Top Arm Identification in Multi-Armed Bandits with Batch Arm Pulls

Kwang-Sung Jun UW-Madison
Kevin Jamieson UC Berkeley
Robert Nowak UW-Madison
Xiaojin Zhu UW-Madison

Abstract

We introduce a new multi-armed bandit (MAB) problem in which arms must be sampled in batches, rather than one at a time. This is motivated by applications in social media monitoring and biological experimentation where such batch constraints naturally arise. This paper develops and analyzes algorithms for batch MABs and top arm identification, for both fixed confidence and fixed budget settings. Our main theoretical results show that the batch constraint does not significantly affect the sample complexity of top arm identification compared to unconstrained MAB algorithms. Alternatively, if one views a batch as the fundamental sampling unit, then the results can be interpreted as showing that the sample complexity of batch MABs can be significantly less than traditional MABs. We demonstrate the new batch MAB algorithms with simulations and in two interesting real-world applications: (i) microwell array experiments for identifying genes that are important in virus replication and (ii) finding the most active users in Twitter on a specific topic.

1 Introduction

We consider the top-k pure-exploration problem for stochastic multi-armed bandits (MAB). Formally, we are given n arms that produce a stochastic reward when pulled. The reward of arm i is an i.i.d. sample from a distribution ν_i whose support is in $[0, 1]$. The bounded-support assumption can be generalized to the σ-sub-Gaussian assumption. Denote by $\mu_i = \mathbb{E}_{X\sim\nu_i} X$ the expected reward of the arm i. We assume a unique top-k set; i.e., $\mu_1 \geq \mu_2 \geq ... \geq \mu_k > \mu_{k+1} \geq ... \geq \mu_n$.

In this work, we introduce a new setting in which arms must be pulled in batches of size b. We also consider the additional constraint that any one arm can be pulled at most $r \leq b$ times in a batch (if $r = b$ then there is no constraint). Said another way, the action space is defined as $\mathcal{A} := \{\mathbf{a} \in \{0, 1, \ldots, r\}^n \,|\, \sum_{i=1}^n a_i \leq b\}$, where $\mathbf{a} \in \mathcal{A}$ indicates the number of arm pulls for each arm. We call this the (b, r)-*batch* MAB setting. Note that this encompasses the standard MAB setting, which corresponds to $b = 1, r = 1$. The general (b, r)-batch setting is fundamentally different, since sampling decisions must be made one batch at a time, rather than one sample at a time. This loss in flexibility could hinder performance, but our main theoretical results show that in most cases there is no significant increase in sample complexity.

There are many real-world applications where the batch constraint arises. For example, suppose the arms represent Twitter users and the goal is to find users who tweet the most about a topic of interest. Using Twitter's free API, one can follow up to $b = 5000$ users at a time, which amounts to a batch of that size. Also, a user is either observed or not, so $r = 1$ in this application. The batch is the fundamental sampling unit in such applications, and thus this paper is interested in methods that aim to reduce the *batch complexity*: the total number of *batches*, or rounds, to correctly identify the top-k arms. As another example, we consider the problem of using microwell array experiments to identify the top-k genes involved in virus replication processes. Here the arms are genes (specifically single knockdown cell strains) and a batch consists of $b = 384$ microwells per array. In this case, we may repeat the same gene (cell strain) in every microwell, so $r = 384$.

Returning to the discussion of theory and algorithms, we consider two settings: fixed confidence and fixed budget. In the fixed confidence setting, given a target failure rate δ the goal is to identify the top-k arms with probability at least $1-\delta$ using the fewest possible number of batches. In the fixed budget setting, given a budget number of batches B, the goal is to identify the top-k arms with as high probability as possible. For the fixed confidence (budget) setting, we propose

Appearing in Proceedings of the 19^{th} International Conference on Artificial Intelligence and Statistics (AISTATS) 2016, Cadiz, Spain. JMLR: W&CP volume 51.

the BatchRacing (BatchSAR) algorithm and prove its theoretical guarantee in Section 3 (Section 4). Our analysis shows that batch MABs have almost the same sample complexity as unconstrained MABs as long as r is not too small relative to b.

An alternative to our batch MAB algorithms is to use standard MAB algorithms as follows. One could use each batch to obtain a sample from a particular arm of interest, and effectively ignore the rest of the data from each batch. This would allow one to use standard MAB algorithms. For example, in the Twitter application we could follow just one user at a time while we can follow 5000 for the same cost (free). Our results show that in most cases this (obviously naive) reduction to the standard MAB would have a batch complexity b times greater than that of our batch MAB algorithms.

In section 5, we validate the analysis of the proposed algorithms with a toy experiment, and evaluate the algorithms on real-world data from the two applications introduced above.

2 Related Work

The pure exploration problem in MAB has a long history dating back to the '50s with the work of [4] and [20]. The top-k exploration, or subset selection, in stochastic MAB has received much attention recently. In the fixed confidence setting including the Probably Approximately Correct (PAC) setup, the Racing algorithm for the top-1 exploration problem was proposed by Maron and Moore [19] and by Even-Dar et al. [6] independently. The Racing algorithm performs uniform sampling on surviving arms while deactivating infeasible arms based on confidence intervals. Racing was generalized to top-k identification by Heidrich-Meisner and Igel [9], and its sample complexity was analyzed in [17]. Median Elimination (ME) [6, 14] runs in stages where each stage eliminates the bottom half of the surviving arms. Under the PAC setup, the sample complexity of ME matches the lower bound. Going beyond elimination-based algorithms, LUCB proposed by Kalyanakrishnan et al. [15] adaptively chooses which arm to pull based on confidence intervals without explicitly removing arms. Kaufmann et al. improved both Racing and LUCB with confidence intervals based on Chernoff information, though restricted to exponentially distributed rewards [17].

In the fixed budget setting, Successive Accepts and Rejects (SAR) proposed by Bubeck et al. [5] runs in $n-1$ phases. Each phase performs uniforms sampling for a predetermined number of rounds. At the end of each phase, SAR either accepts the empirically best arm or rejects the worst arm. UCB-E by Audibert et al. [3] and LUCB-E by Kaufmann et al. [17] run in a more adaptive way with confidence intervals without eliminations, although the problem hardness parameter must be given to the algorithms, which is unrealistic.

Our proposed (b, r)-batch setting overlaps with various existing settings, but is, to the best of our knowledge, never subsumed by any. One popular setting, though studied under the regret setting, is semi-bandit feedback [2] where an action $\mathbf{a} \in \mathcal{A} \subseteq \{0, 1\}^n$ indicates coordinate-wise whether or not ($a_j = 1$ or 0) one pulls an arm. (b,r=1)-batch is a special case of semi-bandit feedback where the action space $\mathcal{A}$ is defined to be all b-sized subsets of arms. This is called multiple plays in [1]. A variant of the multiple plays was studied in [18] where the player pulls b times in a round, but is allowed to pull an arm more than once, up to b times. However, the authors assume that pulling the same arm more than once in a round produces the same reward, whereas in our (b,r=b)-batch setting repeated pulls produce independent rewards.

In the delayed feedback setting [23, 13], the reward at time t is revealed after τ_t time steps. One can relate this setting to our (b, r)-batch setting by assuming that rewards are revealed in blocks after every b rounds: $\tau_t = b - 1 - (t - 1 \mod b)$. If b is known to the player, this construction is exactly the $(b, r = b)$-batch setting. Nevertheless, delayed feedback has only been considered in the regret setting to the best of our knowledge.

Recently, Perchet et al. have proposed a generalized notion of batch arm pulls [22] where each batch can be of different sizes. However, the authors only consider two-armed bandits ($n = 2$) and assume no limitation on the repeated arm pulls ($b = r$). More importantly, the authors only consider the regret setting. Wu et al. [24] proposed a pure exploration MAB framework for combinatorial action space of which (b,r=1)-batch is an instance. However, repeated arm pulls ($r > 1$) are not allowed in their framework.

3 The Fixed Confidence Setting

In the fixed confidence setting, given a target failure rate δ, one tries to find the top-k arms with probability at least $1 - \delta$ in the fewest number of batches as possible. An algorithm must output the correct top-k arms w.p. $\geq 1 - \delta$, and its performance is measured in the batch complexity.

Denote by $\max^{(k)} F$ the k-th largest member of a set F. Let $X_{i,j}$ be the j-th reward of arm i. Define the empirical mean of arm i up to τ samples as $\widehat{\mu}_{i,\tau} = \frac{1}{\tau}\sum_{j=1}^{\tau} X_{i,j}$. The key success of an algorithm in the fixed confidence setting often relies on the confidence bound on the true mean μ_i; the tighter the bounds are,

the less arm pulls we spend. We adopt and simplify a confidence bound proposed in [11] that resembles the law of the iterated logarithm.

Lemma 1. *(non-asymptotic law of the iterated logarithm) [11][1] Let $X_1, X_2, \ldots$ be i.i.d. zero-mean sub-Gaussian random variables with scale $\sigma > 0$; i.e., $\mathbb{E}e^{\lambda X_i} \le e^{\frac{\lambda^2\sigma^2}{2}}$. Let $\omega \in (0, \sqrt{1/6})$. Then,*

$$\mathbb{P}\left(\forall \tau \ge 1, \left|\sum_{s=1}^{\tau} X_s\right| \le 4\sigma\sqrt{\tau \log(\log_2(2\tau)/\omega)}\right) \ge 1 - 6\omega^2. \quad (1)$$

The proof of Lemma 1 is in the supplementary material. Note that a bounded random variable $X \in [a, b]$ is a sub-Gaussian with scale $\sigma = (b-a)/2$. In our case, $\sigma = \frac{1}{2}$. Define a deviation function

$$D(\tau, \omega) := \sqrt{\frac{4\log(\log_2(2\tau)/\omega)}{\tau}}.$$

Let $T_i(t)$ be the number of arm pulls of arm i at round t. We define the lower confidence bound (LCB) $L_i(t,\delta)$ and the upper confidence bound (UCB) $U_i(t,\delta)$ of arm i at round t as

$$\begin{aligned} L_i(t,\delta) &:= \widehat{\mu}_{i,T_i(t)} - D\left(T_i(t), \sqrt{\delta/(6n)}\right) \text{ and} \\ U_i(t,\delta) &:= \widehat{\mu}_{i,T_i(t)} + D\left(T_i(t), \sqrt{\delta/(6n)}\right). \end{aligned} \quad (2)$$

Now that the environment allows the (b,r)-batch arm pull, can we propose an algorithm whose batch complexity achieves b factor reduction from the sample complexity of the state-of-the-art algorithms? Inspired by the Racing algorithm [19, 9], we propose BatchRacing algorithm for the (b,r)-batch setting; see Algorithm 2. The algorithm maintains a set of surviving arms that is initialized as $S_1 = [n]$ before the first round. At round t, the algorithm calls RoundRobin (Algorithm 1) to choose b arm pulls that keeps the pull count of each arm in the surviving set S_t as uniform as possible. Then, the algorithm checks if there is any arm that is confidently top-k or confidently not top-k using the LCB and UCB as follows. Let A_t (R_t) be the set of accepted (rejected) arms at round t and $A_1 = R_1 = \emptyset$. Let $k_t = k - |A_t|$, the remaining number of top arms to identify. Any arm i whose LCB is greater than the UCB of $|S_t| - k_t$ arms is moved to the accept set: $A_{t+1} \leftarrow A_t \cup \{i\}$. Symmetrically, any arm i whose UCB is smaller than the LCB of k_t arms is moved to the reject set $R_{t+1} \leftarrow R_t \cup \{i\}$. Those accepted or rejected arms are removed from the surviving set S_t. The process is repeated until it accepts k arms ($|A_t| = k$), and finally it outputs A_t.

We define the gap Δ_i of arm i and denote by $\sigma(j)$ the arm with j-th smallest gap as follows:

[1] A tighter version can be found in [11].

Algorithm 1 RoundRobin$(A, \{T_i\}_{i\in A}, b, r)$

Input: A: a set of arms, $\{T_i\}_{i\in A}$: arm pull counts, b: batch size, r: repeated pull limit
Output: **a**: pull count vector
$\mathbf{a} \leftarrow \mathbf{0} \in \mathbb{R}^n$
for $\tau = 1 \ldots \min\{b, |A|r\}$ **do**
  $j \leftarrow \arg\min_{\ell \in A: a_\ell \le r} T_\ell + a_\ell$ (break ties arbitrarily)
  $a_j \leftarrow a_j + 1$
end for
Output **a**.

Algorithm 2 BatchRacing

Input: n arms, $\delta \in (0,1)$, k: number of top arms, b: batch size, r: repeated pull limit
Output: k arms.
$t \leftarrow 1$, $S_1 \leftarrow [n]$, $R_1 \leftarrow \emptyset$, $A_1 \leftarrow \emptyset$, $T_i(0) \leftarrow 0, \forall i$
while $S_t \neq \emptyset$ **do**
  $\mathbf{a} \leftarrow$ RoundRobin$(S_t, \{T_i(t-1)\}_{i\in S_t}), b, r)$
  Pull by **a** (pull arm i a_i times, $\forall i$).
  $T_i(t) \leftarrow T_i(t-1) + a_i, \forall i$
  $k_t \leftarrow k - |A_t|$
  $A_{t+1} \leftarrow A_t \bigcup \{i \in S_t \mid L_i(t,\delta) > \max_{j\in S_t}^{(k_t+1)} U_j(t,\delta)\}$
  $R_{t+1} \leftarrow R_t \bigcup \{i \in S_t \mid U_i(t,\delta) < \max_{j\in S_t}^{(k_t)} L_j(t,\delta)\}$
  $S_{t+1} \leftarrow S_t \setminus (R_{t+1} \bigcup A_{t+1})$
  $t \leftarrow t + 1$
end while
Output A_t.

$$\Delta_i := \begin{cases} \mu_i - \mu_{k+1} & \text{if } i \le k \\ \mu_k - \mu_i & \text{if } i > k \end{cases} \quad (3)$$
$$\Delta_{\sigma(1)} = \Delta_{\sigma(2)} \le \Delta_{\sigma(3)} \le \ldots \le \Delta_{\sigma(n)},$$

where $\Delta_{\sigma(1)} = \Delta_{\sigma(2)}$ by definition. Let $\mathcal{E}_i(\delta) = \{\forall t \ge 1, L_i(t,\delta) \le \mu_i \le U_i(t,\delta)\}$ be the event that the LCB and UCB of arm i defined in (2) traps the true mean μ_i for all $t \ge 1$. Let $\omega = \sqrt{\delta/(6n)}$. Define $\overline{T}_i := 1 + \left\lfloor 64\Delta_i^{-2}\log((2/\omega)\log_2(192\Delta_i^{-2}/\omega))\right\rfloor$. Lemma 2 shows how many arm pulls are sufficient to classify an arm to either reject set R_t or accept set A_t.

Lemma 2. *Assume $\cap_{i=1}^n \mathcal{E}_i(\delta)$. In Algorithm 2, let $T'(t) = \min_{i\in S_t} T_i(t)$ and $k_t = k - |A_t|$. Then,*

$$\forall t, \forall i > k, \left(T'(t) \ge \overline{T}_i \implies U_i(t,\delta) < \max_{j\in S_t}^{(k_t)} L_j(t,\delta)\right) \quad (4)$$

$$\forall t, \forall i \le k, \left(T'(t) \ge \overline{T}_i \implies L_i(t,\delta) > \max_{j\in S_t}^{(k_t+1)} U_j(t,\delta)\right). \quad (5)$$

Proof. We use $L_i(t)$ and $U_i(t)$ as shorthands for $L_i(t,\delta)$ and $U_i(t,\delta)$ respectively. It suffices to show for the case where A_t and R_t are empty since otherwise the problem is equivalent to removing rejected or accepted arms from consideration and starting a new problem with $n \leftarrow (n - |A_t| - |R_t|)$ and $k \leftarrow (k - |A_t|)$ while maintaining the samples collected so far.

To prove (4), let $i > k$ and $\widehat{k}$ be the arm with k-th largest empirical mean at t-th round. We prove the contrapositive. Assume the RHS is false: $U_i(t) \geq \max^{(k)}_{j \in S_t} L_j(t)$. Note that using $D(T_i(t), \omega) \leq D(T'(t), \omega))$,

$$U_i(t) \leq \widehat{\mu}_{i,T_i(t)} + D(T'(t), \omega) \leq \mu_i + 2D(T'(t), \omega) \text{ and}$$
$$U_i(t) \geq \max^{(k)}_{j \in S_t} L_j(t) = L_{\widehat{k}}(t) \geq \widehat{\mu}_{\widehat{k},T_{\widehat{k}}(t)} - D(T'(t), \omega),$$

which implies $\mu_i + 2D(T'(t), \omega) \geq \widehat{\mu}_{\widehat{k},T_{\widehat{k}}(t)} - D(T'(t), \omega)$. Using $\widehat{\mu}_{\widehat{k},T_{\widehat{k}}(t)} \geq \mu_k - D(T'(t), \omega)$ that is due to Lemma 3 (see the supplementary material),

$$\begin{aligned} \mu_i + 2D(T'(t), \omega) &\geq \mu_k - 2D(T'(t), \omega) \\ \Delta_i &\leq 4D(T'(t), \omega) \\ &= 4\sqrt{(4/T'(t)) \log(\log_2(2T'(t))/\omega)} \\ T'(t) &\leq 64\Delta_i^{-2} \log(\log_2(2T'(t))/\omega). \end{aligned}$$

Invert this using

$$\tau \leq c \log\left(\frac{\log_2 2\tau}{\omega}\right) \implies \tau \leq c \log\left(\frac{2}{\omega} \log_2\left(\frac{3c}{\omega}\right)\right) \tag{6}$$

with $c = 64\Delta_i^{-2}$ to have $T'(t) \leq 64\Delta_i^{-2} \log((2/\omega) \log_2(192\Delta_i^{-2}/\omega))$. Then, $T'(t) < 1 + \lfloor 64\Delta_i^{-2} \log((2/\omega) \log_2(192\Delta_i^{-2}/\omega)) \rfloor = \overline{T}_i$. This completes the proof of (4). By symmetry, one can show (5) as well. □

Theorem 1 states the batch complexity of the BatchRacing algorithm. Hereafter, all proofs can be found in the supplementary material. Note that even if $r > \lfloor b/2 \rfloor$, BatchRacing pulls the same arm at most $\lfloor b/2 \rfloor$ times in a batch. Thus, our analysis hinges on the effective repeated pull limit $r' = \min\{r, \lfloor b/2 \rfloor\}$ instead of r.

Theorem 1. *If $b \geq 2$, with probability at least $1 - \delta$, Algorithm 2 outputs the top-k arms $\{1, \ldots, k\}$ after at most*

$$\begin{aligned} &\frac{1}{r'}\overline{T}_{\sigma(1)} + \frac{1}{b}\left(\sum_{i=\lfloor b/r' \rfloor + 1}^{n} \overline{T}_{\sigma(i)}\right) + \log n + \frac{n}{b} + \frac{1}{r'} + 2 \\ &= O\left(\frac{1}{r'}\Delta_{\sigma(1)}^{-2} \log\left(\frac{n \log(\Delta_{\sigma(1)}^{-2})}{\delta}\right) + \right. \\ &\quad \left. \frac{1}{b}\left(\sum_{i=\lfloor b/r' \rfloor + 1}^{n} \Delta_{\sigma(i)}^{-2} \log\left(\frac{n \log(\Delta_{\sigma(i)}^{-2})}{\delta}\right)\right) + \log n\right) \end{aligned} \tag{7}$$

batches. In the case of $b = r = 1$, the algorithm does so after at most $\sum_{i=1}^{n} \overline{T}_{\sigma(i)} = O\left(\sum_{i=1}^{n} \Delta_{\sigma(i)}^{-2} \log\left(\frac{n \log(\Delta_{\sigma(i)}^{-2})}{\delta}\right)\right)$ batches.

In the case of b=r=1, BatchRacing is exactly the Racing algorithm, and the batch complexity is equivalent to the sample complexity. The sample complexity of Racing stated in Theorem 1 is the best known insofar as the exact top-k identification problem, although it does not match the best known lower bound $\Omega(\sum_{i=1}^{n} \Delta_i^{-2})$ [17]. If $r \geq \lfloor b/2 \rfloor$, one can verify that the batch complexity (7) reduces to $\frac{1}{b}$ fraction of the sample complexity of Racing except the additive $\log n$ term. The $\log n$ term is the price to pay for performing batch arm pulls, which limits adaptivity. Note that the batch complexity is at least n/b since each arm must be pulled at least once. Thus, unless b is large enough to satisfy $n/b \ll \log(n)$, the additive $\log(n)$ is negligible. For simplicity, we ignore the additive $\log n$ from the discussion. If $r < \lfloor b/2 \rfloor$, r plays an interesting role. The r factor reduction is applied to the largest term that involves $\Delta_{\sigma(1)}^{-2}$. The terms involving arm $\sigma(2), \ldots, \sigma(\lfloor b/r \rfloor)$ disappear. The rest of the terms enjoy b factor reduction.

The contribution of r depends on the gaps $\{\Delta_i\}$. If the smallest gap $\Delta_{\sigma(1)}$ is much smaller than the other gaps, the term $(1/r')\overline{T}_{\sigma(1)}$ becomes dominant, thus making r important. On the other hand, if the gaps are all equal, one can show that the overall b fold reduction is achieved regardless of r using the fact $(1/r')\Delta_{\sigma(1)}^{-2} \log(n \log(\Delta_{\sigma(1)}^{-2})/\delta) \approx (1/b) \sum_{i=1}^{\lfloor b/r' \rfloor} \Delta_{\sigma(i)}^{-2} \log(n \log(\Delta_{\sigma(i)}^{-2})/\delta)$. We empirically verify this case with toy experiments in Section 5.

4 The Fixed Budget Setting

We now consider the problem of identifying the top-k arms with as high a probability as possible within a given number of batches B under the (b, r)-batch setting. An algorithm must terminate after spending no more than B batches and output k arms that are believed to be the top-k. The guarantee is typically made on the misidentification probability $\mathbb{P}(A^* \neq \{1, \ldots, k\})$ where A^* is the output of the algorithm. However, it is often convenient to look at its batch complexity that can be derived from the misidentification probability. Let $H_2 = \max_{i \in [n]} i\Delta_{\sigma(i)}^{-2}$. Bubeck et al. showed that the Successive Accepts and Rejects (SAR) algorithm for the top-k identification problem under the fixed budget setting has the sample complexity of $O(H_2 \log^2 n)$ [5]. One might try running SAR by supplying SAR with bB sample budget. However, such a modification breaks its theoretical guarantee. We propose BatchSAR that extends SAR to allow (b, r)-batch arm pull, and prove its performance guarantee.

The intuition behind BatchSAR is as follows. Imagine the $(b = n, r = 1)$-batch setting where one has

no choice but to pull every arm once at every round. There is no room for adaptivity; eliminating an arm does not yield more arm pulls in return. Let $\tilde{r} = \min\{r, \lceil b/2 \rceil\}$ and $\tilde{n} = \lceil b/\tilde{r} \rceil$. Note $\tilde{n} \geq 2$ by definition. Similarly in the general (b, r)-batch setting, once the surviving number of arms becomes $\tilde{n}$ or less, eliminating an arm does not help. For example, if $r < \lceil b/2 \rceil$ and there are $\tilde{n}-1$ surviving arms, the maximum number of arm pulls one can make in a batch is $r(\tilde{n}-1) < b$, thus wasting $b - r(\tilde{n}-1)$ arm pulls.

Algorithm 3 BatchSAR

Input: n arms, k: the target number of top arms, b: batch size, r: repeated pull limit, B: batch budget
Output: k arms.
Let $\tilde{n} = \max\{\lceil b/r \rceil, 2\}$ and $c_1 = b + \tilde{n}\min\{r, \lceil b/2 \rceil\} + n$
Define $m_s = \begin{cases} \left\lceil \frac{bB - \left(\sum_{i=\tilde{n}+1}^{n} \lceil b/i \rceil\right) - c_1}{\tilde{n}/2 + \sum_{i=\tilde{n}+1}^{n}(1/i)} \frac{1}{n-s+1} \right\rceil & \text{for } s \leq n - \tilde{n} \\ \left\lceil \frac{bB - \left(\sum_{i=\tilde{n}+1}^{n} \lceil b/i \rceil\right) - c_1}{\tilde{n}/2 + \sum_{i=\tilde{n}+1}^{n}(1/i)} \frac{1}{2} \right\rceil & \text{for } s = n - \tilde{n} + 1 \end{cases}$
$t \leftarrow 1$, $S_1 \leftarrow [n]$, $A_1 \leftarrow \emptyset$, $T_i(0) \leftarrow 0, \forall i \in [n]$
for $s = 1, \ldots, (n - \tilde{n} + 1)$ **do**
  while $\min_{i \in S_s} T_i(t-1) < m_s$ **do**
    $\mathbf{a} \leftarrow \text{RoundRobin}(S_s, \{T_i(t-1)\}_{i \in S_s}), b, r)$
    Pull by $\mathbf{a}$ (pull arm i a_i times, $\forall i$).
    $T_i(t) \leftarrow T_i(t-1) + a_i, \forall i$
    $t \leftarrow t + 1$
  end while
  Let $t_s = t-1$, the last round in stage s, and $k' = k - |A_s|$, the remaining number of top arms to identify.
  if $s \leq n - \tilde{n}$ **then**
    Let $\widehat{\mu}_i = \widehat{\mu}_{i, T_i(t_s)}$ and $\rho(i)$ be the arm with i-th largest empirical mean in S_s so that $\widehat{\mu}_{\rho(1)} \geq \ldots \geq \widehat{\mu}_{\rho(n-s+1)}$. Let $\widehat{\Delta}_{\rho(1)} = \widehat{\mu}_{\rho(1)} - \widehat{\mu}_{\rho(k'+1)}$ and $\widehat{\Delta}_{\rho(n-s+1)} = \widehat{\mu}_{\rho(k')} - \widehat{\mu}_{\rho(n-s+1)}$.
    $j_s \leftarrow \arg\max_{i \in \{\rho(1), \rho(n-s+1)\}} \widehat{\Delta}_i$ (break ties arbitrarily).
    Remove arm j_s: $S_{s+1} \leftarrow S_s \setminus \{j_s\}$.
    if $j_s = \rho(1)$ **then**
      $A_{s+1} \leftarrow A_s \cup \{j_s\}$
    else
      $A_{s+1} \leftarrow A_s$
    end if
    (Early exit case 1) **If** $|S_{s+1}| = k - |A_{s+1}|$, **then** accept all the remaining surviving arms $A^* \leftarrow A_{s+1} \cup S_{s+1}$ and exit the for loop.
    (Early exit case 2) **If** $|A_{s+1}| = k$, **then** set $A^* \leftarrow A_{s+1}$ and exit the for loop.
  else
    $A^* \leftarrow A_s \cup \left(\arg k' \max_{i \in S_s} \widehat{\mu}_{i, T_i(t_s)}\right)$ (break ties arbitrarily)
  end if
end for
Output A^*.

Therefore, we design BatchSAR to have two phases: (i) the first $n-\tilde{n}$ stages where an arm is eliminated after each stage just as SAR and (ii) the last stage where uniform sampling is performed without any elimination. BatchSAR is described in Algorithm 3. The algorithm starts with the surviving arm set $S_1 = [n]$ and performs $(n-\tilde{n}+1)$ stages. In each stage s, surviving arms are pulled uniformly by calling RoundRobin (Algorithm 1) until every arm's pull count is at least m_s defined in the algorithm. Then, we choose an arm j_s that is the safest to remove, meaning that the empirical gap $\widehat{\Delta}_i$ defined in the algorithm is the largest. Note j_s is either the empirically best arm or the worst. The arm j_s is then removed from the surviving set and is accepted if j_s is the empirically best arm. We repeat the same until the final stage $s = (n - \tilde{n} + 1)$ where we choose the empirical top-k' arms and add them to the final accept set.

We claim that the algorithm spends no more than B batches. Let t_s be the last round in stage s. It is easy to see that $\min_{i \in S_s} T_i(t_s) \leq m_s + \left\lceil \frac{b}{n-s+1} \right\rceil - 1$ due to the batch effect, and a surviving arm's pull count is either $\min_{i \in S_s} T_i(t_s)$ or $\min_{i \in S_s} T_i(t_s) + 1$. Then,

$$\forall i \in S_s, T_i(t_s) \leq m_s + \left\lceil \frac{b}{n-s+1} \right\rceil. \tag{8}$$

We prove the claim by the fact that $t_{n-\tilde{n}+1} \leq \left\lceil \frac{1}{b} \left(\sum_{i \in S_{n-\tilde{n}+1}} T_i(t_{n-\tilde{n}+1}) + \sum_{s=1}^{n-\tilde{n}} T_{j_s}(t_s) \right) \right\rceil$, which we bound by B using (8) and $\lceil b/\tilde{n} \rceil \leq \tilde{r}$.

Theorem 2 states the misidentification probability of BatchSAR. Define

$$H_3^{<b,r>} = \frac{1}{b} \left(\frac{\tilde{n}}{2} + \sum_{i=\tilde{n}+1}^{n} \frac{1}{i} \right) \max_{i=2, \tilde{n}+1, \tilde{n}+2, \ldots, n} i \Delta_{\sigma(i)}^{-2}. \tag{9}$$

Theorem 2. *Let A^* be the output of BatchSAR. Given a batch budget B, the misidentification probability of BatchSAR under the (b, r)-batch setting satisfies*

$$\mathbb{P}(A^* \neq \{1, \ldots, k\}) \leq 2n^2 \exp\left(-\frac{1}{8} \frac{B - \left(\sum_{i=\tilde{n}+1}^{n} \lceil b/i \rceil\right)/b - c_1/b}{H_3^{<b,r>}} \right).$$

One can derive the batch complexity of BatchSAR as $O\left(H_3^{<b,r>} \log(n) + \frac{1}{b} \sum_{i=\tilde{n}+1}^{n} \left\lceil \frac{b}{i} \right\rceil\right)$. The term $\frac{1}{b} \sum_{i=\tilde{n}+1}^{n} \left\lceil \frac{b}{i} \right\rceil = O(\log(n) + n/b)$ is the price to pay for performing the (b, r)-batch where the algorithm might pull an arm more than necessary. The batch complexity is upper-bounded by $O\left(\left(\frac{1}{\tilde{r}} + \frac{\log n}{b}\right) H_2 \log n + \log n\right)$, where the additive $\log(n)$ is negligible unless $n/b \ll \log(n)$.

In the extreme case of b=r=1, the term $\frac{1}{b} \sum_{i=\tilde{n}+1}^{n} \left\lceil \frac{b}{i} \right\rceil$ is n, which can be ignored since the batch complexity is at least of order n when $b = 1$. Thus, we achieve $O(H_2 \log^2 n)$, which is equivalent to the sample complexity of SAR. If $r \geq \lceil b/2 \rceil$, $H_3^{<b,r>} \log n + \log n \leq$

$\frac{1}{b}(2+\log n)H_2 \log n + \log n = O(\frac{1}{b}H_2 \log^2(n) + \log n)$, which achieves an overall b factor reduction from the sample complexity of SAR except for the additive $\log n$. In the other extreme case where $b = n$ and $r = 1$, BatchSAR reduces to uniform sampling and the batch complexity is $O(\Delta_{\sigma(2)}^{-2} \log n)$.

Practical Considerations In general, the algorithm finishes earlier than B batches. The reason is that m_s is defined to defend against the worst case. In practice, one might want to exhaust the budget B to minimize the error. Let $\overline{\log}(b||a) = \sum_{i=a+1}^{b} \frac{1}{i}$. We propose to replace m_s with

$$m'_s = \left\lceil \frac{bB - \left(\sum_{s'=1}^{s-1} T_{j_{s'}}(t_{s'})\right) - \left(\sum_{i=\tilde{n}+1}^{n-s+1} \lceil \frac{b}{i} \rceil\right) - c'_1}{\left(\tilde{n}/2 + \overline{\log}(n-s+1||\tilde{n})\right)(n-s+1)} \right\rceil \tag{10}$$

where $c'_1 = b + \tilde{n}\tilde{r} + n - s + 1$. For the last stage $s = n-\tilde{n}+1$, we sample the surviving arms until the budget B is exhausted. One can show that $m'_s \geq m_s$ after using (8) to verify $\sum_{s'=1}^{s-1} T_{j_{s'}}(t_{s'}) + \left(\sum_{i=\tilde{n}+1}^{n-s+1} \lceil \frac{b}{i} \rceil\right) + c'_1 \leq \left(\sum_{i=\tilde{n}+1}^{n} \lceil \frac{b}{i} \rceil\right) + c_1 + \frac{(bB - \left(\sum_{i=\tilde{n}+1}^{n} \lceil \frac{b}{i} \rceil\right) - c_1)}{(\tilde{n}/2)+\overline{\log}(n||\tilde{n})} \overline{\log}(n||n-s+1)$. Thus, Theorem 2 still holds true with m'_s.

5 Experiments

We verify our theoretical results with simulation experiments in the fixed confidence setting, then present two important applications of the (b, r)-batch MAB problem. For applications, we focus on the fixed budget setting due to its practicality.

5.1 Toy Experiments

The batch size b and repeated pull limit r contribute to the batch complexity in an intricate way in both the fixed confidence and budget settings. We would like to verify how b and r change the batch complexity empirically and compare it to our theory. Note that the batch complexity shown in the analysis considers the worst case; the number of batches suggested by the theory is often more than what is experienced in practice. However, if such slack is independent of b and r, we can remove it by monitoring the "speedup", see below.

We consider two bandit instances with n=100 arms: (i) Bernoulli arms with expected rewards that linearly decrease, i.e., $\mu_i = (99-i+1)/99$, and (ii) sparsely distributed Bernoulli arms where the expected rewards of 10 arms are 0.5 and the rest are 0.3. We call the former Linear and the latter Sparse. Note we focus on the fixed confidence setting here due to space constraints; the same result can be obtained in the fixed budget setting. We run BatchRacing 10 times with $\delta = 0.1$, k=10, $b \in \{1, 4, 16, 64\}$ and $r \in \{1, 2, 4, 8, 16, 32\}$. For each pair (b', r'), we compute the empirical speedup by dividing the average number of batches in (b=1,r=1) by the average number of batches in (b', r'). Let $M_{b,r}$ be the batch complexity computed by (7). For each pair (b', r'), we also compute the theory-based speedup by $\left(\sum_{i=1}^{n} \overline{T}_{\sigma(i)}\right) / M_{b',r'}$.

Table 1 shows the the speedup of each (b, r)-batch. Overall, the theory-based speedup matches extremely well with the empirical speedup. This implies that the rather complicated contribution of b and r in the batch complexity well explains the reality. Note that r plays an important role in Linear; it is only after $r = b/2$ that the speedup achieves b. This is because $\Delta_{\sigma(1)}^{-2} = (\mu_k - \mu_{k+1})^{-2}$ dominates, which boosts the batch complexity's reliance on the factor $1/r'$. On the other hand, in Sparse the gaps $\{\Delta_i\}$ are uniformly 0.2, and we achieved an overall b fold speedup (almost) regardless of r as predicted in Section 3.

5.2 Application 1: Virus Replication Experiments

Virologists are interested in identifying which genes are important in virus replication. For instance, Hao et al. tested 13k genes of drosophila to find which genes promote influenza virus replication [8]. The authors perform a sequence of batch experiments using a plate with 384 microwells. Each well contains a single-gene knock-down cell strain to which the fluorescing virus is added. They measured the fluorescence levels after an incubation period and obtained an indication of how important the knock-down gene is in virus replication. Since the measurements are noisy, we formulate the problem as a top-k identification problem in MAB. Each arm is a gene, and the reward is its knock-down fluorescence level. The batch arm pull is performed with b=384 wells where the repeated pull limit r is equal to b since one can place the same gene knock-down in multiple wells.

Dataset We simulate the microwell experiments by assuming a Gaussian distribution on each arm, where the mean and variance are estimated from the actual data. Specifically, we use two measurements from each of the $n = 12,979$ [2] gene knock-downs from [8]. For the j-th measurement of arm/gene i, $x_{i,j}$, we remove the plate-specific noise by dividing it by the average control measurement from the same plate. Then, we take the logarithm to effectively make the variance of each fluorescence measurement similar, following [8]. We also flip the sign to call it $z_{i,j}$ since a low fluorescence level implies that the knock-down gene pro-

[2]The original dataset contains 13,071 genes. We found 92 genes whose control was measured incorrectly, so we removed them for the evaluation.

b	r	Linear		Sparse		b	r	Linear		Sparse	
		Theory	Actual	Theory	Actual			Theory	Actual	Theory	Actual
4	1	2.71	2.74	4.00	4.00	64	1	3.16	3.21	63.97	58.28
4	2	4.00	4.00	4.00	4.00	64	2	6.32	6.41	63.97	61.88
16	1	3.14	3.18	16.00	15.83	64	4	12.55	12.74	63.97	63.25
16	2	6.08	6.16	16.00	15.95	64	8	24.31	24.65	63.97	63.73
16	4	10.84	10.96	16.00	15.99	64	16	43.37	43.83	63.97	63.87
16	8	16.00	16.00	16.00	16.00	64	32	64.00	63.99	63.97	63.90

Table 1: Toy experiment result: the speedup in the number of batches in the fixed confidence setting.

motes the replication. We set the mean of arm i as $\mu_i = (z_{i,1} + z_{i,2})/2$. We compute the unbiased estimate of the variance $\widehat{V}_i$ of each gene and compute the average $(1/n)\sum_{i=1}^{n} \widehat{V}_i$ across all arms, which is an unbiased estimate of the variance if all arms have an equal variance. In the end, we simulate measurements for arm i from $\mathcal{N}(\mu_i, 0.1)$. We plot the arm distribution in Figure 1(a), where we see that relatively few genes have large rewards (that facilitate virus replication).

Algorithm 4 Halving

Input: n arms, k: the target number of top arms, b: batch size, r: repeated pull limit, B: batch budget
Output: k arms.
$t \leftarrow 1$, $S_1 \leftarrow [n]$, $T_i(0) \leftarrow 0, \forall i$
for $s = 1$ to $\lceil \log_2 \frac{n}{k} \rceil$ **do**
  for $i = 1$ to $\left\lfloor \frac{B}{\lceil \log_2(n/k) \rceil} \right\rfloor$ **do**
    $\mathbf{a} \leftarrow \text{RoundRobin}(S_s, \{T_i(t-1)\}_{i \in S_s}, b, r)$
    Pull by $\mathbf{a}$ (pull arm i a_i times, $\forall i$).
    $T_i(t) \leftarrow T_i(t-1) + a_i, \forall i$
    $t \leftarrow t + 1$
  end for
  Let S_{s+1} be the set of $\max\{\lceil |S_s|/2 \rceil, \lceil b/r \rceil\}$ arms in S_s with the largest empirical average.
end for
Output $S_{1+\lceil \log_2(n/k) \rceil}$.

Methods We use two algorithms as the baseline: RoundRobin and Halving. RoundRobin repeatedly pull arms according to RoundRobin (Algorithm 1) with no eliminations until the batch budget runs out. Halving adapts Sequential Halving algorithm [16], a top-1 identification algorithm, to perform top-k identification under the batch setting[3], though it lacks theoretical guarantee. We describe Halving in Algorithm 4. Both Halving and BatchSAR could waste the budget if run as is due to the integer operators. Thus, in Halving we randomly spread out the remaining budget over stages, and in BatchSAR we use m'_s defined in (10) in place of m_s.

Result We run the algorithms for $k \in \{100, 200, 400, 800\}$ and $B \in \{100, 200, 400, 800\}$. We run each experiment 20 times and measure false negative rate (FNR). FNR is the proportion of the top-k arms that do not appear in the output of the algorithm: $\frac{|[k] \setminus \{\text{the output of the algorithm}\}|}{k}$. High FNR was pointed out by [7] as the main issue in adaptive experiments. The result is shown in Figure 1 (b-e). We plot the FNR vs. batch budget B for each k, where the error bar is the 95% confidence interval. In all experiments, BatchSAR and Halving significantly outperform RoundRobin. RoundRobin spends the vast majority of arm pulls on arms with small expected rewards and fails to distinguish the top-k arms from those near top-k. BatchSAR outperforms Halving in most cases, especially when the budget is large.

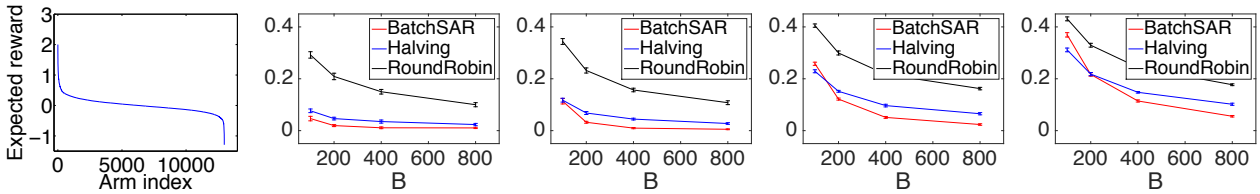
5.3 Application 2: Twitter User Tracking

We are interested in identifying the top-k users in Twitter who talk about a given topic most frequently. For example, in the study of bullying researchers are interested in a longitudinal study on the long term effects of bullying experience [21]. Such a study in Twitter requires first identifying the top-k (e.g., k=100) users with frequent bullying experience, then observing their posts for a long time (e.g., over a year). If one has an unlimited monitoring access to all users (e.g., firehose in Twitter), the first stage of identifying the top-k users would be trivial. However, researchers are often limited by the public API that has rate limits; for instance, user streaming API of Twitter[4] allows one to follow at most 5000 users at a time. In MAB terminology, each user is an arm, one batch pull is to follow 5000 users at a time for a fixed time interval, and the reward is whether or not a user posted tweets about a bullying experience. The latter is decided by a text classifier [25]. Even though it seems that $r = 1$ since we either follow a user or not, we discuss a procedure below where r can be larger.

We observe that the tweet rate of users changes significantly over time of day. Figure 2 (a) shows the daily trend. We plot the number of bullying tweets generated in each 5-minute interval where the count is accumulated over 31 days. To get around this diurnal issue, we define a pull of an arm to be following a user for 5 minutes every hour in a day, totaling two hours. Since there exist twelve 5-minute intervals in an hour, one can follow $b = 12 \cdot 5000 = 60{,}000$ users in a day.

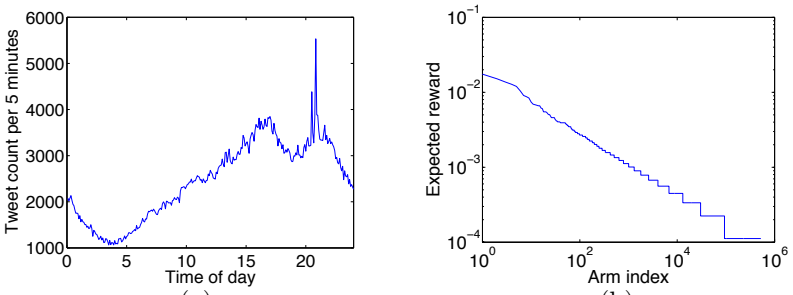
[3] We change the number of stages from $\lceil \log_2 n \rceil$ to $\lceil \log_2(n/k) \rceil$ following [15] and divide stages in terms of the number of batches rather than samples.

[4] `statuses/filter` API with `follow` parameter.



(a) Arm distribution (b) k=100 (c) k = 200 (d) k=400 (e) k=800

Figure 1: Microwell experiment result. (b-e) plots the false negative rate vs. budget B.



(a) (b)

Algorithm	Gain
BatchSAR	0.17 (±0.01)
Halving	**0.21** (±0.02)
RoundRobin	0.09 (±0.01)
Oracle	0.46

(c)

Figure 2: (a) Twitter daily pattern (b) Expected rewards of users (c) Experiment result

Note those 60,000 users need not be unique; one can follow the same user up to 12 times, each in a different 5-minute interval. Thus, r=12. Formally, we assume that the bullying tweet rate of a user i changes every hour of day h, but remains the same within. Let $Y_{i,d,h,c}$ indicate whether or not (1 or 0) the user i generates a bullying tweet at day d hour h interval $c \in [12]$. $Y_{i,d,h,c}$ follows a Bernoulli($p_{i,h}$) distribution, where $p_{i,h}$ is the tweet probability of user i during a 5-minute interval in hour h. We define the reward of an arm i on day d as $(1/24)\sum_{h=1}^{24} Y_{i,d,h,c(h)}$ where $c(h)$ indicates which interval to pick at each hour h. Then, the expected reward μ_i of an arm i is the average Bernoulli probability over a day: $\mu_i = (1/24)\sum_{h=1}^{24} p_{i,h}$.

Dataset The dataset consists of 707,776 bullying tweets collected over 31 days in January 2013. The number of distinct users n is 522,062. We compute the average reward for each user at each hour and take it as the true tweet probability $p_{i,h} := \frac{1}{31}\sum_{d=1}^{31}\frac{1}{12}\sum_{c=1}^{12} Y_{i,d,h,c}$, from which we compute the expected reward μ_i as previously shown. We plot the expected reward in Figure 2(b), where both axes are in $\log_{10}$ scale. The plot nicely follows a power law.

Result We use the same methods from Section 5.2. We set k=100 and run each algorithm 10 times due to randomness in breaking ties. We measure the gain $\sum_{i\in J}\mu_i$, where J is the set of k arms output by an algorithm. The gain measures the quality of the selected arms by the magnitudes of their expected rewards. The result is summarized in Figure 2(c). The number in parentheses is the 95% confidence interval, and Oracle is the gain on the true top-k arms ($J = \{1, \ldots, k\}$), which is the best achievable. Both Halving and BatchSAR significantly outperforms RoundRobin. Halving, though without theoretical guarantee, performs slightly better than BatchSAR. We suspect that the reason is that Halving works well in low budget situation as it was in Section 5.2. A thorough empirical comparison is left as a future work.

6 Future Work

There are numerous opportunities for future works in the (b, r)-batch setting. First, adapting existing pure exploration algorithms to the batch setting and analyzing their batch complexity. Not only Halving presented in this work without guarantee but also LUCB would be interesting to analyze since it is known to perform best in practice [12]. Second, investigating the lower bound on the batch complexity. Specifically, we have observed that the contribution of the repeated pull limit r to the batch complexity is nontrivial. It would be interesting to see if the same form of contribution happens in the lower bound as well. Finally, we would like to extend the batch setting to accommodate different batch sizes as in [22], but for $n > 2$ arms ([22] only pertains to $n = 2$) and in the pure exploration setting rather than the regret setting. Varying batch size MAB problems can arise in clinical trials and in crowdsourcing systems where the crowd size dynamically changes over time.

Acknowledgments

The authors are thankful to the anonymous reviewers for their comments. This work is supported in part by NSF grant IIS-0953219 and IIS-1447449, AFOSR grant FA9550-13-1-0138, ONR awards N00014-15-1-2620 and N00014-13-1-0129. We thank Paul Ahlquist, Michael Newton, and Linhui Hao for providing the virus replication experiment data and their helpful discussion.

References

[1] V. Anantharam, P. Varaiya, and J. Walrand, "Asymptotically efficient allocation rules for the multiarmed bandit problem with multiple plays-part i: I.i.d. rewards," *Automatic Control, IEEE Transactions on*, vol. 32, no. 11, pp. 968–976, 1987.

[2] J. Audibert, S. Bubeck, and G. Lugosi, "Regret in online combinatorial optimization," *Mathematics of Operations Research*, vol. 39, no. 1, pp. 31–45, 2014.

[3] J.-Y. Audibert, S. Bubeck, and R. Munos, "Best arm identification in multi-armed bandits," in *Proceedings of the Conference on Learning Theory (COLT)*, 2010.

[4] R. E. Bechhofer, "A sequential multiple-decision procedure for selecting the best one of several normal populations with a common unknown variance, and its use with various experimental designs," *Biometrics*, vol. 14, no. 3, pp. 408–429, 1958.

[5] S. Bubeck, T. Wang, and N. Viswanathan, "Multiple identifications in multi-armed bandits," in *Proceedings of the International Conference on Machine Learning (ICML)*, 2013, pp. 258–265.

[6] E. Even-dar, S. Mannor, and Y. Mansour, "Pac bounds for multi-armed bandit and markov decision processes," in *Proceedings of the Conference on Learning Theory (COLT)*, 2002, pp. 255–270.

[7] L. Hao, Q. He, Z. Wang, M. Craven, M. A. Newton, and P. a. Ahlquist, "Limited agreement of independent rnai screens for virus-required host genes owes more to false-negative than false-positive factors," *PLoS Computational Biology*, vol. 9, no. 9, p. e1003235, 2013.

[8] L. Hao, A. Sakurai, T. Watanabe, E. Sorensen, C. A. Nidom, M. A. Newton, P. Ahlquist, and Y. Kawaoka, "Drosophila rnai screen identifies host genes important for influenza virus replication," *Nature*, vol. 454, no. 7206, pp. 890–893, 2008.

[9] V. Heidrich-Meisner and C. Igel, "Hoeffding and bernstein races for selecting policies in evolutionary direct policy search," in *Proceedings of the International Conference on Machine Learning (ICML)*, 2009, pp. 401–408.

[10] W. Hoeffding, "Probability inequalities for sums of bounded random variables," *Journal of the American Statistical Association*, vol. 58, no. 301, pp. 13–30, March 1963.

[11] K. G. Jamieson, M. Malloy, R. Nowak, and S. Bubeck, "lil' ucb : An optimal exploration algorithm for multi-armed bandits," in *Proceedings of The 27th Conference on Learning Theory*, 2014, pp. 423–439.

[12] K. G. Jamieson and R. Nowak, "Best-arm identification algorithms for multi-armed bandits in the fixed confidence setting," in *48th Annual Conference on Information Sciences and Systems, CISS*, 2014, pp. 1–6.

[13] P. Joulani, A. György, and C. Szepesvári, "Online learning under delayed feedback." in *Proceedings of the International Conference on Machine Learning (ICML)*, 2013, pp. 1453–1461.

[14] S. Kalyanakrishnan and P. Stone, "Efficient selection of multiple bandit arms: Theory and practice," in *Proceedings of the International Conference on Machine Learning (ICML)*, 2010, pp. 511–518.

[15] S. Kalyanakrishnan, A. Tewari, P. Auer, and P. Stone, "PAC subset selection in stochastic multi-armed bandits," in *Proceedings of the International Conference on Machine Learning (ICML)*, 2012, pp. 655–662.

[16] Z. S. Karnin, T. Koren, and O. Somekh, "Almost optimal exploration in multi-armed bandits." in *Proceedings of the International Conference on Machine Learning (ICML)*, 2013, pp. 1238–1246.

[17] E. Kaufmann, O. Cappé, and A. Garivier, "On the complexity of best-arm identification in multi-armed bandit models," *Journal of Machine Learning Research*, vol. 17, no. 1, pp. 1–42, 2016.

[18] A. Niculescu-Mizil, "Multi-armed bandits with betting," in *COLT'09 Workshop on On-line Learning with Limited Feedback*, 2009.

[19] A. M. Oded Maron, "Hoeffding races: Accelerating model selection search for classification and function approximation," in *Advances in Neural Information Processing Systems (NIPS)*, April 1994, pp. 59–66.

[20] E. Paulson, "A sequential procedure for selecting the population with the largest mean from k normal populations," *Annals of Mathematical Statistics*, vol. 35, no. 1, pp. 174–180, 1964.

[21] A. D. Pellegrini and J. D. Long, "A longitudinal study of bullying, dominance, and victimization during the transition from primary school through secondary school," *British Journal of Developmental Psychology*, vol. 20, no. 2, pp. 259–280, 2002.

[22] V. Perchet, P. Rigollet, S. Chassang, and E. Snowberg, "Batched bandit problems," in *Proceedings of the Conference on Learning Theory (COLT)*, 2015, pp. 1456–1456.

[23] M. J. Weinberger and E. Ordentlich, "On delayed prediction of individual sequences." *IEEE Transactions on Information Theory*, vol. 48, no. 7, pp. 1959–1976, 2002.

[24] Y. Wu, A. Gyrgy, and C. Szepesvri, "On identifying good options under combinatorially structured feedback in finite noisy environments." in *Proceedings of the International Conference on Machine Learning (ICML)*, 2015, pp. 1283–1291.

[25] J.-M. Xu, K.-S. Jun, X. Zhu, and A. Bellmore, "Learning from bullying traces in social media," in *Proceedings of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL HLT)*, 2012, pp. 656–666.