows because the probability of randomly choosing arm k at each epoch j is not more than $cK/(d^2 j) \cdot 1/K = c/(d^2 j)$ and

$$\mathbb{E} T_k^R(r) \leq \sum_{j=1}^r \frac{c}{d^2 j} \leq \frac{c(\ln r + 1)}{d^2}.$$

In order to bound the second term in (19), we use a similar argument as that in [56]. Let $u(r) = \frac{c}{d^2} \ln \frac{r d^2 e^{1/2}}{2cK^2}$ and $s(r) = \lceil r/(2K) \rceil$. Define the following events:

$$\begin{aligned} E_1 &= \left\{ \bar{Y}_k(j) \leq q_k p_k + \frac{\Delta_k}{2}, \text{ for all } j \in [s(r), r] \right\}, \\ E_2 &= \left\{ \bar{Y}_*(j) \geq q_* p_* - \frac{\Delta_k}{2}, \text{ for all } j \in (u(r), r] \right\}, \\ E_3 &= \{ T_*(s(r)) > u(r) \}. \end{aligned}$$

Under the event $E_1 \cap E_2 \cap E_3$, we have for all $i \in [s(r), r]$,

$$r \geq T_*(i) \geq T_*(s(r)) > u(r),$$

which implies that

$$\begin{aligned} \bar{Y}_*(T_*(i)) &\geq q_* p_* - \frac{\Delta_k}{2} \\ &> q_k p_k + \frac{\Delta_k}{2} \\ &\geq \bar{Y}_k(j), \end{aligned} \quad (21)$$

for all $j \in [s(r), r]$. Since $T_k^B(s(r)) \leq s(r)$, we have $T_k^B(r) \leq s(r)$ because otherwise there exists r' such $T_k^B(r') \in [s(r), r]$ and $\bar{Y}_k(T_k(r')) > \bar{Y}_*(T_*(r'))$, which contradicts (21). Therefore, $E_1 \cap E_2 \cap E_3 \subset \{T_k^B(r) \leq s(r)\}$, and

$$\begin{aligned} & \mathbb{P} \left(T_k^B(r) > s(r) \right) \\ &\leq \mathbb{P} (E_1^c \cup E_2^c \cup E_3^c) \\ &\leq \mathbb{P} (E_1^c) + \mathbb{P} (E_2^c) + \mathbb{P} (E_3^c), \end{aligned} \quad (22)$$

where the last inequality is the union bound. We next bound each of the terms on the right hand side of (22) separately. For the first term, we have

$$\begin{aligned} \mathbb{P} (E_1^c) &\leq \sum_{j=s(r)}^r \mathbb{P} \left(\bar{Y}_k(j) > q_k p_k + \frac{\Delta_k}{2} \right) \\ &\leq \sum_{j=s(r)}^r e^{-\frac{\Delta_k^2}{2} j} \\ &\leq \frac{e^{-\frac{\Delta_k^2}{4K} r}}{1 - e^{-\frac{\Delta_k^2}{2}}} \\ &= o(r^{-1}), \end{aligned} \quad (23)$$

where the second inequality follows from Hoeffding's inequality. Similarly, for the second term on the right hand side of (22), we obtain

$$\begin{aligned}\mathbb{P}(E_2^c) &\leq \sum_{j=u(r)}^r \mathbb{P}\left(\bar{Y}_*(j) < q_*p_* - \frac{\Delta_k}{2}\right) \\ &\leq \sum_{j=u(r)}^r e^{-\frac{\Delta_k^2}{2}j} \\ &= O(e^{-\frac{\Delta_k^2 c}{2d^2} \ln r}) \\ &= o(r^{-1}),\end{aligned}\quad (24)$$

since $c > 2$ and $0 < d \leq \min_{k \neq *} \Delta_k$.

From the proof of Theorem 3 in [23], we have $u(r) \leq \frac{1}{2K} \sum_{i=1}^{s(r)} \epsilon_i$ and

$$\begin{aligned}\mathbb{P}(E_3^c) &\leq \mathbb{P}\left(T_*^R(s(r)) \leq u(r)\right) \\ &\leq \mathbb{P}\left(T_*^R(s(r)) \leq \frac{1}{2K} \sum_{i=1}^{s(r)} \epsilon_i\right) \\ &\leq e^{-\frac{1}{5} \frac{1}{2K} \sum_{i=1}^{s(r)} \epsilon_i} \\ &\leq e^{-u(r)/5} \\ &= \left(\frac{2cK^2}{rd^2e^{1/2}}\right)^{\frac{c}{5d^2}} \\ &= o(r^{-1}),\end{aligned}\quad (25)$$

where the third inequality follows from Bernstein's inequality (see (13) of [23] for a detailed proof).

Applying (23) to (25) on the right hand side of (22), we obtain $\mathbb{P}(T_k^B(r) > s(r)) = o(r^{-1})$. From (19) and (20), we then have

$$\mathbb{P}\left(T_k(r) \geq \frac{r}{K}\right) = o(r^{-1}),$$

which yields

$$\begin{aligned}\mathbb{P}\left(T_*(r) \leq \frac{r}{K}\right) &= \mathbb{P}\left(\sum_{k \neq *} T_k(r) \geq \frac{r(K-1)}{K}\right) \\ &\leq \sum_{k \neq *} \mathbb{P}\left(T_k(r) \geq \frac{r}{K}\right) \\ &= o(r^{-1}),\end{aligned}\quad (26)$$

and Lemma 1 is proved.

From (17) and Lemma 1, we obtain

$$\begin{aligned}r(n) &\geq nq_*p_* - O(\sqrt{n}) - \sum_{r=K+1}^M (2\alpha r - \alpha - 1)^+ O(r^{-1}) \\ &\geq np_*q_* - O(\sqrt{n}) - O(M) \\ &\geq np_*q_* - O(\sqrt{n}),\end{aligned}$$

since $M = O(\sqrt{n})$. The proof of Theorem 2 is now complete. $\square$

4.2 Uniform-Pulling Recommendation Strategy

In this subsection, we propose another strategy, which we call the Uniform-Pulling Recommendation (UR) strategy. We again divide time into epochs, with the r -th epoch

Algorithm 2 UR

Inputs: $\alpha > 0$.

Initialization: Recommend one gold task from each arm.

Set $r = 2$.

1: **loop**

2: Choose arm $c_r = \arg \max_k \bar{Y}_k(r)$.

3: Choose K time steps randomly from epoch r and recommend gold tasks one from each of the K arm at these K time steps. Recommend a non-gold task drawn from arm c_r at each of the other $\tau(r) - \tau(r-1)$ time steps.

4: Set $r = r + 1$.

5: **end loop**

containing $K + \tau(r) - \tau(r-1)$ time steps. In the r -th epoch, where $r > 1$, we choose arm $c_r = \arg \max_{1 \leq k \leq K} \bar{Y}_k(T_k(r))$, where now $T_k(r) = r$ for all $k \in [1, K]$. Next we recommend K gold tasks at K time steps that are randomly chosen from this epoch, one from each of the K task categories. In the other $\tau(r) - \tau(r-1)$ time steps, we recommend a non-gold task from arm c_r at each time step. The UR strategy is summarized in Algorithm 2.

Theorem 3. UR has regret of order $O(\sqrt{n})$, where n is the number of time steps.

Proof: The proof uses a similar argument and the same notations as that in the proof of Theorem 2. For $r > 1$, we have

$$\begin{aligned}P_{*,r+1} &\geq 1 - \sum_{k \neq *} \mathbb{P}(\bar{Y}_k(r) \geq \bar{Y}_*(r)) \\ &= 1 - \sum_{k \neq *} \mathbb{P}(\bar{Y}_k(r) - \bar{Y}_*(r) + \Delta_k \geq \Delta_k) \\ &\geq 1 - \sum_{k \neq *} e^{-2\Delta_k^2 r} \\ &\geq 1 - o(r^{-1}),\end{aligned}\quad (27)$$

where the penultimate inequality follows from Hoeffding's inequality.

We choose M be the largest integer satisfying $K + \sum_{r=2}^M (\tau(r) - \tau(r-1) + K) = \lceil \alpha M^2 \rceil - \lceil \alpha^2 \rceil + KM \leq n$ so that $M = O(\sqrt{n})$, we have from (15) and (16),

$$\begin{aligned}r(n) &\geq \sum_{r=2}^M q_*(2\alpha r - \alpha - 1)^+ \left(p_* P_{*,r} - \mathbb{E} \left[\frac{\beta \sigma_*^2}{T_*(r)} \right] \right) \\ &\geq \sum_{r=2}^M q_* p_* (2\alpha r - \alpha - 1)^+ (1 - o(r^{-1})) \\ &\quad - \sum_{r=2}^M (2\alpha r - \alpha - 1)^+ \frac{\beta \sigma_*^2}{r} \\ &\geq q_* p_* \alpha M^2 - O(M) - \sum_{r=2}^M (2\alpha r - \alpha - 1)^+ \frac{\beta \sigma_*^2}{r} \\ &\geq nq_*p_* - O(\sqrt{n}).\end{aligned}$$

where the second inequality follows from (27). Therefore, $R(n) \leq O(\sqrt{n})$, and the proof of Theorem 3 is complete. $\square$

4.3 ϵ -First Recommendation Strategy

Algorithm 3 ϵ -First

Inputs: $H = \lfloor \sqrt{n} \rfloor$.

- 1: Recommend H gold tasks from each of the K arms.
 - 2: Choose arm $c_H = \arg \max_k \bar{Y}_k(H)$.
 - 3: Recommend a non-gold task drawn from arm c_H at each of the following $n - KH$ time steps.
-

The ϵ -first strategy was designed to deal with the budget-limited MAB problem [57], where the budget refers to the total number of time steps. In the ϵ -first strategy, the first ϵ fraction of the budget is used to explore the reward distributions of all arms, while the remaining $1 - \epsilon$ fraction is used spent on the empirically best arm found in the initial exploration phase. To apply the ϵ -first strategy to our task recommendation problem, we let $\epsilon = \Theta(1/\sqrt{n})$. Uniform pulling of arms is used in the exploration phase. The ϵ -first strategy is summarized in Algorithm 3, where $\epsilon = KH/n$.

Theorem 4. *If $\epsilon = \Theta(1/\sqrt{n})$, then the ϵ -first strategy has regret of order $O(\sqrt{n})$, where n is the number of time steps.*

Proof: Let $P_{k,H} = \mathbb{P}(c_H = k)$. Similar to (27) in the proof of Theorem 3, it can be shown that $P_{*,H} \geq 1 - o(H^{-1})$. From (4) and (16), we obtain

$$\begin{aligned}
 r(n) &\geq \mathbb{E} \left[q_{c_H} (n - KH) \left(p_{c_H} - \frac{\beta \sigma_{c_H}^2}{H q_{c_H}} \right)^+ \right] \\
 &\geq \mathbb{E} \left[q_* (n - KH) \left(p_{c_*} - \frac{\beta \sigma_*^2}{H q_*} \right) \mathbf{1}_{\{c_H = *\}} \right] \\
 &\geq q_* (n - KH) \left(p_* P_{*,H} - \frac{\beta \sigma_*^2}{H q_*} \right) \\
 &\geq n q_* p_* - O(\sqrt{n}), \tag{28}
 \end{aligned}$$

since $H = O(\sqrt{n})$. Therefore, $R(n) \leq O(\sqrt{n})$, and the proof is complete. $\square$

4.4 Discussions

The ϵ -first strategy assumes that the total number of tasks n to be recommended to each worker is known beforehand, while the other two strategies do not need such an assumption. If both UR and the ϵ -first strategy recommend the same total number of gold tasks, then the ϵ -first strategy achieves a smaller regret than UR. This is because the ϵ -first strategy recommends all the gold tasks in the initial exploration phase, leading to a better estimation of the reward distributions before any non-gold tasks are recommended. However, in a practical crowdsourcing platform, knowing the total number of tasks for a worker may not be feasible as a worker is not guaranteed to be active or to remain in the system. An alternative is to use a hybrid UR and ϵ -first strategy, where the initial exploration phase in each epoch of the UR strategy is set to be an ϵ fraction of the total number of time steps in that epoch.

On the other hand, UR can become more costly than GR when the number of arms K is large. If UR and GR both recommend the same total number of gold tasks to

the worker, GR tends to recommend more gold tasks from the best task category in the long run, leading to a smaller estimation variance of (2) for the best category and a smaller asymptotic regret than UR.

In both the GR and UR strategies, we choose epoch function $\tau(r) = \Theta(r^2)$, which determines how frequently gold tasks are recommended to the worker. A more general choice is $\tau(r) = \Theta(r^\gamma)$ for $\gamma \geq 1$. We now show that $\gamma = 2$ is essentially the only choice that gives optimal order regret in GR and UR. Let $f(n)$ be the number of gold tasks recommended till time n . Then, since GR and UR recommends a fixed number of gold tasks in each epoch, we have $f(n) = \Theta(r)$ if n is within epoch r . The number of time steps up to and including epoch r is $\Theta(\tau(r))$, which implies that $f(n) = \Theta(n^{1/\gamma})$. Then from (8), we have

$$\begin{aligned}
 R(n) &\geq \min \left\{ \frac{an}{b_1 n^{1/\gamma}} + q_* p_* b_2 n^{1/\gamma} - a, n q_* p_* \right\} \\
 &= \min \left\{ \frac{a}{b_1} n^{1-1/\gamma} + q_* p_* b_2 n^{1/\gamma} - a, n q_* p_* \right\},
 \end{aligned}$$

for some positive constants a, b_1 , and b_2 . Therefore, if $\gamma \neq 2$, $R(n)/\sqrt{n} \rightarrow \infty$ as $n \rightarrow \infty$.

5 SIMULATIONS AND PERFORMANCE EVALUATION

As our MAB framework and strategies are *online* strategies, there are no real data sets that can be used to verify the performance of our proposed approaches. Therefore, simulations are necessary to provide insights into different aspects of our strategies that we are not able to show theoretically. In this section, we perform simulations using GR, UR, and the ϵ -first strategy. For comparison, we include the quality check mechanism named honeypot used in [14] as the baseline, where one gold task is recommended to the worker randomly per 100 time steps. To avoid any doubts, the full honeypot procedure is shown in Algorithm 4. We also compare with the UR strategy with epoch function $\tau(r) = \lceil \alpha r^\gamma \rceil$, for $\gamma > 0$, which we call the UR(γ) strategy. The UR(2) strategy is equivalent to the UR strategy in Algorithm 2.

Algorithm 4 Honeypot

Inputs: $\alpha > 0$.

Initialization: Recommend one gold task from each arm. Set $b_0 = K$. Select a random sequence $b_i, i \geq 1$, where each b_i is drawn uniformly and randomly from $100i + K - 99$ to $100i + K$. Set $j = 0$.

- 1: **loop**
 - 2: Select the empirical best arm $c_j = \arg \max_k \bar{Y}_k$.
 - 3: From time step $b_j + 1$ to $b_{j+1} - 1$, recommend non-gold tasks from task category c_j .
 - 4: At time step b_{j+1} , randomly choose one task category and recommend one gold task from it.
 - 5: Set $j = j + 1$.
 - 6: **end loop**
-

To ensure that simulations are realistic and representative, similar to [6], [10], [16], [58], we simulate the worker's reliabilities according to worker behavior types studied in

[10], [15]. Specifically, following the statistical result reported in [10], we simulate 3 types of worker behaviors in different task categories:

- 1) a reliable worker in 43% of task categories where her reliabilities are drawn from $\mathcal{U}(0.6, 1)$,
- 2) a sloppy worker in 32% of task categories where her reliabilities are drawn from $\mathcal{U}(0.2, 0.6)$, and
- 3) a spammer in 25% of task categories where her reliabilities are drawn from $\mathcal{U}(0, 0.2)$,

with $\mathcal{U}(a, b)$ denoting the uniform distribution on the interval $[a, b]$. The worker's behaviors and the task category information are summarized in Table 1, where the numbers of task categories have been rounded to integers. In all of our simulations, we use different settings shown in Table 1, except the simulations in Section 5.5 where we mimic cheating behaviors [8] in crowdsourcing with deteriorated reliabilities.

We track the regret as our performance measure. In each simulation, we perform 2000 trials, each with 10^3 time steps. To compare the performance of different approaches, we compute the empirical average regret, over all the trials. For convenience, this average is referred to as the *average regret*.

In Theorem 2, we show that for GR, a sufficient condition, under which the order optimal regret is achieved, is $c > \max\{5d^2, 2\}$. However, in simulations we find that a smaller c can lead to better performance for all settings listed in Table 1. Therefore, in the simulations, we set $c = 0.05$ in GR. In practice, the parameter d in GR can be set to be a fixed small value like 0.1. We set the parameter d in GR to be 0.1. In all simulations except those in Section 5.3, we set $\alpha = 0.1$ for GR, UR and $\text{UR}(\gamma)$ for all γ . In Section 5.3, we test the effect of parameter α , which controls the frequency of gold tasks. In our strategies, fewer gold tasks are recommended with larger α . We use $\beta = 50$ in all simulations except those simulations in Section 5.3 where we study the effect of varying β and α .

5.1 Comparison of Different Strategies

We first compare the performance of different strategies. In Figs. 1 and 2, we use the setting $K = 10$ in Table 1 for our simulations. We observe the following.

- GR, UR, and the ϵ -first strategy outperform honeypot, UR(1.5) and UR(15) when the number of time steps n is large. In honeypot, gold tasks are recommended uniformly without leveraging on the information learned up to each time step, leading to a large average regret. In UR(1.5), gold tasks are recommended more frequently than GR and UR. Therefore, less non-gold tasks are recommended to the worker. On the other hand, UR(15) corresponds to the case where gold tasks are recommended infrequently compared to GR and UR. In Fig. 2, we see that $\gamma = 2$ achieves the optimal regret order, in line with our observations in Section 4.4.
- The ϵ -first strategy performs best among all the strategies, but it requires knowledge of the total number of time steps n beforehand.
- GR outperforms UR. One reason is the exploration of all arms at every epoch in UR reduces the cumulative reward with fewer non-gold tasks assigned. Another reason is even if GR and UR have recommended the

Fig. 1. Performance comparison between different strategies using the setting $K = 10$ in Table 1 with $\beta = 50$.

Fig. 2. Average regret order of $\text{UR}(\gamma)$ under the setting $K = 10$ in Table 1 with $\beta = 50$.

same total number of gold tasks to the worker, the number of gold tasks recommended from the best category using UR is less than that using GR. Therefore, the variance of the empirical mean (2) for the best category using UR is larger than that using GR, leading to a larger average regret.

5.2 Varying Total Number of Task Categories K

We next show the performance of honeypot, UR, GR and the ϵ -first strategy for different number of arms K using

TABLE 1
Simulation settings

Task category types	Total number of task categories K			Worker's behaviors	
	$K = 5$	$K = 10$	$K = 20$	Reliabilities p_k	Preferences q_k
Reliable	2	4	9	$\mathcal{U}(0.6, 1.0)$	$\mathcal{U}(0, 1.0)$
Sloppy	2	3	6	$\mathcal{U}(0.2, 0.6)$	$\mathcal{U}(0, 1.0)$
Spammer	1	3	5	$\mathcal{U}(0, 0.2)$	$\mathcal{U}(0, 1.0)$

the settings listed in Table 1. From Fig. 3, we observe the following:

- GR outperforms UR in the setting $K = 10$ and $K = 20$, while UR performs slightly better than GR in the setting $K = 5$. This is because when K is small, the cost of uniformly recommending gold tasks from all task categories in GR is smaller with better estimations on the more unreliable task categories.
- The average regret with more task categories is larger than that with fewer task categories, which is expected.
- In all cases, honeypot performs worst and ϵ -first performs best. The explanation is similar to that given in Section 5.1.

Fig. 3. Performance comparison between different strategies with $K = 5, 10$ and 20 using settings in Table 1 with $\beta = 50$.

5.3 Varying Leading Coefficient α of the Epoch Function

In this section, we vary the parameter α in GR and UR separately with various values of β and K . We first fix $\beta = 50$ and study the effect of α for different K . The results are shown in Figs. 4 and 5, respectively. We observe the following:

- We note that a moderate α is optimal for both UR and GR. When α is too large, the strategies recommend gold tasks too infrequently, leading to a larger estimation variance. On the other hand, a small α results in too many gold tasks.

- The optimal α for GR is smaller than that for UR. This is because if α is set to be the same for UR and GR, UR tends to recommend more gold tasks than GR, which makes UR more costly as we have discussed in Section 5.1.
- The optimal α in GR tends to decrease with increasing K . This is because with more task categories, GR needs more explorations to determine the best category using more gold tasks with a smaller α . In contrast, the optimal α in UR tends to increase with increasing K . One possible reason is that in principle with more task categories, we need more explorations to determine the best category using more gold tasks, but the cost of UR's uniform recommendation behavior has canceled out the benefits coming from better estimations with more gold tasks. Therefore, UR tends to use less gold tasks when K is large.

We next fix $K = 10$ and study the effect of α for different β . The results are shown in Figs. 6 and 7, respectively. We observe that the optimal α UR and GR both tend to decrease with increasing K . This is because large β emphasizes the importance of the estimation accuracy of $\bar{X}_k(t)$. The platform designer is supposed to set β larger and consequently reduces the uncertainty of the estimator with more gold tasks recommended to the worker.

Fig. 4. Performance of GR with $\beta = 50$ and different values of the parameter α under different settings in Table 1.

Fig. 5. Performance of UR with $\beta = 50$ and different values of the parameter α under different settings in Table 1.

Fig. 7. Performance of UR with different values of the parameter α and different β under the setting $K = 10$ in Table 1.

Fig. 6. Performance of GR with different values of the parameter α and different β under the setting $K = 10$ in Table 1.

Fig. 8. Performance of different strategies considering either preferences or reliabilities under the setting $K = 10$ in Table 1.

5.4 Learning Only Partial Information

We next consider a crowdsourcing platform that learns only either a worker's preferences or reliabilities but not both, when performing task recommendations. The simulation results are shown in Fig. 8 using the setting $K = 10$ in Table 1. We see that learning either only the preferences or reliabilities results in a lower cumulative reward, which is expected. The loss in cumulative reward for GR, UR or the ϵ -first strategy learning either only the preferences or reliabilities widens compared to GR, UR or the ϵ -first strategy learning both preferences and reliabilities, respectively

when the number of time steps n becomes large.

5.5 Robustness under Cheating

The paper [8] noted that workers who have passed a qualification test may choose to complete their tasks like spammers. We mimic this cheating behavior using the setting $K = 10$ in Table 1. We now assume the worker adopts a spammer behavior in task categories where she originally performs the best after time step $F = K = 10$, i.e., after the initial calibration period. From Fig. 9, we observe the following:

Fig. 9. Performance of different strategies under cheating with the setting $K = 10$ in Table 1.

- We note that the regret curves of UR, GR and the ϵ -first strategy are still sub-linear. This agrees with our analysis in Section 4. In fact, even if the worker chooses to cheat in the initial gold tasks, our sequential learning strategies guarantee optimal regret order with more accurate estimation of her true preferences and reliabilities over time.
- Honeypot has the worst performance because it recommends gold tasks in a random fashion and does not optimize based on information learned up to each time step, unlike the MAB based approaches.

6 CONCLUSION

We have formulated the task recommendation problem in crowdsourcing as a MAB problem in which the reward is unobservable. To learn the worker's preferences and reliabilities for different task categories, we have included gold tasks whose solutions are known *a priori*, but which however do not produce any rewards. We showed that the optimal regret order is $O(\sqrt{n})$, where n is the number of time steps. We proposed three strategies and showed that all achieve the optimal regret order. Simulations verify the efficiency of our proposed strategies. We believe that the theoretical analysis and extensive simulations can guide platform designers in building more efficient recommendation systems in crowdsourcing.

In this paper, we have assumed that the worker's preferences and reliabilities for different task categories are independent. However, in practice, preferences and reliabilities for different categories may be dependent, and such correlation information can be used to enhance the exploration phase. A future research direction is to consider the case where workers' reliabilities and preferences for different task categories may be correlated, and to study how to recommend tasks from different categories to multiple

workers. In this paper, we have demonstrated that even with cheating behavior, our strategies perform well. However our strategies cannot detect complicated state changes. The worker may choose to cheat within some time intervals and then go back to her normal state. Combining techniques from quickest change detection [59] may be useful in this case.

REFERENCES

- [1] CrowdFlower Company. *CrowdFlower Website*. (2007) [Online]. Available: <https://www.crowdflower.com/>, Accessed on: Feb. 25, 2018.
- [2] Amazon Company. *Amazon Mechanical Turk (MTurk) Website*. (2005) [Online]. Available: <https://www.mturk.com/>, Accessed on: Feb. 25, 2018.
- [3] CrowdSPRING LLC. *crowdspring website*. (2008) [Online]. Available: <https://www.crowdspring.com/>, Accessed on: Feb. 25, 2018.
- [4] J. Ross, L. Irani, M. Silberman, A. Zaldivar, and B. Tomlinson, "Who are the crowdworkers? Shifting demographics in Mechanical Turk," in *Proc. SIGCHI Conf. Hum. Factors Comput. Syst.*, 2010, pp. 2863–2872.
- [5] L. R. Varshney, "Privacy and reliability in crowdsourcing service delivery," in *Proc. Annu. SRII Global Conf.*, 2012, pp. 55–60.
- [6] A. Vempaty, L. R. Varshney, and P. K. Varshney, "Reliable crowdsourcing for multi-class labeling using coding theory," *IEEE J. Sel. Topics Signal Process.*, vol. 8, no. 4, pp. 667–679, Aug. 2014.
- [7] Q. Kang and W. P. Tay, "Sequential multi-class labeling in crowdsourcing," *IEEE Trans. Knowl. Data Eng.*, vol. 31, no. 11, pp. 2190–2199, Nov. 2019.
- [8] O. Alonso and R. Baeza-Yates, "Design and implementation of relevance assessments using crowdsourcing," in *Proc. Eur. Conf. Inform. Retrieval*, 2011, pp. 153–164.
- [9] C. Grady and M. Lease, "Crowdsourcing document relevance assessment with mechanical turk," in *Proc. Annu. Conf. North Amer. Chapter Assoc. Comput. Linguistics*, 2010, pp. 172–179.
- [10] J. Vuurens, A. P. de Vries, and C. Eickhoff, "How much spam can you take? an analysis of crowdsourcing results to increase accuracy," in *Proc. ACM SIGIR Workshop Crowdsourcing Inform. Retrieval*, 2011, pp. 21–26.
- [11] A. Kittur, E. H. Chi, and B. Suh, "Crowdsourcing user studies with mechanical turk," in *Proc. SIGCHI Conf. Human Factors Comput. Syst.*, 2008, pp. 453–456.
- [12] D. Oleson, A. Sorokin, G. P. Laughlin, V. Hester, J. Le, and L. Biewald, "Programmatic gold: Targeted and scalable quality assurance in crowdsourcing," in *AAAI Workshop on Human Computation*, 2011, pp. 43–48.
- [13] L. Spitzner, "Honeypots: Catching the insider threat," in *Proc. Annu. Comput. Security Appl. Conf.*, 2003, pp. 170–179.
- [14] M. Rokicki, S. Zerr, and S. Siersdorfer, "Group sourcing: Team competition designs for crowdsourcing," in *Proc. Int. Conf. World Wide Web*, 2015, pp. 906–915.
- [15] G. Kazai, J. Kamps, and N. Milic-Frayling, "Worker types and personality traits in crowdsourcing relevance labels," in *Proc. Int. Conf. Inform. Knowledge Manage.*, 2011, pp. 1941–1944.
- [16] N. Q. V. Hung, N. T. Tam, L. N. Tran, and K. Aberer, "An evaluation of aggregation techniques in crowdsourcing," in *Proc. Int. Conf. Web Inform. Syst. Eng.*, 2013, pp. 1–15.
- [17] M. Safran and D. Che, "Real-time recommendation algorithms for crowdsourcing systems," *Appl. Comput. Inform.*, vol. 13, no. 1, pp. 47–56, Jan. 2017.
- [18] The size of the mechanical turk marketplace. Accessed: Jun. 25, 2019. [Online]. Available: <https://www.pewresearch.org/internet/2016/07/11/the-size-of-the-mechanical-turk-marketplace/>
- [19] M. L. Gray and S. Suri, *How to Stop Silicon Valley from Building a New Global Underclass*. New York, USA: Houghton Mifflin Harcourt, 2019.
- [20] W. Mason and D. J. Watts, "Financial incentives and the performance of crowds," in *Proc. ACM SIGKDD Workshop Human Comput.*, 2009, pp. 77–85.
- [21] G. Kazai, "In search of quality in crowdsourcing for search engine evaluation," in *Proc. Eur. Conf. Inform. Retrieval*, 2011, pp. 165–176.
- [22] T. L. Lai and H. Robbins, "Asymptotically efficient adaptive allocation rules," *Advances Appl. Math.*, vol. 6, no. 1, pp. 4–22, Mar. 1985.

- [23] P. Auer, N. Cesa-Bianchi, and P. Fischer, "Finite-time analysis of the multiarmed bandit problem," *Mach. Learning*, vol. 47, no. 2, pp. 235–256, May 2002.
- [24] D. R. Karger, S. Oh, and D. Shah, "Budget-optimal task allocation for reliable crowdsourcing systems," *Oper. Res.*, vol. 62, no. 1, pp. 1–24, Feb. 2014.
- [25] —, "Efficient crowdsourcing for multi-class labeling," in *Proc. ACM Inter. Conf. Measurement Modeling Comput. Syst.*, 2013, pp. 81–92.
- [26] F. Basik, B. Gedik, H. Ferhatosmanoglu, and K.-L. Wu, "Fair task allocation in crowdsourced delivery," *IEEE Trans. Services Comput.*, 2018, in press.
- [27] J. Shu, X. Jia, K. YANG, and H. Wang, "Privacy-preserving task recommendation services for crowdsourcing," *IEEE Trans. Services Comput.*, 2018, in press.
- [28] A. Khetan and S. Oh, "Reliable crowdsourcing under the generalized Dwid-Skene model," *arXiv preprint arXiv:1602.03481*, 2016.
- [29] D. Zhou, Q. Liu, J. C. Platt, C. Meek, and N. B. Shah, "Regularized minimax conditional entropy for crowdsourcing," *arXiv preprint arXiv:1503.07240*, 2015.
- [30] O. Cappé and E. Moulines, "Online EM algorithm for latent data models," *J. Royal Statist. Soc.: Series B (Statist. Methodology)*, vol. 71, no. 3, pp. 593–613, Feb. 2009.
- [31] Q. Kang and W. P. Tay, "Sequential multi-class labeling in crowdsourcing: A Ulam-Rényi game approach," in *Proc. Int. Conf. Web Intell.*, 2017, pp. 245–251.
- [32] L. Massoulié, M. I. Ohannessian, and A. Proutière, "Greedy-Bayes for targeted news dissemination," in *Proc. Int. Conf. Meas. Modelling Comput. Syst.*, 2015, pp. 285–296.
- [33] C.-J. Ho, S. Jabbari, and J. W. Vaughan, "Adaptive task assignment for crowdsourced classification," in *Proc. Int. Conf. Mach. Learning*, 2013, pp. 534–542.
- [34] C.-J. Ho and J. W. Vaughan, "Online task assignment in crowdsourcing markets," in *Proc. AAAI Conf. Artificial Intell.*, 2012, pp. 45–51.
- [35] L. Tran-Thanh, S. Stein, A. Rogers, and N. R. Jennings, "Efficient crowdsourcing of unknown experts using bounded multi-armed bandits," *Artificial Intell.*, vol. 214, no. 1, pp. 89–111, May 2014.
- [36] H. Kellerer, U. Pferschy, and D. Pisinger, *Knapsack problems*. 2004. Berlin: Springer, 2003.
- [37] Y. Tong, J. She, B. Ding, L. Wang, and L. Chen, "Online mobile micro-task allocation in spatial crowdsourcing," in *Proc. IEEE Int. Conf. Data. Eng.*, 2016, pp. 49–60.
- [38] J. Fan, G. Li, B. C. Ooi, K.-I. Tan, and J. Feng, "iCrowd: An adaptive crowdsourcing framework," in *Proc. SIGMOD Int. Conf. Manage. Data*, 2015, pp. 1015–1030.
- [39] S. K. 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, May 2018.
- [40] Y. Liu and M. Liu, "An online learning approach to improving the quality of crowd-sourcing," *IEEE/ACM Trans. Netw.*, vol. 25, no. 4, pp. 2166–2179, Aug. 2017.
- [41] S. Jain, B. Narayanaswamy, and Y. Narahari, "A multiarmed bandit incentive mechanism for crowdsourcing demand response in smart grids," in *Proc. AAAI Conf. Artificial Intell.*, 2014, pp. 721–727.
- [42] Y. Chen and X. Yin, "Stable job assignment for crowdsourcing," in *Proc. Global Commun. Conf.*, 2017, pp. 51–56.
- [43] M.-C. Yuen, I. King, and K.-S. Leung, "TaskRec: A task recommendation framework in crowdsourcing systems," *Neural Process. Lett.*, vol. 41, no. 2, pp. 223–238, Feb. 2015.
- [44] —, "Task recommendation in crowdsourcing systems," in *Proc. Inter. Workshop Crowdsourcing and Data Mining*, 2012, pp. 22–26.
- [45] S. Vakili and Q. Zhao, "Risk-averse multi-armed bandit problems under mean-variance measure," *IEEE J. Sel. Topics Signal Process.*, vol. 10, no. 6, pp. 1093–1111, Sep. 2016.
- [46] —, "Mean-variance and value at risk in multi-armed bandit problems," in *Proc. Annu. Allerton Conf. Commun. Control, and Comput.*, 2015, pp. 1330–1335.
- [47] A. Sani, A. Lazaric, and R. Munos, "Risk-aversion in multi-armed bandits," in *Proc. Advances Neural Inf. Process. Syst.*, 2012, pp. 3275–3283.
- [48] M. C. Steinbach, "Markowitz revisited: Mean-variance models in financial portfolio analysis," *SIAM review*, vol. 43, no. 1, pp. 31–85, Feb. 2001.
- [49] H. Markowitz, "Portfolio selection," *J. Finance*, vol. 7, no. 1, pp. 77–91, Mar. 1952.
- [50] D. Geiger, S. Seedorf, T. Schulze, R. C. Nickerson, and M. Schader, "Managing the crowd: Towards a taxonomy of crowdsourcing processes," in *Proc. Amer. Conf. Inform. Syst.*, 2011, [Online]. Available: http://aisel.aisnet.org/amcis2011_submissions/430/.
- [51] N. Kaufmann, T. Schulze, and D. Veit, "More than fun and money. Worker motivation in crowdsourcing – A study on mechanical turk," in *Proc. Amer. Conf. Inform. Syst.*, 2011, [Online]. Available: http://aisel.aisnet.org/amcis2011_submissions/340.
- [52] J. Jarrett, M. B. Blake, and I. Saleh, "Crowdsourcing, mixed elastic systems and human-enhanced computing—a survey," *IEEE Trans. Services Comput.*, vol. 11, no. 1, pp. 202–214, May 2018.
- [53] H. Li and B. Yu, "Error rate bounds and iterative weighted majority voting for crowdsourcing," *arXiv preprint arXiv:1411.4086*, 2014.
- [54] Y. Koren, R. Bell, and C. Volinsky, "Matrix factorization techniques for recommender systems," *Comput.*, vol. 42, no. 8, pp. 30–37, Aug. 2009.
- [55] H. Wang, Q. Wu, and H. Wang, "Learning hidden features for contextual bandits," in *Proc. Int. Conf. Inform. Knowl. Manag.*, Indianapolis, USA, Oct. 2016, pp. 1633–1642.
- [56] J.-Y. Audibert, R. Munos, and C. Szepesvári, "Exploration–exploitation tradeoff using variance estimates in multi-armed bandits," *Theoretical Comput. Sci.*, vol. 410, no. 19, pp. 1876–1902, Oct. 2009.
- [57] L. Tran-Thanh, A. Chapman, J. E. Munoz De Cote Flores Luna, A. Rogers, and N. R. Jennings, " ϵ -First policies for budget-limited multi-armed bandits," in *Proc. AAAI Conf. Artificial Intell.*, 2010, pp. 1211–1216.
- [58] Q. V. H. Nguyen, C. T. Duong, T. T. Nguyen, M. Weidlich, K. Aberer, H. Yin, and X. Zhou, "Argument discovery via crowdsourcing," *Int. J. Very Large Data Bases*, vol. 26, no. 4, pp. 511–535, Apr. 2017.
- [59] T. S. Lau and W. P. Tay, "Quickest change detection in the presence of a nuisance change," *IEEE Trans. Signal Process.*, vol. 67, no. 20, pp. 5281–5296, Oct. 2019.

Qiyu Kang received the B.S. degree in Electronic Information Science and Technology from University of Science and Technology of China in 2015, and the Ph.D. degree in Electrical and Electronic Engineering from Nanyang Technological University in 2020. He is currently a research fellow in the School of Electrical and Electronic Engineering at Nanyang Technological University. His research interests include sequential learning, collaborative computing, and social network.

Wee Peng Tay (S'06 M'08 SM'14) received the B.S. degree in Electrical Engineering and Mathematics, and the M.S. degree in Electrical Engineering from Stanford University, Stanford, CA, USA, in 2002. He received the Ph.D. degree in Electrical Engineering and Computer Science from the Massachusetts Institute of Technology, Cambridge, MA, USA, in 2008. He is currently an Associate Professor in the School of Electrical and Electronic Engineering at Nanyang Technological University, Singapore. His research in-

terests include information and signal processing over networks, distributed inference and estimation, information privacy, machine learning, information theory, and applied probability. Dr. Tay received the Tan Chin Tuan Exchange Fellowship in 2015. He is a coauthor of the best student paper award at the Asilomar conference on Signals, Systems, and Computers in 2012, and the IEEE Signal Processing Society Young Author Best Paper Award in 2016. He was an Associate Editor for the IEEE Transactions on Signal Processing (2015 – 2019), and is currently an Associate Editor for the IEEE Transactions on Signal and Information Processing over Networks, an Editor for the IEEE Transactions on Wireless Communications and IEEE Open Journal of Vehicular Technology. He serves on the Machine Learning for Signal Processing Technical Committee of the IEEE Signal Processing Society.