Fast Pure Exploration via Frank-Wolfe

Anonymous Author(s)
Affiliation
Address
email

Abstract

We study the problem of active pure exploration with fixed confidence in generic stochastic bandit environments. The goal of the learner is to answer a query about the environment with a given level of certainty while minimizing her sampling budget. For this problem, instance-specific lower bounds on the expected sample complexity reveal the optimal proportions of arm draws an Oracle algorithm would apply. These proportions solve an optimization problem whose tractability strongly depends on the structural properties of the environment, but may be instrumental in the design of efficient learning algorithms. We devise Best Challenger (BC), a simple algorithm whose sample complexity matches the lower bounds for a wide class of pure exploration problems. The algorithm is computationally efficient as, to learn and track the optimal proportion of arm draws, it relies on a single iteration of Frank-Wolfe algorithm applied to the lower-bound optimization problem. We apply BC to various pure exploration tasks, including best arm identification in unstructured, thresholded, linear, and Lipschitz bandits. Despite its simplicity, BC outperforms existing algorithms.

1 Introduction

Pure exploration in stochastic bandits [25] refers to the task of answering a given question about the reward distributions of the different arms, using as few arm pulls (or samples) as possible. The task may correspond to identifying the best arm [13], the top-m arms [38], all ϵ-good arms [28], a set of arms whose expected rewards exceed a given threshold [27], etc. To reduce the sample complexity of such a task, the learner needs to leverage as much as possible the information available about reward distributions, which typically comes as known structural properties of the mapping of arms to their expected rewards. Exploiting particular structures (e.g., unimodal, Lipschitz, convex, linear) has been thoroughly studied in the regret minimization setting (see [6], and references therein), but less in the pure exploration framework, where most efforts have focused on linear structures [36, 20, 40, 37, 10, 18, 9].

In this paper, we investigate a generic learning problem proposed in [8] and covering the aforementioned pure exploration tasks with or without structure. Consider K arms whose reward distributions $(\nu_1, \dots, \nu_K)$ come from a one-dimensional exponential family and are of unknown means $\boldsymbol{\mu} = (\mu_1, \dots, \mu_K)$. The parameter $\boldsymbol{\mu}$ is known to belong to $\Lambda \subset \mathbb{R}^K$, the set of possible instances. For each $\boldsymbol{\mu} \in \Lambda$, we assume that there is a unique true answer $i^\star(\boldsymbol{\mu})$ that belongs to the finite set $\mathcal{I}$ of possible answers[1] (e.g., for the best arm identification task, $i^\star(\boldsymbol{\mu}) = \arg\max_k \mu_k$). We consider pure exploration tasks in the *fixed confidence* setting where the learner wishes, for any possible $\boldsymbol{\mu} \in \Lambda$, to discover $i^\star(\boldsymbol{\mu})$ with a certain level of confidence $1 - \delta$, for some $\delta \in (0, 1)$. The learner's strategy is defined by (i) an adaptive sampling rule dictating the sequence of arm pulls, (ii) a stopping rule defining τ, the round where, based on the data gathered so far, the learner decides to stop pulling

[1] Scenarios with several correct answers require a more involved analysis, see [7].

Submitted to 35th Conference on Neural Information Processing Systems (NeurIPS 2021). Do not distribute.

arms, and (iii) a decision rule specifying her answer. The goal is to devise a δ-PAC (it outputs the right answer with probability at least $1-\delta$ for any $\boldsymbol{\mu} \in \Lambda$) strategy minimizing the expected sample complexity $\mathbb{E}_{\boldsymbol{\mu}}[\tau]$.

Using the same arguments as those used in [13] for classical MAB problems, we may derive a lower bound of the expected sample complexity satisfied by any δ-PAC strategy. This lower bound, whose proof can be found in Appendix B for completeness, is given by $T^\star(\boldsymbol{\mu})\mathrm{kl}(\delta, 1-\delta)$, where the characteristic time $T^\star(\boldsymbol{\mu})$ is defined through the following optimization problem:

$$T^\star(\boldsymbol{\mu})^{-1} = \sup_{\boldsymbol{\omega} \in \Sigma} \inf_{\boldsymbol{\lambda} \in \mathrm{Alt}(\boldsymbol{\mu})} \sum_{k=1}^{K} \omega_k d(\mu_k, \lambda_k), \tag{1}$$

where Σ is the $(K-1)$-dimensional simplex, $\mathrm{Alt}(\boldsymbol{\mu})$ is the set of *confusing* parameters $\boldsymbol{\lambda} \in \Lambda$ such that $i^\star(\boldsymbol{\mu}) \neq i^\star(\boldsymbol{\lambda})$, $\mathrm{kl}(a, b)$ is the KL divergence between two Bernoulli distributions of means a and b, and $d(\mu_k, \lambda_k)$ denotes the KL divergence of arm-k reward distributions under prameters $\boldsymbol{\mu}$ and $\boldsymbol{\lambda}$. A solution $\omega^\star(\boldsymbol{\mu})$ of (1) can be interpreted as an optimal *allocation*, in the sense that pulling each arm i a proportion of round equal to $\omega_i^\star(\boldsymbol{\mu})$ (in expectation) constitutes an optimal sampling rule.

Most existing algorithms achieving an asymptotically (when δ goes to 0) minimal sample complexity leverage a Track-and-Stop (TaS) framework [13]. In each round t, they plug $\hat{\boldsymbol{\mu}}(t)$ the estimated expected arm rewards in the lower bound optimization problem (1), and track the allocation $w^\star(\hat{\boldsymbol{\mu}}(t))$. As already noticed in [29], the main drawback of the Track-and-Stop framework is that it requires a recurrent access to an Oracle able to solve (1) (actually existing analyses usually assume that the Oracle outputs the exact solution for any $\boldsymbol{\mu}$). (1) is a concave program but can become difficult to solve depending the underlying structure Λ. Indeed, for complex structures, identifying the most confusing parameters leading to the objective function $\inf_{\boldsymbol{\lambda} \in \mathrm{Alt}(\boldsymbol{\mu})} \sum_{k=1}^{K} \omega_k d(\mu_k, \lambda_k)$ can be hard.

Contributions. 1) Instead of solving (1) in each round as in the TaS framework, we propose an online iterative method to approach the optimal allocation of arm pulls. Specifically, we devise Best Challenger (BC), a computationally efficient algorithm that just relies, in each round, on a single iteration Frank-Wolfe (FW) algorithm applied to (1) instantiated at $\hat{\boldsymbol{\mu}}(t)$.
2) For a wide class of pure exploration problems with or without structure, we derive an upper bound of the expected sample complexity of BC for any certainty level δ, and show that this bound matches the lower bound $T^\star(\boldsymbol{\mu})\mathrm{kl}(\delta, 1-\delta)$ asymptotically as δ goes to 0.
3) We illustrate the performance of BC on various pure exploration problems, including best arm identification in unstructured, linear, and Lipschitz bandits. In all tested scenarios, and despite its simplicity, BC outperforms existing algorithms.

The use of the FW algorithm has been suggested in [13] in the case of best arm identification problem in unstructured bandits. In this case, FW iterations take a very simple and intuitive form (see Example 1 introduced in §3). The corresponding sampling rule is referred to as Best Challenger in [13] (hence the name of our algorithm), and leads to algorithms with remarkably low sample complexity empirically – sometimes lower than that of TaS algorithms solving (1) in each round. So far however, as discussed in [29], the analysis of FW-based sampling rules, and even their convergence, have eluded researchers. Towards the design of the BC algorithm, we devise a simple variant of the FW algorithm that yields a sampling rule whose sample complexity can be analyzed. We confirm the asymptotic optimality of BC as well as its empirical superiority, not only for the case of best arm identification in unstructured bandits as predicted by [13], but also for a wide class of pure exploration problems. We believe that our analysis also brings interesting solutions to the three important obstacles we needed to tackle to devise and analzye a FW-based sampling rule: (i) the objective function in (1) is not smooth; (ii) its curvature becomes infinite in general close the boundary of Σ; and (iii) the estimate $\hat{\boldsymbol{\mu}}(t)$ is evolving and might be far from $\boldsymbol{\mu}$.

2 Related work

Best Arm Identification (BAI) has recently received a lot of attention, either in unstructured bandit problems, see [13, 34], or in problems with various kinds of structure, e.g., linear [36, 20, 40, 37, 10, 18, 9, 32], combinatorial [24, 19, 33], spectral [22], monotone [14], cascading [42]. For BAI in unstructured bandits with fixed confidence, [13] developed the celebrated Track-and-Stop framework leading to algorithms able to asymptotically converge towards the optimal allocation of arm draws,

and in turn, to achieve the lowest sample complexity possible in the high confidence regime (as δ goes to 0). It is possible to apply the TaS framework to specific structures, as this was proposed in [18] for linear bandits. However, for more involved structures, this might become computationally too difficult. Indeed TaS requires the learner to repeatedly solve the optimization problem (1).

The authors of [8] propose and exploit an interpretation of the lower bound optimization problem (1) as the solution of a 2-players game – the $\boldsymbol{\omega}$-player playing the 'sup' and the $\boldsymbol{\lambda}$-player playing the 'inf'. The algorithm presented in [8] combines two zero-regret algorithms applied sequentially by the two players, and converge to an optimal allocation. Interestingly, the algorithm uses the optimism-in-front-of-uncertainty principle to remove the need of forced exploration (the $\boldsymbol{\omega}$-player is fed with upper-confidence bounds on her rewards). As shown later, the algorithm does not perform as well as BC. The applicability of the framework used in [8] remains unclear to us: in [9] and in [19], the authors claim that the framework cannot be applied to linear and combinatorial bandits, respectively.

In [29], the author proposes a solution close to ours. His algorithm, LMA (Lazy Miror Ascent), just runs in each round one iteration of a sub-gradient ascent algorithm applied to (1). Fortunately, the projection step usually involved in such algorithm is simple. Numerically, as illustrated later in the paper, we found that LMA may not be as efficient as TaS or BC. We could try to explain this by remarking that LMA has similarities with the Exponential Weights algorithm (see Appendix F in [29]), an algorithm designed for adversarial online optimization problem, and may be too conservative in a stochastic setting.

As already mentioned in the introduction, Frank-Wolfe-based algorithms for BAI in unstructured bandits have been mentioned first in [13] for their simplicity and good performance. Applying FW as if the objective function was smooth may fail at converging [29] experimentally. We believe that we manage to make, in our algorithm, the minimal modification of the FW algorithm so that convergence and asymptotic optimality are guaranteed. Finally note that [2] uses FW in a regret minimization problem but with a smooth objective function.

We conclude this section by mentioning existing works on the FW algorithm when applied to optimizing non-smooth functions. The proposed solutions consist by either smoothing objective function or enlarging the set of differential (this is the second approach we chose). [11, 15] apply FW on the randomly smoothed surrogate instead of the original non-smooth objective. However, computing the gradient at each iteration requires to query many time on the objective function, which may not be practical. [1, 30] use a proximal operator to replace the objective function, but as pointed out in [4], the smoothing parameters of the proximal operator are not trivial to tune. Our solution is close to those developed in [31, 4]. There, inspired by the approximate subdifferential [39], the authors propose to collect the set of the gradients in the neighborhood at each round. They show that these collection is continuous even when the objective functions is non-smooth, which allows for the use of FW. The way we deal with the non-smoothness issue is similar but simplified by the fact that the specific form of our objective function.

3 Preliminaries

We consider the pure exploration task described in the introduction. This section presents the additional assumptions made towards the design and analysis of our algorithm. These assumptions are here illustrated for the classical Best Arm Identification (BAI) task in unstructured bandits (see Example 1); they will be verified for all other examples of pure exploration problems presented in Section 5. This section also provides useful properties of the lower bound optimization problem (1), and finally describes our choice of stopping and decision rules.

3.1 Assumptions and properties of the lower bound optimization problem

The answer map $i^\star : \Lambda \to \mathcal{I}$ allows us to decompose Λ into a union of non-overlapping sets: $\Lambda = \cup_{i\in\mathcal{I}}\mathcal{S}_i$, where $\mathcal{S}_i = \{\boldsymbol{\mu} \in \Lambda : i^\star(\boldsymbol{\mu}) = i\}$ for all $i \in \mathcal{I}$. The answer map is known (i.e., knowing $\boldsymbol{\mu}$ is enough to output the right answer), and hence without loss of generality, we can assume that $\mathcal{S}_i \neq \emptyset$ for all $i \in \mathcal{I}$. Using this notation, the set of confusing parameters can be written as $\mathrm{Alt}(\boldsymbol{\mu}) = \cup_{i\neq i^\star(\boldsymbol{\mu})}\mathcal{S}_i$.

Assumption 1. *For each $i \in \mathcal{I}$, $\mathcal{S}_i$ is an open set and the complementary of $\mathcal{S}_i$ is a finite union of convex sets. Namely, there exists a finite collection $\mathcal{J}_i$ of convex sets $\mathcal{C}_j^i$ s.t. $\Lambda \setminus \mathcal{S}_i = \cup_{j \in \mathcal{J}_i} \mathcal{C}_j^i$.*

Example 1. The BAI task in unstructured bandits with Bernoulli rewards. For this task, we have $\Lambda = (0,1)^K, \mathcal{I} = \{1, \dots, K\}$, and for all arm i, the set of parameters for which arm i is the best arm is $\mathcal{S}_i = \{\boldsymbol{\mu} \in \Lambda : \mu_i > \mu_k, \forall k \neq i\}$. We have: $\Lambda \setminus \mathcal{S}_i = \cup_{j \in \mathcal{J}_i} \mathcal{C}_j^i$ where $\mathcal{J}_i = \mathcal{I} \setminus \{i\}$ is the set of arms different than i and $\mathcal{C}_j^i = \{\boldsymbol{\mu} \in \Lambda : \mu_j > \mu_i\}$ is the convex set of parameters for which arm j is better than arm i. □

Now under Assumption 1, we can decompose the lower bound optimization problem as follows: $T^\star(\boldsymbol{\mu})^{-1} = \sup_{\boldsymbol{\omega} \in \Sigma} F_{\boldsymbol{\mu}}(\boldsymbol{\omega})$ where $F_{\boldsymbol{\mu}}(\boldsymbol{\omega}) = \min_{j \in \mathcal{J}_{i^\star(\boldsymbol{\mu})}} f_j(\boldsymbol{\omega}, \boldsymbol{\mu})$ and for all $j \in \mathcal{J}_{i^\star(\boldsymbol{\mu})}$,

$$f_j(\boldsymbol{\omega}, \boldsymbol{\mu}) = \inf_{\boldsymbol{\lambda} \in \mathcal{C}_j^{i^\star(\boldsymbol{\mu})}} \sum_{k=1}^K \omega_k d(\mu_k, \lambda_k). \tag{2}$$

Note that (2) is convex program (by convexity of the KL divergence), and that f_j is a concave function in $\boldsymbol{\omega}$ (as the minimum of concave functions). As a consequence, the objective function $F_{\boldsymbol{\mu}}$ is also concave, but not smooth. The following proposition summarizes insightful properties of the functions f_j, $j \in \mathcal{J}_{i^\star(\boldsymbol{\mu})}$. It is a consequence of the envelope theorem and proved in Appendix K.2.

Proposition 1. *Let $j \in \mathcal{J}_{i^\star(\boldsymbol{\mu})}$. Define for all $(\boldsymbol{\omega}, \boldsymbol{\mu}) \in \Sigma \times \mathcal{S}_{i^\star(\boldsymbol{\mu})}$,*

$$\overline{\boldsymbol{\lambda}_j(\boldsymbol{\omega}, \boldsymbol{\mu})} = \arg \min_{\boldsymbol{\lambda} \in \mathrm{cl}(\mathcal{C}_j^{i^\star(\boldsymbol{\mu})})} \sum_{k=1}^K \omega_k d(\mu_k, \lambda_k), \tag{3}$$

where $\mathrm{cl}(\mathcal{C}_j^{i^\star(\boldsymbol{\mu})})$ is the closure of $\mathcal{C}_j^{i^\star(\boldsymbol{\mu})}$. Then under Assumption 1, $\overline{\boldsymbol{\lambda}_j(\boldsymbol{\omega}, \boldsymbol{\mu})}$ is unique for all $(\boldsymbol{\omega}, \boldsymbol{\mu}) \in \mathring{\Sigma} \times \mathcal{S}_{i^\star(\boldsymbol{\mu})}$, where $\mathring{\Sigma}$ is the interior of Σ. In addition, f_j is continuously differentiable on $\mathring{\Sigma} \times \mathcal{S}_{i^\star(\boldsymbol{\mu})}$, and $\forall (\boldsymbol{\omega}, \boldsymbol{\mu}) \in \mathring{\Sigma} \times \mathcal{S}_{i^\star(\boldsymbol{\mu})}$,

$$\nabla_{\boldsymbol{\omega}} f_j(\boldsymbol{\omega}, \boldsymbol{\mu}) = \sum_{k=1}^K d(\mu_k, \overline{\boldsymbol{\lambda}_j(\boldsymbol{\omega}, \boldsymbol{\mu})}_k) \boldsymbol{e}_k, \tag{4}$$

where $\boldsymbol{e}_k$ denotes the K-dimensional vector whose k-th coordinate is 1 and whose other coordinates are 0.

A key insight from the above result is that the objective function $F_{\boldsymbol{\mu}}$ is the minimum of a finite number of continuously differentiable functions. This observation will make the use of a slightly modifed FW algorithm possible (remember that the FW algorithm is known to converge for smooth functions only). We use an additional assumption on the gradient and curvature of f_j. A controlled curvature is an essential ingredient when analyzing the convergence of FW-based algorithms, see e.g. [17]. Define $\Sigma_\gamma = \{\boldsymbol{\omega} \in \Sigma : \min_k \omega_k \geq \gamma\}$ for any $\gamma \in (0, 1/K)$. Following [17], we define $C_\psi(\mathcal{K})$, the curvature constant of the concave differentiable function $\psi : \mathcal{K} \to \mathbb{R}$ with respect to the compact set $\mathcal{K}$, as

$$C_\psi(\mathcal{K}) = \sup_{\substack{\boldsymbol{x}, \boldsymbol{z} \in \mathcal{K} \\ \alpha \in (0,1] \\ \boldsymbol{y} = \boldsymbol{x} + \alpha(\boldsymbol{z} - \boldsymbol{x})}} \frac{1}{\alpha^2} \left[\psi(\boldsymbol{x}) - \psi(\boldsymbol{y}) + \langle \boldsymbol{y} - \boldsymbol{x}, \nabla \psi(\boldsymbol{x}) \rangle \right]. \tag{5}$$

Refer to [17], for the intuition behind this defintion and examples.

Assumption 2. *For all $\boldsymbol{\mu} \in \Lambda$,*
(i) there exists $L > 0$ such that $\forall j \in \mathcal{J}_{i^\star(\boldsymbol{\mu})}$, $\|\nabla_{\boldsymbol{\omega}} f_j(\boldsymbol{\omega}, \boldsymbol{\mu})\|_\infty \leq L$;
(ii) there exists $D > 0$ such that $\forall \gamma \in (0, 1/K)$ and $\forall j \in \mathcal{J}_{i^\star(\boldsymbol{\mu})}$, $C_{f_j(\cdot, \boldsymbol{\mu})}(\Sigma_\gamma) \leq \frac{D}{\gamma}$.

There is a simple way to verify whether a pure exploration problem satisfies Assumption 2, by looking at the second derivative of the function $y \mapsto d(x, y)$ at the points $(\mu_k, (\overline{\boldsymbol{\lambda}_j(\boldsymbol{\omega}, \boldsymbol{\mu})})_k)$ for all k. Refer to Appendix C for details.

Example 1 (cont'd). For unstructured bandits with Bernoulli rewards, we can easily compute f_j and its gradient [13]: for all $j \neq i^\star(\boldsymbol{\mu})$ and all $\boldsymbol{\omega} \in \mathring{\Sigma}$, define $m_j(\boldsymbol{\omega}, \boldsymbol{\mu}) = \frac{\omega_{i^\star(\boldsymbol{\mu})}\mu_{i^\star(\boldsymbol{\mu})} + \omega_j \mu_j}{\omega_{i^\star(\boldsymbol{\mu})} + \omega_j}$. Then $\overline{\boldsymbol{\lambda}_j(\boldsymbol{\omega}, \boldsymbol{\mu})}_k = \mu_k$ if $k \notin \{i^\star(\boldsymbol{\mu}), j\}$ and $\overline{\boldsymbol{\lambda}_j(\boldsymbol{\omega}, \boldsymbol{\mu})}_k = m_j(\boldsymbol{\omega}, \boldsymbol{\mu})$ otherwise. As a consequence:

$$\begin{cases} f_j(\boldsymbol{\omega}, \boldsymbol{\mu}) = \omega_{i^\star(\boldsymbol{\mu})} d(\mu_{i^\star(\boldsymbol{\mu})}, m_j(\boldsymbol{\omega}, \boldsymbol{\mu})) + \omega_j d(\mu_j, m_j(\boldsymbol{\omega}, \boldsymbol{\mu})), \\ \nabla f_j(\boldsymbol{\omega}, \boldsymbol{\mu}) = d(\mu_{i^\star(\boldsymbol{\mu})}, m_j(\boldsymbol{\omega}, \boldsymbol{\mu})) \boldsymbol{e}_{i^\star(\boldsymbol{\mu})} + d(\mu_j, m_j(\boldsymbol{\omega}, \boldsymbol{\mu})) \boldsymbol{e}_j. \end{cases} \tag{6}$$

For this example, we can verify that Assumption 2 holds, either directly or using the tool described in Appendix C. □

3.2 Stopping and decision rules

Next we present the two last components of the BC algorithm, namely the stopping and decision rules. These components are standard and borrowed from the existing literature. We need a few notations. For any $t \geq 1$, let A_t denote the arm selected in round t. Define $N_k(t) = \sum_{s=1}^t \mathbb{1}\{A_s = k\}$ the number of times arm k has been selected up to round t, and by $\omega_k(t) = N_k(t)/t$ the corresponding empirical proportion of draw. The empirical average reward of arm k up to round t is denoted by $\hat{\mu}_k(t) = \sum_{s=1}^t X_k(s)\mathbb{1}\{A_s = k\}/N_k(t)$, where $X_k(s)$ is the random reward received from pulling arm k in round s.

Let us denote by τ, the stopping time defining when the algorithm stops exploring and has to output a decision. Our decision rule is obviously to output the best empirical answer: $\hat{\imath}_\tau = i^\star(\hat{\boldsymbol{\mu}}(\tau))$.

For the stopping rule, as in other existing algorithms, we leverage a Generalized Likelihood Ratio Test (GLRT). Our test boils down to comparing $tF_{\hat{\boldsymbol{\mu}}(t)}(\boldsymbol{\omega}(t))$ to a threshold $\beta(t, \delta)$ (recall that $F_{\boldsymbol{\mu}}$ is the objective function of the lower bound optimization problem):

$$\tau = \inf\{t \geq 1 : tF_{\hat{\boldsymbol{\mu}}(t)}(\boldsymbol{\omega}(t)) \geq \beta(t, \delta)\}. \tag{7}$$

Many thresholds $\beta(t, \delta)$ have been proposed in the literature [21, 13, 18, 29]. For BC and its analysis, we just need that the threshold statisfies the two following properties:

$$\forall t \geq 1, \quad \left(tF_{\hat{\boldsymbol{\mu}}(t)}(\boldsymbol{\omega}(t)) \geq \beta(t, \delta)\right) \Longrightarrow \left(\mathbb{P}_{\boldsymbol{\mu}}\left[i^\star(\hat{\boldsymbol{\mu}}(t)) \neq i^\star(\boldsymbol{\mu})\right] \leq \delta\right), \tag{8}$$

$$\exists c_1(\Lambda), c_2(\Lambda) > 0 \ : \ \forall t \geq c_1(\Lambda), \ \ \beta(t, \delta) \leq \log\left(\frac{c_2(\Lambda)t}{\delta}\right). \tag{9}$$

The first of the above properties will naturally imply that BC returns the true answer with probability at least $1 - \delta$ when stopping, whereas the second will be instrumental in the sample complexity analysis (there, $c_1(\Lambda), c_2(\Lambda)$ may depend on the set of possible instances, and on the reward distributions). In [21], the authors manage to provide, for any generic pure exploration task, a single threshold satisfying (8)-(9)). Unless otherwise mentioned, we will use the stopping rule implementing this threshold.

4 The BC Algorithm and its Sample Complexity

In the BC algorithm, we use the FW algorithm to learn an optimal allocation $\boldsymbol{\omega}^\star(\boldsymbol{\mu})$. In each round, an iteration of FW updates the allocation that the BC algorithm aims at approaching using some tracking procedure. We describe this learning and tracking procedure below.

4.1 Adapting Frank-Wolfe to the non-smooth function $F_{\boldsymbol{\mu}}$

The FW algorithm [12] solves smooth convex programs by linearizing, in each iteration, the objective function and moving towards a minimizer of this linear function. Compared to the projected gradient and proximal methods, FW is computationally more efficient (e.g. it avoids the projection step), and is particularly well-suited when optimizing over polyhedra [3] (which is our case here). For a contemporary treatment of FW, refer to [17]. FW was suggested in [13] for BAI in unstructured bandits to update the allocation to be tracked. For this BAI problem, an iteration of the FW algorithm takes an intuitive form (see also Appendix A2 in [29]):

Example 1 (cont'd). For BAI in unstructured bandits, the optimal allocation $\boldsymbol{\omega}^\star(\boldsymbol{\mu})$ is the maximizer of the function $\boldsymbol{\omega} \mapsto F_{\boldsymbol{\mu}}(\boldsymbol{\omega}) = \min_j f_j(\boldsymbol{\omega}, \boldsymbol{\mu})$. $F_{\boldsymbol{\mu}}$ is smooth at points when the minimum is realized at a single arm $j^\star = \arg\min_j f_j(\boldsymbol{\omega}, \boldsymbol{\mu})$, and there, in view of (6), its gradient

is $\nabla F_{\boldsymbol{\mu}}(\boldsymbol{\omega}) = d(\mu_{i^\star(\boldsymbol{\mu})}, m_{j^\star}(\boldsymbol{\omega}, \boldsymbol{\mu}))\boldsymbol{e}_{i^\star(\boldsymbol{\mu})} + d(\mu_{j^\star}, m_{j^\star}(\boldsymbol{\omega}, \boldsymbol{\mu}))\boldsymbol{e}_{j^\star}$. Now in an iteration of the FW algorithm, one would follow the direction given by $\arg\max_{\boldsymbol{\omega}' \in \Sigma} \boldsymbol{\omega}'^\top \nabla F_{\boldsymbol{\mu}}(\boldsymbol{\omega})$. This direction is $\boldsymbol{e}_{j^\star}$ if $d(\mu_{j^\star}, m_{j^\star}(\boldsymbol{\omega}, \boldsymbol{\mu})) > d(\mu_{i^\star(\boldsymbol{\mu})}, m_{j^\star}(\boldsymbol{\omega}, \boldsymbol{\mu}))$, and $\boldsymbol{e}_{i^\star(\boldsymbol{\mu})}$ otherwise. This is precisely what the BC sampling rule suggested in [13] is doing: in round $(t+1)$, the best challenger is defined as $j^\star = \arg\min_j f_j(\boldsymbol{\omega}(t), \hat{\boldsymbol{\mu}}(t))$, and the arm selected corresponds to the direction given by $\arg\max_{\boldsymbol{\omega}' \in \Sigma} \boldsymbol{\omega}'^\top \nabla F_{\hat{\boldsymbol{\mu}}(t)}(\boldsymbol{\omega}(t))$, i.e., it is either the best challenger $j^\star$ or the best empirical arm $i^\star(\hat{\boldsymbol{\mu}}(t))$. □

The convergence analysis of FW usually requires that the objective function is smooth, and that its curvature can be controlled. When applying FW-type algorithms to design an optimal sampling rule (a rule that converges to the allocation $\boldsymbol{\omega}^\star(\boldsymbol{\mu})$ maximizing $F_{\boldsymbol{\mu}}$), we face three issues: (i) $F_{\boldsymbol{\mu}}$ is not smooth; (ii) $F_{\boldsymbol{\mu}}$ has an unbounded curvature close to the boundary of Σ; (iii) $\boldsymbol{\mu}$ is unknown initially, so the FW iteration in round t can be applied to $F_{\hat{\boldsymbol{\mu}}(t)}$ only. We discuss below how we circumvent these issues in the design of our algorithm.

(i) Non-smoothness of $F_{\boldsymbol{\mu}}$. In view of Proposition 1, $F_{\boldsymbol{\mu}}$ is the minimum of a finite number of smooth concave functions f_j. Hence at points where two of these functions are equal in $\boldsymbol{\omega}$, $F_{\boldsymbol{\mu}}$ is not differentiable in $\boldsymbol{\omega}$. The FW algorithm has been adapted to cope with non-smooth functions, see e.g. [31]. Typically, one constructs continuous approximations of the gradient close to non-smooth points of the functions. This construction often involves the r-subdifferential [16][2], which would be too costly to compute for $F_{\boldsymbol{\mu}}$. Instead, we can leverage the fact $F_{\boldsymbol{\mu}}$ is the minimum of concave functions, and construct the called *r-subdifferential subspace*: for $r \in (0,1)$,

$$H_{F_{\boldsymbol{\mu}}}(\boldsymbol{\omega}, r) = \mathrm{cov}\left\{\nabla f_j(\boldsymbol{\omega}) : j \in \mathcal{J}_{i^\star(\boldsymbol{\mu})}, f_j(\boldsymbol{\omega}) < F_{\boldsymbol{\mu}}(\boldsymbol{\omega}) + r\right\}, \tag{10}$$

where $\mathrm{cov}(S)$ denotes the convex hull of the set S. This choice greatly simplifies because it does not require to compute the gradient of f_j in a neighborhood of $\boldsymbol{\omega}$. Since the f_j are continuously differentiable, we can prove that $\boldsymbol{\omega} \mapsto H_{F_{\boldsymbol{\mu}}}(\boldsymbol{\omega}, r)$ is a continuous (i.e. upper- and lower-hemicontinuous). Using the r-subdifferential subspace, the modified FW update is given as follows. Let $\boldsymbol{x}(t)$ be the estimated optimizer of $F_{\boldsymbol{\mu}}$ in round t. In round $(t+1)$, it is updated as:

$$\begin{cases} \boldsymbol{z}(t+1) = \mathrm{argmax}_{\boldsymbol{z} \in \Sigma} \min_{h \in H_{F_{\boldsymbol{\mu}}}(\boldsymbol{x}(t), r_t)} \langle \boldsymbol{z} - \boldsymbol{x}(t), h \rangle, \\ \boldsymbol{x}(t+1) = \frac{t}{t+1}\boldsymbol{x}(t) + \frac{1}{t+1}\boldsymbol{z}(t+1). \end{cases} \tag{11}$$

Of course in the BC algorithm, $\boldsymbol{\mu}$ is unknown, and will be simply replaced by $\hat{\boldsymbol{\mu}}(t)$ is the above update. The way we choose the sequence of parameters $\{r_t\}_{t \geq 1}$ will be discussed later. Computing $\boldsymbol{z}(t)$ is equivalent to solving a zero-sum game, which can be further formulated as a LP [41] (Chapter 20). Refer to Appendix H for a detailed description of this LP.

(ii) Unbounded curvature of $F_{\boldsymbol{\mu}}$ and (iii) unknown $\boldsymbol{\mu}$. These two issues are solved by a single trick. We impose that in the FW iterations, the update directions $\boldsymbol{z}(t)$ cover all $\boldsymbol{e}_k, k = 1, \ldots, K$ sufficiently often. This ensures that the target allocation $\boldsymbol{x}(t)$ stays away from the boundary of Σ, which in turn allows us to control the curvature of $F_{\hat{\boldsymbol{\mu}}(t)}$ thanks to Assumption 2. This imposed constraint can be seen as a sort of forced exploration, and further implies (thanks to our tracking procedure) that each arm is played often enough. Now, with this kind of forced exploration, $\hat{\boldsymbol{\mu}}(t)$ will concentrate around the true $\boldsymbol{\mu}$.

4.2 Algorithm

The BC algorithm proceeds as follows. BC maintains a target allocation, denoted by $\boldsymbol{x}(t)$, its empirical allocation $\boldsymbol{\omega}(t)$, and the empirical average rewards $\hat{\boldsymbol{\mu}}(t)$ after round t. After an initialization phase (K rounds where each arm is selected), BC alternates between forced exploration and FW updates. More precisely:

Forced exploration occurs at rounds t where $\sqrt{\lfloor t/K \rfloor}$ is an integer and at those where $\hat{\boldsymbol{\mu}}(t-1) \notin \Lambda$ (in this case, we cannot compute the objective function). In forced exploration round t, the target allocation is updated towards the center of the simplex: $\boldsymbol{x}(t) = \frac{t-1}{t}\boldsymbol{x}(t-1) + \frac{1}{t}(1/K, \ldots, 1/K)$.

FW updates happen in other rounds. There, the target allocation is updated according to our adapted version of FW (11), where in round t the unknown $\boldsymbol{\mu}$ is replaced by $\hat{\boldsymbol{\mu}}(t-1)$. In the successive FW

[2] For $r \in (0,1)$, the r-subdifferential of $\psi : \mathcal{K} \to \mathbb{R}$ (where $\mathcal{K} \subset \mathbb{R}^K$ is compact and convex) is defined as $\partial_r \psi(\boldsymbol{x}) = \{h \in \mathbb{R}^K : \psi(\boldsymbol{y}) < \psi(\boldsymbol{x}) + \langle \boldsymbol{y} - \boldsymbol{x}, h \rangle + r$ for all $\boldsymbol{y} \in \mathcal{K}\}$.

updates, we use r-subdifferential subspaces with varying parameter r. For the analysis of BC, we will select a sequence of parameters $\{r_t\}_{t\geq 1}$ with an appropriate decay rate.

After the target allocation is updated in round t, the algorithm tracks this allocation by selecting the arm maximizing over k the ratio $x_k(t)/\omega_k(t-1)$. Finally, BC, whose pseudo-code is presented below, uses the stopping and decision rules described in §3.2.

Algorithm 1: The Best Challenger algorithm

Input: Confidence level δ, sequence $\{r_t\}_{t\geq 1}$
Initialization: Sample each arm once and update $\boldsymbol{\omega}(K)$, $\boldsymbol{x}(K) = (\frac{1}{K}, \dots, \frac{1}{K})$, and $\hat{\boldsymbol{\mu}}(K)$
$t \leftarrow K$
While $(tF_{\hat{\boldsymbol{\mu}}(t)}(\boldsymbol{\omega}(t)) < \beta(\delta, t))$
  $t \leftarrow t+1$
  If $(\sqrt{\lfloor t/K \rfloor} \in \mathbb{N}$ or $\hat{\boldsymbol{\mu}}(t-1) \notin \Lambda)$ (*forced exploration*) $\boldsymbol{z}(t) \leftarrow (\frac{1}{K}, \dots, \frac{1}{K})$
  Else (*FW update*)

$$\boldsymbol{z}(t) \leftarrow \operatorname*{argmax}_{\boldsymbol{z}\in\Sigma} \min_{h \in H_{F_{\hat{\boldsymbol{\mu}}(t-1)}}(\boldsymbol{x}(t-1), r_t)} \langle \boldsymbol{z} - \boldsymbol{x}(t-1), h\rangle$$

  Update $\boldsymbol{x}(t) \leftarrow \frac{t-1}{t}\boldsymbol{x}(t-1) + \frac{1}{t}\boldsymbol{z}(t)$
  Sample the arm $A_t \leftarrow \operatorname{argmax}_k x_k(t)/\omega_k(t-1)$ (*ties broken arbitrarily*)
  Update $\boldsymbol{\omega}(t)$ and $\hat{\boldsymbol{\mu}}(t)$
Output: $i^\star(\hat{\boldsymbol{\mu}}(t))$

4.3 Sample complexity

In the following theorem, we establish the asymptotic optimality of BC.

Theorem 1. *Consider the BC algorithm with a sequence $\{r_t\}_{t\geq 1}$ of strictly positive reals satisfying (i) $\lim_{t\to\infty} \frac{1}{t}\sum_{s=1}^{t} r_s = 0$, and (ii) $\lim_{t\to\infty} tr_t = \infty$. Under Assumptions 1, 2, the algorithm terminates in finite time almost surely and is δ-PAC. Its sample complexity τ satisfies:*

$$\forall \boldsymbol{\mu} \in \Lambda,\ \ \mathbb{P}_{\boldsymbol{\mu}}\left[\limsup_{\delta\to 0} \frac{\tau}{\log(1/\delta)} \leq T^\star(\boldsymbol{\mu})\right] = 1, \ \ \textit{and} \ \ \limsup_{\delta\to 0} \frac{\mathbb{E}_{\boldsymbol{\mu}}[\tau]}{\log(1/\delta)} \leq T^\star(\boldsymbol{\mu}).$$

The proof is given in Appendix I. We sketch the proof of the guarantees in expectation. The proof relies on classical concentration results, but more critically combines continuity arguments (developed in Appendix K) to account for the varying $\hat{\boldsymbol{\mu}}(t)$, and tools to analyze the convergence of the modified FW algorithm (reported in Appendix L).
1. First using concentration inequalities and the fact that BC includes forced exploration rounds, we can define, for round t, a "good" event $\mathcal{E}_t$ under which $\hat{\boldsymbol{\mu}}(t)$ is very close to $\boldsymbol{\mu}$ and such that $\sum_{t=1}^{\infty} \mathbb{P}_{\boldsymbol{\mu}}[\mathcal{E}_t^c] < \infty$. Then, several continuity arguments have to be made. In Lemma 6 (Appendix K) we show that $\boldsymbol{\mu} \mapsto F_{\boldsymbol{\mu}}$ is continuous (w.r.t. the uniform convergence norm). In Theorem 3 (Appendix K) we also prove that the solution $\boldsymbol{z}(t+1)$ of the FW update (11) is continuous in $\boldsymbol{\mu}$. The arguments above allow us to analyze the convergence of the FW updates almost as if $\hat{\boldsymbol{\mu}}(t)$ was replaced by $\boldsymbol{\mu}$ provided that the event $\mathcal{E}_t$ occurs.
2. Now we can study under the event $\mathcal{E}_t$, the impact of the FW update on the target allocation. The main step of our proof is Theorem 6 (Appendix L) characterizing how $F_{\boldsymbol{\mu}}(\boldsymbol{x}(t))$ get closer to $F_{\boldsymbol{\mu}}(\boldsymbol{\omega}^\star(\boldsymbol{\mu}))$ in each FW update. We then deduce that after a time T_1, $F_{\hat{\boldsymbol{\mu}}(t)}(\boldsymbol{x}(t))$ is a good approximation of $F_{\boldsymbol{\mu}}(\boldsymbol{\omega}^\star(\boldsymbol{\mu}))$.
3. We conclude the proof using similar arguments as those in [13]. According to our stopping rule, $t > \tau$ if and only if $tF_{\hat{\boldsymbol{\mu}}(t)}(\boldsymbol{\omega}(t)) > \beta(t,\delta)$. Hence $\mathbb{E}_{\boldsymbol{\mu}}[\tau] = \sum_{t=1}^{\infty} \mathbb{P}_{\boldsymbol{\mu}}[\tau > t] = \sum_{t=1}^{\infty} \mathbb{P}_{\boldsymbol{\mu}}\left[tF_{\hat{\boldsymbol{\mu}}(t)}(\boldsymbol{\omega}(t)) \leq \beta(t,\delta)\right]$ which can be approximately upper bounded by $T_1 + \sum_{t=T_1}^{\infty} \mathbb{P}_{\boldsymbol{\mu}}[\mathcal{E}_t^c] + \sum_{t=1}^{\infty} \mathbb{P}_{\boldsymbol{\mu}}[tF_{\boldsymbol{\mu}}(\boldsymbol{\omega}^\star(\boldsymbol{\mu})) \leq \beta(t,\delta)]$. The proof is concluded by remarking that in view of the property (9) of our stopping threshold, the last sum is close to $T^\star(\boldsymbol{\mu})\log(1/\delta)$ as $\delta \to 0$.
Note that our proof of Theorem 1 accounts for the possibility in certain structures (e.g. linear) of

having multiple optimal allocations (these allocations form a convex set). We just reason in terms of the objective function (as in [18] for linear bandits).

Under the following additional assumption, we can derive non-asymptotic sample complexity upper bound for BC. The proof of the following theorem is presented in Appendix N.

Assumption 3. *For any $\boldsymbol{\mu} \in \Lambda$, there exist constants $\kappa, E > 0$, s.t. if $\|\boldsymbol{\pi} - \boldsymbol{\mu}\|_\infty \leq \kappa$, then $\boldsymbol{\pi} \in \mathcal{S}_{i^\star(\boldsymbol{\mu})}$, $\forall \boldsymbol{\omega} \in \mathring{\Sigma},\ j \in \mathcal{J}_{i^\star(\boldsymbol{\mu})}$, $\nabla_{\boldsymbol{\pi}} d(\pi_k, \overline{\boldsymbol{\lambda}_j(\boldsymbol{\omega}, \boldsymbol{\pi})}_k)$ is continuous and $\left\|\nabla_{\boldsymbol{\pi}} d(\pi_k, \overline{\boldsymbol{\lambda}_j(\boldsymbol{\omega}, \boldsymbol{\pi})}_k)\right\|_1 \leq E,\ \forall k = 1, \ldots, K$.*

Theorem 2. *Consider the BC algorithm with a sequence $\{r_t\}_{t \geq 1}$ as in Theorem 1. Under Assumptions 1, 2, and 3, the sample complexity τ of the algorithm satisfies: for any $\boldsymbol{\mu} \in \Lambda, \delta \in (0, 1)$, and two positive constants $\epsilon < \min\{\kappa E/2, 1\}$, $\tilde{\epsilon} < 1$,*

$$\mathbb{E}_{\boldsymbol{\mu}}[\tau] \leq \frac{1+\tilde{\epsilon}}{F_{\boldsymbol{\mu}}(\boldsymbol{\omega}^\star(\boldsymbol{\mu})) - 6\epsilon}\left[\log\left(\frac{(1+\tilde{\epsilon})c_2(\Lambda)e}{\delta(F_{\boldsymbol{\mu}}(\boldsymbol{\omega}^\star(\boldsymbol{\mu})) - 6\epsilon)}\right) + \log\log\left(\frac{(1+\tilde{\epsilon})c_2(\Lambda)}{\delta(F_{\boldsymbol{\mu}}(\boldsymbol{\omega}^\star(\boldsymbol{\mu})) - 6\epsilon)}\right)\right] + \Psi(K, D, E, L, c_1(\Lambda), \epsilon) + T_{\epsilon,L}^{\frac{5}{4}},$$

where $T_{\epsilon,L}$ is a constant such if $t \geq T_{\epsilon,L}$, then $\sum_{s=1}^t r_s < t\epsilon$ and $tr_t > L$. The constant Ψ is polynomial in $(D, E, L, c_1(\Lambda), 1/\epsilon)$ and exponential in K. The precise definition of Ψ is given in Appendix N.

5 Examples and Experiments for Linear Bandits

5.1 Examples

Our framework can be applied to many pure exploration problems, including BAI in unstructured (see Example 1), linear, Lipschitz bandits. It further covers threshold bandits (the problem of identifying all arms with rewards greater than a threshold), linear threshold bandits, top-m bandits (where we wish to identify the best m arms), and dueling bandits. All these examples are presented in Appendix. Using numerical experiments, we show that BC outperforms existing algorithms for BAI in unstructured, linear, and Lipschitz bandits, see Appendices D-E-F, respectively. To the best of our knowledge, we report the first results for BAI in Lipschitz bandits. We quote some of our results for BAI in linear bandits below.

When facing a new pure exploration problem, one can check whether it falls into our framework, by first directly verifying Assumption 1. In Appendix C, we provide a simple sufficient condition ensuring that Assumption 2 holds, and explain why all the aforementioned pure exploration problems satisfy this condition.

5.2 BAI in linear bandits

Linear bandits constitute arguably the most popular and important bandit problems with structure, and have found many applications [26, 5]. BAI in linear bandits has received a lot of attention recently, see §2. To model linear bandits, we slightly modify our framework. The reason for this modification is that the linear structure is so strong that using our initial framework, the set Λ would be small, and we would have problems ensuring that $\hat{\boldsymbol{\mu}}(t) \in \Lambda$ after some reasonable time t. Alternatively (rather than modifying the framework), we could modify the BC algorithm so that $\hat{\boldsymbol{\mu}}(t)$ is projected onto Λ.

Consider a set of K arms. Arm k is attached a d-dimensional feature vector $\boldsymbol{a}_k$ and its average reward $\langle \boldsymbol{a}_k, \boldsymbol{\mu} \rangle$, where $\boldsymbol{\mu} \in \mathbb{R}^d$ is unknown. Without loss of generality, we assume that $\{\boldsymbol{a}_k\}_{k \in [K]}$ spans $\mathbb{R}^d$. We modify the definition of Λ as follows: $\Lambda = \left\{\boldsymbol{\mu} \in \mathbb{R}^d : \exists k \in [K] \text{ s.t.} \langle \boldsymbol{a}_k - \boldsymbol{a}_i, \boldsymbol{\mu} \rangle > 0, \forall i \neq k\right\}$. Hence $\boldsymbol{\mu}$ parametrizes the average rewards of the arms, but μ_k is not the average reward of arm k. The true answer is $i^\star(\boldsymbol{\mu}) = \operatorname{argmax}_k \langle \boldsymbol{a}_k, \boldsymbol{\mu} \rangle$. The lower bound optimization problem (1) becomes: $\sup_{\boldsymbol{\omega} \in \Sigma} F_{\boldsymbol{\mu}}(\boldsymbol{\omega})$ where $F_{\boldsymbol{\mu}}(\boldsymbol{\omega}) = \inf_{\boldsymbol{\lambda} \in \text{Alt}(\boldsymbol{\mu})} \frac{1}{2}(\boldsymbol{\mu} - \boldsymbol{\lambda})^\top \sum_k \omega_k \boldsymbol{a}_k \boldsymbol{a}_k^\top (\boldsymbol{\mu} - \boldsymbol{\lambda})$ and $\text{Alt}(\boldsymbol{\mu}) = \{\boldsymbol{\lambda} \in \Lambda : \exists k \neq i^\star(\boldsymbol{\mu}) \text{ s.t.} \langle \boldsymbol{a}_k - \boldsymbol{a}_{i^\star(\boldsymbol{\mu})}, \boldsymbol{\lambda} \rangle > 0\}$, see e.g. [18]. From there, we can reproduce our framework: for Assumption 1, for all $j \neq i^\star(\boldsymbol{\mu})$, $\mathcal{C}_j^{i^\star(\boldsymbol{\mu})} = \{\boldsymbol{\lambda} \in \Lambda : \langle \boldsymbol{a}_j - \boldsymbol{a}_{i^\star(\boldsymbol{\mu})}, \boldsymbol{\lambda} \rangle > 0\}$; as for

the functions f_j, they are defined through:

$$\overline{\boldsymbol{\lambda}_j(\boldsymbol{\omega}, \boldsymbol{\mu})} = \boldsymbol{\mu} + \left(\frac{\langle \boldsymbol{a}_{i^\star(\boldsymbol{\mu})} - \boldsymbol{a}_j, \boldsymbol{\mu} \rangle}{\left\| \boldsymbol{a}_{i^\star(\boldsymbol{\mu})} - \boldsymbol{a}_j \right\|^2_{V_{\boldsymbol{\omega}}^{-1}}} V_{\boldsymbol{\omega}}^{-1} \right) \left(\boldsymbol{a}_j - \boldsymbol{a}_{i^\star(\boldsymbol{\mu})} \right), \tag{12}$$

where $V_{\boldsymbol{\omega}} = \sum_k \omega_k \boldsymbol{a}_k \boldsymbol{a}_k^\top$. In the BC algorithm for linear bandits, we use the Least-Squares Estimator (LSE) $\hat{\boldsymbol{\mu}}(t)$ given past observations, see [18] or Appendix E for an explicit expression. It can be readily seen that this slight modification of our framework does not affect the validity of Theorem 1. We just need to use the concentration inequalities derived in [18] for $\hat{\boldsymbol{\mu}}(t)$ in the first step of its proof.

Numerical experiments. We consider the example proposed by [36]. The unknown parameter $\boldsymbol{\mu} = \boldsymbol{e}_1$ and there are $d+1$ arms, $\boldsymbol{e}_1, \cdots, \boldsymbol{e}_d, \cos(\phi)\boldsymbol{e}_1 + \sin(\phi)\boldsymbol{e}_2$ in $\mathbb{R}^d$, where $(\boldsymbol{e}_1, \cdots, \boldsymbol{e}_d)$ form the standard orthonormal basis. We set $d = 6$ and $\phi = 0.1$. To assess the performance of the BC algorithm, we compare with the following algorithms: the Lazy Track and Stop algorithm (`LT`) from [18]; LineGame-C (`CG-C`) and LineGame (`Lk-C`) from [9] and implemented by [35]; the XY-Adaptive algorithm (`XY-A`) from [36]. For information, we also run the Round Robin algorithm `RR` selecting each equally. For comparison, we finally compute the sample complexity lower bound $\mathtt{LB}_{\mathtt{lin}}(\delta)$ (equal to $T^\star(\boldsymbol{\mu})\mathrm{kl}(\delta, 1-\delta)$).

Except for `LT`, `LT-H`, and `XY-A`, all algorithms implement the same stopping rule defined in (7) with threshold $\beta(t, \delta) = \log((\log(t)+1)/\delta)$ (this threshold was initially suggested in [13], and is also used in [35] for `CG-C` and `Lk-C`). For `LT`, `LT-H`, and `XY-A`, we use the stopping rule advocated in the corresponding papers. Refer to Appendix E for the detailed implementations.

In Table 1, we present the sample complexity (the number of samples gathered before the algorithm stops) averaged over 1000 runs for the various algorithms and for different confidence levels $\delta \in \{0.1, 0.01, 0.001, 0.0001\}$. In Appendix E, we provide detailed results, e.g. including box-plots (to show how confident we are about the values displayed in Table 1), as well as the empirical allocations achieved under the various algorithms. In linear bandits, the gain achieved by our algorithm BC is rather spectacular. In BAI in other structured bandits, BC still outperforms other algorithms, but without such an impressive performance difference.

Table 1: Sample complexity for the linear bandit benchmark example of [36], averaged over 1000 runs. Refer to Appendix E for details, including box-plots.

	BC	LT	LT-H	CG-C	Lk-C	XY-A	RR	$\mathtt{LB}_{\mathtt{lin}}(\delta)$
$\delta = 0.1$	**1 030**	7 651	3 319	2 498	2 319	7 016	5 451	359
$\delta = 0.01$	**1 614**	8 290	3 907	3 501	3 431	7 779	8 814	920
$\delta = 0.001$	**2 229**	8 952	4 527	4 324	4 326	9 090	12 101	1 408
$\delta = 0.0001$	**2 839**	9 575	5 120	5 118	5 120	9 723	15 314	1 881

6 Conclusion

We have developed BC, a computationally and statistically efficient algorithm for active pure exploration in bandit problems with fixed confidence. In each round, BC performs a single iteration of a modified FW algorithm to approach an optimal allocation of arm draws predicted by the asymptotic lower bound. In the BC algorithm, the FW iterations aim at maximizing a non-smooth function. Our main contribution is here to adapt the design of FW so that its convergence can be analyzed even for this non-smooth function. FW-based pure exploration algorithms have been discussed in the literature, with the belief that they would perform well. We confirm this belief, and even establish the asymptotic optimality of BC in wide class of pure exploration problems.

Many interesting research directions could be investigated. Our analysis of the sample complexity in the moderate confidence regime has the advantage of being applicable to generic pure exploration problems, but may not be always tight. For bandits with specific structures, we may refine the analysis in this regime to get better upper bounds. We are also interested in investigating whether the iterative approach used in the BC algorithm can be extended to more complex problems such as learning an optimal policy in MDPs, as well as to regret minimization problems. There, instance-specific regret lower bounds and the corresponding optimal exploration process are characterized by the solution of an optimization problem, just as in pure exploration problems.

References

[1] Andreas Argyriou, Marco Signoretto, and Johan Suykens. Hybrid conditional gradient-smoothing algorithms with applications to sparse and low rank regularization. *Regularization, Optimization, Kernels, and Support Vector Machines*, 2014.

[2] Quentin Berthet and Vianney Perchet. Fast rates for bandit optimization with upper-confidence frank-wolfe. In *Proc. of NeurIPS*, 2017.

[3] Venkat Chandrasekaran, Benjamin Recht, Pablo A. Parrilo, and Alan S. Willsky. The convex geometry of linear inverse problems. 2012.

[4] Edward Cheung and Yuying Li. Solving separable nonsmooth problems using frank-wolfe with uniform affine approximations. In *Proc. of IJCAI*, 2018.

[5] Wei Chu, Lihong Li, Lev Reyzin, and Robert Schapire. Contextual bandits with linear payoff functions. In *Proc. of AISTATS*, 2011.

[6] Richard Combes, Stefan Magureanu, and Alexandre Proutiere. Minimal exploration in structured stochastic bandits. In *Proc. of NeurIPS*, 2017.

[7] Rémy Degenne and Wouter M Koolen. Pure exploration with multiple correct answers. In *Proc. of NeurIPS*, 2019.

[8] Rémy Degenne, Wouter M Koolen, and Pierre Ménard. Non-asymptotic pure exploration by solving games. In *Proc. of NeurIPS*, 2019.

[9] Rémy Degenne, Pierre Ménard, Xuedong Shang, and Michal Valko. Gamification of pure exploration for linear bandits. In *Proc. of ICML*, 2020.

[10] Tanner Fiez, Lalit Jain, Kevin G Jamieson, and Lillian Ratliff. Sequential experimental design for transductive linear bandits. In *Proc. of NeurIPS*, 2019.

[11] Abraham D Flaxman, Adam Tauman Kalai, and H Brendan McMahan. Online convex optimization in the bandit setting: gradient descent without a gradient. In *Proc. of SODA*, 2005.

[12] Marguerite Frank and Philip Wolfe. An algorithm for quadratic programming. *Naval Research Logistics Quarterly*, 1956.

[13] Aurélien Garivier and Emilie Kaufmann. Optimal best arm identification with fixed confidence. In *Proc. of COLT*, 2016.

[14] Aurélien Garivier, Pierre Ménard, Laurent Rossi, and Pierre Menard. Thresholding bandit for dose-ranging: The impact of monotonicity. *arXiv preprint arXiv:1711.04454*, 2017.

[15] Elad Hazan and Satyen Kale. Projection-free online learning. In *Proc. of ICML*, 2012.

[16] Jean-Baptiste Hiriart-Urruty and Claude Lemaréchal. *Convex analysis and minimization algorithms I: Fundamentals*. Springer science & business media, 2013.

[17] Martin Jaggi. Revisiting frank-wolfe: Projection-free sparse convex optimization. In *Proc. of ICML*, 2013.

[18] Yassir Jedra and Alexandre Proutiere. Optimal best-arm identification in linear bandits. In *Proc. of NeurIPS*, 2020.

[19] Marc Jourdan, Mojmír Mutnỳ, Johannes Kirschner, and Andreas Krause. Efficient pure exploration for combinatorial bandits with semi-bandit feedback. In *Proc. of ALT*, 2021.

[20] Zohar S Karnin. Verification based solution for structured mab problems. In *Proc. of NeurIPS*, 2016.

[21] Emilie Kaufmann and Wouter Koolen. Mixture martingales revisited with applications to sequential tests and confidence intervals. *arXiv preprint arXiv:1811.11419*, 2018.

[22] Tomáš Kocák and Aurélien Garivier. Best arm identification in spectral bandits. In *Proc. of IJCAI*, 2020.

[23] Wouter Koolen. tidnabbil: Julia library for structured bandit models. `https://bitbucket.org/wmkoolen/tidnabbil/src/master/`, 2021. [Online; accessed 09-May-2021].

[24] Yuko Kuroki, Junya Honda, and Masashi Sugiyama. Combinatorial pure exploration with full-bandit feedback and beyond: Solving combinatorial optimization under uncertainty with limited observation. *arXiv preprint arXiv:2012.15584*, 2020.

[25] Tze Leung Lai and Herbert Robbins. Asymptotically efficient adaptive allocation rules. *Advances in applied mathematics*, 1985.

[26] Lihong Li, Wei Chu, John Langford, and Robert E Schapire. A contextual-bandit approach to personalized news article recommendation. In *Proc. of WWW*, 2010.

[27] Andrea Locatelli, Maurilio Gutzeit, and Alexandra Carpentier. An optimal algorithm for the thresholding bandit problem. In *Proc. of ICML*, 2016.

[28] Blake Mason, Lalit Jain, Ardhendu Tripathy, and Robert Nowak. Finding all ϵ-good arms in stochastic bandits. In *Proc. of NeurIPS*, 2020.

[29] Pierre Ménard. Gradient ascent for active exploration in bandit problems. *arXiv*, 2019.

[30] Federico Pierucci, Zaid Harchaoui, and Jérôme Malick. *A smoothing approach for composite conditional gradient with nonsmooth loss*. PhD thesis, INRIA Grenoble, 2014.

[31] Sathya N Ravi, Maxwell D Collins, and Vikas Singh. A deterministic nonsmooth frank wolfe algorithm with coreset guarantees. *Informs Journal on Optimization*, 2019.

[32] Clémence Réda, Emilie Kaufmann, and Andrée Delahaye-Duriez. Top-m identification for linear bandits. In *Proc. of AISTATS*, 2021.

[33] Idan Rejwan and Yishay Mansour. Top-k combinatorial bandits with full-bandit feedback. In *Proc. of ALT*, 2020.

[34] Daniel Russo. Simple bayesian algorithms for best arm identification. In *Annual Conference on Learning Theory*. PMLR, 2016.

[35] Xuedong Shang. Linbai: Gamification of pure exploration for linear bandits. `https://github.com/xuedong/LinBAI.jl`, 2021. [Online; accessed 09-May-2021].

[36] Marta Soare, Alessandro Lazaric, and Rémi Munos. Best-arm identification in linear bandits. In *Proc. of NeurIPS*, 2014.

[37] Chao Tao, Saúl Blanco, and Yuan Zhou. Best arm identification in linear bandits with linear dimension dependency. In *Proc. of ICML*, 2018.

[38] Tengyao Wang, Nitin Viswanathan, and Sébastien Bubeck. Multiple identifications in multi-armed bandits. In *Proc. of ICML*, 2013.

[39] DJ White. Extension of the frank-wolfe algorithm to concave nondifferentiable objective functions. *Journal of optimization theory and applications*, 1993.

[40] Liyuan Xu, Junya Honda, and Masashi Sugiyama. A fully adaptive algorithm for pure exploration in linear bandits. In *Proc. of AISTATS*, 2018.

[41] Petyon Young and Shmuel Zamir. *Handbook of game theory*. Elsevier, 2014.

[42] Zixin Zhong, Wang Chi Cheung, and Vincent Tan. Best arm identification for cascading bandits in the fixed confidence setting. In *Proc. of ICML*, 2020.

Checklist

1. For all authors...
   (a) Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope? [Yes] The claims match our results clearly.
   (b) Did you describe the limitations of your work? [Yes] They are clearly described in Sections 1 and 3.
   (c) Did you discuss any potential negative societal impacts of your work? [Yes] We have not identified any potential negative impact (mainly due to the theoretical nature of our work).
   (d) Have you read the ethics review guidelines and ensured that your paper conforms to them? [Yes]
2. If you are including theoretical results...
   (a) Did you state the full set of assumptions of all theoretical results? [Yes] The assumptions can be found in Sections 1,3 and 4.
   (b) Did you include complete proofs of all theoretical results? [Yes] All the detail of proofs are provided in Appendix due to space constraints.
3. If you ran experiments...
   (a) Did you include the code, data, and instructions needed to reproduce the main experimental results (either in the supplemental material or as a URL)? [Yes] Please refer to `README.md` in our attached code for instructions.
   (b) Did you specify all the training details (e.g., data splits, hyperparameters, how they were chosen)? [Yes] We provide high-level descriptions in the supplementary material while the more technical details such as random seeds are specified in our attached code.
   (c) Did you report error bars (e.g., with respect to the random seed after running experiments multiple times)? [Yes] The sample complexity of the compared methods are also shown in box-plots for this purpose.
   (d) Did you include the total amount of compute and the type of resources used (e.g., type of GPUs, internal cluster, or cloud provider)? [Yes] Please see our supplementary material.
4. If you are using existing assets (e.g., code, data, models) or curating/releasing new assets...
   (a) If your work uses existing assets, did you cite the creators? [Yes] For part of the baselines, we use the code implementation by [23, 35], as described in the experimental section in the supplementary material.
   (b) Did you mention the license of the assets? [Yes] Our code is distributed under MIT license.
   (c) Did you include any new assets either in the supplemental material or as a URL? [Yes] Our code is attached in the supplementary material.
   (d) Did you discuss whether and how consent was obtained from people whose data you're using/curating? [N/A]
   (e) Did you discuss whether the data you are using/curating contains personally identifiable information or offensive content? [N/A]
5. If you used crowdsourcing or conducted research with human subjects...
   (a) Did you include the full text of instructions given to participants and screenshots, if applicable? [N/A]
   (b) Did you describe any potential participant risks, with links to Institutional Review Board (IRB) approvals, if applicable? [N/A]
   (c) Did you include the estimated hourly wage paid to participants and the total amount spent on participant compensation? [N/A]