Understanding Exploration in Bandits with Switching Constraints: A Batched Approach in Fixed-Confidence Pure Exploration

Newton Mwai
Department of Computer Science and Engineering
Chalmers University of Technology and University of Gothenburg
SE-41296 Gothenburg, Sweden
mwai@chalmers.se

Milad Malekipirbazari *
Department of Computer Science and Engineering
Chalmers University of Technology and University of Gothenburg
SE-41296 Gothenburg, Sweden
milad.maleki@gmail.com

Fredrik D. Johansson
Department of Computer Science and Engineering
Chalmers University of Technology and University of Gothenburg
SE-41296 Gothenburg, Sweden
frejohk@chalmers.se

Abstract

Most multi-armed bandit algorithms focus on efficient exploration, often oblivious to constraints tied to exploration like the cost of switching between arms. Switching costs arise in real-world applications such as personalized medicine, in which changes in treatment may require a wash-out period where the patient is not taking any drug; or in industrial applications where reconfiguring production is costly. Unfortunately, controlling for switching is significantly understudied outside of regret minimization. In this work, we present a formulation of the fixed-confidence pure exploration problem with constraints on the arm switching frequency. We show how this problem lends itself to batched bandits and give a lower bound on the exploration time for any such algorithm. We translate this idea into two algorithms inspired by the track-and-stop framework, adapted to batch plays with a limited number of arm switches per batch. Finally, we demonstrate empirically that our approach achieves quick stopping times, comparable to unconstrained algorithms, even when constrained to a minimal switching limit.

1 Introduction

Sequential decision-making algorithms promise to improve outcomes in diverse applications, from healthcare to e-commerce, by systematically exploring alternative policies for action. Classically, an effective algorithm strikes a good balance between exploration and exploitation [24, 7, 21],

*Work conducted during a postdoctoral affiliation at Chalmers University of Technology.

18th European Workshop on Reinforcement Learning (EWRL 2025).

converging as quickly as possible to a good or optimal policy. However, bandit and reinforcement learning algorithms typically assume that actions can be freely switched during exploration.

However, many applications come with costs tied to *switching* actions, and decision-making agents are incentivized to use the same action repeatedly. For example, in healthcare settings such as in clinical trials [4] and treatment personalisation for chronic diseases [17], switching treatments has costs for the patient: every time a treatment is changed, the patient has to weave off their current therapy and get used to the new treatment and its potential side effects. In the personalization of web pages or apps, switching content or interface frequently may be inconveniencing or annoying to users, and in industrial applications, switching actions could mean high costs of reconfiguring production setups. It is therefore desirable to limit the frequency of arm switches, even if they make exploration more efficient.

In the multi-armed bandit (MAB) problem, an agent sequentially samples actions from a set of unknown distributions, and it aims to sample (*explore*) them in a manner that helps it to learn about the underlying distributions; either quickly, or with high confidence given an exploration budget (*pure exploration*) [5, 13, 10], or in order to minimize the cumulative cost of choosing sub-optimal actions (*regret minimization*) [26, 11, 20, 22]. Switching in multi-armed bandits has been extensively studied in the regret minimization setting, [3, 8, 25, 2, 23] but it is less studied for pure exploration, possibly because satisfying fixed-confidence correctness is difficult while minimizing the total number of switches. Several works on regret minimization with switching costs use ideas around batching the action selection in time. Although there are works on batched bandits for pure exploration [16, 1, 18, 6], the area is still relatively under-explored.

Main contributions. 1) We propose a formulation of the fixed-confidence pure exploration problem with a constraint on the frequency of arm switching (Section 2). **2)** We provide a lower bound for the search time of any bandit algorithm that solves this problem by limiting the arm switching frequency using batches of uniform size (Section 3). **3)** We present two tracking-based algorithm variants, Sparse-Projected Batch C-Tracking (SPB C-Tracking) and Sparse Batch Configurations (SBC) (Section 4), and give an asymptotic upper bound for the exploration time of both. **4)** We present empirical results from a simulation study showing that our algorithms identify the best arm in time comparable to track-and-stop algorithms without switching constraints and more quickly than existing successive-elimination batch algorithms (Section 5).

2 Problem Formulation

We study fixed-confidence pure exploration multi-armed bandits with a limit on the rate of arm switches. Let $\mathcal{A} = \{1, ..., K\}$ be a set of arms and $\mu_a \in \mathbb{R}$ the expected reward for arm $a \in \mathcal{A}$, with $\mu^* = \max_{a \in \mathcal{A}} \mu_a$. A bandit algorithm ϕ plays an arm a_t in successive rounds $t = 1, 2, ...$, before terminating according to a stopping criterion at time τ and recommending the arm $\hat{a}_\tau$. The total number of arm switches S_τ is the number of successive plays where the arms differ, $S_\tau = \sum_{t=2}^{\tau} [a_t \neq a_{t-1}]$. Our goal is to design a search strategy ϕ *to minimize the expected number of arm plays* τ *required to identify an optimal arm with confidence at least* $1 - \delta$ *for a given* $\delta > 0$*, while limiting the expected rate of switching arms to* $\alpha \in [0, 1]$.

$$\begin{aligned} \underset{\phi}{\text{minimize}} \quad & \mathbb{E}_\phi[\tau] \\ \text{subject to} \quad & \mathbb{P}(\mu_{\hat{a}_\tau} < \mu^*) \leq \delta \\ & \mathbb{E}_\phi[S_\tau] \leq \alpha \mathbb{E}_\phi[\tau] \end{aligned} \tag{1}$$

This formulation, however, poses a challenge: the switching rate constraint depends on the expected stopping time $\mathbb{E}_\phi[\tau]$, which is the very optimization target, creating a circular dependency. We resolve this as a reformulation with a *batched* variant of the problem that provides a practical and well-defined alternative to the original constraint.

In batched bandits [9, 14, 16, 6], arm plays are planned in a sequence at the start of a batch, and the rewards for all plays are given at the end for the bandit to update its model. Using batches of fixed size B allows us to ensure that all plays for a specific arm are made in sequence within the batches, and the number of switches in a batch S^b is determined by the number of distinct arms, see Figure 1. With this, the number of switches in exploration will be attributed either to switching between arms *within* the batches when changing from one successive arm play segment to the next, or to changing arms *between* batches.

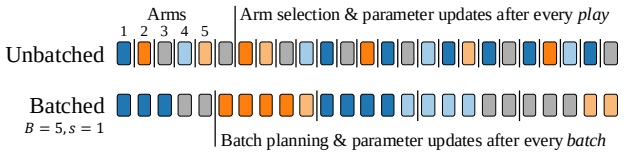


Figure 1: Illustration of batched arm plays used to limit the arm switching frequency in a 5-arm problem. The number of plays of each arm is the same.

Let S^b denote the number of switches in batch b. If $S^b \leq s$ for all b, we can bound the expected number of switches by $\mathbb{E}_\phi[S_\tau] \leq \mathbb{E}_\phi[\beta](s+1) - 1$; where $\mathbb{E}_\phi[\beta]$ is the expected number of batches played by ϕ before terminating. The bound covers the number of switches within a batch and the switches from one batch to the next. As a result, the second constraint in our objective above can be satisfied by keeping the switches in the batches low, requiring that the constraint in Eq. (1) holds for the right-hand side of the inequality above, $\forall b : \mathbb{E}_\phi[\beta](s+1) - 1 \leq \alpha \mathbb{E}_\phi[\tau] = \alpha B \mathbb{E}_\phi[\beta]$. If this holds, with $\lfloor \cdot \rfloor$ the floor operator,

$$\forall b : S^b \leq s := \lfloor \alpha B - 1 \rfloor \implies \mathbb{E}_\phi[S_\tau] \leq \alpha \mathbb{E}_\phi[\tau] \,. \tag{2}$$

We can now re-formulate our goal to be *to minimize the expected number of batches β required to identify an optimal arm, with confidence at least $1-\delta$, while limiting the arm switches within the batch to be at most $s \in \{0, ..., \min(K-1, B-1)\}$*,

$$\begin{aligned} \underset{\phi}{\text{minimize}} \quad & \mathbb{E}_\phi[\beta] \\ \text{subject to} \quad & \mathbb{P}\left(\mu_{\hat{a}_\beta} < \mu^*\right) \leq \delta \\ & S^b \leq s, \ \forall b \in \mathbb{N} \end{aligned} \tag{3}$$

In this work, all batches are planned deterministically. Randomized algorithms could yield non-integer *expected* numbers of switches but we do not explore that.

3 Lower Bounding the Number of Batches in Pure Exploration

For fixed-confidence pure exploration, [10] presented a general lower bound for the expected stopping time $\mathbb{E}[\tau]$ of any δ-PAC multi-armed bandit algorithm, i.e., one that returns the best arm with probability at least $1-\delta$, for some $\delta > 0$,

$$\mathbb{E}[\tau] \geq T^*(\mu) \, \mathrm{kl}(\delta, 1-\delta) \,. \tag{4}$$

$$\text{where } T^*(\mu)^{-1} := \sup_{w \in \Sigma^K} \inf_{\lambda \in \mathrm{Alt}(\mu)} \left(\sum_{a=1}^{K} w_a d(\mu_a, \lambda_a) \right).$$

Here, $d(.)$ is the KL-divergence, and $\Sigma^K := \{w \in \mathbb{R}_+^K : \sum_{a=1}^K w_a = 1\}$ is the simplex of possible arm playing proportions. This lower bound is derived by considering the optimal allocation of arm pulls w^* to minimize the worst-case stopping time specific to the instance μ while ensuring that the probability of incorrectly identifying the best arm does not exceed a pre-specified confidence level δ. The term $T^*(\mu)$ represents the inverse of the exploration time associated by the best-case (supremum) playing proportions w and the worst-case (infimum) alternative bandit model λ (that differs from μ in its optimal arm), $\mathrm{Alt}(\mu) = \{\lambda \in \mathbb{R}^K : \arg\max_a \lambda_a \neq \arg\max_a \mu_a\}$.

We apply the same idea to bound the number of batches necessary for exploration. Let $\mathbb{E}[\beta]$ denote the expected number of batches β necessary for any δ-PAC algorithm. Seemingly, we can extend the reasoning of the lower bound in Eq. (4),

$$\mathbb{E}[\beta] \geq T_b^*(\mu) \, \mathrm{kl}(\delta, 1-\delta) \tag{5}$$

$$T_b^*(\mu)^{-1} := \sup_{w \in \Sigma^K} \inf_{\lambda \in \mathrm{Alt}(\mu)} \left(\sum_{a=1}^{K} \lceil w_a B \rceil \cdot d(\mu_a, \lambda_a) \right)$$

where w_a represents the fraction of times that arm a is played out of the total expected number of arm plays, and $\lceil w_a B \rceil$ an upper bound on the expected number of times arm a is played in each batch, with $\lceil \cdot \rceil$ the ceiling operator. However, for small batch sizes, playing according to this distribution of arm plays every batch is infeasible. And even when batch sizes are large, this does not respect the arm switching limit.

3.1 A Lower Bound on the Number of Batches with Switching Constraints

The lower bound in Eq. (5) may not generally be attainable by algorithms that obey the arm switching constraint in Eq. (1) since this limitation is not represented in the bound. Incorporating the switching constraint introduces a new dimension to the problem where planning the plays over successive batches becomes crucial. To match the lower bound, *we want to end up having played according to the maximizer of Eq. (5), but we can't play according to these proportions every batch*. Arms must be scheduled in a way that minimizes the exploration time *across* batches, while respecting the given constraints. To represent this in a lower bound, we will study the optimal proportions of played *sparse batch configurations* instead.

Given that the batch size is fixed and known, we can index all possible configurations c of integer arm plays in a batch that satisfy the desired switching limit. For a given number of arms K, batch size B and switching limit s, we denote this set $\mathcal{C}_{B,s}^K$,

$$\mathcal{C}_{B,s}^K := \left\{ c \in \mathbb{N}^K : \sum_{a=1}^K c_a = B, \;\; \|c\|_0 \leq s+1 \right\} . \tag{6}$$

Here, $\|\cdot\|_0$ denotes the ℓ_0-norm, which counts the number of nonzero elements in the vector, $\|x\|_0 := \sum_{i=1}^K \mathbb{1}[x_i \neq 0]$. Each element $c = [c_1, ..., c_K]^\top \in \mathcal{C}_{B,s}^K$ represents a configuration that can be executed in a single batch and each coordinate c_a represents the number of times arm a will be played in the batch. We say that c is *sparse* if there are arms a such that $c_a = 0$.

Building on the above definition, define p_c to be the proportion of batches that use a configuration c during the bandit algorithm run. The total plays up of arm a until batch β can be expressed as:

$$\mathbb{E}[N_a(\beta)] = \sum_{b=1}^{\beta} \sum_{c \in \mathcal{C}_{B,s}^K} p_c c_a \; .$$

We can now state a lower bound for batch-playing bandits that obey the switching constraint.

Theorem 1. *Let $\Sigma^{\mathcal{C}} := \Sigma^{|\mathcal{C}_{B,s}^K|-1}$ be the simplex over batch configurations of size B that use fewer than s switches. Given a confidence level $\delta \in (0,1)$, for any algorithm that returns the best arm with probability at least $1-\delta$, and for any bandit problem $\mu \in \mathbb{R}^K$, the following holds:*

$$\mathbb{E}_\mu[\beta] \geq T_{bc}^*(\mu) \cdot \mathrm{kl}(\delta, 1-\delta), \tag{7}$$

where the characteristic time $T_{bc}^*(\mu)$ is given by

$$T_{bc}^*(\mu)^{-1} := \sup_{p \in \Sigma^{\mathcal{C}}} \inf_{\lambda \in \mathrm{Alt}(\mu)} \sum_{a=1}^K \sum_{c \in \mathcal{C}_{B,s}^K} p_c c_a d(\mu_a, \lambda_a). \tag{8}$$

A proof is given in Appendix A. In Eq. (8), the supremum is computed over the possible probabilities of sparse configurations, thereby incorporating the switch limit into the batch play optimization.

Although similar in form and derivation, our lower bound differs from the result in [10] in several key ways. First, arm plays are confined to batches with switching constraints, meaning only a limited number of unique arms are played in each batch. Their result has no constraints on how arms can be played in sequence. Second, the playing proportions in our result are defined for entire batch configurations $\mathcal{C}_{B,s}^K$ of integer arm plays. As we will see in the next section, unlike in the non-batched setting, this does not translate as easily to a bandit algorithm.

Our approach to combining batch plays with switching constraints is related to combinatorial bandits with semi-bandit feedback, see e.g., Jourdan et al. [15]. Solving this problem involves making decisions over combinations of several arms at a time. Similarly, our method involves managing combinations of arm plays within each batch, taking into account the sparsity constraint to limit switching between arms. The main difference with this setting is that our goal is to identify a *single optimal arm*, not an optimal combination.

The configuration set $\mathcal{C}_{B,s}^K$ introduced in our method is typically very high-dimensional, scaling exponentially with the number of arms. Restricting batches to contain *at most* s switches, we have $\left|\mathcal{C}_{B,s}^K\right| = \sum_{i=0}^s \binom{K}{i+1}\binom{B-1}{i}$ configurations. See Appendix Tables 2,3 for examples.

The large dimensionality of $\mathcal{C}_{B,s}^K$ means that solving Eq. (8) by enumerating all configurations is practically infeasible. Moreover, the solution may not be unique since the combination of several configurations with different proportions may yield the same expected number of arm plays. Consider, for example, a problem with $K = 3$, $B = 2$, $s = 1$ and $c_1 = [2, 0, 0]^\top, c_2 = [0, 0, 2]^\top$, and $c_3 = [1, 0, 1]^\top$. Playing with proportions $p = [0.5, 0, 0.5]^\top$ and $p' = [0, 0, 1]^\top$ yields the same expected number of plays of each arm. As a result, although configurations take the place of arms in the lower bound, the implications for algorithm design are quite different from the non-batched case.

4 Tracking Algorithms

Inspired by their analysis, Garivier and Kaufmann [10] introduced the idea of *track-and-stop* algorithms, designed to *track* the optimal arm playing proportions $w^*(\hat{\mu})$ of the lower bound in Eq. (4),

$$w^*(\hat{\mu}) := \arg\max_{w \in \Sigma^K} \inf_{\lambda \in \mathrm{Alt}(\hat{\mu})} \left(\sum_{a=1}^{K} w_a d(\hat{\mu}_a, \lambda_a) \right) . \tag{9}$$

based on an estimate $\hat{\mu}$ of the arm parameters, continuously updated as more data is collected. A track-and-stop algorithm plays arms following a *tracking rule* aiming for an overall arm proportion as close to the optimal proportions as possible, combined with a *stopping rule* for terminating exploration. The stopping rule is a statistical test of whether the past observations indicate, with a risk of at most δ, that one arm has a higher average reward than the others.

Applying the track-and-stop framework in our setting requires imposing a switching constraint in the tracking rule. We cannot impose sparsity in the tracked proportions w^* without destroying the solution to Eq. (9). If an arm a is never played, $w_a = 0$, the adversary λ can exploit this and differ arbitrarily for that arm, rendering the lower bound infinite. This is also evident from Lemma 4 in Garivier and Kaufmann [10] which would be violated if $\exists a : w_a^* = 0$. Neither is it a good idea to play configurations to track the proportions p^* that solve Eq. (8). The solution is not necessarily unique and, even if it is, the sheer number of possible configurations makes exploring (tracking) all of them infeasible. Moreover, the number of batches where a configuration is played is not itself of interest, only that the resulting distribution of arm plays is optimal.

Instead, we can attempt to track the optimal arm proportions with suitably chosen batches. Suppressing superscripts and subscripts for convenience, we let $\mathcal{C} = \mathcal{C}_{B,s}^K$.

Observation 1. *If the optimal arm allocation w^* in Eq. (9) is "realizable" under $\mathcal{C}$, i.e., $\exists p^* \in \Sigma^{\mathcal{C}}$ such that $\sum_{c \in \mathcal{C}} p_c^* c = w^*(\hat{\mu})$, then p^* are minimizers of Eq. (8).*

This is clear since Eq. (8) is more constrained than Eq. (9). Whenever every single-arm configuration is feasible, that is, $\forall a : \mathbb{1}_a B \in \mathcal{C}$, where $\mathbb{1}_a$ is the one-hot binary vector at a, the condition in Observation 1 is true. This argument motivates constructing batch configurations that together track the optimal proportions of arm plays given by Eq. (9).

4.1 Tracking Arm Proportions with Batches

We present an algorithm and two batch selection rules for tracking optimal proportions of arm plays. Algorithm 1 provides a pseudocode outline of both variants. The general strategy for tracking algorithms is to establish a set of goal proportions $\bar{w}$ and select arms to minimize the deficit of played arm proportions to the goal. In the C-tracking procedure[10], the goal proportion is the cumulative sum of tracking weights over batches,

$$\bar{w}(b) = B \sum_{i=0}^{b-1} w^{\epsilon_i}(\hat{\mu}_i) \tag{10}$$

where $w^\epsilon(\hat{\mu})$ is the L_∞-projection of $w^*(\hat{\mu})$ in Eq. (9) onto $\Sigma_\epsilon^K = \{w \in \mathbb{R}_+ : \sum_a w_a = 1, \min_a w_a \geq \epsilon\}$. The deficit for arm a in batch b is then $d_a(b) := \bar{w}_a(b) - N_a(b)$, and the vector of deficits is $d(b) = [d_1(b), ..., d_K(b)]^\top$. We aim to minimize the total positive deficit $D(b) := \sum_{a=1}^K (d_a(b))_+$, where $(x)_+ = \mathbb{1}[x > 0]x$. To this end, we define the selection rule,

$$\tilde{c} \in \arg\min_{c \in \mathcal{C}} \sum_{a=1}^{K} \Big(d_a(b) - c_a \Big)_+ \tag{11}$$

where c_a are the number of plays of arm a in the batch configuration c. In Section 4.2, we prove that this rule results in an upper bound on the number of batches necessary to find the best arm with high probability, in the high-certainty asymptotic regime, $\delta \to 0$.

Greedy batch filling: Sparse Batch Configurations (SBC) C-Tracking algorithm

Unlike the lower bound problem Eq. (8), Eq. (11) can actually be solved in polynomial time through a greedy algorithm (see Appendix C). We call this variant of our algorithm *Sparse Batch Configurations* (SBC) C-Tracking (see Algorithm 1). However, the solution is not unique. For example, if more than $s+1$ arms have positive deficit, there are cases where the allocations to the selected arms in the batch can be decided partially arbitrarily. Once the deficit of selected arms has been removed, the choice of how to distribute remaining plays between them won't alter Eq. (11). The algorithm in Appendix C puts the remaining allocation on the arm with the largest remaining fractional deficit. Next, we consider another algorithm variant that constructs batches proportional to the arm deficits.

Proportional batch filling: Sparse Projected Batch (SPB) C-Tracking algorithm

The Sparse-Projected Batch Tracking algorithm is an alternative method of selecting batch configurations that minimize total arm play deficits. The allocations in the batch are now distributed proportionally to the deficits of the selected arms. The idea is to project the normalized positive deficits between expected and actual plays $(\bar{d}(b))_+ = \frac{(d(b))_+}{\sum_{a\in\mathcal{A}}(d_a(b))_+}$ onto an *(s+1)-sparse simplex* and construct the batch configuration according to the resulting sparse proportions. Kyrillidis et al. [19] showed that projection on a sparse simplex (e.g. $s+1$-sparse) is solved exactly using the polynomial greedy selector and simplex projector (GSSP) by *selecting the largest $(s+1)$ items* and then re-normalizing.

Let $N(b) = [N_1(b), ..., N_K(b)]^\top$ be the vector comprising the number of plays of each arm until batch b and define the s switch-constrained ($(s+1)$-sparse) simplex,

$$\Sigma_{s+1}^K = \left\{ w \in \mathbb{R}_+^K : \sum_{a=1}^K w_a = 1,\ \|w\|_0 \le s+1 \right\} .$$

We compute batch proportions by projecting (with GSSP) the normalized positive deficits onto Σ_{s+1}^K:

$$\hat{w}^{s+1}(b) \in \underset{w\in\Sigma_{s+1}^K}{\arg\min} \left\| w - (\bar{d}(b))_+ \right\|_2 . \tag{12}$$

The configurations are then obtained as $\tilde{c} = \text{integer}(\hat{w}^{s+1}(b) * B)$ after rounding arm proportions, using the procedure described in Appendix E.

Arm Selection and Stopping Rules

Within each batch, the algorithms iteratively select the arm a_t with the highest number of plays in the configuration, plays it $\tilde{c}_{a_t}$ times and observe the rewards $(r_t, ..., r_{t+\tilde{c}_{a_t}-1})$. After the batch, the arm estimates are updated. The process repeats for batches until a criterion based on a confidence threshold is met. We use Chernoff's stopping rule as presented in [10] with statistic

$$Z_b = \max_{a\in\mathcal{A}} \min_{\tilde{a}\neq a} N_a(b) d_{\hat{\mu}(b)}(a,\tilde{a}) + N_{\tilde{a}}(b) d_{\hat{\mu}(b)}(\tilde{a}, a) \tag{13}$$

where $d_{\hat{\mu}}(a,\tilde{a}) = \text{KL}(\hat{\mu}_a, \hat{\mu}_{a,\tilde{a}})$ and $\hat{\mu}_{a,\tilde{a}}$ is the weighted average of $\hat{\mu}_a$ and $\hat{\mu}_{\tilde{a}}$, weighted by their arm plays: $\hat{\mu}_{a,\tilde{a}}(t) = \frac{N_a(t)}{N_a(t)+N_{\tilde{a}}(t)}\hat{\mu}_a(t) + \frac{N_{\tilde{a}}(t)}{N_a(t)+N_{\tilde{a}}(t)}\hat{\mu}_{\tilde{a}}(t)$. We use the threshold $Z_b > \log\left(\frac{\log(bB)+1}{\delta}\right)$. At the stopping batch β, the estimated best arm $\hat{a}_\beta = \arg\max_a \hat{\mu}_{\beta,a}$ is returned.

4.2 An Upper Bound on the Stopping Batch

By proving that batch configurations selected according to Eq. (11) track the optimal arm proportions in Eq. (9), we show that both SBC and SPB C-Tracking in Algorithm 1 match the lower bound in Eq. (4) in the high-certainty limit.

Algorithm 1 Sparse Batch Configurations (SBC) and Sparse Projected Batch (SPB) C-Tracking

Input: K arms, $\delta \in (0,1)$, B: batch size
Input: s: batch switch limit
Output: β, $\hat{a}_\beta$

$b \leftarrow 1$, $t \leftarrow 1$, $Z_1 \leftarrow 0$, $\hat{\mu}_0 \leftarrow \mathbf{0}$, $N(1) \leftarrow \mathbf{0} \in \mathbb{R}^K$
while $Z_b \leq \log\left(\frac{\log(bB)+1}{\delta}\right)$ **do**
  Let $\epsilon_b \leftarrow (K^2 + bB)^{-1/2}/2$ ▷ set $\epsilon_b = 1/K$ if $bB < 3K^2$
  Compute $w^{\epsilon_{b-1}}(\hat{\mu}_{b-1})$ ▷ See Eq. (10)
  Compute $d(b) = B\sum_{i=0}^{b-1} w^{\epsilon_i}(\hat{\mu}_i) - N(b)$
  if SBC C-Tracking **then**
    Let $\tilde{c} \in \arg\min_{c \in \mathcal{C}^K_{B,s}} \sum_{a=1}^{K} (d_a(b) - c_a)_+$
    ▷ Greedy batch filling (App. C)
  else if SPB C-Tracking **then**
    Let $\hat{w}^{s+1}(b) \in \arg\min_{w \in \sum^K_{s+1}} \left\| w - (\bar{d}(b))_+ \right\|_2$
    and $\tilde{c} = \text{integer}(\hat{w}^{s+1} * B)$
    ▷ Proportional filling (Eq. (15), App. E)
  end if
  while $t \leq bB$ **do**
    Let $\bar{a} \leftarrow \arg\max_{a \in \mathcal{A}} \tilde{c}$ and $\bar{c} = \tilde{c}_{\bar{a}}$
    Play $a_t, ..., a_{t+\bar{c}-1}$ with arm $\bar{a}$
    Observe rewards $(r_t, ..., r_{t+\bar{c}-1})$
    $N_{a_t}(b+1) \leftarrow N_{a_t}(b) + \bar{c}$
    $\hat{\mu}_{b,a_t} \leftarrow \frac{1}{N_{\bar{a}}(b+1)} \sum_{j=1}^{t+c_{\bar{a}}-1} \mathbb{1}[a_j = \bar{a}] r_j$
    $t \leftarrow t + \bar{c}$ and $\tilde{c}_{\bar{a}} \leftarrow 0$
  end while
  $b \leftarrow b + 1$
  Compute Z_b ▷ See Eq. (13)
end while
Return $\hat{a}_\beta = \arg\max_a \hat{\mu}_{\beta,a}$

Theorem 2. *Let $\mu \in \mathbb{R}^K$, $B \geq 1$, $s \in [B-1]$ be an instance of the switch-constrained pure exploration problem with confidence $\delta > 0$. Assume that the optimal arm allocation $w^*(\mu)$ is realizable under $\mathcal{C}^K_{B,s}$ (Observation 1) and let $\alpha \in [1, e/2]$ and $\rho(t) = O(t^\alpha)$. Then, using Algorithm 1, and Chernoff's stopping rule with threshold* $\log(\rho(t)/\delta)$,

$$\limsup_{\delta \to 0} \frac{\mathbb{E}[\beta_\delta]}{\log 1/\delta} \leq \alpha T^*_{bc}(\mu) \, .$$

where T^*_{bc} is the batch characteristic time from Eq. (8).

The result is proven in Appendices B–D: Lemmas 1–2 prove that the deviation of any tracking rule minimizing Eq. (11) is bounded and are used in Appendix B.3 to give a general upper bound on $\mathbb{E}[\beta_\delta]$. In Appendices C–D, we prove that SBC and SPB C-tracking minimize Eq. (11).

Tightness of the bound Theorem 2 matches the lower bound in Theorem 1. The parameters s and B are not visible in the bound but affect the batch characteristic time T^*_{bc}. Moreover, in the non-asymptotic case, the tightness of the bound depends on s, B, and K. For any non-zero δ, Lemmas 1–2 (see Appendix B) used to prove Theorem 2 establish that there is a batch number $\beta_\epsilon \leq ((2K+1)/(2\epsilon))^4$ after which the proportion $N_a(b)/(bB)$ of plays of each arm differs by at most $3(K-1)\epsilon$ from the optimal proportion $w^*_a(\mu)$, with ϵ the C-tracking forced-exploration parameter. In this regime, the bound is tighter for small s since the bound on β_ϵ assumes that plays of an important arm can be delayed by a whole batch due to the switching constraint.

Comparison to the unconstrained setting. In the asymptotic setting, the upper bound on the *number of batches* (Theorem 2) matches the bound on the *number of arm plays* in the non-batched case. This means we pay up to a factor B to reach the same guarantee as in the unconstrained case. This should not be surprising—in the worst case, $s = 0$, we can play at most one unique arm in each batch but the optimal proportions may be uniform. This is illustrated in our experiments, where the switching constraint only limits performance in the high-batch size setting (see Figure 3b).

Extension to unrealizable allocations. Theorem 2 is restricted to the realizable case for presentation since SPB and SBC C-tracking are designed for this case. The bound can be generalized to the case where unconstrained arm allocations are not representable by $\mathcal{C}^K_{B,s}$ by replacing Σ^K in Eq. (9) by $\tilde{\Sigma}^K = \{w : \exists p \in \Sigma^{\mathcal{C}}, \sum_{c\in\mathcal{C}} p_c c = w\}$ and carrying forward to Eq. (11). The remaining analysis is unchanged.

5 Simulation Experiments

In our experiments, we are foremost interested in investigating if we can achieve a low stopping time $\mathbb{E}[\tau] = B\mathbb{E}[\beta]$ while conforming to a given switching limit s during exploration. We investigate the effects of different combinations of s and the batch size B. As discussed in Section 2, s and B are inherently tied, $s \coloneqq \lfloor \alpha B - 1 \rfloor$. For different batch sizes, the choice of s will have a large impact on which arm configurations can be played in a batch, and therefore on the nature of exploration. We aim to validate this empirically.

We compare our SBC and SPB C-Tracking algorithms to BatchRacing [16], a batched racing algorithm that successively eliminates arms across rounds, starting with the full set of available arms as a surviving set $S_1 = \mathcal{A}$. In each batch of size B, it uses a round-robin algorithm to determine plays in the batch uniformly. It is designed for top-k best-arm identification; our setting is top-1. In each round, BatchRacing uses upper-confidence bounds (resp. lower-confidence bounds) to determine if there is any arm that is confidently top-k (resp. not), and moves the arms to an accepted, A_t (resp rejected, R_t) set. The arms moved to the accepted or rejected sets are removed from the surviving sets and this repeats until k arms are accepted, whereby it ends, outputting the accepted set A_τ. In our setting $|A_\tau| = 1$.

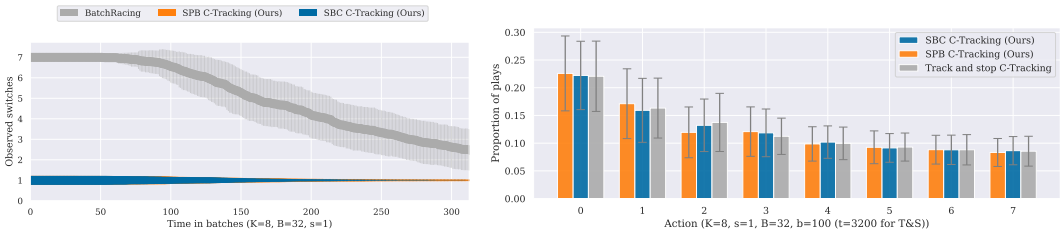
5.1 Experimental Setup

The simulation setting presented comprises a set of 8 arms from Gaussian distributions, with means $\mu = \{0.8, 0.65, 0.6, 0.55, 0.5, 0.45, 0.4, 0.35\}$ and standard deviation $\sigma = 1$. The algorithms (SBC and SPB C-Tracking, BatchRacing and Track-and-Stop C-Tracking [10]) are run with $\delta = 0.01$, and a pull limit of 20,000. We run experiments to investigate the effect of varying a switching limit s in SPB C-Tracking. In these, we *set* the number of switches $S^b = s$ for all batches b, rather than let them be bounded by s from above. We also compare the effect of the batch size in SPB C-Tracking and the BatchRacing baseline. We use the stopping time, $\mathbb{E}[\tau] = B\mathbb{E}[\beta]$, as the evaluation metric and we also compare the stopping times of the batched algorithms to the stopping time of the un-batched Track-and-Stop C-Tracking. All experiments are done for 500 repetitions and results are presented with means and standard deviation across runs.

5.2 Results and Discussion

Limiting switching and optimality. In Figure 2a, thicker points in the horizon indicate that more of the repeated runs have not stopped until that point. We see that both our algorithms achieve faster stopping compared to BatchRacing, even when constrained to a minimal switching limit ($s = 1$). We see that the BatchRacing algorithm starts with the maximum possible switches in early batches and eventually decreases in later batches, until $s = 1$. Successive elimination algorithms, including BatchRacing, which comprise the bulk of the limited work in batched bandits in pure exploration [16, 1, 18, 6] will always exhibit this switching behaviour. These algorithms are expected to have a high number of switches during exploration, as they always start with the whole set of arms as the feasible exploration set.

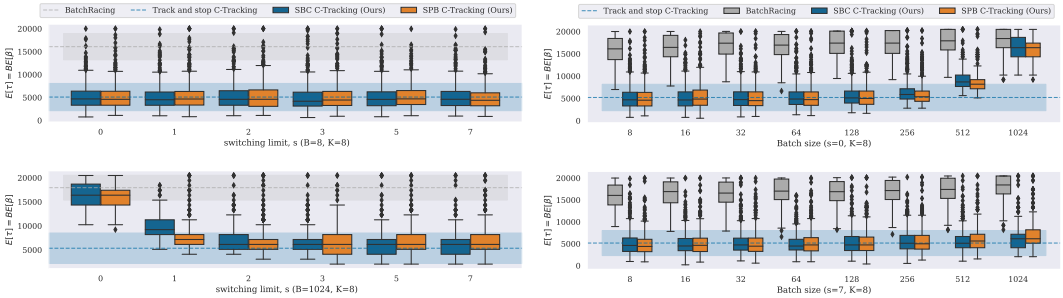
The stopping times in Figure 3a show that, as expected, SBC and SPB C-Tracking always outperform the BatchRacing baseline. This can be explained by the algorithms' characteristic of tracking the optimal playing proportions from the lower bound, Eq. (9). The stopping times for batch



(a) Observed switches and stopping, along time in batches for SBC and SPB C-Tracking (Ours) with $s = 1, B = 32$ vs BatchRacing (Baseline).

(b) Proportions of arm plays for SBC and SPB (Ours) after 100 batches (3200 plays, with $B = 32$, $s = 1$) and C-Tracking after 3200 plays.

Figure 2: SBC and SPB stop quicker even with a restrictive switching limit and match well to the *optimal, unbatched* Track-and-Stop C-tracking baseline in tracking proportions.



(a) Comparison of stopping times over switching limits $s \in \{0, 1, 2, 3, 5, 7\}$ in SBC and SPB, and BatchRacing, with $B \in \{8, 1024\}$.

(b) Comparison of stopping times over batch sizes for SBC and SPB C-Tracking ($s \in \{0, 7\}$), and BatchRacing, with $B \in \{8, 16, 32, 64, 128, 256, 512, 1024\}$.

Figure 3: SBC and SPB only suffer with highly restrictive switching and large batch sizes

sizes $B \leq 256$ also match those of the unbatched (standard) Track-and-Stop C-Tracking which is unconstrained in switching, so it switches almost every play during exploration. The matching in stopping times is consistent across different switching limits for the same batch size (Figure 3a). The proportions of arm plays also align closely (Figure 2b), which aligns with our theoretical result in Theorem 2.

In Figures 3a and 3b, we also see that restricting arm switching hurts exploration only when the batch size is large, for restrictive switching limits. Furthermore, with less restriction in switching ($s = K - 1$), our results provide evidence that it is possible to batch and still achieve comparable stopping times with un-batched optimal pure exploration algorithms like Track and Stop, even with large batch sizes. These results are expanded in Appendix F with more combinations of batch size B, and switching limit s. A simulation with 16 arms is also included, and the results are consistent.

Algorithms batch filling characteristics. In Appendix F with Examples 1 and 2, we discuss the differences in SBC and SPB, which are in batch filling (*greedily for SBC* and *proportionally for SPB*) and illustrate that in Appendix Figures 4 and 5.

6 Discussion

We have studied the problem of controlling arm-switching in fixed-confidence pure-exploration and from our results, we learn that it is possible to impose a switching constraint for exploration by using batching, and then restricting how often arms can be switched inside the batches. We derived a lower bound (Theorem 1) for this, by considering feasible batch configurations respecting switching constraints. While our lower bound can theoretically provide algorithms that explicitly track the proportions p_c of the configurations, these proportions are defined for entire batch configurations, $\mathcal{C}^K_{B,s}$, preventing us from applying an exact tracking rule. However, we observe (Observation 1) that we can still leverage the core tracking ideas for algorithm design by constructing optimal batch configurations. Constructing configurations turns out to be computationally feasible, as it reduces to a polynomial-time problem.

We further presented the batched algorithms SBC and SPB C-Tracking that constrain the switching frequency by playing constructed optimal batch configurations. Our algorithms empirically show that it is possible to achieve efficient exploration even when constraining the arm-switching frequency, and they perform well except under extreme conditions, specifically when the batch size is large and the switching constraint is stringent. We also provided an upper bound (Theorem 2) matching our lower bound, but an open question from our result is whether we can derive a tighter upper bound that explicitly accounts for switching. A limitation of our approach is that the batch size of our algorithms is not given by the problem, but is only a means to control switching. We use a static batch size that may not be optimal for efficient exploration and whose selection is not obvious, as the optimal choice is problem-dependent, so adaptively tuning the batch size is an interesting direction for future work.

Bibliography

[1] Arpit Agarwal, Shivani Agarwal, Sepehr Assadi, and Sanjeev Khanna. Learning with limited rounds of adaptivity: Coin tossing, multi-armed bandits, and ranking from pairwise comparisons. In *Conference on Learning Theory*, pages 39–75. PMLR, 2017.

[2] Idan Amir, Guy Azov, Tomer Koren, and Roi Livni. Better best of both worlds bounds for bandits with switching costs. *Advances in neural information processing systems*, 35:15800–15810, 2022.

[3] Raman Arora, Ofer Dekel, and Ambuj Tewari. Online bandit learning against an adaptive adversary: from regret to policy regret. *arXiv preprint arXiv:1206.6400*, 2012.

[4] Maryam Aziz, Emilie Kaufmann, and Marie-Karelle Riviere. On multi-armed bandit designs for dose-finding trials. *Journal of Machine Learning Research*, 22(14):1–38, 2021.

[5] Sébastien Bubeck, Rémi Munos, and Gilles Stoltz. Pure exploration in multi-armed bandits problems. In *Algorithmic Learning Theory: 20th International Conference, ALT 2009, Porto, Portugal, October 3-5, 2009. Proceedings 20*, pages 23–37. Springer, 2009.

[6] Shengyu Cao, Simai He, Ruoqing Jiang, Jin Xu, and Hongsong Yuan. Best arm identification in batched multi-armed bandit problems. *arXiv preprint arXiv:2312.13875*, 2023.

[7] Herman Chernoff. Sequential design of experiments. *The Annals of Mathematical Statistics*, 30 (3):755–770, 1959. ISSN 00034851.

[8] Ofer Dekel, Jian Ding, Tomer Koren, and Yuval Peres. Bandits with switching costs: T 2/3 regret. In *Proceedings of the forty-sixth annual ACM symposium on Theory of computing*, pages 459–467, 2014.

[9] Zijun Gao, Yanjun Han, Zhimei Ren, and Zhengqing Zhou. Batched multi-armed bandits problem. *Advances in Neural Information Processing Systems*, 32, 2019.

[10] Aurélien Garivier and Emilie Kaufmann. Optimal best arm identification with fixed confidence. In Vitaly Feldman, Alexander Rakhlin, and Ohad Shamir, editors, *29th Annual Conference on Learning Theory*, volume 49 of *Proceedings of Machine Learning Research*, pages 998–1027, Columbia University, New York, New York, USA, 23–26 Jun 2016. PMLR.

[11] J. Gittens and Michael Dempster. Bandit processes and dynamic allocation indices. *Journal of the Royal Statistical Society. Series B: Methodological*, 41:148–177, 02 1979.

[12] Chester Holtz, Chao Tao, and Guangyu Xi. BanditPyLib: a lightweight python library for bandit algorithms. Online at: `https://github.com/Alanthink/banditpylib`, 2020. URL `https://github.com/Alanthink/banditpylib`. Documentation at `https://alanthink.github.io/banditpylib-doc`.

[13] Kevin Jamieson, Matthew Malloy, Robert Nowak, and Sébastien Bubeck. lil'ucb: An optimal exploration algorithm for multi-armed bandits. In *Conference on Learning Theory*, pages 423–439. PMLR, 2014.

[14] Tianyuan Jin, Jing Tang, Pan Xu, Keke Huang, Xiaokui Xiao, and Quanquan Gu. Almost optimal anytime algorithm for batched multi-armed bandits. In *International Conference on Machine Learning*, pages 5065–5073. PMLR, 2021.

[15] Marc Jourdan, Mojmír Mutnỳ, Johannes Kirschner, and Andreas Krause. Efficient pure exploration for combinatorial bandits with semi-bandit feedback. In *Algorithmic Learning Theory*, pages 805–849. PMLR, 2021.

[16] Kwang-Sung Jun, Kevin Jamieson, Robert Nowak, and Xiaojin Zhu. Top arm identification in multi-armed bandits with batch arm pulls. In *Artificial Intelligence and Statistics*, pages 139–148. PMLR, 2016.

[17] Newton Mwai Kinyanjui, Emil Carlsson, and Fredrik D Johansson. Fast treatment personalization with latent bandits in fixed-confidence pure exploration. *Transactions on Machine Learning Research*, 2023.

[18] Junpei Komiyama, Kaito Ariu, Masahiro Kato, and Chao Qin. Optimal simple regret in bayesian best arm identification. *arXiv preprint arXiv:2111.09885*, 2021.

[19] Anastasios Kyrillidis, Stephen Becker, Volkan Cevher, and Christoph Koch. Sparse projections onto the simplex. In *International Conference on Machine Learning*, pages 235–243. PMLR, 2013.

[20] T.L Lai and H Robbins. Asymptotically efficient adaptive allocation rules. *Advances in Applied Mathematics*, 6:4–22, 1985.

[21] Tor Lattimore and Csaba Szepesvári. *Bandit algorithms*. Cambridge University Press, 2020.

[22] Lihong Li, Wei Chu, John Langford, and Robert E Schapire. A contextual-bandit approach to personalized news article recommendation. In *Proceedings of the 19th international conference on World wide web*, pages 661–670, 2010.

[23] Yingying Li, James A Preiss, Na Li, Yiheng Lin, Adam Wierman, and Jeff S Shamma. Online switching control with stability and regret guarantees. In *Learning for Dynamics and Control Conference*, pages 1138–1151. PMLR, 2023.

[24] Herbert Robbins. Some aspects of the sequential design of experiments. *Bulletin of the American Mathematical Society*, 58(5):527–535, 1952.

[25] Chloé Rouyer, Yevgeny Seldin, and Nicolo Cesa-Bianchi. An algorithm for stochastic and adversarial bandits with switching costs. In *International Conference on Machine Learning*, pages 9127–9135. PMLR, 2021.

[26] William R. Thompson. On the likelihood that one unknown probability exceeds another in view of the evidence of two samples. *Biometrika*, 25(3/4):285–294, 1933. ISSN 00063444.

Appendix

Table 1: Table of commonly used notation

Symbol	Description
K	Number of actions/arms
$\mathcal{A}$	Set of arms, $\mathcal{A} = \{1, ..., K\}$
a	A single arm, $a \in \mathcal{A}$
μ_a	Expected reward of arm a
a^*	Optimal arm, $a^* = \arg\max_a \mu_a$
μ^*	Expected reward of optimal arm, a^*
τ	Stopping time (in number of arm plays)
$\hat{a}_\tau$	Recommended arm at stopping time
S_τ	The number of arm switches until time τ
α	Constraint on the arm switching rate
δ	Confidence parameter, $\delta \in (0, 1)$
B	Batch size, $B \geq 1$
β	Stopping batch (in number of batches)
S^b	Number of arm switches inside batch b
s	In-batch arm switching limit
T^*	Characteristic time
Σ^K	Simplex over K arms, $\Sigma^K = \{w \in \mathbb{R}^K : \sum_a w_a = 1\}$
w_a	Arm playing proportion for arm a
λ_a	Alternative bandit model
$\text{Alt}(\mu)$	Set of alternative bandit models with optimal arm that differs from that of μ
c	Configuration of a single batch, $c = [c_1, ..., c_K]^\top$
$\mathcal{C}_{B,s}^K$	Set of configurations of size as B integer plays of K arms, limited to s switches
$N_a(\beta)$	Number of plays of arm a until batch β
$\Sigma^{\mathcal{C}}$	Simplex over elements in set $\mathcal{C}$
$\hat{\mu}$	Estimated arm parameters
$\bar{w}(b)$	Desired (cumulative) arm proportions at batch b
$d_a(b)$	Play deficit for arm a at batch b
$w^\epsilon(\mu)$	Proportions L_∞-projected onto simplex where each element has weight at least ϵ
$\tilde{c}$	Chosen batch configuration

A Proof of Theorem 1

Proof of Theorem 1: Consider $\delta \in (0, 1)$ and a bandit model $\mu \in \mathbb{R}^K$, along with a δ-PAC strategy. For each block $b \geq 1$, let $N_a(b)$ represent the number of times arm a is drawn up to the end of block b. According to Garivier and Kaufmann [10, Lemma 1], the expected number of draws for each arm and the Kullback-Leibler divergence between two bandit models with distinct optimal arms are related to the error probability δ:

$$\forall \lambda \in \mathbb{R}^K \ : \ a^*(\lambda) \neq a^*(\mu),$$
$$\sum_{a=1}^{K} \mathbb{E}_\nu[N_a(\beta)] d(\mu_a, \lambda_a) \geq \text{kl}(\delta, 1-\delta).$$

Rather than selecting a specific λ for each arm a to provide a lower bound on $\mathbb{E}_\mu[\beta]$, we integrate the inequalities from all alternative λs:

$$\text{kl}(\delta, 1-\delta) \leq \inf_{\lambda \in \text{Alt}(\mu)} \sum_{a=1}^{K} \mathbb{E}_\mu[N_a(\beta)] d(\mu_a, \lambda_a)$$

Let $\mathcal{C}_{B,s}^K$ be the available integer playing configurations for plays corresponding to the desired sparsity and $\Sigma^{\mathcal{C}} \coloneqq \Sigma_1^{|\mathcal{C}_{B,s}^K|}$ be the simplex over sparse batch configurations. Then, we have

$$\mathrm{kl}(\delta, 1-\delta) \leq \inf_{\lambda \in \mathrm{Alt}(\mu)} \sum_{a=1}^{K} \mathbb{E}_\mu \left[\sum_{b=1}^{\beta} \sum_{c \in \mathcal{C}_{B,s}^K} c_{a,b} \right] d(\mu_a, \lambda_a) = \inf_{\lambda \in \mathrm{Alt}(\mu)} \sum_{a=1}^{K} \mathbb{E}_\mu[\beta] \mathbb{E}_\mu \left[\sum_{c \in \mathcal{C}_{B,s}^K} c_a \right] d(\mu_a, \lambda_a)$$

$$\leq \mathbb{E}_\mu[\beta] \sup_{p \in \Sigma^{\mathcal{C}}} \inf_{\lambda \in \mathrm{Alt}(\mu)} \sum_{a=1}^{K} \sum_{c \in \mathcal{C}_{B,s}^K} p_c c_a \, d(\mu_a, \lambda_a),$$

where the last inequality arises because the probabilities of arm draws specific to each batch are less than or equal to their maximum values. This substitution is made to derive a bound that applies to any δ-PAC algorithm. ■

B An upper bound on the expected stopping time of the tracking algorithms

We repeat Theorem 2 below.

Theorem 3. *Let $\mu \in \mathbb{R}^K, B \geq 1, s \in [B-1]$ be an instance of the switch-constrained pure exploration problem with confidence $\delta > 0$. Let $\alpha \in [1, e/2]$ and $\rho(t) = O(t^\alpha)$. Using Algorithm 1 with any selection rule that solves Eq. (11), and Chernoff's stopping rule with threshold $\log(\rho(t)/\delta)$,*

$$\limsup_{\delta \to 0} \frac{\mathbb{E}[\beta_\delta]}{\log 1/\delta} \leq \frac{\alpha}{B} T^*(\mu) \, .$$

where T^* is the characteristic time in the non-batched setting, Eq. (4).

To prove this result, our strategy will be to show that, despite planning arm plays in batches, we will track the optimal proportions for the non-batched settings. To do this, we need to generalize two key lemmas due to Garivier and Kaufmann [10] for the batched setting. First, we will show that playing according to a selection rule like that of Algorithm 1 maintains an error that is upper bounded by a constant w.r.t. the batch index. Second, we use this result to show that the tracking rule ensures that the deviation of historical plays from the optimal playing proportion is bounded. At this point, we have established all we need to follow the remainder of the proof of Theorem 14 in Garivier and Kaufmann [10].

B.1 Proportion Tracking with Batch Plays and a Switching Limit

Lemma 1. *Let K and B be positive integers, and let Σ_K be the simplex of dimension $K-1$. For each arm $a \in \{1, \ldots, K\}$, define the expected cumulative number of plays after batch b as $BP_a(b)$ where for every $b \leq n$, $P_a(b) = p_a(1) + \ldots + p_a(b)$. Let $N_a(b)$ denote the actual cumulative number of plays after batch b.*

At each batch b, choose the batch configuration $\tilde{c}_{b+1} \in \mathcal{C}_{B,s}^K$ (a feasible configuration of arm plays that respects the batch size B and switching limit s) such that:

$$\tilde{c}_{b+1} = \arg\min_{c \in \mathcal{C}_{B,s}^K} \sum_{a=1}^{K} \left(B \sum_{i=0}^{b-1} p_a(i) - N_a(b) - c(a) \right)_+ ,$$

where $\sum_{i=0}^{b-1} p_a(i)$ represents the cumulative weight estimate for arm a up to batch $b-1$.

Then, the maximum deviation between the actual and expected number of plays for any arm is bounded as:

$$\max_{1 \leq a \leq K} |N_a(b) - BP_a(b)| \leq B(K-1).$$

Proof. We intend to prove the same bound for the two edge cases, which is when $s = 0$, implying that $\tilde{c}_{b+1}$ contains only one arm played B times, or $s \geq B-1$, meaning that any number of switches is allowed. Since the bounds are the same, the intermediate cases $s \in (0, B-1)$ follow.

First, we prove by induction on b that:

$$\max_{1\le a\le K} N_a(b) - BP_a(b) \le B.$$

At batch $b = 0$, no plays have been made, so $N_a(0) = 0$ for all arms a, and $P_a(0) = 0$ for all arms a. Therefore, the base case is trivially satisfied:

$$\max_{1\le a\le K} N_a(0) - BP_a(0) = 0.$$

Assume that this holds for some $b \ge 0$. Then, for $a \notin \tilde{c}_{b+1}$, $N_a(b+1) - BP_a(b+1) = N_a(b) - B(P_a(b) + p(b)) \le B(1 - p_a(b)) \le B$ and for $a \in \tilde{c}_{b+1}$, $N_{a\in\tilde{c}_{b+1}}(b+1) - BP_{a\in\tilde{c}_{b+1}}(b+1) = \tilde{c}_{b+1}(a) + (N_{a\in\tilde{c}_{b+1}}(b) - BP_{a\in\tilde{c}_{b+1}}(b+1))$. Using the fact that that $\sum_a BP_a(b+1) - N_a(b) = 0$, we know that, for any selected arm $a \in \tilde{c}_{b+1}$, $N_{a\in\tilde{c}_{b+1}}(b) - BP_{a\in\tilde{c}_{b+1}}(b+1) \le 0$ or there was a non-selected arm $a \notin \tilde{c}_{b+1}$ that had a smaller value in the criterion. But if such an arm exists, it would have been selected in place of a. This is true whether only one arm can be selected ($s = 0$) or any arm can be selected ($s = B-1$). Hence, for all $a \in \tilde{c}_{b+1}$, $\tilde{c}_{b+1}(a) + (N_{a\in\tilde{c}_{b+1}}(b) - BP_{a\in\tilde{c}_{b+1}}(b+1)) \le B + 0 \le B$.

It follows that for all terms:

$$\begin{aligned}\max_{1\le a\le K} |N_a(b) - BP_a(b)| &= \max\left\{\max_{1\le a\le K} BP_a(b) - N_a(b), \max_{1\le a\le K} N_a(b) - BP_a(b)\right\} \\ &\le \max\left\{\sum_{a=1}^{K} (BP_a(b) - N_a(b))_+, B\right\}\end{aligned}$$

To complete the proof, we introduce the auxiliary variable, for every $b \in \{1, ..., n\}$,

$$r_b = \sum_{a=1}^{K} (BP_a(b) - N_a(b))_+ .$$

and prove by induction on b that

$$r_b \le B(K-1).$$

To start, we note that the base case $b = 1$ holds trivially. Next, we'll assume that the statement holds for some $b > 1$ and prove that $r_{b+1} \le B(K-1)$. First, recall that the plays in batch $b+1$ are given by the selection rule

$$\tilde{c}_{b+1} = \operatorname*{arg\,min}_{c\in\mathcal{C}^K_{B,s}} \sum_{a=1}^{K} (BP_a(b+1) - N_a(b) - c(a))_+ .$$

We'll use the short-hand $c = \tilde{c}_{b+1}$ for the remainder of this proof.

Define the play deficit for arm a at time $b+1$ as $d_a \coloneqq BP_a(b+1) - N_a(b+1)$. We can separate the terms in the sum making up r_{b+1} as follows,

$$r_{b+1} = \sum_{a:c(a)=0} (d_a)_+ + \sum_{\substack{a:c(a)>0\\ d_a>0}} d_a + \sum_{\substack{a:c(a)>0\\ d_a\le 0}} 0 .$$

For convenience, let c_+ represent the terms in the second sum, $c_+ = \{a \in [K] : c(a) > 0, BP_a(b+1) - N_a(b+1) > 0\}$ and c_- the terms in the third, $c_- = \{a \in [K] : c(a) > 0, BP_a(b+1) - N_a(b+1) < 0\}$. We also use the convention $a \in c \Leftrightarrow c(a) > 0$. We will consider two cases, one where $c_- = \emptyset$ and one where it is not.

Case I: $c_- = \emptyset \Leftrightarrow \forall a \in c : BP_a(b+1) - N_a(b+1) \geq 0$

In this case, all played arms have a remaining deficit, i.e., they have not yet caught up to the expected number of plays, $BP_{a\in c}(b+1)$. We can re-write r_{b+1} as a function of r_b as follows.

$$\begin{aligned} r_{b+1} &\leq r_b + \sum_{a=1}^{K} \left[Bp_a(b+1) - c(a)\right] \mathbb{1}_{(BP_a(b+1) \geq N_a(b+1))} \\ &= r_b + \sum_{a=1}^{K} p_a(b+1)\mathbb{1}_{(BP_a(b+1) \geq N_a(b+1))} - \sum_{a\in c} c(a)\mathbb{1}_{(BP_{a\in c}(b+1) \geq N_{a\in c}(b+1))} \\ &\leq r_b + B - \sum_{a=1}^{K} c(a) = r_b + B - B = r_b \leq B(K-1)\ . \end{aligned}$$

The second to last inequality holds because, by assumption $c = c_+$ and $c(a) = 0$ for all $a \notin c$.

In the general case, some played arms may have no remaining deficit, and some do.

Case II: $|c_-| > 0 \Leftrightarrow \exists a \in c : BP_a(b+1) - N_a(b+1) < 0$

In this case, the number of plays $N_a(b+1)$ for at least one played arm $a \in c$ has surpassed the expected number of plays $BP_a(b+1)$. This implies that there is no longer a positive deficit for the arm being played.

By construction, we can write r_{b+1} as follows.

$$r_{b+1} = \sum_{a\notin c} (BP_a(b+1) - N_a(b))_+ + \sum_{a\in c_+} \left(BP_a(b+1) - N_a(b) - c(a)\right)\ .$$

For any arm not selected, $a \notin c$, the deficit before selection, $BP_a(b+1) - N_a(b)$, must have been smaller than the pre-selection deficit for any selected arm, including those in c_-. This holds true in both edge cases, $s = 0$ and $s = B-1$. Otherwise, a play for arm a would have been selected for the batch instead of a play for an arm in c_-. Thus,

$$\sum_{a\notin c} (BP_a(b+1) - N_a(b))_+ \leq \sum_{a\notin c} \min_{a'\in c_-} \left(BP_{a'}(b+1) - N_{a'}(b)\right)_+\ .$$

Further, for any arm $a' \in c_-$, $BP_{a'}(b+1) - N_{a'}(b+1) = BP_{a'}(b+1) - N_{a'}(b) - c(a') \leq 0$ by definition, and so $BP_{a'}(b+1) - N_{a'}(b) \leq c_{a'} \leq B$. Thus,

$$\sum_{a\notin c} (BP_a(b+1) - N_a(b))_+ \leq \sum_{a\notin c} B\ .$$

A similar argument can be used for any arm in c_+. The remaining deficit after deciding on the plays must be smaller for arms in c_+ than what the deficit would have been if an arm play was moved from c_+ to an arm in c_-,

$$\begin{aligned} \forall a \in c_+ : BP_a(b+1) - N_a(b) - c_a &\leq \min_{a'\in c_-} BP_{a'}(b+1) - N_{a'}(b) - c_{a'} + 1 \\ &\leq \min_{a'\in c_-} c_{a'} - c_{a'} + 1 = 1 \end{aligned}$$

Thus, whenever $|c_-| > 0$, $r_{b+1} \leq \sum_{a\notin c} B + \sum_{a\in c_+} 1 \leq B(K-1)$. This last inequality is loose in general, but if only one arm is played and no played arm have remaining deficit, it is tight. ■

B.2 Tracking rule convergence

Lemma 2. *For any batch $b \geq 1$ and $a \in \mathcal{A}$, any tracking rule that is optimal with respect to Eq. (11) (the problem in Lemma 1) ensures that $N_a(b) \geq B(\sqrt{bB + K^2} - 2K + 1)$ and that*

$$\max_{1\leq a\leq K} \left| N_a(b) - B\sum_{i=0}^{b-1} w_a^*(\hat{\mu}_i) \right| \leq B(K-1).$$

where w^* are the solution to Eq. (9).

To obtain Lemma 2, we start by applying Lemma 1 with the batch configuration and weights $p(b) = w^{\epsilon_{b-1}}(\hat{\mu}(b-1))$, so that

$$P(b+1) = \sum_{i=0}^{b} w^{\epsilon_i}(\hat{\mu}_i).$$

This gives us the following deviation bound for the number of plays after batch b:

$$\max_{1\le a\le K} \left| N_a(b) - B\sum_{i=0}^{b-1} w_a^{\epsilon_i}(\hat{\mu}_i) \right| \le B(K-1).$$

Moreover, by the definition of $w^{\epsilon}(i)$, we can express the difference between the weighted actual plays and the optimal allocation as:

$$\max_{1\le a\le K} \left| B\sum_{i=0}^{b-1} w_a^{\epsilon_i}(\hat{\mu}_i) - B\sum_{i=0}^{b-1} w_a^{*}(\hat{\mu}_i) \right| \le BK\epsilon_i.$$

Now, with $\epsilon_b = (K^2 + bB)^{-1/2}/2$, we get

$$\sqrt{bB+K^2} - K = \int_0^{bB} \frac{ds}{2\sqrt{K^2+i}} \le \sum_{i=0}^{bB-1} \epsilon_i \le \int_{-1}^{bB-1} \frac{ds}{2\sqrt{K^2+i}} = \sqrt{bB+K^2-1} - \sqrt{K^2-1}.$$

This yields the following bound on the deviation:

$$\max_{1\le a\le K} \left| N_a(b) - B\sum_{i=0}^{b-1} w_a^{*}(\hat{\mu}_i) \right| \le B(K-1) + BK\left(\sqrt{bB+K^2-1} - \sqrt{K^2-1}\right) \le BK(1+\sqrt{bB}).$$

We now derive the lower bound for $N_a(b)$. From the previous results, it follows that:

$$N_a(b) \ge B\sum_{i=0}^{b-1} \epsilon_i - B(K-1).$$

Using the integral approximation for $\sum_{i=0}^{b-1} \epsilon_i$, we obtain:

$$N_a(b) \ge B\left(\sqrt{bB+K^2} - K\right) - B(K-1).$$

Simplifying this gives:

$$N_a(b) \ge B(\sqrt{bB+K^2} - 2K + 1)$$

We can further apply the techniques from Lemma 20 in Garivier and Kaufmann [10]. Let $h(\beta) = \beta^{1/4}$. For all $b > \sqrt{\beta}$ and all arms a, with ϵ the C-tracking constant used in the L_∞ projection.

$$\begin{aligned}
\left|\frac{N_a(b)}{bB} - w_a^*(\mu)\right| &\le \left|\frac{N_a(b)}{bB} - \frac{1}{b}\sum_{i=0}^{b-1} w_a^*(\hat{\mu}_i)\right| + \left|\frac{1}{b}\sum_{i=0}^{b-1} w_a^*(\hat{\mu}_i) - w_a^*(\mu)\right| \\
&\le \frac{BK(1+\sqrt{bB})}{bB} + \frac{h(\beta)}{b} + \frac{1}{b}\sum_{i=h(\beta)}^{b-1} |w_a^*(\hat{\mu}_i) - w_a^*(\mu)| \\
&\le \frac{2K+1}{\beta^{1/4}} + \epsilon\,.
\end{aligned}$$

Thus, for any $\beta \ge ((2K+1)/(2\epsilon))^4$,

$$\left|\frac{N_a(b)}{bB} - w_a^*(\mu)\right| \le 3\epsilon\,.$$

Note that this adds a factor B to the lower bound on the corresponding stopping time, $T = B\beta$.

■

B.3 Prooof of Theorem 2

Once Lemmas 1–2 have been established, we can exploit the proof of Theorem 14 in Garivier and Kaufmann [10] to show that any tracking rule that select batches that solve Eq. (11) satisfies the statement in Theorem 2. Proposition 1 shows that SBC-C tracking solves Eq. (11).

Here, Lemma 2 takes the role of "Lemma 7" in their case and the other key component, "Lemma 19" applies directly also in our setting. The rest of the proof follows the same steps as the proof of Theorem 14. In the end, terms that are constant w.r.t. δ vanish in the division $1/\delta$ when $\delta \to 0$. This includes the waiting time for the played proportions to converge to the tracked proportions, discussed in the previous result. Applying Theorem 14 yields a bound on the stopping time in *number of plays* $\mathbb{E}[\tau_\delta]$, which immediately yields a stopping time on the number of batches $\mathbb{E}[\beta_\delta] = \mathbb{E}[\tau_\delta]/B$ since the batch size is fixed to B and stopping is only performed when completing a full batch. Finally, when there exists $p^* \in \Sigma^{\mathcal{C}^K_{B,s}}$ such that $\sum_{c \in \mathcal{C}^K_{B,s}} p^*_c c_a = w^*(\mu)$, it holds that $T^*(\mu) = BT^*_{bc}(\mu)$, where $T^*(\mu)$ is the characteristic time (number of arm plays) in the non-batched case Eq. (4), since the optimal arm allocations are feasible to construct from batch configurations.

■

C A greedy algorithm for constructing configurations

The problem in Eq. (11) can be abstracted to the following form. Given demands $d_a \in \mathbb{R}$ for $a = 1, .., K$, a batch size $B \in \mathbb{N}$ and a switching constraint $s \in \{0, ..., B-1\}$, the sparse batch deficit minimization problem is,

$$\begin{aligned} \underset{c \in \mathbb{N}^K}{\text{minimize}} \quad & \sum_a (d_a - c_a)_+ \\ \text{subject to} \quad & \sum_{a=1}^K c_a = B \\ & \|c\|_0 \leq s + 1 \end{aligned} \tag{14}$$

We now give a greedy algorithm to solve Eq. (14) and prove that it is optimal.

Proposition 1. *Algorithm 2 returns an optimal solution to Eq. (14).*

Proof. Each round of Phase I of the algorithm allocates a play that results in a reduction of 1 in the overall deficit or moves on to the next arm. The arm is changed only if the remaining deficit for the current arm is smaller than 1 and the switching limit has not yet been reached. It is easy to see that, after Phase 1, any arms a that are assigned at least one play, $c_a \geq 1$, will have deficit $d_a - c_a \in [0, 1)$ or the batch limit has been filled. It is also easy to see that any unplayed arm, $c_a = 0$, has a smaller initial demand d_a than any played arm or there were no arms with a positive demand. If this were true, any configuration would be optimal.

We can now argue for the result by contradiction. Assume that the solution is suboptimal. That means that there exists at least one arm a' such that increasing $c_{a'}$ by 1 and reducing c_a for a selected arm a results in a better solution. Since the total deficit removed by the batch can be expressed as the sum of deficit removed with each play, it is sufficient to consider one such change at a time.

Case (i) $c_{a'} = 0$. If a' was not played, increasing $c_{a'}$ would increase $\|c\|_0$ which would require removing all plays of a played arm a if the switching constraint was active. If the switching constraint was *not* active, Phase 1 terminated with a non-active constraint as well. This means that either (i.i) all plays in the batch were already allocated or (i.ii) all arms had already been examined and a' was ignored. In case (i.i), there would be no use substituting a' for any played arm since any selected plays remove a deficit of 1 per round, and the total removed for any arm is at least as large as what could be removed for a', due to the initial sorting. In case (i.ii), arm a' must have had a total deficit $d_{a'}$ smaller than 1, or the arm would have been added. But since $c_{a'} = 0$ also after the entire algorithm, a' must have been ignored also in Phase II. But if the switching constraint was not active, this must mean that other arms had larger remaining deficits or a' would have been selected.

If the switching constraint *was* active, it could have been activated either in Phase I or Phase II. If it happened in Phase II, a' must have had $d_{a'} < 1$ since, otherwise, it would have been selected

Algorithm 2 Greedy batch filling

Input K arms, B : batch size, s : batch switch limit, $d \in \mathbb{R}^K$: arm demands

Phase I: Removing integer deficit
Create an index $\{a_i\}$ of arms ordered by d in descending order such that $i < j \Rightarrow d_{a_i} \geq d_{a_j}$ for all $i, j \in [K]$.
Initialize the batch configuration $c = [0, ..., 0]^\top \in \mathbb{N}^K$.
Set $i = 1$
while $\sum_{a=1}^K c_a < B$ and $i \leq K$ **do**
    if $d_{a_i} - c_{a_i} \geq 1$ **then**
        $c_{a_i} = c_{a_i} + 1$
    else if $\|c\|_0 < s + 1$ **then**
        $i = i + 1$
    else
        Break
    end if
end while

Phase II: Removing remaining fractional deficit
Sort $d - c$ in descending order such that $i < j \Rightarrow d_{a_i} - c_{a_i} \geq d_{a_j} - c_{a_j}$ for all $i, j \in [K]$.
Set $i = 1$
while $\sum_{a=1}^K c_a < B$ and $i \leq K$ **do**
    if $(d_{a_i} - c_{a_i} \geq 0)$ and $(c_{a_i} > 0$ or $\|c\|_0 < s + 1)$ **then**
        $c_{a_i} = c_{a_i} + 1$
        $d_{a_i} = d_{a_i} - 1$
    else
        $i = i + 1$
    end if
end while
Distribute any remaining plays arbitrarily among previously played arms. *(E.g. Greedily onto the arm with the largest fractional deficit in the selected arms)*

in Phase I. But this must mean that $d_{a'}$ was smaller than the demand for an arm that activated the constraint in Phase II. If the constraint was made active in Phase I, there must have been an arm a with at least as large demand $d_a \geq d_{a'}$ that was added instead. Switching a' for any such a could not reduce the remaining deficit.

Case (ii) $c_{a'} > 0$. If arm a' was already allocated plays, its allocation could be increased without concern for the switching constraint. By construction, interchanging plays between selected arms during Phase I makes no difference to the total removed deficit. This means that the allocation must change during Phase II to improve the objective. But since Phase II proceeds in order of remaining deficit, either a' was assigned an additional play in Phase II, in which case increasing $c_{a'}$ further would not reduce the total deficit since any deficit in Phase II is < 1, or Phase II terminated before increasing $c_{a'}$. Since $c_{a'} > 0$, this can only happen if other arms given allocations in Phase II had higher remaining deficit than a'.

In summary, there is no arm a' for which an increased allocation, at the cost of reducing the allocation for another arm a, would reduce the total deficit more than the solution by Algorithm 2. ■

D Sparse-Projected Batch (SPB) Algorithm for Constructing Configurations

Let $d \in \mathbb{R}_+^K$ denote the vector of deficits for K arms, where $d_a \geq 0$ for $a = 1, \ldots, K$. Given a batch size $B \in \mathbb{N}$ and a switching constraint $s \in \{0, \ldots, \min(K-1, B-1)\}$, we consider the following batch configuration problem:

Choose

$$c \in \mathbb{N}^K \quad \text{such that} \quad \sum_{a=1}^{K} c_a = B \quad \text{and} \quad \|c\|_0 \le s+1,$$

so as to minimize the total remaining (positive) deficit

$$D(c) = \sum_{a=1}^{K} \Big(d_a - c_a\Big)_+,$$

or equivalently, to maximize the total deficit removal

$$R(c) = \sum_{a=1}^{K} \min\{c_a, d_a\}.$$

The *Sparse-Projected Batch (SPB)* algorithm proceeds as follows:

1. **Compute the Normalized Positive Deficits:**

$$\bar{d} = \frac{(d)_+}{\sum_{a=1}^{K}(d_a)_+}.$$

2. **Project onto the $(s+1)$-Sparse Simplex:**

$$\Sigma_{s+1}^{K} = \Big\{ w \in \mathbb{R}_+^K : \sum_{a=1}^{K} w_a = 1,\ \|w\|_0 \le s+1 \Big\}.$$

That is, solve

$$\hat{w}^{s+1} \in \arg\min_{w \in \Sigma_{s+1}^K} \|w - \bar{d}\|_2. \tag{15}$$

The GSSP algorithm [19] showed that this projection can be performed by *selecting the largest $s+1$ arms in $\bar{d}$* and then re-normalizing.

3. **Form the Continuous Configuration:**

$$\hat{c} = B\,\hat{w}^{s+1}.$$

4. **Round to an Integer Configuration:** Obtain $\tilde{c} \in \mathbb{N}^K$ from $\hat{c}$ via the rounding procedure described in Appendix E.

Next, we give a results that when combined with Lemma 1 shows that SPB C-tracking attains the upper bound of Theorem 2.

Proposition 2. *Let $d \in \mathbb{R}_+^K$ be the vector of arm deficits, and let $B \in \mathbb{N}$ and $s \in \{0, \dots, \min(K-1, B-1)\}$ be the batch size and switching limit, respectively. Then the SPB algorithm—which selects the continuous configuration $\hat{c} = B\hat{w}^{s+1}$ with $\hat{w}^{s+1}$ given by Eq. (15) and rounds it as described in Appendix E—returns a batch configuration $\tilde{c}$ that minimizes*

$$D(c) = \sum_{a=1}^{K} \Big(d_a - c_a\Big)_+$$

over all $c \in \mathbb{N}^K$ satisfying $\sum_{a=1}^{K} c_a = B$ and $\|c\|_0 \le s+1$.

Proof. We now prove Proposition 2 showing that the SPB algorithm minimizes Eq. (14) and thus enjoys the tracking guarantee.

Consider the batch configuration construction problem at a given batch b, where we select

$$c \in \mathcal{C}_{B,s}^{K} = \{c \in \mathbb{N}^K : \sum_{a=1}^{K} c_a = B,\ \|c\|_0 \le s+1\}$$

to minimize

$$D(c) = \sum_{a=1}^{K} \Big(d_a - c_a \Big)_+ .$$

Any algorithm that minimizes $D(c)$ automatically satisfies the tracking guarantee required by Theorem 2. We now show that the SPB algorithm minimizes $D(c)$ by the following steps.

Step 1. Optimality via Maximum Deficit Removal.

Minimizing

$$D(c) = \sum_{a=1}^{K} \Big(d_a - c_a \Big)_+$$

is equivalent to maximizing the total deficit removal

$$R(c) = \sum_{a=1}^{K} \min\{c_a, d_a\} ,$$

since

$$D(c) = \sum_{a=1}^{K} d_a - R(c).$$

This is because $\sum_{a=1}^{K} d_a$ is independent of the allocation c, so minimizing $D(c)$ is equivalent to maximizing $R(c)$:

$$\arg\min_c D(c) = \arg\max_c R(c).$$

Hence, an allocation that removes the largest fraction of the total positive deficit $\sum_{a=1}^{K} (d_a)_+$ is optimal for Eq. (14).

Step 2. Decomposition into Integer and Fractional Deficit Removal. For each arm a, decompose the deficit as

$$d_a = \lfloor d_a \rfloor + \{d_a\} ,$$

where $0 \le \{d_a\} < 1$. Any play allocated to arm a first removes one full unit until $c_a = \lfloor d_a \rfloor$ plays are allocated; additional plays then remove only the fractional part $\{d_a\}$. Therefore, if two configurations remove the same number of integer deficit units, they are equivalent in terms of integer deficit removal. We call a configuration *integer-optimal* if it removes the maximum possible number of integer deficit units. For such configurations, the only difference in total deficit removal comes from the removal of the remaining (fractional) deficit.

Step 3. Maximizing Integer Deficit Removal. Since each play on arm a reduces the deficit by one unit until $c_a = \lfloor d_a \rfloor$ plays are allocated, any configuration that does not allocate plays to the arms with the largest deficits cannot remove more integer deficit.

We now prove by contradiction that any configuration which does not allocate all B plays solely to the arms with the largest integer parts of the deficits (i.e. the top $s+1$ arms in terms of $\lfloor d_a \rfloor$) cannot be integer-optimal.

Suppose, for the sake of contradiction, that there exists a configuration

$$c \in \mathbb{N}^K \quad \text{with} \quad \sum_{a=1}^{K} c_a = B \quad \text{and} \quad \|c\|_0 \le s+1,$$

which is integer-optimal (i.e. it maximizes

$$\sum_{a=1}^{K} \min\{c_a, \lfloor d_a \rfloor\}$$

over all feasible configurations) but which does *not* allocate all B plays to the top $s+1$ arms. This means there exists an arm j that is *not* among the top $s+1$ arms (with respect to $\lfloor d_a \rfloor$) for which $c_j > 0$.

If the configuration uses at most $s+1$ arms and arm j is not among the top $s+1$ arms, there must exist an arm i among the top $s+1$ arms for which either c_i is less than its capacity to remove integer deficit (i.e. $c_i < \lfloor d_i \rfloor$) or even $c_i = 0$.

Now, define a new configuration c' by transferring one play from arm j to arm i:

$$c'_i = c_i + 1, \quad c'_j = c_j - 1, \quad c'_a = c_a \quad \text{for all } a \notin \{i, j\}.$$

Because $c_i < \lfloor d_i \rfloor$, we have

$$\min\{c'_i, \lfloor d_i \rfloor\} = c_i + 1 > c_i = \min\{c_i, \lfloor d_i \rfloor\}.$$

For arm j, note that reducing c_j by one unit cannot increase the term $\min\{c_j, \lfloor d_j \rfloor\}$; indeed, if $c_j \leq \lfloor d_j \rfloor$ then the removal decreases by one unit, and if $c_j > \lfloor d_j \rfloor$ the term remains $\lfloor d_j \rfloor$. Therefore,

$$\min\{c'_j, \lfloor d_j \rfloor\} \leq \min\{c_j, \lfloor d_j \rfloor\}.$$

Consequently, the total integer deficit removal in configuration c' satisfies

$$\sum_{a=1}^{K} \min\{c'_a, \lfloor d_a \rfloor\} > \sum_{a=1}^{K} \min\{c_a, \lfloor d_a \rfloor\}.$$

This contradicts the assumption that c was integer-optimal.

Thus, by contradiction, any integer-optimal configuration must allocate all B plays to a subset of at most $s+1$ arms with the highest $\lfloor d_a \rfloor$ values. By design, the SPB algorithm selects these top $s+1$ deficit arms, thereby maximizing the number of integer deficit units removed.

Step 4. Maximizing Fractional Deficit Removal among Integer-Optimal Configurations. Once the integer deficit is maximized, the remaining difference among configurations is solely due to the removal of the fractional parts $\{d_a\}$. Since $\{d_a\} < 1$ for every arm, the optimal strategy is to allocate any extra plays in decreasing order of the fractional deficit.

Via the rounding procedure in Appendix E, the SPB algorithm produces an integer configuration that allocates any remaining plays in decreasing order of the fractional remainders. Therefore, among all integer-optimal configurations, the SPB-selected configuration removes the maximum possible fractional deficit.

Step 5. Concluding Optimality. Combining Steps 3 and 4, we conclude that the SPB algorithm:

(i) Selects an allocation that is integer-optimal (i.e., it removes the maximum number of integer deficit units by playing only the top $s+1$ arms).

(ii) Among integer-optimal configurations, allocates any extra plays so as to maximize the removal of the fractional deficits.

Thus, any configuration different from the SPB-selected configuration must either remove fewer integer deficit units or (if integer-optimal) remove a smaller sum of fractional deficits. In either case, the total remaining deficit $D(c)$ would be larger than that achieved by the SPB algorithm.

Since minimizing $D(c)$ is equivalent to obtaining the tracking guarantee (and any algorithm minimizing Eq. (14) attains the same upper bound as the SBC algorithm), we conclude that the SPB algorithm minimizes $D(c)$ and enjoys the same upper bound.

Conclusion. In summary, we have shown that:

- Any batch configuration that removes the largest fraction of the total positive deficit is optimal with respect to Eq. (14).
- Among configurations that remove the same number of integer deficit units (i.e., among integer-optimal configurations), the one that removes the maximum fractional deficit is best.
- The SPB algorithm, by selecting the top $s+1$ arms and using a rounding procedure that allocates extra plays in decreasing order of fractional deficit, simultaneously maximizes both the integer and fractional deficit removal.

Thus, the SPB algorithm minimizes $D(c)$ and consequently enjoys the tracking guarantee as stated in Theorem 2. ■

E Rounding Procedure

To convert the continuous solution $\hat{c} = B \cdot \hat{w}^{s+1}$ (with $\hat{w}^{s+1} \in \Sigma_{s+1}^K$) into a valid integer allocation $\tilde{c} \in \mathbb{N}^K$, we proceed as follows:

1. **Flooring:** For each arm a, set
$$\tilde{c}_a = \lfloor \hat{c}_a \rfloor .$$
2. **Computing Remainders:** For each arm a, compute the fractional remainder
$$f_a = \hat{c}_a - \lfloor \hat{c}_a \rfloor .$$
3. **Allocating Remaining Plays:** Let
$$R = B - \sum_{a=1}^{K} \tilde{c}_a .$$
Distribute the remaining R plays to the arms in decreasing order of f_a (breaking ties arbitrarily).

F Appendix: Additional experimental results

Table 2 contains the number of configurations for 8 arms, batch size $B = 128$ for various values of the switching constraint s.

s	At most s switches	Exactly s switches
7	138432467745	89356415775
6	49076051970	41355035400
5	7721016570	7118489700
4	602526870	578739000
3	23787870	23336250
2	451620	448056
1	3564	3556
0	8	8

Table 2: The number of configurations $\left|\mathcal{C}_{B,s}^K\right|$ for different s values $K = 8$, $B = 128$.

K	At most s switches	Exactly s switches
4	366145	333375
8	23787870	23336250
16	611238316	606742500
32	12027912984	11988165000
64	212152083760	211818474000
128	3559176881760	3556444500000

Table 3: The number of configurations $\left|\mathcal{C}_{B,s}^K\right|$ for different K values $s = 3$, $B = 128$.

Example 1: Tracking characteristics Consider an illustrative example with $K = 8, B = 64$ and $s = 2$ where until batch $b = 10$, both algorithms have accumulated the same number of arm plays and deficits (rounded):

$$\bar{w}(b)/B = [1.6, 1.7, 1.6, 1.8, 1.1, 1.1, 1.1, 1.1]^\top$$
$$N(b) = [86, 99, 85, 136, 54, 52, 64, 64]^\top$$
$$d(b) = [15.6, 9.1, 17.9, -18.7, 14.5, 16.5, 4.5, 4.5]^\top$$

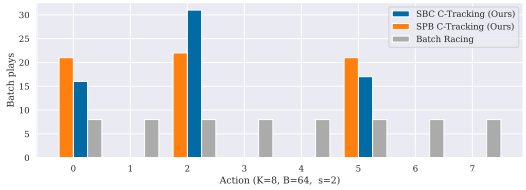


Figure 4: Illustrative example showing arm selection for SPB and SBC starting from the same accumulated arm plays $N(b)$ and desired proportions $\bar{w}(b)$ after $b = 10$ batches. BatchRacing included for reference.

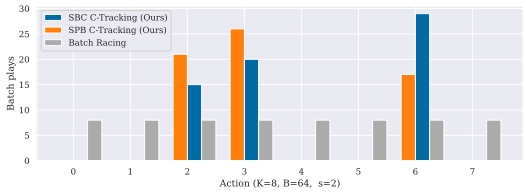


Figure 5: Illustrative example comparing arm selection for SPB and SBC starting from the same accumulated arm plays $N(b)$ and desired proportions $\bar{w}(b)$ after $b = 10$ batches. BatchRacing included for reference.

In deciding the next plays, SBC and SPB respectively yield the following configurations, see Figure 4:

$$\text{SPB: } \tilde{c} = [21, 0, 22, 0, 0, 21, 0, 0]^\top$$
$$\text{SBC: } \tilde{c} = [16, 0, 31, 0, 0, 17, 0, 0]^\top$$

Both SBC and SPB select the arms with the largest deficit to be included in the next batch (both are *integer-optimal*, see Appendix D). SBC, with the greedy batch filling subroutine (Appendix C), yields configurations where the remaining batch allocation after removing integer deficits is allocated greedily onto the arm with the largest fractional deficit in the selected arms. For SPB, the batch is filled proportionally to the deficits in the selected arms. BatchRacing is also shown in Figure 4 with uniform plays across all arms, from the round-robin procedure given that all arms are still in the feasible set. See Appendix F, Example 2 for another example.

Example 2: Tracking characteristics Consider an illustrative example with $K = 8, B = 64$ and $s = 2$ where until a batch $b = 10$, both algorithms have accumulated the same number of arm plays, as well as the deficits:
$\sum_{i=0}^{b-1} w^{\epsilon_i}(\hat{\mu}_i) = [2.918, 1.643, 1.696, 1.082, 0.708, 0.894, 1.291, 0.769]^\top$,
$N(b) = [177, 103, 94, 50, 43, 61, 72, 40]^\top$,
$d(b) = B \sum_{i=0}^{b-1} w^{\epsilon_i}(\hat{\mu}_i) - N(b) = [9.752, 2.152, 14.544, 19.248, 2.312, -3.784, 10.624, 9.216]^\top$.

In deciding the next plays, SBC and SPB respectively yield the following configurations, also shown in Figure 5:
SPB: $\tilde{c} = [0, 0, 21, 26, 0, 0, 17, 0]^\top$
SBC: $\tilde{c} = [0, 0, 15, 20, 0, 0, 29, 0]^\top$

F.1 Additional results

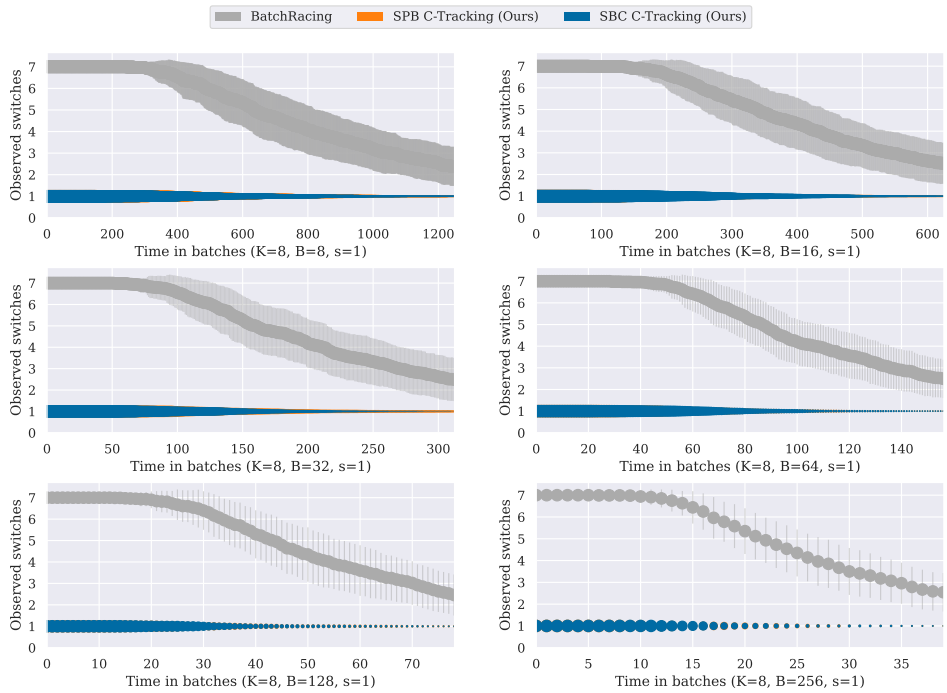


Figure 6: Comparison trace of observed switches along time in batches for SBC and SPB C-Tracking (Ours) vs BatchRacing (Baseline) with $s = 1$ for different batch sizes $B \in \{8, 16, 32, 64, 128, 256\}$

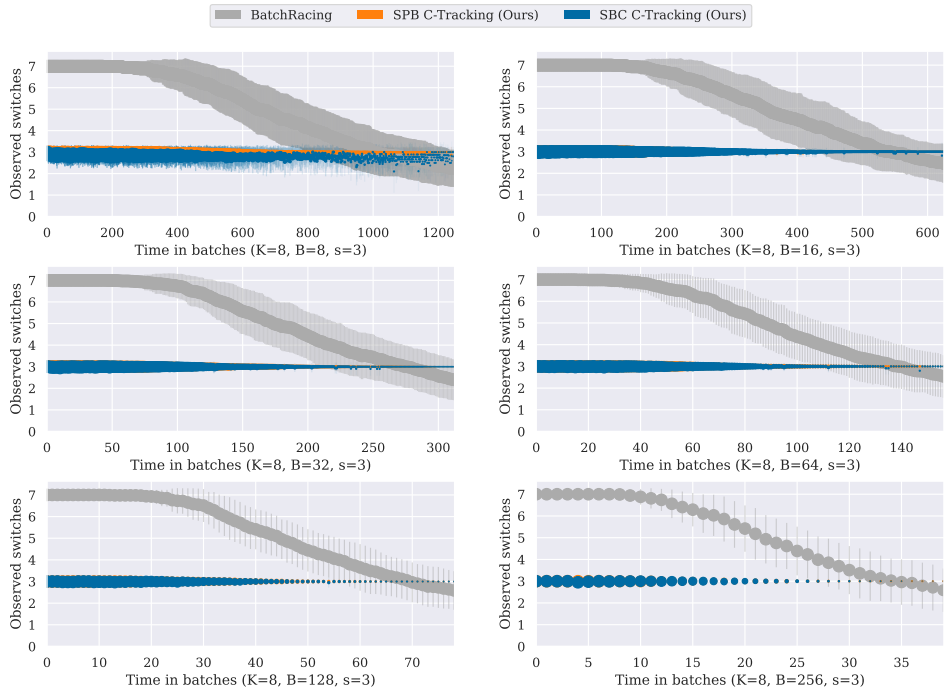


Figure 7: Comparison trace of observed switches along time in batches for SBC and SPB C-Tracking (Ours) vs BatchRacing (Baseline) with $s = 3$ for different batch sizes $B \in \{8, 16, 32, 64, 128, 256\}$

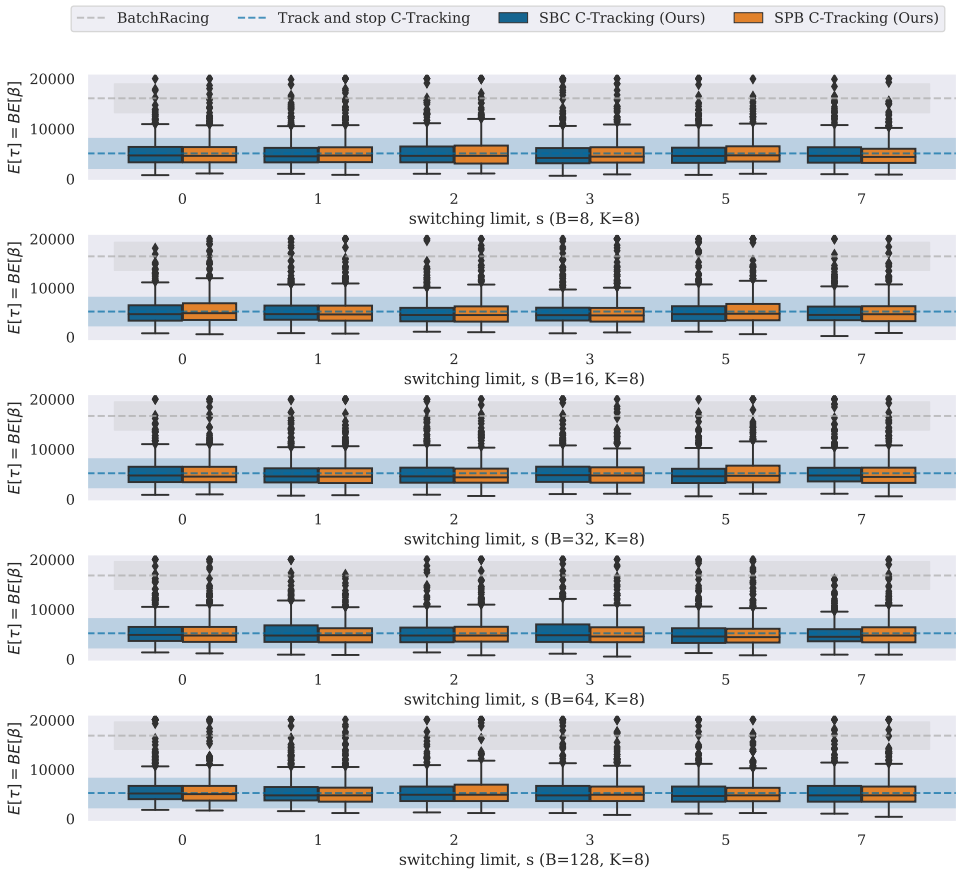


Figure 8: Comparison of stopping times over switching limits $s \in \{0, 1, 2, 3, 5, 7\}$ in SPB C-Tracking (Ours) with different batch sizes $B \in \{8, 16, 32, 64128\}$. Track-and-stop C-tracking is not batched.

F.2 A second set of 16 arms

A second simulation setting comprises a set of 16 arms from Gaussian distributions, with means $\mu = \{0.80, 0.69, 0.66, 0.62, 0.59, 0.55, 0.52, 0.48, 0.45, 0.41, 0.38, 0.34, 0.31, 0.27, 0.24, 0.20\}$ and standard deviation $\sigma = 0.3$. The results are consistent with the first simulation and are shown in Figures 10, 11 and 12.

G Code

All code in Python for the algorithms implementation and reproducing graphs is included in the supplemental files as zip file. The algorithm modules were written as extension modules to Bandit-PyLib by Holtz et al. [12] and they can be found in the $banditpylib/learners/mab_fcbai_learner$ directory. Notebooks for running the experiments with Slurm and for reproducing the graphs are also included in the $examples/$ folder.



Figure 9: Effect of batch size on the stopping times for BatchRacing and SPB C-Tracking $s \in \{0, 1, 3, 5, 7\}$, $B \in \{8, 16, 32, 64, 128, 256, 512, 1024\}$.

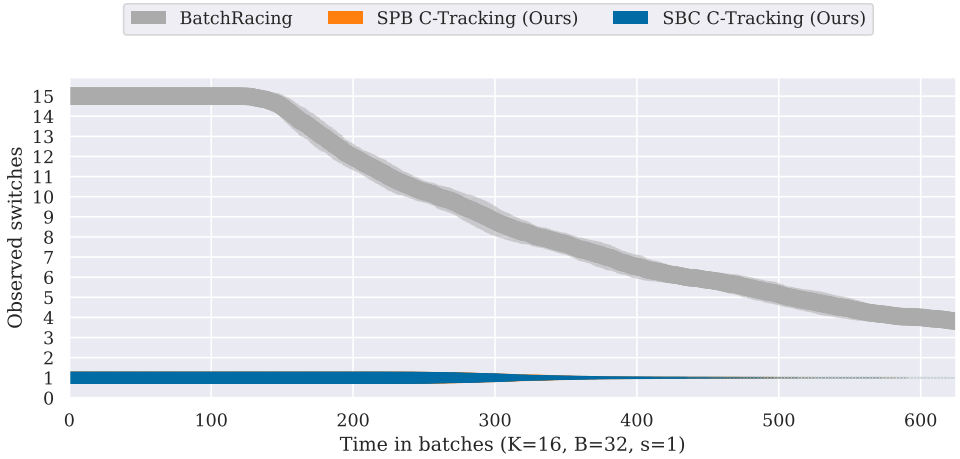


Figure 10: Comparison trace of observed switches along time in batches for SBC and SPB C-Tracking (Ours) with $s = 1, B = 32$ vs BatchRacing (Baseline).

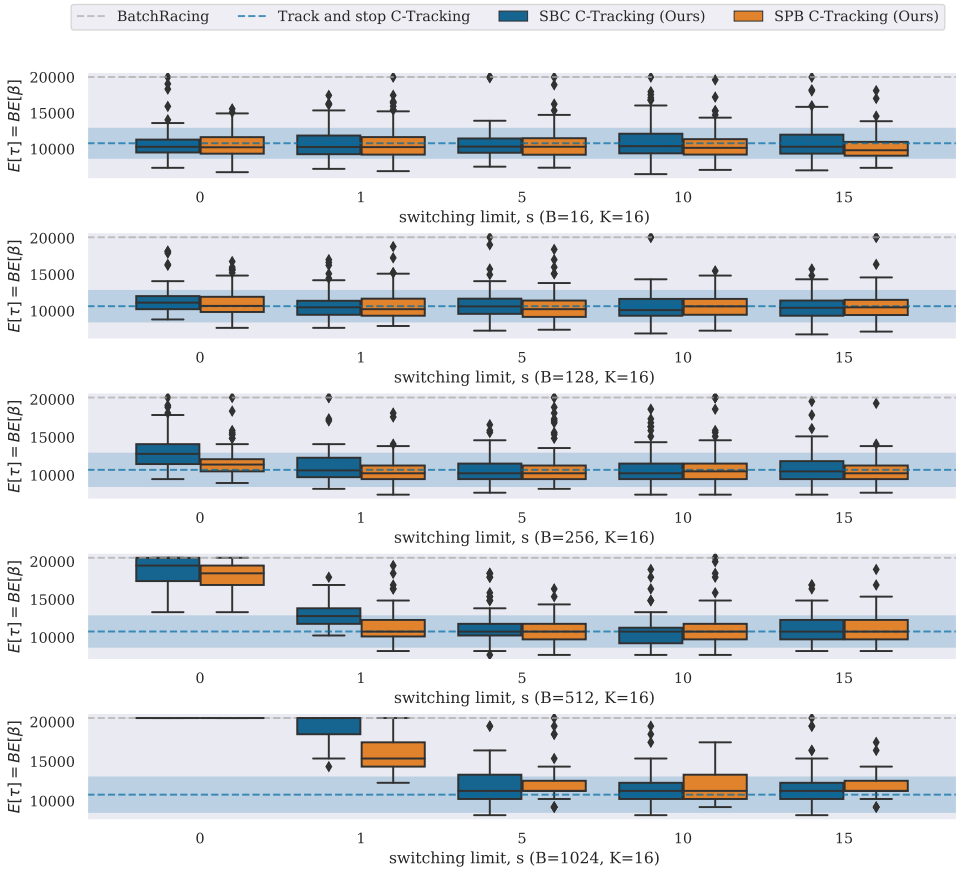


Figure 11: Comparison of stopping times over switching limits $s \in \{0, 1, 5, 10, 15\}$ in SPB C-Tracking (Ours) with different batch sizes $B \in \{16, 128, 256, 512, 1024\}$. Track-and-stop C-tracking is not batched.

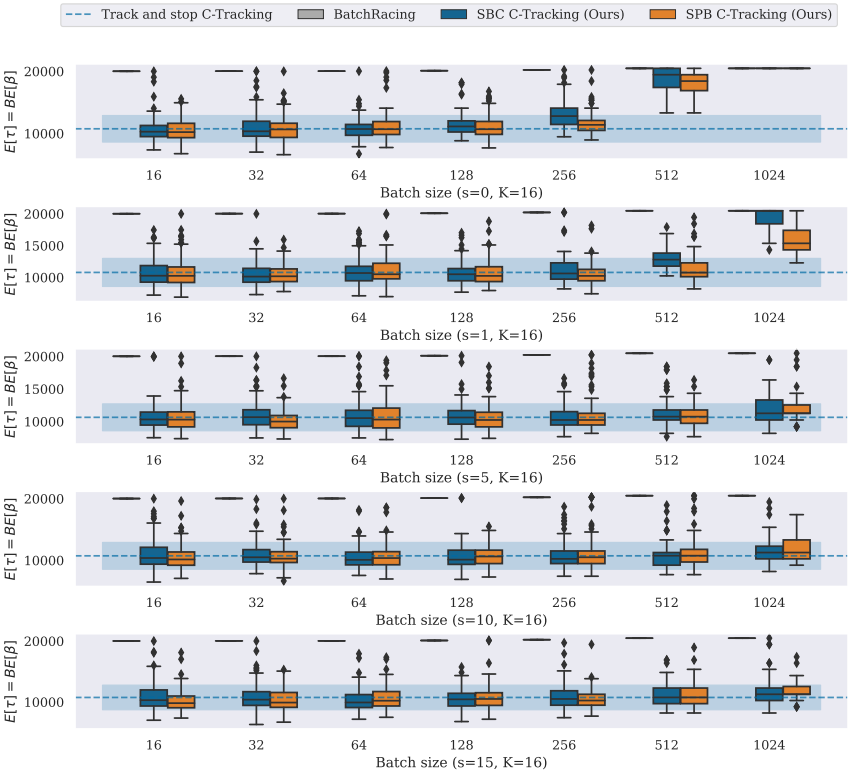


Figure 12: Effect of batch size on the stopping times for BatchRacing and SPB C-Tracking $s \in \{0, 1, 5, 10, 15\}$, $B \in \{16, 32, 64, 128, 256, 512, 1024\}$.