Adaptive Crowdsourcing for Temporal Crowds

L. Elisa Celis
Xerox Research Center India
elisa.celis@xerox.com

Koustuv Dasgupta
Xerox Research Center India
koustuv.dasgupta@xerox.com

Vaibhav Rajan
Xerox Research Center India
vaibhav.rajan@xerox.com

ABSTRACT

Bandit problems embody in essential form a conflict evident in all human action: information versus immediate payoff.
–P. Whittle (1989)

Crowdsourcing is rapidly emerging as a computing paradigm that can employ the collective intelligence of a distributed human population to solve a wide variety of tasks. However, unlike organizational environments where workers have set work hours, known skill sets and performance indicators that can be monitored and controlled, most crowdsourcing platforms leverage the capabilities of *fleeting* workers who exhibit changing work patterns, expertise, and quality of work. Consequently, platforms exhibit significant variability in terms of performance characteristics (like response time, accuracy, and completion rate). While this variability has been folklore in the crowdsourcing community, we are the first to show data that displays this kind of changing behavior. Notably, these changes are not due to a distribution with high variance; rather, the *distribution itself is changing over time.*

Deciding which platform is most suitable given the requirements of a task is of critical importance in order to optimize performance; further, making the decision(s) adaptively to accommodate the dynamically changing crowd characteristics is a problem that has largely been ignored. In this paper, we address the changing crowds problem and, specifically, propose a multi-armed bandit based framework. We introduce the simple ε-smart algorithm that performs robustly. Counterfactual results based on real-life data from two popular crowd platforms demonstrate the efficacy of the proposed approach. Further simulations using a random-walk model for crowd performance demonstrate its scalability and adaptability to more general scenarios.

Categories and Subject Descriptors

H.4 [**Information Systems Applications**]: Miscellaneous

Keywords

Crowdsourcing, Multi-armed Bandits, Adaptive Optimization, Online Algorithms

1. INTRODUCTION

The multi-armed bandit problem models online decision making under uncertainty where an agent simultaneously attempts to *explore*, i.e., acquire new knowledge, and *exploit*, i.e., optimize decisions based on existing knowledge. Each *arm* represents a different option which will result in a random *reward* specified by an *unknown reward function.* This problem has a long and rich history [3], and has been used to design effective algorithms for a large number of real-world problems[16, 6].

Platforms like Amazon Mechanical Turk, CrowdFlower and MobileWorks allow *requesters* to post *microtasks* for workers to complete for relatively small amounts of money. Despite the small size of payments, crowdsourcing markets attract a diverse pool of workers from around the world, who participate both to pick up extra cash and beat boredom [9]. Matching the terminology of a multi-armed bandit, the requester gets a *reward* in the form of the *quality of work* returned (as measured by whichever metric the requester chooses). The reward, as above, can be captured by a random unknown reward function. Each *arm* then corresponds to a potential *job posting*, which is described by the platform, payment amount, worker restrictions, deadline, and other relevant parameters.[1] Our goal, as in the usual multi-armed bandit problem, is to choose the arm that maximizes our reward.

By choosing an arm, we effectively observe a sample from that arm's reward function. In the *static* multi-armed bandit problem, the reward functions are *time invariant.* Hence, an algorithm for arm-selection can eventually obtain almost perfect information and adapt its strategy so that its reward converges to that of an optimal strategy; essentially, it converges to a state where it need only exploit. However, due to high turnover and other effects, the crowd workers change often [10], and with it, as shown in Section 2, the reward changes as well. Hence, a static multi-armed bandit model cannot fully capture the behavior in crowdsourcing markets. Instead, we could consider the non-static situation where reward functions are *chosen by an adversary.* However, this worst-case model would not be representative of the *gradual* or *stochastic* change we expect to observe. In

[1]Note that while changing some aspects of the job posting (e.g., payment amount) may be easy to automate, other aspects (e.g., which platform) may have significant up front cost. While this may, in practice, be prohibitively cumbersome for an individual researcher, with the emergence of Enterprise Crowdsourcing (e.g., [5]) it is relevant, and hence we do not incorporate this up-front cost in our calculations.

Section 2, we show data that suggests crowd behavior indeed varies temporally but not arbitrarily. Hence, a better model is one where reward functions are *dynamic* but reasonably behaved. In this setting, an agent must continuously balance explore and exploit steps in order to adapt to the dynamic reward functions, but need not be as defensive as against an adversary. To this end, we describe a *random-walk model* in Section 3 which we then employ in our simulations.

The quality of an algorithm is measured by the *expected regret* incurred. There are two common notions of regret; 1) *strong regret* measures the expected difference between our performance and that of the optimal (hypothetical) omniscient algorithm which always chooses the arm with best expected reward. 2) *weak regret* measures the expected difference between our performance and that of the optimal (hypothetical) omniscient algorithm that is restricted to using the same arm at every time step. Strong regret is particularly relevant for static reward functions since one can converge to the optimal strategy. However, in the dynamic (i.e., temporal) case, this is not possible; we must continuously balance explore and exploit steps. In this case, we simply hope to *not lose too much*. However, with regard to weak regret, we have some hope of outperforming the best single-arm strategy. This is particularly relevant since in a crowdsourcing setting, where typically a single task description (i.e., platform, payment, etc) is chosen - thus, an algorithm that can *adapt* to the changing landscape by changing its choice of task description could provide dramatic benefits.

Our Contributions

We are the first, to our knowledge, to bring to light data that suggests that temporal variations in crowd performance indeed occurs. We present our data and methodology in Section 2. We further propose a multi-armed bandit approach to explore and exploit these temporal trends. Prior art in the emerging area of crowdsourcing has not addressed this problem of adaptation w.r.t crowd platform selection under dynamic variations - in fact, the focus has been almost entirely on optimizing the selection of a *single* platform. We believe a multi-armed bandit framework would be useful in general, and present a specific ε-smart algorithm which makes the simple observation that at a given time step, *not every arm is worth exploring*. Additionally, we consider four natural existing comparator algorithms: a classic ε-*greedy* algorithm inspired by the static case, a version of $EXP3$ inspired by the adversarial case [17], and our baselines - a *Random* algorithm that randomly chooses on the first time step and sticks with it, and a slightly more informed *Bootstrap* algorithm that first samples each arm and then sticks with the best observation. The latter two algorithms capture current practice where either an arbitrary choice or one based on some initial experiments is made. We run counterfactual experiments on our data, presented in Section 5.1, and discuss the performance of the various algorithms. Since our data is limited, we additionally introduce a random walk model in Section 3 that captures temporal variation. We run additional simulations on this model in order to test the algorithms for robustness in varied environments. Thus far, there has been no study of how to address temporal variations in crowd performance and further optimize performance under dynamically varying conditions. To this end, we believe that this paper sheds key insights on this problem and some interesting solution approaches. As we collect data for longer time periods, we expect the proposed approaches to be increasingly useful in long-term adaptations to changing crowd behavior.

Related Work

The multi-armed bandit framework found early applications in the area of clinical trials and adaptive routing efforts for minimizing delays in a network. In economics, experimental consumption is a leading example of an allocation problem where the tradeoff between current payoff and value of information plays a key role. Multi-armed bandit solutions have been suggested for matching advertisements to users, displaying content such as news articles to web viewers, and scheduling multi-threaded processors(refer to survey [12] for a thorough introduction into bandit applications and algorithms). In crowdsourcing in particular, multi-armed bandit algorithms have been used for learning and/or matching the skills of specific workers (e.g., [8] and [18]). However, these papers do not address the task of crowd (platform)selection, and more significantly, assume a static model.

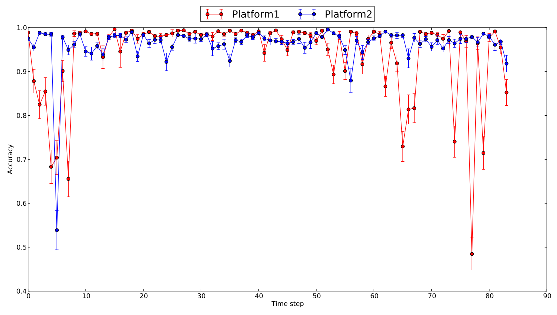
A number of models have been proposed for capturing the dynamic aspect of the MAB problem. Motivated by task scheduling, [6] considered the case where only the state of the active arm (the arm currently being played) can change in a given step, giving an optimal policy for the Bayesian formulation with time discounting. This seminal result gave rise to several rich lines of work, including restless bandits introduced in [19], where states of all arms can change in each step according to a known (but arbitrary) stochastic transition function. While Guha et al. [17] have made progress on certain special cases, restless bandits are notoriously intractable, and computing an (approximately) optimal strategy is PSPACE-hard [14]. Most relevantly, Upfal and Slivkins [17] consider a Brownian motion model for restless bandits and propose several algorithms, giving theoretical bounds on the strong regret they incur. In particular, we compare against their best algorithm, a variant of $EXP3$, defined in Section 4.

Several forms of outsourcing have become a crucial component of business processes [15]. However, selecting which outsourcing options to use has thus far been an ad-hoc process. Often, this is done arbitrarily; at best, benchmarked data is used to select a reasonable platform and configuration [2]. Clearly, a benchmarked-type solution relies on the assumption that reward functions are static. Furthermore, even with benchmarking, the stochasticity in the system means that over time, the optimal platform will change, necessitating an adaptive solution. The approach proposed in this paper is a first-of-its-kind that strives to leverage this temporal variance in crowd platforms and use the information for (platform) selection.

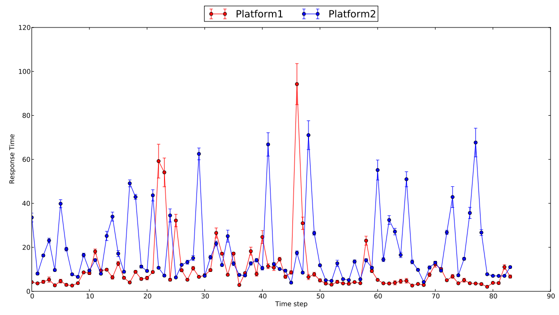
2. CROWDS CHANGE: DATA

In this section we describe our model and results with data collected from two popular crowd platforms[2] that provide efficient APIs to (i) programmatically inject crowd tasks into the platforms based on user specifications, and (ii) curate performance characteristics that affect the decision-making. Further, the platforms themselves claim differences in the inherent mix of crowd workers: e.g., workers on Platform 1

[2]Names omitted for anonymity.



(a) **Accuracy**. Which platform is better changes 27 times in 84 time steps. Platforms have average accuracies of 96.27 and 94.01 percent respectively.



(b) **Response Time**. Which platform is better changes 26 times in 84 time steps. Platforms have average response times of 18.99 and 10.18 seconds respectively.

Figure 1: Performance statistics of two platforms over one week, plotted at 2hr intervals, resulting in 84 time steps.

are predominantly from US and India, while those on Platform 2 are mainly from developing nations including India, Philippines, Bangladesh, and Morocco.

We chose to analyze digitization tasks since they are prevalent in crowdsourcing and hence workers are familiar with their format. In particular, we considered an insurance form of a healthcare provider[3] that was hand-filled by volunteers with ficticious user data. The handwritten forms were presented in the form of crowd tasks, and workers were asked to digitize three fields: "patient's legal name", "identification number", and "briefly describe injury". Each task paid one cent (USD) for the extraction of all three fields. A corpus of more than 1000 forms were used. For each of the forms processed by the crowd, we consider the following performance characteristics:

Accuracy accuracy of the response received (based on gold data, i.e., the ground truth which was available to us).

Response Time: measured from time of posting the task to the time when results are received.

Batches of 50 tasks were posted every hour of the day to each platform. The data was collected over a period of one week resulting in a total of approximately 10,000 tasks on each platform. Performance characteristics are plotted in Figure 1. Note that Platform 1 performs better with respect to accuracy, while Platform 2 performs better with respect to response time.

3. CROWDS CHANGE: A MODEL

While we will test our algorithms on the above data, the data set is admittedly limited. Not only is the time frame small, but we only observe two platforms on two different metrics. While we hope to build a general data-based model, our intent here it to begin with a simple model that captures temporal change which we can use in a first attempt to verify the behavior of various algorithms.[3] Gradual and stochastic change is traditionally modeled using random walks or Brownian motion, and hence here we use a random walk model.

Consider a family of probability distributions $\mathcal{A} = \{\mathcal{F}(\mu)\}$ parameterized by their mean μ (and potentially other parameters). An arm in the *random walk model* on $\mathcal{A}$ will be defined by a sequence of means $\mu_0, \mu_1, \ldots$, where the arm's reward at time t is drawn from $\mathcal{F}(\mu_t)$. The μ_ts will form a lazy random walk with reflecting boundaries. To define this precisely, let $M = [\mu_{\min}, \mu_{\max}] \in \mathbb{R}$ be the range of the walk. Let $\mathcal{G}$ be an arbitrary distribution on M, let $p \in [0, 1]$, and consider $\Delta \in \mathbb{R}$. Then, $\mu_0 \sim \mathcal{G}$, and for each t, with probability $1 - p$, $\mu_{t+1} = \mu_t$. With the remaining probability p, let $x = \mu_t \pm \Delta$ where the sign is positive or negative with equal probability. If $x \in M$, then $\mu_{t+1} = x$. Otherwise, if $x > \mu_{\max}$, we let $\mu_{t+1} = 2\mu_{\max} - x$ and if $x < \mu_{\min}$ we let $\mu_{t+1} = 2\mu_{\min} - x$. Hence, this is a lazy random walk with step size Δ and reflecting boundaries defined by M.

Note that the means, unless $M = \mathbb{R}$, do not form a martingale due to the reflecting boundaries. However, they do form a simple (potentially infinite) Markov chain. Contrary to what was stated in the discussion of a similar model in [Slivkins & Upfal 2008], the stationary distribution may not be uniform on M, again, due to the boundary condition. However a stationary distribution exists since the Markov chain is strongly connected and aperiodic. Additionally, due to the symmetry, it is easy to show that the stationary distribution will have expectation $\mu_{\max} - \mu_{\min}/2$. Note that when we select an arm at time t, we do not observe μ_t itself; rather, we observe a sample from $\mathcal{F}(\mu_t)$.

4. CHANGE CROWDS: ALGORITHMS

The multi-armed bandit problem, originally described by Robbins (1952), is a statistical decision model of an agent trying to optimize his decisions while simultaneously improving his information. The inspiration for this problem comes from a gambler, who must decide which of the k different slot machines to play in a sequence of trials so as to maximize his reward. This classical problem has received much attention because of the simple model it provides of the trade-off between exploration (trying out each arm to find the best one) and exploitation (playing the arm believed to give the best payoff). Each choice of an arm results in an immediate random payoff, but the process determining these payoffs can change during the play of the bandit. The distinguishing feature of bandit problems is that the rewards from one arm do not depend on the rewards obtained from other arms. In our scenario, each arm is a potential task description, and the reward is a function of cost, accuracy, and response time or other relevant measurable parameters.

[3]Note specifically that we will not try to fit our parameters to the above data.

Any multi-armed bandit algorithm, on a high level, proceeds as follows:

1. Let $\mathbf{w}$ be some distribution over arms.
2. Choose an arm according to $\mathbf{w}$ and send tasks to a.
3. Receive a *reward* given by the selected arm.[4]
4. Compute a new distribution $\mathbf{w}$ that puts more weight on the arm if it had high reward and less on an arm if it had low reward.
5. Repeat steps 2-4.

From a practical point of view, this type of algorithm learns the benefit of the various services over time, and, more importantly, implicitly adapts its decisions if the services change *without input* from a human counterpart.

4.1 Regret

We now formally define regret; the standard measure for how well a bandit algorithm will perform against hypothetical omniscient algorithms. Consider a family of probability distributions $\mathcal{A} = \{\mathcal{F}(\mu)\}$ parameterized by their mean μ. Let $\{\mu^i\}_{t=1}^{\infty}$ be an infinite sequence where μ_t^i corresponds to the mean of arm i at time t and let $\mathcal{F}_t^i = \mathcal{F}(\mu_t^i)$ be the corresponding probability distributions. More precisely, if we choose arm i at time t, then we get a reward $r_t \sim \mathcal{F}_t^i$. Recall, however, that these reward functions and μs are not known to us a priori. However, for comparison, we will consider a very strong algorithm that is omniscient and, hence, can base its decisions on the μs.

Let $\mu_t^* = \max_i\{\mu_t^i\}$ and $i_t^* = \text{argmax}\{\mu_t^i\}$. Then, the optimal omncient algorithm is the one that plays arm i_t^* at time t. Note that its expected reward at time t is exactly μ_t^*. We now measure the *average strong regret* (henceforth strong regret), i.e., the expected difference between our algorithm's average reward and the omniscient algorithm's reward. After T time steps, if our algorithm A chooses arm i_t^A in the tth time step, strong regret is given by

$$\frac{\sum_{t=1}^{T} \mathbb{E}_A\left[\mu_t^{i_t^A}\right] - \mu_t^*}{T}.$$

Note that we have no hope of comparing favorably against such an algorithm. We can at best demonstrate that we do not behave *too* poorly. However, considering such regret is relevant as it gives the worst-case performance.

We also consider *average weak regret* (henceforth weak regret), which restricts the omniscient algorithm to select a *single arm* and stick with it. This metric is popular in the study of adversarial bandits where there is no hope of comparing against an all-powerful algorithm *and* adversary. However, it is also particularly relevant for us since current practice precisely involves choosing a single option. Since the algorithm is omniscient, it can choose the arm with best expected overall mean, and hence is a strong comparator. More precisely, for T time steps, let $\mu^+ = \max_i\{\sum_{t=1}^{T} \mu_t^i/T\}$ and $i^+ = \text{argmax}_i\{\sum_{t=1}^{T}\mu_t^i\}$. Then, weak regret is given by

$$\frac{\sum_{t=1}^{T} \mathbb{E}_A[\mu_t^{i_t^A}]}{T} - \mu^+.$$

We will compare our algorithms using both of these benchmarks.

[4]This can be a function of various parameters such as cost, response time, and accuracy.

4.2 Multi-Armed Bandit Algorithms

In general, we propose a multi-armed bandit framework for attacking the problem of temporal variability in crowd performance. In order to test this hypothesis, we consider five different multi-armed bandit algorithms. The first two we present, *Random* and *Bootstrap*, correspond to standard practice in the crowdsourcing community; e.g., either selecting a crowd and parameters at random, or with some limited (and, as time progresses, irrelevant) set of statistics. We then consider variants two state of the art algorithms, ε-greedy and $EXP3m$, designed originally for static and adversarial bandits respectively. Finally, we present a new algorithm, ε-smart, which leverages the gradual nature of the temporal change in order to ensure it only explores arms that have *some probability* of providing a higher reward.

Random

To compare against the most naïve case, we consider an agent that simply, at the beginning of time, randomly selects an arm from a probability distribution $\mathcal{D}$ over arms[5] and sticks with it. In this case, the expected reward is initially the expectation of $\mathcal{D}$, and converges to $(\mu_{\max}+\mu_{\min})/2$ as argued in Section 4. Note that *any* $\mathcal{D}$ will converge to this expected reward, including the one that has probability 1 of selecting a specific arm. Hence, for any k and random walk model parameters, the average weak regret of this algorithm will converge to 0. Note that average strong regret at time t is $(\mu_{\max}+\mu_{\min})/2 - \mathbb{E}_{\mathcal{D}}[\max_{i\in[k]}\{\mu_t^k\}]$. Specifically, as $k \to \infty$, we will get $\max_{i\in[k]}\{\mu_t^k\} \to \mu_{\max}$ for all t. Hence, the average strong regret will converge to $(\mu_{\min}-\mu_{\max})/2$ as $k \to \infty$.

Bootstrap

A slightly less naïve algorithm is a bootstrapped version where each arm is first *tested out* b times in some (potentially random) order. The agent then selects the best performing arm and sticks with it. This corresponds to the common practice of collecting data from various platforms in order to make an informed decision at the beginning, but conducting no further experimentation. Note that as $t \to \infty$, both *Random* and *Bootstrap* behave identically since, as above, the random walks will both *mix* and have expected reward $(\mu_{\max}+\mu_{\min})/2$. Hence, bootstrapping can at best give an initial boost to our regret. This could, however, be relevant, especially for short time spans (as in our data), and hence we keep both algorithms as comparators.

ε-greedy

Consider the static situation, which corresponds to the fully-lazy ($p = 0$) version of our model. In this case, there is a classic algorithm, $\varepsilon - decreasing$, that will converge to the optimal arm. This algorithm takes a simple approach, at time t, it chooses the arm with best *sample mean* with probability $1-\varepsilon_t$, and chooses an arm uniformly at random with probability ε_t. Since the arms are static, the sample mean will converge to the actual mean. Hence, we can decrease the probability of exploration since, as the number of samples grows, we become increasingly *certain* of our estimates. For static arms, the algorithm is optimized when $\varepsilon_t \sim 1/\sqrt{t}$,

[5]Such a distribution could potentially be biased based on platform popularity, budget, etc.

and gives tight polylogarithmic strong regret [Cesa-Bianchi & Fischer 1998].

Unfortunately, for $p > 0$, we no longer have the luxury of either taking the sample mean (old samples are no longer relevant) or of decreasing ε to zero (the distributions will keep changing). Hence, we consider a *windowed ε-greedy* algorithm that keeps a constant ε and uses a sample mean based only on the last ℓ observations. Now, with probability $1 - \varepsilon$ the arm with the best *windowed sample mean* is chosen, and an arm is chosen uniformly at random otherwise. Pseudocode is given in Figure 2 for $\ell = 1$, which is the case we consider in our simulation results.

```
ε-greedy Algorithm
input $k \in \mathbb{Z}_+$, $\varepsilon \in [0,1]$
for $t = 1, 2, \dots$
    $\mu^* = \max_i\{\widehat{\mu}^i\}$, $i^* = \mathrm{argmax}_i\{\widehat{\mu}^i\}$
    $i_t = \begin{cases} i^* & \text{with probability } 1-\varepsilon \\ i \neq i^* & \text{with probability } \varepsilon/k-1. \end{cases}$
    $r \sim \mathcal{F}_t^{i_t}$, $\widehat{\mu}^{i_t} = r$
```

Figure 2: Pseudocode for ε-greedy for a k-armed bandit. The parameter ε determines how much we explore. Note that $\ell = 1$, i.e., we simply take the last observation for arm i as our estimate of its mean. A more sophisticated estimate, potentially incorporating more samples, could be used in its place.

EXP3m

In the worst-case model, an *adversary* who *knows our algorithm* gets to decide the reward for each arm at each time step right before we make our selection. While such malice is not observed in practice, this type of analysis is relevant because it gives us a benchmark; if our algorithm performs well in this hopeless-sounding situation, then *no matter what* the real situation is, our algorithm is *not too bad.* The adversary can punish us arbitrarily badly, incurring arbitrarily bad strong regret. However, there exists a multiplicative-weight algorithm that guarantees approximately $\sqrt{\log k/T}$ weak regret when there are k arms after T time steps [Auer et. al. 2002, McMahan & Streeter 2009]. Note that for this algorithm, T is fixed in advance, which allows us to optimize parameters. A very relevant paper [Slivkins & Upfal 2008] considers Brownian motion bandits, a related concept to our random walk version, and provides a variant of EXP3 where the algorithm is re-started every m time steps for a carefully chosen m. The update parameter η can now be optimized by setting $T = m$. Interestingly, Slivkins & Upfal are then able to prove a bound on the average strong regret on their model. We use precisely this algorithm in our experiments, and the pseudocode is given in Figure 3.

ε-smart

We now present our algorithm, a variant of ε-greedy, that relies on a very simple observation: *Not every arm is worth exploring.*[6] Again we have parameters ε and ℓ as before.

[6]Note that [Slivkins & Upfal 2008] consider a similar Brownian motion model for bandits where one can also make this natural observation. However, they provide a different algorithm and give a theoretical analysis for the special case where one can observe μ_t directly.

```
EXP3m Algorithm
input $k \in \mathbb{Z}_+$, $\eta \in [0,1]$, $m \in \mathbb{Z}_+$.
$t = 1$
while true
    $\mathbf{w}_t = (1/k, 1/k, \dots, 1/k)$
    for $s \in [m]$
        $i_t \sim \mathbf{w}_t$, $r \sim F_t^{i_t}$
        $\widehat{\mathbf{r}} = \begin{cases} r/w_{t,i} & \text{if } i = i_t \\ 0 & \text{otherwise.} \end{cases}$
        for $i \in [k]$
            $w_{t+1,i} = \frac{w_{t,i}\exp(-\eta(\mu_{\max}-\widehat{r}))}{\sum_{i=1}^k w_{t,i}\exp(-\eta(\mu_{\max}-\widehat{r}))}$
        $t = t + 1$
```

Figure 3: Pseudocode for EXP3m for a k-armed bandit. Note that the algorithm *restarts* every m time steps, and the parameter η determines how much we weight each observation. The full vector of rewards r is unknown, only the reward r that corresponds to i_t is observed. Hence, for our update step, we use $\widehat{\mathbf{r}}$, which is an unbiased estimator for r.

However, we note that *since each arm is a random walk*, we can often determine, with high probability, that an arm will not outperform our current best choice.

Let $\widehat{\mu}_t^i$ be the ℓ-windowed sample mean for arm i at time t, i.e., the mean of the last ℓ times it has been chosen. Let $\mu_t^* = \max\{\widehat{\mu}\}$ be our current estimate of the best arm $a_t^* = \mathrm{argmax}\{\widehat{\mu}\}$. Let τ_i be the *last time* arm i was pulled, and hence $\mu_{\tau_i}^i$ (or μ_τ^i with some abuse of notation) is our estimate of the mean reward at this time step. Now, since we are assuming μ^i is performing a (lazy) random walk, we know that in t time steps, with high probability, its mean will change by at most $\gamma\sqrt{t}$, where γ can be carefully chosen to provide the desired probability guarantee. Hence, we know that if

$$\mu_t^* - \mu_\tau^i > \gamma\sqrt{t - \tau_i},$$

then *with high probability*, arm i will not outperform our current choice, and we say it is *inactive.* Hence, our algorithm can leverage this, and only choose to explore *active* arms. More precisely, at time t, with probability ε, we choose an active arm at at random if one exists. Otherwise, we choose a_t^*. Pseudocode is given in Figure 4 for $\ell = 1$, which is the case we consider in our simulation results.

5. EXPERIMENTAL RESULTS

We now compare the above algorithms, first with a counterfactual experiment using our data and then on the random walk model.

5.1 Counterfactual Experiment

We begin by considering the accuracy and response time measured on two platforms in 2-hour intervals during a one week period as described in Section 2. We consider the five algorithms described in Section 4, and run the algorithms using the following parameters.
Random: Uniform distribution over platforms.
Bootstrap: $b = 1$, uniform order over platforms.
ε-greedy: $\varepsilon = .03$, $\ell = 1$.

```
ε-smart Algorithm
input: $k \in \mathbb{Z}_+$, $\varepsilon \in [0,1]$, $\gamma \in \mathbb{R}_+$.
$\tau = 0$, $\widehat{\mu}_{\mathbf{0}} = 0$
for $i \in [k]$:
    $r \sim \mathcal{F}_i^i$, $\tau_i = i$.
    $\widehat{\mu}_i^j = \begin{cases} r & \text{if } j = i \\ \widehat{\mu}_{\tau_j}^j & \text{otherwise.} \end{cases}$
for $t = k, k+1, \ldots$
    $\mu^* = \max_i\{\widehat{\mu}^i\}$, $i^* = \mathrm{argmax}_i\{\widehat{\mu}^i\}$
    for $i \in [k]$:
        let $a_i = \begin{cases} 1 & \text{if } \mu^* - \mu_{\tau_i}^i \le \gamma\sqrt{t - \tau_i}, \\ 0 & \text{otherwise.} \end{cases}$
    $i_t = \begin{cases} i^* & \text{with probability } 1-\varepsilon \\ i \text{ s.t. } a_i = 1 & \text{with probability } \varepsilon/\sum_i a_i. \\ i \text{ s.t. } a_i = 0 & \text{with probability } 0. \end{cases}$
```

Figure 4: Pseudocode for ε-smart for a k-armed bandit. The parameter ε determines how much we explore, however, since we only consider *active* arms, any exploration is *targeted* instead of *random*. The *activation parameter* γ can be tuned. Note that i^* will always be active so the algorithm is well-defined.

EXP3m: $\eta = 1/7$ for accuracy, $\eta = 1/250$ for response time. $m = 10$.

ε-smart: $\varepsilon = .1$, $\gamma = 0.01$, $\ell = 1$.

Note that these parameters have not been tuned or optimized for the data since we would not expect to know in practice, a priori, which parameters to optimize for. Of course, these could be learned over time, but given the limited scope of our data set we chose to consider the most restrictive scenario.[7] The one exception is EXP3m where we set η and m optimally as described in [Auer et. al. 2002, Slivkins & Upfal 2008] using a rough estimate of $\mu_{\max}$ and σ^2. We give EXP3m this advantage since we expected it to be the strongest competitor to our ε-smart algorithm and do not wish to create any bias in our favor. As we see below, even with this tuning, EXP3m does not perform well counterfactually. For the remaining algorithms, we simply plug in the parameters that were determined using the simulations in Section 5.2. Since the model we use there was not fit to the data this does not violate the conditions of our counterfactual experiment.

Accuracy

When measuring accuracy, which platform is better changes 27 times in the 84 time steps (see Table 1). The average maximum accuracy (in percentage) is 0.978 with standard deviation 0.034 while the average accuracy for Platform 1 is 0.963 with standard deviation 0.051 and the average accuracy for Platform 2 is 0.940 with standard deviation 0.095. Hence, both platforms are comparable, although Platform 1 has slightly better average performance. The counterfactual results are presented in Table 1 – note that in this case we want to *maximize* accuracy.

For accuracy, the *Bootstrap* and *Random* algorithms have essentially identical performance since, for this data set, the order in which we test out the platforms in the first two time steps determines which one we choose; hence reducing *Bootstrap* to *Random*. Interestingly, these algorithms outperform *EXP3m*, which, while it has provable strong regret guarantees, does not excel in this setting. The best performer is ε-smart, which only explores when necessary.

[7]In general, one can (and would improve performance) by learning and tuning the parameters over time, but this is outside the scope of this current work.

	strong	σ_{strong}	weak 1	weak 2
ε-smart	-0.0144	0.0018	-0.0008	0.0235
ε-greedy	-0.0174	0.0058	-0.0022	0.0204
Bootstrap	-0.03	0.012	-0.0148	0.0078
Random	-0.0303	0.0131	-0.0151	0.0075
EXP3m	-0.0316	0.0023	-0.0159	0.0071

Table 1: Accuracy. Average strong and weak regret against Platforms 1 and 2. We run the counterfactual experiment 300 times for each of the 5 algorithms we consider. The results are sorted from best to worst regret. The units of regret are in the percentage of accuracy.

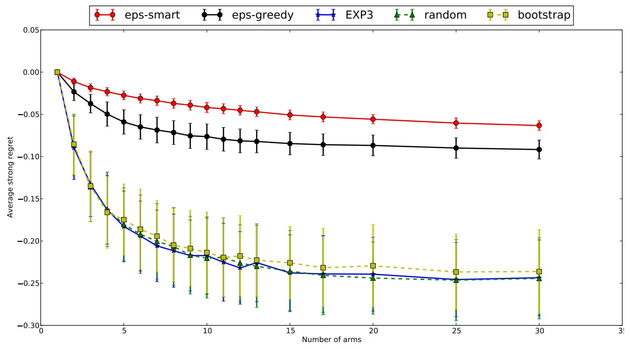
Response Time

When measuring response time, which platform is better changes 26 times in the 84 time steps (see Table 2). The average minimum response time (in seconds) is 6.79 with standard deviation 3.69, while the average response time for Platform 1 is 18.99 with standard deviation 16.03 and the average accuracy for Platform 2 is 10.18 with standard deviation 13.43. The counterfactual results are presented in Table 1 – note that in this case we want to *minimize* the response time.

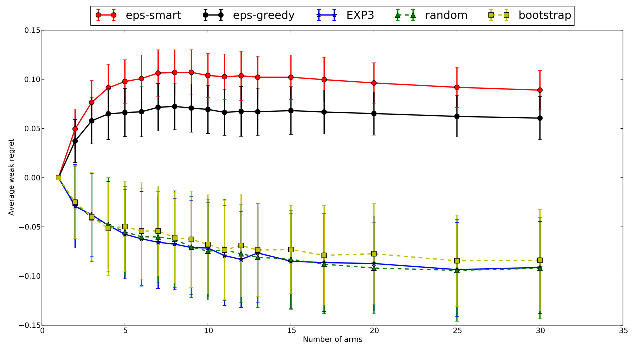
	strong	σ_{strong}	weak 1	weak 2
Random	3.385	4.12	-8.815	0
Bootstrap	3.735	0	-8.465	0.350
ε-smart	4.367	1.285	-7.832	0.983
ε-greedy	7.309	2.293	-4.891	3.924
EXP3	7.682	0.432	-4.993	4.178

Table 2: Response Time. Average strong and weak regret against Platforms 1 and 2. We run the counterfactual experiment 300 times for each of the 5 algorithms we consider. The results are sorted from best to worst regret. The units of regret are in seconds.

For response time, unlike accuracy, the ε-smart algorithm no longer outperforms the rest. In fact, the *Bootstrap* algorithm performs remarkably well. While this may at first sound surprising, the fact that Platform 2 *significantly* outperforms Platform 1 means that, in this case, choosing and sticking to Platform 2 is (in retrospect) a good strategy. Additionally, since the bootstrapping occurred in the first two time steps where Platform 2, conveniently, also outperformed Platform 1, the *Bootstrap* algorithm always chooses Platform 2. Hence its exceptional performance. Note that the variance of our performance is significant since the data itself is also quite noisy, and in fact directly contributes to the decline in the performance of ε-smart. Since we are using a window of size 1, the high variability means that we could be quite wrong about a platform's performance. Hence, we will mistakenly think it is not worth exploring for many time steps.



(a) Strong Regret.



(b) Weak Regret.

Figure 5: Strong and weak regret vs the number of arms in the random walk model $\mathcal{X}_i^t = Gaussian(\mu_i^t, .05)$ where μ_i^t is the bounded lazy random walk on $[.5, 1]$. We vary the number of arms from 1 to 30 with 300 trials. As expected, performance deteriorates with the number of arms, particularly for strong regret (where the optimal omniscient algorithm approaches accuracy 1 as the number of arms increases). In both cases, ε-smart algorithm outperforms the other algorithms, and is the only one that attains positive weak regret.

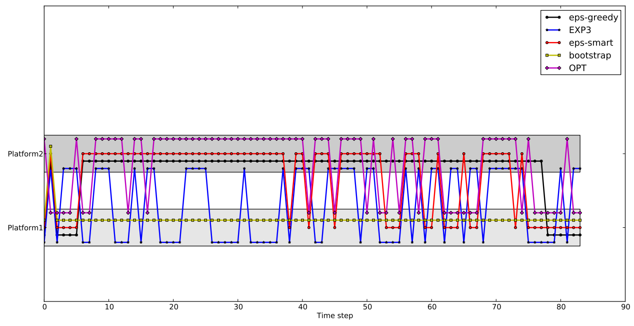


Figure 6: Depicts the switching behavior of various algorithms on the return-time data set. The optimum algorithm (OPT) knows the best platform *a priori* and always chooses that platform.

Switching platforms.

Figure 6 gives a simple example of how, on a given run, each of the algorithms switch from one platform to another over time[8]. For conciseness, we only show the graph for accuracy values. The figures also show the hypothetical "optimum" algorithm as defined for strong regret.

5.2 Random-Walk Model Simulation

We now consider the lazy random walk model described in Section 3, and consider two different experiments, one varying the number of arms and the other varying σ. The number of arms is very relevant as not only are there many crowdsourcing platforms available, but each platform, depending on the task description (which specifies the payment amount, time of day, worker specifications, etc), can be used in many different ways. Hence, it is crucial that whichever algorithm we use be scalable against the number of arms so we are able to optimize over a wide class of task descriptions. Additionally, since we do not know much about platform behavior, it is important that whichever algorithm we choose to perform well even in the presence of noise.

Note that since our data is limited, we will not attempt to fit the random walk model to the data. Instead, we consider a simple special case and use it to gain insight into the effectiveness of bandit algorithms. Specifically, we consider the family $\mathcal{A}$ of Gaussian distributions where $M = [.5, 1]$. Our initial points μ_i^0 are chosen uniformly at random from the set $\{.5, .55, \ldots, 1\}$, and the arms evolve with parameters $p = .5$ and $\Delta = .05$. We show the average strong regret and the minimum average weak regret after running each algorithm for 1000 time steps, averaged over 300 runs in Figures 5 and **??**. In this case, we first tune the parameters for $\sigma = .05$ and $k = 4$ using simulations. This gave the following parameters which were used for our results.

Random: Uniform distribution over platforms.
Bootstrap: $b = 1$, uniform ordering.
ε-greedy: $\varepsilon = .03$, $\ell = 1$.
EXP3m: $\eta = .1$, $m = 10$.
ε-smart: $\varepsilon = .1$, $\gamma = 1$, $\ell = 1$.

Regret vs. # of Arms

We vary the number of arms (or task descriptions) from 0 to 40. As the number of arms increase, our task becomes increasingly difficult since the performance of the optimal algorithm can only improve; i.e., our comparator for regret measurement only becomes stronger. Using the argument from Section 4 on the parameters chosen for this random walk model, we expect average strong regret for both *Random* and *Bootstrap* to converge to $-.25$, and this is indeed observed. The two perform essentially identically, as would be expected since $t = 1000$ gives sufficient time for the walks to mix. More surprisingly, $EXP3m$ does not outperform *Random* and *Bootstrap*, even given the optimization of its parameters. On the other hand, ε-greedy performs better than these other three algorithms, especially for low k where it even hovers close to 0 weak regret. However, its performance becomes indistinguishable from the other three as k increases. The best performer, and the only one that is con-

[8]The Random algorithm is not shown as it is trivial; namely, choose a platform at random and stick to it

sistently better in both strong and weak regret for the full range of k observed, is ε-smart. Additionally, it is the only algorithm to attain positive weak regret, and maintains it for all observed k. This is a strong indicator that, in practice, the ε-smart algorithm will scale well.

Regret vs. Variance

We vary the parameter σ in the random walk model from 0 (which corresponds to the deterministic random walk model) to .4 (where we primarily observe noise). As we increase the variance, our task becomes increasingly difficult since we cannot learn much from any observation. However, the task becomes just as difficult for the (hypothetical) optimal algorithms since, even though their *expected* average at a given time step may be higher, the distributions have so much overlap for high σ that in the *observations* the distinction is no longer detectable. In this case, in expectation, both strong and weak regret converges to 0 for all algorithms. This is, part of what we observe in our counterfactual experiments for response time; the variance in both platforms is very high, and we cannot cope effectively. However, for low variance we again see ε-smart outperforming the other platforms and attaining positive weak regret. Recall that this implies ε-smart outperforms the best (hypothetical) algorithm that chooses a single arm, and hence is extremely relevant in practice. This suggests that, even, or perhaps especially, for large numbers of arms we can and should exploit the dynamic nature of the crowd to obtain better performance.

6. CONCLUSION

We take the first step towards studying temporal variations in crowdsourcing platforms and propose a multi-armed bandit approach for platform selection in order to cope with this changing behavior. We compare a number of existing and new algorithms; specifically, we introduce $\epsilon-$smart, that performs well in both real and simulated scenarios. While the results presented in this paper are based on a relatively small dataset, it motivates us to further explore the efficacy of bandit approaches in crowd platform selection, particularly over longer time scales. In fact, we expect our algorithm to be more useful in the long-term, where we expect to observe even more stochastic drift. Our ongoing efforts are focused on collecting real-world data, both in terms of additional platforms as well as longer periods, to verify this drift experimentally and test and develop bandit algorithms further for these scenarios.

Acknowledgments

The authors wish to thank Saraschandra Karanam of Xerox Research Centre India for his insights drawn from platform data.

References

[**1**] P. Auer, N. Cesa-Bianchi, Y. Freund and R. E. Schapire, "The non-stochastic multi- armed bandit problem.", in SIAM JoC, 32:48-77, 2002.

[**2**] C. Balamurugan, S. Roy and S. Gujar, "Sustainable Employment in India by Crowdsourcing Enterprise Tasks", ACM Dev, 2013.

[**3**] D. A. Berry and B. Fristedt, "Bandit problems: sequential allocation of experiments", Chapman and Hall, 1985.

[**4**] N. Cesa-Bianchi and P. Fischer, "Finite-time regret bounds for the multiarmed bandit problem", ICML, pages 100-108, 1998.

[**5**] K. Dasgupta, V. Rajan, S. Karanam, C. Balamurugan and N. Piratla, "CrowdUtility: Know The Crowd That Works For You", CHI, 2013.

[**6**] J. C. Gittins, "Bandit Processes and Dynamic Allocation Indices", Journal of the Royal Statistical Society. Series B, Vol. 41, No. 2, 1979.

[**7**] S. Guha and K. Munagala, "Approximation algorithms for partial-information based stochastic control with Markovian rewards", FOCS, 2007.

[**8**] C. J. Ho and J. W. Vaughan, "Online Task Assignment in Crowdsourcing Markets", AAAI, 2012.

[**9**] P. G. Ipeirotis, "Analyzing the Amazon Mechanical Turk marketplace", ACM XRDS 17(2):16-21, 2010.

[**10**] S. Karanam and K. Dasgupta. "Do Crowdsourcing platforms differ in their performance characteristics?", XIG White Paper, 2012.

[**11**] V. Kuleshov and D. Precup, "Algorithms for the multi-armed bandit problem", JML, 1-48, 2000.

[**12**] A. Mahajan and D. Teneketzis, "Multi-Armed Bandit Problems", in Foundations and Applications of Sensor Managment, Ed. A. Hero, D. Castaron, D. Cochran and K. Kastella. Chapter 6. 2007.

[**13**] B. McMahan and M. Streeter, "Tighter Bounds for Multi-Armed Bandits with Expert Advice", COLT, 2009.

[**14**] C. H. Papadimitriou and J. N. Tsitsiklis, "The complexity of optimal queueing network control", Structure in Complexity Theory, pages 318-322, 1994.

[**15**] A. Quinn and B. Bederson, "Human computation: a survey and taxonomy of a growing field", CHI, 2011.

[**16**] H. Robbins, "Some Aspects of the Sequential Design of Experiments" Bulletin of the AMS, vol 58, 1952.

[**17**] A. Slivkins and E. Upfal, "Adapting to a Stochastically Changing Environment", COLT, 2008.

[**18**] L. Tran-Thanh, S. Stein, A. Rogers and N. R. Jennings, "Efficient Crowdsourcing of Unknown Experts using Multi-Armed Bandits", ECAI, 2012.

[**19**] P. Whittle. "Restless bandits: Activity allocation in a changing world", J. of Appl. Prob., 25A:287-298, 1988.