Multitasking Bandits: Learning Fail-Safe Action Combinations

Damon Falck
Worcester College
University of Oxford

Supervised by
Thomas Orton, Varun Kanade

A dissertation submitted in partial fulfilment
of the requirements for the degree of
Master of Mathematics and Computer Science

Trinity Term 2022

Abstract

The *multi-armed bandit* problem is a classic online learning scenario where an agent repeatedly picks between several actions, receiving at each round a hitherto-unknown reward depending on its choice. The agent's goal is to minimize its *regret* relative to the best single action in the long term. This is now a well-studied problem, with a number of successful algorithms in the literature and with constantly-emerging applications ranging from medical trials and COVID-19 testing to viral marketing and portfolio optimization. But what if you can take *more than one* action at each round and get the reward from the best one? This extension is natural in many practical settings—particularly with the availability of parallel processing—but has received little attention. In this work we propose a number of novel algorithms for this problem in various contexts based on a generalization of 'Follow the Perturbed Leader'. We provide theoretical guarantees on our algorithms' performance relative to the best *single* action (unlike related works) and corresponding computational lower bounds. Next, we show that our algorithms can be used to improve on current methods in submodular function optimization. Finally, we study our algorithms' empirical performance on an online deep learning hyperparameter selection problem based on the 2020 NeurIPS Black-Box Optimization Challenge, as well as in a number of synthetic scenarios.

Contents

List of Algorithms

Chapter 1

Introduction

Situations frequently arise in practice where a choice must repeatedly be made between several alternatives—each time resulting in some observed outcome—and it is necessary to learn *on the go* how to maximize long-term success. For instance, when conducting a clinical trial one may wish to decide which treatment option to give each new participant based on current success rates; when running a business one may wish every day to set new prices for your products; when designing a lifelong learning agent one may wish to choose which of several learning methods to apply to each new problem.[1]

These situations are captured by the *multi-armed bandit* problem, in which an agent must at each of T time-steps choose an action a from some fixed finite set $\mathscr{A}$, and this choice receives some reward (or equivalently incurs some cost); the goal is to maximize the cumulative reward over all T rounds (or minimize the cumulative cost). First formulated in the mid-twentieth century [Rob52], the ubiquity of this problem has since generated consistent attention, with many successful solutions having been proposed and analysed in the literature under a variety of assumptions on the nature of the rewards/costs, which may be sampled from a distribution or chosen by an adversary, and from both Bayesian and frequentist perspectives. More recently, the wide applicability of this framework to modern data-rich settings has caused a resurgence of interest, with both new theoretical approaches [Sli19] and new applications of classical bandit algorithms [BRA20] featuring prominently at major conferences.

Indeed, multi-armed bandit algorithms have been subject to a plethora of generalizations and adaptations. Some of these deal with varying decision structures: *linear bandits* allow each action to be a vector in $\mathbb{R}^d$, with the received reward some linear function of this decision [AK08; MB04]; *combinatorial bandits* allow actions within some subset of $\{0,1\}^d$ [CBL12; BCB12]. Others handle varying feedback regimes: does the algorithm see the rewards for all possible actions after each round, some subset of them, or just the action it took? Still more tackle the addition of *contextual information* available at each round [TM17]. Indeed, work exists in problem settings involving many combinations of the above.

One interesting variant of this problem, however, has received little attention, despite its natural applicability to a large class of practical scenarios. Suppose, as in the original bandit problem, that an agent can choose from a fixed action set $\mathscr{A}$, but that now it is given an *action budget* B and can pick up to B separate actions to take together at each round; the agent receives the

[1] All of these scenarios involve a trade-off between *exploration* and *exploitation*: choose the option you think is best at each stage and you may never discover better ones you haven't tried yet, but spend the whole time trying out new things and you'll ruin your long-term performance.

largest reward from any of the individual actions it takes. Intuitively, now the agent must learn how to guarantee that *at least one* of the B actions it picks will be good (a 'fail-safe' combination of actions).

This scenario models the leveraging of additional resources at each round (i.e. so that more than one action can be performed) to increase performance; three obvious ways such additional resources may arise are given in Table 1.1. See Table 1.2 for a more detailed list of potential applications of this maximum-of-rewards structure.

It is this problem variant—which we call the *multitasking bandit problem*—that we tackle in this work.

1.1 Contributions and structure

Our main original contributions are as follows:

- We propose a new efficient algorithm to solve the full feedback multitasking bandit problem and prove various regret bounds for this algorithm (Chapter 3).
- We adapt our algorithm to the partial feedback setting in various ways and prove regret bounds for each, as well as considering a less efficient version of our algorithm that competes better in anticorrelated-reward settings (Chapter 4).
- We provide initial lower bounds on the regret achievable by any multitasking bandit algorithm, which our full feedback algorithm can match asymptotically in $|\mathscr{A}|$ (Chapter 5).
- We use our algorithm to create a new hybrid version of an existing online submodular function maximization algorithm, and we examine the performance of both on the multitasking bandit problem (Chapter 6).
- We apply our algorithms empirically to an online hyperparameter optimization problem based on the 2020 NeurIPS Black-Box Optimization Challenge [Tur+21] (Chapter 7).
- We empirically compare the various algorithms we've discussed in a number of synthetic-reward environments (Chapter 8).

We start in Chapter 2 by providing a brief introduction to the classical multi-armed bandit problem and two bandit algorithms that we will refer to, and to close in Chapter 9 we discuss our findings and possible research directions for the future.

1.2 Related work

Although what we call the multitasking bandit problem has not been considered before in this exact form, several works have dealt with generalizations or variants of it, and we review these briefly here.

In the most relevant work, Streeter and Golovin [SG08] propose a greedy $(1-\mathrm{e}^{-1})$-approximation algorithm for an *online submodular function maximization* problem of which the multitasking bandit problem is a special case. Their algorithm has regret guarantees relative to the best-in-hindsight set of B actions; in our work we focus instead on regret bounds with respect to the best-in-hindsight set of N actions for $N < B$, which better models the *addition* of extra resources inherent in many of the applications just discussed. Indeed, our algorithms **(a)** have much smaller regret in T than theirs for the $N = 1$ case we focus on, as well as in certain cases for $N > 1$, and **(b)** often outperform theirs in practice, as explored later. We discuss their work further in Chapter 6

Table 1.1: Families of scenarios the multitasking bandit problem applies to.

- **Parallel processing.** If each action is not easily parallelizable but B processors are available for computation, it makes sense to take B actions in parallel and use the results from the best one.
- **Time availability.** If there is some pre-determined time available for each round before moving onto the next, and executing one action takes much less than the available time, it makes sense to take as many actions as possible before the next round starts and use the results from the best one.
- **Trading off resources intentionally.** Even if neither of the above apply, in many cases it may still be beneficial to voluntarily spend more compute/time on a round than just one action needs so as to increase the chances of success—especially if high performance at each round is more important than, or of similar importance to, fast/efficient completion of all rounds. It is this scenario that motivates much of the theoretical analysis in our work.

Table 1.2: Some applications of the multitasking bandit problem.

- **Online algorithm portfolios:** Arguably the most attractive application is to choosing between multiple available algorithms/methods for completing each of a series of tasks. With the availability of parallel processing, or even without (as mentioned above), it may be desirable to try B algorithms on each task instead of just one and take the best results. Some instances of this are:
  - **Hyperparameter optimization.** Increasingly advanced algorithms are emerging for hyperparameter search when training deep learning models. In many cases no one algorithm is best across all cases; we may want to on each of a series of learning tasks apply B different hyperparameter selection algorithms and take the best hyperparameters found by any of them. We apply our algorithms to this setting in Chapter 7.
  - **Reinforcement learning.** Laroche and Feraud [LF17] applied bandit techniques to selecting one of several available reinforcement learning policies to apply on each episode of an episodic task. In some situations it may be possible and appropriate to instead execute B policies on each episode and lock in the results from the best one.
  - **Lifelong learning.** The most general and appealing version of this setting is in a generalized learning agent which must tackle many different learning problems throughout its lifetime. This agent may have a large repertoire of learning techniques it has acquired but finite resources to apply to each problem, so it may want to be able to choose B of its methods to try on each task.
- **Network routing.** One use of classical bandit algorithms is adaptive data routing [AK08], where a router forwards packets to their destinations using varying network routes with the goal of minimizing time taken (and packet loss). We may wish to instead forward each packet using B routes for redundancy.
- **Advert placement.** Another common use of bandit algorithms is to choose which of several ads should be displayed on a website each time a user visits it, hoping to maximize clicks [Ava+21]. In the restricted feedback setting where the algorithm only sees its overall maximum reward at each round, the multitasking bandit problem models displaying B adverts and the user clicking on the best one.
- **Online bidding.** A situation where an auctioneer repeatedly chooses B participants to bid for an item and the item is sold to the highest bidder is modelled well by the multitasking bandit problem.

(where we also propose a hybridization of their algorithm with ours that can improve on theirs in the general submodular function setting), and we evaluate their algorithm and our modification of it empirically against ours in Chapters 7 and 8.

The multitasking bandit problem is also a special case of the *combinatorial bandit problem*, where an algorithm chooses a bitstring from some $\mathscr{X} \subseteq \{0,1\}^d$ at each round and receives a reward based on this choice: just set $d := |\mathscr{A}|$ and $\mathscr{X} := \left\{(b_1, \dots, b_d) \in \{0,1\}^d : \sum_{i=1}^d b_i \leqslant B\right\}$.[2] Several works consider such combinatorial bandit problems where the overall reward is a *linear* function of the chosen tuple [CBL12; BCBK12; ABL14], i.e. the sum of the relevant single-action rewards, but in our case the reward function would be

$$\texttt{reward}(b_1, \dots, b_d) := \max\{r_1 b_1, \dots, r_d b_d\} \tag{1.1}$$

for single-action rewards $r_1, \dots, r_d$, which is not linear.

There has been some work on combinatorial bandits with a general, non-linear reward function, but none quite applies to our problem. Combes et al. [Com+15], Kveton et al. [Kve+15], Chen et al. [Che+18], and Chen, Wang, and Zhou [CWZ18] all consider versions in the *stochastic* setting, where rewards are sampled i.i.d. from some distribution; the *adversarial* setting which we are interested in in this work—where rewards may be chosen arbitrarily by an adversary—is considerably more general and quite different in nature. Of particular note, Chen et al. [Che+16] propose an algorithm called 'Stochastically Dominant Confidence Bound' that solves the combinatorial bandit problem with general reward functions and they apply it to a stochastic version of the problem we consider in this work, which they note the importance of and call the K-MAX bandit problem.[3] A recent follow-up work [HWC21] has tackled the adversarial setting of the combinatorial bandit problem with a general reward function, but with a more restrictive feedback regime: only the *overall* calculated reward is revealed to the algorithm after each round, not the method of calculation (in our case the single-action rewards).

1.3 Attribution and publication

The initial ideas for a number of algorithms, results and proofs in this thesis as well as assistance with various details are due to my supervisor Thomas Orton; much of this work is the result of collaboration between us, and a jointly-authored conference paper based in part on some of the results in this thesis is currently under review.

1.4 Notation

Some standard notation used throughout the document is listed in Table 1.3.

[2] So, for example, given $|\mathscr{A}| = 5$ available actions and a budget $B = 2$, one possible choice would be $(0, 0, 1, 0, 1)$, corresponding to taking the third and fifth of the five available actions together.

[3] They also compare their algorithm for the K-MAX bandit problem to Streeter and Golovin's more general algorithm, demonstrating theoretical superiority (arbitrary approximation accuracy) and a substantial empirical improvement (but this was in the stochastic setting, which is orthogonal to ours.)

Table 1.3: Semantics, unless otherwise specified, of some frequently used notation (note that some symbols not mentioned here have overloaded meanings in different proofs.)

Symbol	Meaning
$\mathbb{N}$	The set $\{1, \dots\}$, i.e. natural numbers *excluding* zero
$[n]$	The set $\{1, \dots, n\}$
$T \in \mathbb{N}$	Number of rounds in a bandit game
$\mathscr{A}$	Set of actions available
$B \in \mathbb{N}$	Action budget
$N \in \mathbb{N}$	Benchmark set size
$t, s \in \mathbb{N}$	Round indices
$i, j \in \mathbb{N}$	Action indices
$\mathscr{E}_t$	Variously refers to events or algorithms
$r_t(a) \in [0, 1]$	Reward for action $a \in \mathscr{A}$ at round $t \in [T]$
$r_t(S)$	Maximum reward $\max_{a \in S} r_t(a)$ of action set $S \subseteq \mathscr{A}$ at round $t \in [T]$
$R_t(\cdot)$	Cumulative reward $\sum_{s=1}^{t} r_s(\cdot)$ up to round $t \in [T]$ (by convention $R_0 = 0$)
$R_t^\star(\cdot)$	Cumulative reward $\sum_{s=0}^{t} r_s(\cdot)$ *including a non-zero initial reward*
$p_t(a) \in \mathbb{R}$	Random perturbation applied at round $t \in [T]$ for action $a \in \mathscr{A}$
$\tilde{R}_t(a)$	*Perturbed* cumulative reward $R_t(a) + p_{t+1}(a)$ of action $a \in \mathscr{A}$ up to round $t \in [T]$
$\hat{r}_t(a)$	An *estimated* reward for action $a \in \mathscr{A}$ at round $t \in [T]$
$\hat{R}_t(a)$	Estimated cumulative reward $\sum_{s=1}^{t} \hat{r}_s(a)$ for action $a \in \mathscr{A}$ up to round $t \in [T]$
$a^\star$	Best-in-hindsight single action $\arg\max_{a \in \mathscr{A}} R_T(a)$
$S_N^\star$	Best-in-hindsight fixed size-N action set $\arg\max_{S \subseteq \mathscr{A} : \lvert S \rvert = N} R_T(S)$
$\mathscr{R}_N$	N-regret $R_T(S_N^\star) - \sum_{t=1}^{T} r_t(S_t)$ of a sequence of action sets $(S_t)_{t \in [T]}$
$\mathscr{F}_t$	The σ-algebra generated by all observations and choices up to (and including) round t

Chapter 2

Background: Multi-Armed Bandits

We start with a very brief summary of some relevant classical bandit theory.

2.1 Overview

First introduced by Robbins [Rob52] in 1952 and named thus in reference to choosing between an array of 'one-armed bandit' slot machines at a casino, the multi-armed bandit problem and its variants have generated a large literature over the last seventy years. Formally, the basic problem is as follows:[1]

Definition 2.1 (Multi-armed bandits)**.** *The* multi-armed bandit problem *(or bandit problem) is defined as follows. Suppose you are playing a game consisting of T rounds. There is a fixed finite set $\mathscr{A}$ of actions at your disposal; at each round $t \in [T]$ you may select one action $a_t \in \mathscr{A}$ to take. Before the game starts, an adversary picks a hidden* reward[2] *$r_t(a) \in [0,1]$ associated with each action $a \in \mathscr{A}$ at each round $t \in [T]$; the reward received by the action you take is revealed after your selection. The goal is to maximize your cumulative reward $\sum_{t=1}^{T} r_t(a_t)$ over all T rounds.*

A multitasking bandit algorithm *(or just a bandit algorithm) is any algorithm that solves the multi-armed bandit problem.*

The version of the problem stated above uses what's called *bandit feedback*, where you only see the reward of the action you actually take at each round; in constrast, in the *full feedback* version of the problem (often called the *experts* problem) you get to see the reward of every possible action after you make each choice.

Write $R_t(a) := \sum_{s=1}^{t} r_s(a)$ for the cumulative reward of an action $a \in \mathscr{A}$. The most common way to talk about the performance of a bandit algorithm is how low its *regret* is:[3]

Definition 2.2 (Regret)**.** *The* 1-regret*, or just* regret*, of a bandit algorithm that takes actions $a_1, \dots, a_T$ is*

$$\mathscr{R}_1 := R_T(a^\star) - \sum_{t=1}^{T} r_t(a_t), \tag{2.1}$$

[1] The literature often refers to actions as *arms* and frames the problem as minimizing *costs* $c_t(a) := 1 - r_t(a)$ instead of maximizing rewards.

[2] The restriction $r_t(a) \in [0,1]$ is not necessary but we assume it throughout our work.

[3] There are many other possible notions of regret, some of which we'll use in the more general problem settings we consider later in this work.

where $a^\star = \arg\max_{a \in \mathscr{A}} R_T(a)$.

An algorithm that always receives reward 0 at every round would incur worst-case regret $\mathscr{R}_1 = T$, so the aim is normally to produce algorithms with regret guaranteed to be *sublinear in* T.

2.2 WMR and Exp3

One algorithm proposed by Littlestone and Warmuth [LW94] for the full feedback bandit problem is called 'Randomized Weighted Majority' (Algorithm 1), sometimes referred to as *Hedge*.[4] Intuitively, at each round **WMR** decreases the chance of picking an action by an factor that's greater the smaller its reward at that round was, so in the long run actions that frequently get large rewards will have the highest probability of being chosen.

Algorithm 1: WMR

Input: *parameter* $\varepsilon \in (0, 1/2)$*, available actions* $\mathscr{A}$*, number of rounds* T*, rewards* $(r_t(a))_{a \in \mathscr{A}}$ *revealed after each round* t*.*

Output: *actions* $a_1, \dots, a_T$*.*

Initialize weights $w_1(a) := 1$ for each $a \in \mathscr{A}$. Then for each round $t \in [T]$:

1. Define $q_t(a) := \frac{w_t(a)}{\sum_{a' \in \mathscr{A}} w_t(a')}$ for each action $a \in \mathscr{A}$.
2. Sample an action a_t from distribution $(q_t(a))_{a \in \mathscr{A}}$.
3. Observe rewards $(r_t(a))_{a \in \mathscr{A}}$.
4. Update the weights as $w_{t+1}(a) := w_t(a) \cdot (1 - \varepsilon)^{1 - r_t(a)}$ for each a.

It is shown that this algorithm satisfies the following worst-case regret bound:

Proposition 2.3. ***WMR*** *with parameter* $\varepsilon = \sqrt{\ln |\mathscr{A}|/T}$ *incurs expected regret*

$$\mathbb{E}[\mathscr{R}_1] = \mathscr{O}(\sqrt{T \log |\mathscr{A}|}). \tag{2.2}$$

In the *bandit feedback* setting, some kind of estimate must be made of the unobserved rewards at each round; the problem is how to make these best to balance *exploration* (trying actions you don't know much about) with *exploitation* (taking actions you think are good). A generalization of **WMR** that handles bandit feedback called **Exp3**, for *exploration, exploitation, exponentiation*, was proposed in a seminal 2002 paper [Aue+02]—see Algorithm 2. This algorithm satisfies the following regret bound:

Proposition 2.4. ***WMR*** *with parameters* $\gamma < 1/2T$ *and* $\varepsilon = \sqrt{(1 - \gamma) \ln |\mathscr{A}|/3|\mathscr{A}|T}$ *incurs expected regret*

$$\mathbb{E}[\mathscr{R}_1] = \mathscr{O}(\sqrt{T|\mathscr{A}| \log |\mathscr{A}|}). \tag{2.4}$$

[4] In the *stochastic* case where rewards for each action are drawn i.i.d. from some distribution at each round, deterministic bandit algorithms may do well, but in the *adversarial* case we consider where no such assumption applies, it is easy to see that some amount of randomization is necessary to be able to get a sublinear worst-case bound on the regret.

Algorithm 2: Exp3

Input: *parameter $\varepsilon \in (0, 1/2)$, exploration probability $\gamma \in (0, 1/2)$, available actions $\mathscr{A}$, number of rounds T, reward $r_t(a_t)$ revealed after each round t only for the action taken.*

Output: *actions $a_1, \ldots, a_T$.*

For each round $t \in [T]$:

1. Call **WMR** and receive the probability distribution $(q_t(a))_{a \in \mathscr{A}}$.
2. Sample an action a'_t from this distribution.
3. With probability $1-\gamma$ take action $a_t := a'_t$, otherwise pick an action $a_t \in \mathscr{A}$ uniformly at random to take.
4. Observe the reward reward $r_t(a_t)$ for the chosen action.
5. Define estimated rewards

$$\hat{r}_t(a) := \begin{cases} \frac{r_t(a_t)}{\gamma/|\mathscr{A}|+(1-\gamma)q_t(a)} & \text{if } a = a_t, \\ 0 & \text{otherwise} \end{cases} \tag{2.3}$$

   for each $a \in \mathscr{A}$.
6. Return the estimated rewards $(\hat{r}_t(a))_{a \in \mathscr{A}}$ to **WMR**.

2.3 Follow the Perturbed Leader

Another approach to the full feedback bandit problem was rediscovered[5] by Kalai and Vempala [KV05]. Motivated by the natural idea of taking the action that's done best overall so far, but also the necessity of introducing randomization to compete with a worst-case adversary, their algorithm is called 'Follow the Perturbed Leader' (Algorithm 3).

They proved that **FPL** achieves the same asymptotic regret bound as **WMR** of $\mathscr{O}(\sqrt{T \log |\mathscr{A}|})$ in expectation when run with perturbation rate $\varepsilon = \sqrt{\ln |\mathscr{A}|/T}$.[6]

FPL has since been adapted to the partial feedback setting in a couple of ways, and we use these for inspiration later.

2.4 Some recent applications

There have been many applications of bandit algorithms in the literature, especially in the last decade. A good survey is provided by Bouneffouf, Rish, and Aggarwal [BRA20]. Some examples are given in Table 2.1.

[5] It was proposed initially by Hannan [Han57] in a different context and with a more complex analysis.
[6] It is the technique they used for this proof which we will generalize later in this work.

Algorithm 3: FPL

Input: *perturbation rate $\varepsilon > 0$, available actions $\mathscr{A}$, number of rounds T, rewards $(r_t(a))_{a\in\mathscr{A}}$ revealed after each round t.*

Output: *actions $a_1, \dots, a_T$.*

For each round $t \in [T]$:

1. Sample independent exponential perturbations $p_t(a) \sim \mathrm{Exp}(\varepsilon)$ for each action $a \in \mathscr{A}$.
2. Define the perturbed cumulative reward $\tilde{R}_{t-1}(a) := R_{t-1}(a) + p_t(a)$ for each action $a \in \mathscr{A}$.
3. Pick $a_t := \arg\max_{a\in\mathscr{A}} \tilde{R}_{t-1}(a)$ to be the action with the largest perturbed cumulative reward.
4. Observe the rewards $(r_t(a))_{a\in\mathscr{A}}$ (and update $R_t(a) := R_{t-1}(a) + r_t(a)$ for each a).

Table 2.1: Some recent applications of bandit algorithms.

- Durand et al. [Dur+18] apply bandit techniques to adaptive allocation for data collection in clinical trials, gathering more data dynamically for promising treatments.
- Huo and Fu [HF17] incorporate *risk-awareness* into the classical bandit setting to apply bandit algorithms to online portfolio selection in financial markets.
- Kerkouche et al. [Ker+18] demonstrate the use of multi-armed bandit algorithms to optimize the performance of long-range wide area network technology by allowing each node to select its communication parameters adaptively.
- One particularly recent and interesting application of bandit algorithms has been to COVID-19 testing. In the summer of 2020, the Greek government for three months allocated limited PCR tests at every port of entry using a Bayesian bandit algorithm; an analysis showed that their algorithm detected up to 4 times more asymptomatic infected travellers during peak times than simpler testing policies used by other nations would have [Bas+21].

Chapter 3

Multiple Actions: Generalizing 'Follow the Perturbed Leader'

We turn now to our new generalized problem setting:

Definition 3.1 (Multitasking bandits)**.** *The* multitasking bandit problem *is defined as follows. Suppose you are playing a game consisting of T rounds.*[1] *There is a fixed finite set $\mathscr{A}$ of actions at your disposal, and you are given an* action budget *$B \in [|\mathscr{A}|]$; at each round $t \in [T]$ you may select up to B actions $a \in \mathscr{A}$ to take at once, i.e. you may choose any set $S_t \subseteq \mathscr{A}$ s.t. $|S_t| \leqslant B$. Before the game starts, an adversary picks a hidden* reward *$r_t(a) \in [0,1]$ associated with each action $a \in \mathscr{A}$ at each round $t \in [T]$. The rewards received by each action you actually take are revealed after your selection and* you *receive as your payoff the maximum $r_t(S_t) := \max_{a \in S_t} r_t(a)$ of these rewards, i.e. you are rewarded for the best single action out of those you chose. The goal is to maximize your cumulative reward (payoff) $\sum_{t=1}^{T} r_t(S_t)$ over all T rounds.*[2]

A multitasking bandit algorithm *is any algorithm that solves the multitasking bandit problem.*

How best to measure an algorithm's performance is now less obvious—what benchmark should we compare to? Writing $R_t(\cdot) := \sum_{s=1}^{t} r_s(\cdot)$ for cumulative reward up to round t, let's define the following family of regret functions:

Definition 3.2 (N-regret)**.** *For any $N \in [|\mathscr{A}|]$, the N-*regret *incurred by a multitasking bandit algorithm that chooses action sets $S_1, \dots, S_T$ is*

$$\mathscr{R}_N := R_T(S_N^\star) - \sum_{t=1}^{T} r_t(S_t) \tag{3.1}$$

where $S^\star := \max_{S \subseteq |\mathscr{A}| \,:\, |S| = N} R_T(S)$ is the best fixed action set of size N.

For a multitasking bandit algorithm with budget B there are two obvious candidate regret notions: the B-regret and the 1-regret. The former measures how well the algorithm does compared to the best fixed use of the same resources; the latter rather captures the benefit of *adding* the additional resources and enables a more meaningful comparison to classical $B = 1$ bandit algo-

[1] In practice the total number of rounds T may be unknown (or the game may run indefinitely), in which case simple adaptations to any algorithms can be made exactly as in the classical bandit setting—for example, by using the *doubling trick* [BK18].

[2] Note that in the case $B = 1$ this reduces to the classical multi-armed bandit setting discussed in Chapter 2.

rithms. Motivated by the applications discussed in the introduction, we choose to focus primarily on this second metric.

In this chapter we thus seek polynomial-time algorithms with asymptotically small *1-regret* as a function of T, $|\mathscr{A}|$ and also B. We will start with the *full feedback* case, a slightly simpler version of the problem where after each round t the reward $r_t(a)$ for *every* action $a \in \mathscr{A}$ is revealed to the algorithm. Later, in Chapter 4, we will consider variants of our algorithms that can deal with the more general *partial feedback* setting defined above.[3]

3.1 A new algorithm

We proceed by generalizing the 'Follow the Perturbed Leader' (FPL) algorithm from Chapter 2 to a full feedback multitasking bandit algorithm we call 'Follow the Perturbed Multiple Leaders' (FPML). Like **FPL**, **FPML** stochastically perturbs the cumulative rewards at each rounds, but it then takes the top B actions by perturbed cumulative reward rather than just the top one. See Algorithm 4.

Algorithm 4: FPML(ε)

Input: *rate parameter $\varepsilon \in (0, \infty]$, available actions $\mathscr{A}$, budget $B \in [|\mathscr{A}|]$, rewards $(r_t(a))_{a \in \mathscr{A}}$ received after each round $t \in [T]$.*

Output: *action sets $S_1, \dots, S_T$ all of size at most B.*

For each $t \in [T]$:

1. Independently draw exponential random perturbations $p_t(a) \sim \mathrm{Exp}(\varepsilon)$ for each action $a \in \mathscr{A}$, and define $\tilde{R}_{t-1}(a) := R_{t-1}(a) + p_t(a)$ to be the perturbed cumulative reward for action a so far.
2. Enumerate the actions $a \in \mathscr{A}$ in decreasing order of $\tilde{R}_{t-1}(a)$ and choose S_t to be the set containing the first B actions in this list.

Our analysis of this algorithm is based on a generalization of the argument used by Kalai and Vempala [KV05] and rests on a simple observation in the case $\varepsilon = \infty$:[4]

Lemma 3.3. *The 1-regret experienced by algorithm **FPML**(∞) is at most the number of times that an action currently* not *in the top B actions (by cumulative reward) becomes the best action so far on the next round.*

Proof. Let $m_t := \max_{a \in \mathscr{A}} R_t(a)$ for each $t \in [T]$, so that in particular $m_T = R_T(a^\star)$ where $a^\star := \arg\max_{a \in \mathscr{A}} R_T(a)$ is the best-in-hindsight single action. At each round $t \in [T]$ the actions S_t taken by **FPML**(∞) are the top B by cumulative reward; let $a_t^\star := \arg\max_{a \in \mathscr{A}} R_{t-1}(a)$ be the best of them, and define the event $\mathscr{E}_t = \{a_{t+1}^\star \notin S_t\}$.

[3] Another, orthogonal, direction in the problem space controls how much information our *adversary* receives. In the formulation defined above, which we focus on, we assume an *oblivious* adversary who chooses all rewards in advance before the game starts (or equivalently chooses each action's reward at each round without seeing our previous choices). In contrast, an *adaptive* adversary sees what we do at each round and may use this to inform their reward choices for the next round. Our full feedback algorithms do in fact work well in this case too, and this is proven in the accompanying conference paper.

[4] Note that using $\varepsilon = \infty$ simply chooses the B best actions so far without applying any randomness.

Fix t and assume $\neg\mathscr{E}_t$. We may write

$$r_t(a^\star_{t+1}) = R_t(a^\star_{t+1}) - R_{t-1}(a^\star_{t+1}) \tag{3.2}$$
$$= m_t - R_{t-1}(a^\star_{t+1}) \qquad \text{since } a^\star_{t+1} \text{ is the leader} \tag{3.3}$$
$$\geqslant m_t - m_{t-1} \tag{3.4}$$

by definition of m_t, m_{t-1}, so since $a^\star_{t+1} \in S_t$ by assumption,

$$r_t(S_t) = \max_{a \in S_t} r_t(a) \geqslant r_t(a^\star_t) \geqslant m_t - m_{t-1}. \tag{3.5}$$

Hence, defining $\mathscr{I} := \{t \in [T] : \mathscr{E}_t \text{ holds}\}$, the total reward received by the algorithm is

$$\sum_{t=1}^{T} r_t(S_t) \geqslant \sum_{t \in \mathscr{I}^c} r_t(S_t) \geqslant \sum_{t \in \mathscr{I}^c} (m_t - m_{t-1}) = \sum_{t=1}^{n} (m_t - m_{t-1}) - \sum_{t \in \mathscr{I}} (m_t - m_{t-1}) \tag{3.6}$$
$$= m_T - m_0 - \sum_{t \in \mathscr{I}} (m_t - m_{t-1}) \tag{3.7}$$
$$\geqslant m_T - m_0 - \sum_{t \in \mathscr{I}} 1 = m_T - |\mathscr{I}| \tag{3.8}$$

(taking $m_0 = 0$). Thus the 1-regret is

$$\mathscr{R}_1 = R_T(a^\star) - \sum_{t=1}^{T} r_t(S_t) \leqslant R_T(a^\star) - m_T + |\mathscr{I}| = |\mathscr{I}|. \tag{3.9}$$

□

This result can be quickly extended to the general case including random perturbations:

Lemma 3.4. *For any $\varepsilon \in (0, \infty)$, algorithm **FPML**($\varepsilon$) experiences 1-regret satisfying*

$$\mathbb{E}[\mathscr{R}_1] \leqslant \frac{\ln |\mathscr{A}|}{\varepsilon} + \mathbb{E}[|\mathscr{I}|] \tag{3.10}$$

where expectations are over the random perturbations and $\mathscr{I} \subseteq [T]$ is the set of rounds at which an action not currently in the top B (after perturbation) becomes the (perturbed) leader.

Proof outline. The key step is to note that perturbing each action's cumulative reward at each round is equivalent in expectation to choosing a random 'initial' reward for each action to start with before the game commences. The proof of Lemma 3.3 is then adapted to use these random initializations; the new $\ln |\mathscr{A}|/\varepsilon$ term comes from an upper bound on the expected difference between the random initialization of the best action and the largest random initialization of any action. See Appendix A for the details. □

This allows us to nicely bound the overall expected regret by arguing that $\mathscr{I}$ is usually small, similarly to in Kalai and Vempala [KV05].

Proposition 3.5. *For any $\varepsilon \in (0, \infty)$, algorithm **FPML**($\varepsilon$) experiences 1-regret satisfying*

$$\mathbb{E}[\mathscr{R}_1] \leqslant \frac{\ln |\mathscr{A}|}{\varepsilon} + T(1 - \mathrm{e}^{-\varepsilon})^B. \tag{3.11}$$

Proof outline. The proof argues that $\mathbb{E}[\mathscr{I}] \leqslant T(1-\mathrm{e}^{-\varepsilon})^B$ roughly as follows. First, at any round t, it is shown that for an action not in S_t to become the best (perturbed) action at this round, every action $a \in S_t$ must be overtaken (in perturbed cumulative reward) by some action not in S_t. Next, it is argued that for this overtaking event to happen to an action it is necessary that the action's current cumulative reward be at most one greater than that of the $(B+1)^{\text{th}}$ best action so far. A probabilistic lemma is applied to show that whether this is the case is conditionally independent from any other actions given the actions in S_t and their rewards, and the probability of this event is bounded using the memoryless property of the exponential distribution. The full proof is rather nice and given in Appendix A; the reader is encouraged to take a look! □

It remains to make an attractive choice of the perturbation rate ε.

3.1.1 Choosing the perturbation rate

We consider two choices of ε here; the first gives an algorithm with asymptotic regret bound essentially optimal in $|\mathscr{A}|$ (see Chapter 5), and the second gives an algorithm with good behaviour in $|\mathscr{A}|$, T and B.

Theorem 3.6. *Algorithm **FPML**(ε) with $\varepsilon = \ln\ln|\mathscr{A}|$ achieves expected 1-regret*

$$\mathscr{O}\left(\frac{\log|\mathscr{A}|}{\log\log|\mathscr{A}|} + T\exp\left[-\frac{B}{\log|\mathscr{A}|}\right]\right). \tag{3.12}$$

In particular, when $B = \Omega(\log|\mathscr{A}|\log T)$ the expected regret is $\mathscr{O}(\log|\mathscr{A}|/\log\log|\mathscr{A}|)$.

Proof. Setting $\varepsilon = \ln\ln|\mathscr{A}|$, the bound in Proposition 3.5 gives

$$\mathbb{E}[R] \leqslant \frac{\ln|\mathscr{A}|}{\ln\ln|\mathscr{A}|} + T(1-\mathrm{e}^{-\ln\ln|\mathscr{A}|})^B = \frac{\ln|\mathscr{A}|}{\ln\ln|\mathscr{A}|} + T\left(1-\frac{1}{\ln|\mathscr{A}|}\right)^B \tag{3.13}$$

and applying the approximation $1-x \leqslant \mathrm{e}^{-x}$ to the second term gives the result. □

The next result comes from using a slightly weaker approximation, $1-\mathrm{e}^{-x} \leqslant x$ instead of $\leqslant \mathrm{e}^{\mathrm{e}^{-x}}$:

Theorem 3.7. *Algorithm **FPML**(ε) with $\varepsilon = (\ln|\mathscr{A}|/T)^{1/(B+1)}$ achieves expected 1-regret*

$$\mathscr{O}\left(T^{\frac{1}{B+1}}(\log|\mathscr{A}|)^{\frac{B}{B+1}}\right).^5 \tag{3.14}$$

In particular, when $B = \Omega(\log T)$ the expected regret is $\mathscr{O}(\log|\mathscr{A}|)$.

Proof. Direct from Proposition 3.5, using the approximation $1-\mathrm{e}^{-x} \leqslant x$. This value of ε was chosen by setting the two terms of that bound approximately equal. □

From now on we will use **FPML** to mean this second instantiation of the algorithm with $\varepsilon = (\ln|\mathscr{A}|/T)^{1/(B+1)}$.[6]

[5] Note that in the case $B = 1$ this reduces to the standard regret bound $\mathscr{O}(\sqrt{T\log|\mathscr{A}|})$ for 'Follow the Perturbed Leader' (c.f. Chapter 2).

[6] Another interesting version of the algorithm results from using $\varepsilon = \alpha\ln|\mathscr{A}|$ for some $\alpha > 0$, giving expected regret $\mathscr{O}(T\mathrm{e}^{-B/|\mathscr{A}|^\alpha})$ which is constant for $B = \Omega(|\mathscr{A}|^\alpha\log T)$. It can be shown that in this case the constant upper regret bound is $1/(1+\alpha)$, meaning α controls the trade-off between size of the constant regret bound and required

3.2 Higher-order regret bounds

So we've given attractive upper bounds on the expected 1-regret incurred by our new algorithm. Can we do the same for the N-regret, where $N > 1$? The proof techniques used for the $N = 1$ case lead to the following instance-dependent parametric regret bound:

Proposition 3.8. *Algorithm **FPML**(ε) achieves expected N-regret*

$$\mathbb{E}[\mathscr{R}_N] \leqslant \frac{1 - N^{-1} + \ln(|\mathscr{A}|/N)}{\varepsilon} + T \sum_{k=0}^{N-1} \binom{B}{k} \mathrm{e}^{-k\varepsilon}(1 - \mathrm{e}^{-\varepsilon})^{B-k} + \mathtt{err}_N \tag{3.15}$$

where $\mathtt{err}_N := R_T(S_N^\star) - R_T(S_T|_{\varepsilon=\infty})$ is the difference in reward between the best-in-hindsight set of N actions and the set of the top N actions in hindsight on the given problem instance.

In particular, the expected B-regret is

$$\mathbb{E}[\mathscr{R}_B] \leqslant \frac{1 - B^{-1} + \ln(|\mathscr{A}|/B)}{\varepsilon} + T(1 - \mathrm{e}^{-\varepsilon B}) + \mathtt{err}_B. \tag{3.16}$$

Proof outline. Adapt the $N = 1$ argument, using "an action not in the top B enters the best N-set" as the event of interest; use the harmonic series form of the expectation of the max of exponential random variables to get a lower bound, and use a binomial counting argument to bound the probability of the event. □

The error term here is worst-case linear in T. This result does show, however, that **FPML** is competitive with the set of the top B fixed actions, and on instances where this set is similar to the fixed best set of B actions this is a useful result:

Corollary 3.9. *Algorithm **FPML**(ε) achieves expected regret at most $(1 - B^{-1} + \ln(|\mathscr{A}|/B))/\varepsilon + T(1 - \mathrm{e}^{-\varepsilon B})$ relative to the set of the top B actions.*

3.2.1 Using super-actions

One naïve way to obtain a decent N-regret bound is to run **FPML** treating arbitrary subsets of N actions as unrelated 'super-actions' and selecting the best B/N such super-actions;[7] this gives us a natural regret bound against the best single super-action:[8]

Proposition 3.10. *For general $N \in \mathbb{N}$, algorithm **FPML**^{N} achieves expected N-regret*

$$\mathbb{E}[\mathscr{R}_N] = \mathscr{O}\left(T^{\frac{N}{N+B}}(N \log|\mathscr{A}|)^{\frac{B}{N+B}}\right). \tag{3.17}$$

In particular, when $B = \Omega(N \log T)$ the expected N-regret is $\mathscr{O}(T \log|\mathscr{A}|)$.

time resources.

[7] We assume for convenience that N divides B.

[8] This algorithm is not computationally efficient; the expensive step is repeatedly finding the top B/N sets of N actions under cumulative reward. Since cumulative reward is a submodular function, this is a special case of finding the k best sets for maximizing a monotone submodular set function, which is an NP-hard problem (as discussed in Streeter and Golovin [SG08]). For $k = 1$ this problem is efficiently $(1 - \mathrm{e}^{-1})$-approximable using the algorithm given in Nemhauser, Wolsey, and Fisher [NWF78], and as with any discrete optimization problem a $k > 1$ version may then be efficiently generated using the procedure from Lawler [Law72]. However, the approximation ratio of the resulting bandit algorithm is likely to be fairly poor.

Algorithm 5: FPML^N

Input: *available actions $\mathscr{A}$, budget B, benchmark N, rewards $(r_t(a))_{a\in\mathscr{A}}$ received after each round $t \in [T]$.*

Output: *action sets $S_1, \dots, S_T \subseteq \mathscr{A}$ all of size B.*

1. Define $\mathscr{A}' := \{S \subseteq \mathscr{A} : |S| = N\}$.
2. Run algorithm **FPML** with action set $\mathscr{A}'$ and budget B/N.
3. At each round t, **FPML** chooses an action set $S'_t \subseteq \mathscr{A}'$; define $S_t := \bigcup S'_t$ and choose this action set for execution. We receive rewards $(r_t(a))_{a\in\mathscr{A}}$; pass on **FPML** the rewards $r'_t(S) := \max_{a\in S} r_t(a)$ for each $S \in \mathscr{A}'$.

Proof. This is immediate from applying Theorem 3.7 to the internal instance of **FPML** and noting that $|\mathscr{A}'| = \mathscr{O}(|\mathscr{A}|^N)$. □

Chapter 4

Partial Feedback Versions

We now return to the original *partial feedback* version of the multitasking bandit problem as defined at the start of Chapter 3, where the algorithm discovers the rewards only for the B actions it actually takes at each round.[1]

As in the classical bandit setting, techniques that handle partial feedback will usually involve making estimates of the unobserved rewards at each round and running a full feedback algorithm on these estimated rewards. We start by providing a general bound on **FPML**'s performance when executed with any unbiased reward estimators, before discussing two different ways to obtain these reward estimates.

4.1 General result with unbiased estimators

Suppose we have a method of obtaining *bounded unbiased* estimates $\hat{r}_t(a)$ of the true rewards $r_t(a)$ for each round $t \in [T]$ and action $a \in \mathscr{A}$. Formally, for each $t \in [T]$ let $\mathscr{F}_t$ denote the σ-algebra generated by all observations and choices up to and including round t; then $(\hat{r}_t(a))_{a\in\mathscr{A}}$ are bounded real-valued $\mathscr{F}_t$-measurable (but not $\mathscr{F}_{t-1}$-measurable) random variables with

$$\mathbb{E}[\hat{r}_t(a) \mid \mathscr{F}_{t-1}] = r_t(a), \qquad a \in \mathscr{A}. \tag{4.1}$$

We start by proving that the same regret decomposition shown in Lemma 3.4 still applies to **FPML**(ε) when run on these estimated rewards instead of the true rewards.

Proposition 4.1. *Algorithm **FPML**(ε) run on the estimated rewards $(\hat{r}_t(a))_{t\in[T],a\in\mathscr{A}}$ achieves expected 1-regret*

$$\mathbb{E}[\mathscr{R}_1] \leqslant \frac{\ln|\mathscr{A}|}{\varepsilon} + T\beta(1 - \mathrm{e}^{-(\alpha+\beta)\varepsilon})^B, \tag{4.2}$$

where $\alpha, \beta \geqslant 0$ are deterministic bounds such that $\hat{r}_t(a) \in [-\alpha, \beta]$ for all $t \in [T]$, $a \in \mathscr{A}$.

Proof. Shown by closely adapting the proofs of Lemma 3.4; see Appendix A. □

Now we need only come up with methods of producing the estimates $(\hat{r}_t(a))_{t\in[T],a\in\mathscr{A}}$ and adjust the regret bound we just showed accordingly.

[1] This scenario occurs more commonly in practice and will be the basis of our experiments later in Chapters 7 and 8.

4.2 A uniform-exploration adaptation of FPML

A first idea is to use some of our action budget to *explore*, gathering information to make our estimates, and only run **FPML** with the remaining action slots available. One way of doing this is to choose C actions uniformly at each round, combining these with the actions selected by **FPML** with budget $B - C$. This algorithm is detailed in Algorithm 6.[2]

Algorithm 6: $\textbf{FPML}_{\textbf{unif}}(C,\varepsilon)$ (an adaptation of **FPML**(ε) to the partial feedback case using *uniform exploration*).

> ***Input:*** *parameter $C \in \mathbb{N}$, parameter $\varepsilon > 0$, available actions $\mathscr{A}$, budget $B > C$, rewards $r_t(a)$ received after each round $t \in [T]$* ***only for the actions taken by the algorithm****.*
>
> ***Output:*** *action sets $S_1, \dots, S_T$ all of size at most B.*
>
> For each $t \in [T]$:
>
> 1. Independently draw random perturbations $p_t(a) \sim \mathrm{Exp}(\varepsilon)$ for each action $a \in \mathscr{A}$, and define $\tilde{R}_{t-1}(a) := p_t(a) + \sum_{s=1}^{t-1} \hat{r}_s(a)$ to be the *estimated* perturbed total reward for action a so far (see step 5).
> 2. Enumerate the actions $a \in \mathscr{A}$ in decreasing order of $\tilde{R}_{t-1}(a)$ and choose S_t^{exploit} to be the set containing the first $B - C$ actions in this list.
> 3. Uniformly choose a subset $S_t^{\text{explore}} \subseteq \mathscr{A}$ of size C. Let $S_t := S_t^{\text{explore}} \cup S_t^{\text{exploit}}$.
> 4. Receive feedback $r_t(a)$ for the actions $a \in S_t$.
> 5. For each action $a \in \mathscr{A}$, estimate the reward for this round as:
>
> $$\hat{r}_t(a) := \begin{cases} r_t(a)\frac{|\mathscr{A}|}{C} & \text{if } a \in S_t^{\text{explore}}, \\ 0 & \text{otherwise} \end{cases} \tag{4.3}$$
>
> for use in future rounds.

Proposition 4.2. *Algorithm $\textbf{FPML}_{\textbf{unif}}(C,\varepsilon)$ achieves expected 1-regret*

$$\mathbb{E}[\mathscr{R}_1] \leqslant \frac{\ln|\mathscr{A}|}{\varepsilon} + \frac{T|\mathscr{A}|}{C}\left(1 - \mathrm{e}^{-\varepsilon|\mathscr{A}|/C}\right)^{B-C}. \tag{4.4}$$

Proof. The probability of inclusion of any particular action in the uniform sample at each round is

$$\mathbb{P}(a \in S_t^{\text{explore}}) = \frac{\binom{|\mathscr{A}|-1}{C-1}}{\binom{|\mathscr{A}|}{C}} = \frac{C}{|\mathscr{A}|}. \tag{4.5}$$

So the estimates $\hat{r}_t(a)$ are unbiased, as

$$\mathbb{E}[\hat{r}_t(a)] = r_t(a)\frac{|\mathscr{A}|}{C} \cdot \mathbb{P}(a \in S_t^{\text{explore}}) + 0 \cdot \mathbb{P}(a \notin S_t^{\text{explore}}) = r_t(a)\frac{|\mathscr{A}|}{C}\frac{C}{|\mathscr{A}|} = r_t(a), \tag{4.6}$$

[2] While more difficult to analyse, in practice it may be attractive to, rather than using a fixed exploration budget C at each round, instead use each of the B available action slots for exploration independently with some probability γ at each round (and for exploitation otherwise); this is equivalent to sampling $C \sim \mathrm{Bin}(B, \gamma)$ freshly at each round. One advantage of this approach is finer control over the propensity to explore, particularly when B is very small. We explore this 'probabilistic exploration' algorithm empirically in Chapter 7.

and they are nonnegative and bounded above by $\frac{|\mathscr{A}|}{C}$.

The result then follows from Proposition 4.1, applied with reduced time budget $B - C$ available to the actual **FPML** algorithm. □

As in the full feedback case, various values of ε may be appropriate; a good choice is given below, in an analogous result to Theorem 3.7:

Theorem 4.3. *The algorithm* $\textbf{FPML}_{\textbf{unif}}(C,\varepsilon)$ *run with* $\varepsilon = \frac{C}{|\mathscr{A}|}\left(\frac{\ln|\mathscr{A}|}{T}\right)^{1/(B-C+1)}$ *achieves expected 1-regret*

$$\mathscr{O}\left(T^{\frac{1}{B-C+1}} C^{-1} |\mathscr{A}| (\log|\mathscr{A}|)^{\frac{B-C}{B-C+1}}\right). \tag{4.7}$$

In particular, for $C = 1$ *this is*

$$\mathscr{O}\left(T^{\frac{1}{B}} |\mathscr{A}| (\log|\mathscr{A}|)^{\frac{B-1}{B}}\right) \tag{4.8}$$

and for $B = \Omega(\log T)$ *this becomes* $\mathscr{O}(|\mathscr{A}| \log|\mathscr{A}|)$.

Proof. Immediate from Proposition 4.2 by applying the approximation $1 - e^{-x} \leqslant x$; the value $\varepsilon = \frac{C}{|\mathscr{A}|}\left(\frac{\ln|\mathscr{A}|}{T}\right)^{1/(B-C+1)}$ is chosen by setting the resulting two terms equal. □

We will use $\textbf{FPML}_{\textbf{unif}}$ to refer to $\textbf{FPML}_{\textbf{unif}}(C,\varepsilon)$ run with this value of ε and with $C = 1$.

4.3 Using data from non-exploration actions

So far to inform our estimates we've only used the data gathered from actions explicitly taken for exploration; intuitively it is desirable to also make use of the abundant reward data from the remaining actions (chosen for exploitation).

One obvious way to do this is to adapt the 'inverse propensity' estimates used in the last section to use the *overall* probability of an action being taken, including any exploration:

$$\hat{r}_t(a) := \begin{cases} \frac{r_t(a)}{\mathbb{P}(a \in S_t)} & \text{if } a \in S_t, \\ 0 & \text{otherwise.} \end{cases} \tag{4.9}$$

It is simple to check that this gives an unbiased estimator. The problem is that the selection probabilities $\mathbb{P}(a \in S_t)$ implied by **FPML** are not expressible in a simple closed form, and are computationally expensive to numerically calculate. An alternative is to estimate them empirically by repeating the selection at each round many times and observing how frequently each action is taken. This problem was addressed by Neu and Bartók [NB13] for the classical **FPL** algorithm in the combinatorial linear bandit setting; they proposed a technique called *geometric resampling* that improves on this second option by vastly reducing the number of re-runs necessary at each round, and we adapt this technique here.

Consider $\textbf{FPML}(\varepsilon)$ running directly on reward estimates as in Section 4.1. For each round $t \in [T]$ and each action $a \in \mathscr{A}$ let $q_{t,a} := \mathbb{P}(a \in S_t \mid \mathscr{F}_{t-1})$. Consider the following estimates:

$$\hat{r}_t(a) := \begin{cases} 1 - (1 - r_t(a)) \min(Z_{t,a}, M) & \text{if } a \in S_t, \\ 1 & \text{otherwise,} \end{cases} \tag{4.10}$$

where $Z_{t,a}$ is a Geom($q_{t,a}$)-distributed random variable and $M \in \mathbb{N}$ is a fixed hyperparameter. We first show that $\min(Z_{t,a}, M)$ is a good guess at $1/q_{t,a}$ and so these estimators are close to unbiased.[3]

Remark. These estimators approximate 'ideal' estimators of the form

$$1 - \hat{r}_t(a) := \begin{cases} \frac{1-r_t(a))}{\mathbb{P}(a \in S_t)} & \text{if } a \in S_t, \\ 0 & \text{otherwise,} \end{cases} \tag{4.11}$$

instead of as in Eq. (4.9), i.e. we're replacing $r_t(a)$ with $1 - r_t(a)$. Equivalently, we're approximating estimators of the form in Eq. (4.9) applied to *costs* $c_i(a) := 1 - r_i(a)$ instead of rewards.

There is an important reason for this. Both Eq. (4.9) and Eq. (4.11) are unbiased estimators (for either cost or reward), but they have subtly different properties, and crucially, the cost-based estimators in Eq. (4.11) have a self-stabilizing character: if an action is not taken for several rounds its estimated cumulative reward will gradually exceed those of actions that *have* been taken, incentivizing the algorithm to once again take it. This forms a sort of 'automatic exploration', and we will see that we can actually prove an asymptotically small regret bound for **FPML** when using (our approximations to) these estimators even without performing any explicit exploration.[4] This is an elegant and surprising symmetry break, as in all other respects using rewards and costs are equivalent; this phenomenon was commented on briefly by Poland [Pol05] in the context of classical **FPL** with standard statistical resampling.

Lemma 4.4. *For each round t and action a, given the history $\mathscr{F}_{t-1}$ the estimated reward in Eq.* (4.10) *has conditional expectation*

$$\mathbb{E}[\hat{r}_t(a) \mid \mathscr{F}_{t-1}] = r_t(a) + (1 - q_{t,a})^M (1 - r_t(a)). \tag{4.12}$$

Proof. Deferred to Appendix A; the argument rests on showing that

$$\mathbb{E}[\min(Z_{t,a}, M) \mid \mathscr{F}_{t-1}] = \frac{1 - (1 - q_{t,a})^M}{q_{t,a}}. \tag{4.13}$$

□

The random variables $Z_{t,a}$ can be sampled by repeatedly reapplying random (exponential) perturbations to every action and stopping when action a is in the top B perturbed actions; see Algorithm 7. Indeed, on any of the repetitions in step 4 of the algorithm, the probability of a particular action a being in the top B is $q_{t,a}$, and so the $Z_{t,a}$ indeed must be geometrically distributed with parameter $q_{t,a}$ if $M = \infty$; thus $\min(Z_{t,a}, M)$ has the truncated geometric distribution needed, so Lemma 4.4 applies. This allows us to prove a nice regret bound:

Proposition 4.5. *Algorithm **FPML**$_{\mathbf{GR}}(M,\varepsilon)$ achieves expected 1-regret*

$$\mathbb{E}[\mathscr{R}_1] \leqslant \frac{\ln|\mathscr{A}|}{\varepsilon} + T(1 - \mathrm{e}^{-\varepsilon M})^B + \frac{T|\mathscr{A}|}{\mathrm{e}M}. \tag{4.15}$$

[3] Taking the minimum with M introduces bias but is necessary to avoid potentially unbounded sampling time for $Z_{t,a}$ in Algorithm 7.

[4] Formally, if our geometric resampling estimators were instead based on Eq. (4.9), the corresponding result to Lemma 4.4 would instead prove that each estimator is a slight *under*estimate and the argument at the very beginning of the proof of Proposition 4.5 would then fail.

Algorithm 7: FPML$_{\mathbf{GR}}$ (M,ε) (an adaptation of **FPML**(ε) to the partial feedback case using *geometric resampling*).

Input: *parameter $M \in \mathbb{N}$, parameter $\varepsilon > 0$, available actions $\mathscr{A}$, budget B, rewards $r_t(a)$ received after each round $t \in [T]$* ***only for the actions taken by the algorithm****.*

Output: *action sets $S_1, \dots, S_T$ all of size at most B.*

For each $t \in [T]$:

1. Independently draw random perturbations $p_t(a) \sim \mathrm{Exp}(\varepsilon)$ for each action $a \in \mathscr{A}$, and define $\tilde{R}_{t-1}(a) := p_t(a) + \hat{R}_{t-1}(a)$ to be the *estimated* perturbed total reward for action a so far.
2. Enumerate the actions $a \in \mathscr{A}$ in decreasing order of $\tilde{R}_{t-1}(a)$ and choose S_t to be the set containing the first B actions in this list.
3. Receive feedback $r_t(a)$ for the actions $a \in S_t$.
4. For $k = 1, \dots, M$, repeat steps 1 and 2 above (re-drawing the random perturbations) and let $\mathscr{X}_k$ be the set of B actions chosen. If at any point every action in S_t has been re-chosen at least once we may move straight to the next step.
5. For each action $a \in S_t$ define $Z_{t,a}$ to be the first k such that $a \in \mathscr{X}_k$, or ∞ if no such k exists.
6. For each action $a \in \mathscr{A}$, estimate the reward for this round as:

$$\hat{r}_t(a) := \begin{cases} 1 - (1 - r_t(a)) \min(Z_{t,a}, M) & \text{if } a \in S_t, \\ 1 & \text{otherwise} \end{cases} \tag{4.14}$$

   (as in Eq. (4.10)) for use in future rounds.

In particular, when run with $\varepsilon = \left(\frac{\ln|\mathscr{A}|}{T} \left(\frac{\ln|\mathscr{A}|}{T|\mathscr{A}|} \right)^B \right)^{1/(2B+1)}$ and $M = \left(|\mathscr{A}| \left(\frac{T|\mathscr{A}|}{\ln|\mathscr{A}|} \right)^B \right)^{1/(2B+1)}$ this regret is of order

$$\mathbb{E}[\mathscr{R}_1] = \mathscr{O}\left(T^{\frac{B+1}{2B+1}} (|\mathscr{A}| \log|\mathscr{A}|)^{\frac{B}{2B+1}} \right).^5 \tag{4.16}$$

Proof. Deferred to Appendix A. □

[5] In the classical case $B = 1$ this bound is $\mathscr{O}(T^{2/3}(|\mathscr{A}| \log|\mathscr{A}|)^{1/3})$. Compare this to the regret bound obtained in Neu and Bartók [NB13] for **FPL** with geometric resampling of $\mathscr{O}(\sqrt{T|\mathscr{A}| \log|\mathscr{A}|})$. Ours is worse in T but better in $|\mathscr{A}|$ (the latter stems from using the cap M to bound the estimates, whereas they used a variance argument that did not incorporate M; we've essentially shown that the choice of M is a trade-off between bias and variance of the estimators, and that picking a smaller value can actually be beneficial). It is quite possible that a tighter analysis using round-specific bounds on the estimated rewards rather than the deterministic global bounding interval $[1 - M, M]$ would yield a slightly improved result in T. Further thinking along this line would be an interesting direction for future work.

Chapter 5

Lower Regret Bounds

We move now to some initial lower bounds on the achievable regret for the multitasking bandit problem.

5.1 Definitions

Let's start by formalizing the notion of an environment (problem instance) and policy (algorithm).[1]

Definition 5.1. *An* environment *E is a set of fixed rewards $r_t(a) \in [0,1]$ indexed by $t \in [T]$ and $a \in \mathscr{A}$ for some* horizon *$T \geqslant 1$ and* action set *$\mathscr{A}$. Formally, E is a mapping[2] $\mathscr{A} \times [T] \longrightarrow [0,1]$.*

We write $\mathscr{E}(T, \mathscr{A})$ to refer to the set of all environments with horizon T and action set $\mathscr{A}$.

Definition 5.2. *A* B-policy *π is a function from a horizon T and action set $\mathscr{A}$ to a sequence of T mappings*

$$(S_1, (r_1(a))_{a \in \mathscr{A}}, \dots, S_{t-1}, (r_{t-1}(a))_{a \in \mathscr{A}}) \mapsto P_t \tag{5.1}$$

for $t \in [T]$, where each S_t is a subset of $\mathscr{A}$ of size at most B and each P_t is a distribution over subsets S_t of size at most B.[3]

We write $\Pi(B)$ for the set of all B-policies.

5.2 Lower-bounding the 1-regret

We start by bounding the 1-regret; we'll then prove a lower bound for $N > 1$ by reducing to this case.

Lemma 5.3. *The maximum of k independent* $\mathrm{Bin}(n, 1/n)$ *random variables has expected value $\Theta(\log k / \log\log k)$ for large n.*

[1] In what follows we will consider 'randomized' environments, which are formally distributions over environments. If there is a distribution over environments on which an algorithm incurs high regret in expectation, there certainly must be some specific environment in which the algorithm incurs high (expected) regret. Similarly, a randomized policy is simply a distribution over deterministic policies (as defined above), so for proving lower bounds it will suffice to focus on the case of deterministic policies; the results will generalize immediately to randomized ones.

[2] Note this models the setting of an *oblivious adversary*. This is the main setting we're interested in but it could be interesting to generalize our earlier results to adaptive adversaries.

[3] This policy definition models full feedback algorithms; in the partial feedback setting we allow dependence only on $(S_1, (r_1(a))_{a \in S_1}, \dots, S_{t-1}, (r_{t-1}(a))_{a \in S_{t-1}})$.

Proof. Partly based on an argument from Gnedenko [Gne43] (in the proof of Theorem 1); see Appendix A for the details. □

Theorem 5.4. *For any $\pi \in \Pi(B)$, any horizon T such that $B = \Omega(\log T)$, and any action set $\mathscr{A}$ there exists an environment $E \in \mathscr{E}(T, \mathscr{A})$ in which π has 1-regret*

$$\mathbb{E}[\mathscr{R}_1] = \Omega\left(\frac{\log|\mathscr{A}|}{\log\log|\mathscr{A}|} - \log T\right).^{4} \tag{5.2}$$

Proof. Draw i.i.d. Bern$(1/T)$ rewards $(r_t(a))_{t\in[T],a\in\mathscr{A}}$. The 1-regret is

$$\mathscr{R}_1 = \max_{a\in\mathscr{A}} \sum_{t=1}^{T} r_t(a) - \sum_{t=1}^{T} r_t(S_t) \tag{5.3}$$

where $(S_t)_{t\in[T]}$ are the action-sets chosen by π. The first term is the maximum of $|\mathscr{A}|$ independent Bin$(T, 1/T)$ random variables, so by Lemma 5.3 has expectation $\Theta(\log|\mathscr{A}|/\log\log|\mathscr{A}|)$ for large T. The second term has expectation

$$T\mathbb{P}(\exists a \in S_1 \,:\, r_1(a) = 1) = T[1 - (1 - 1/T)^B]. \tag{5.4}$$

For $B = \log T$ this is $\Theta(\log T)$ (from the Puiseux series at infinity). So the policy regret is $\mathbb{E}[\mathscr{R}_1] = \Theta(\log|\mathscr{A}|/\log\log|\mathscr{A}| - \log T)$.[5] □

Remark. This is more an 'example' proof than anything, and illustrates that this standard technique for lower bounds in the classical bandit setting has more limited utility when $B > 1$. Other, better lower bounds are possible (some are given in the accompanying conference paper but omitted here for lack of space), and in general this is an an important direction for future work.

5.3 Lower-bounding the N-regret

We can reduce to the above case as follows (and this argument applies to any lower bound on the 1-regret):

Theorem 5.5. *If all the rewards are chosen i.i.d.* Bern$(T, 1/T)$ *then any B-policy will have N-regret*

$$\mathbb{E}[\mathscr{R}_N] = \Omega\left(N\frac{\log(|\mathscr{A}|/N)}{\log\log(|\mathscr{A}|/N)} - N\log T\right) \tag{5.5}$$

when $B = \Omega(N \log T)$.

Proof. Draw i.i.d. Bern$(1/T)$ rewards $(r_t(a))_{t\in[T],a\in\mathscr{A}}$. So every B-set chosen by the algorithm will receive reward at round t with probability $1 - (1 - 1/T)^B$, so the algorithm's expected reward is $T(1 - (1 - 1/T)^B)$ which is $\Theta(N \log T)$ for $B = N \log T$.

[5] It is possible using Bern$(T, 1/BT)$ to achieve a lower bound that grows as the maximum of $|\mathscr{A}|$ independent Bin$(T, 1/BT)$ random variables (without any $-\log T$ term), which may be a more useful result from some perspectives.

[5] The upper regret bound for the full feedback **FPML**(ε) algorithm with $\varepsilon = \ln\ln|\mathscr{A}|$ shown in Theorem 3.6 matches this asymptotically in $|\mathscr{A}|$. It would be interesting to find a parameterization of the algorithm that matches a lower bound in T too.

Now, loosely following the proof of Theorem 14 in Streeter and Golovin [SG08], partition $\mathscr{A}$ into N bins, each containing $|\mathscr{A}|/N$ actions; let $a_i^\star$ be the best action in the i^{th} bin and write $S^\star := \{a_1^\star, \dots, a_N^\star\}$. Let $S = \{a_1, \dots, a_N\}$ contain a randomly selected action from each bin.

For each $i \in [N]$ randomly mark $x_i := R_T(a_i^\star) - R_T(a_i)$ of the rounds at which $a_i^\star$ *did* receive a reward; let M_i be the marked rounds and U_i be the unmarked rounds (still only at which a reward was received). Noting $|U_i| = R_T(a_i)$, for any round t we have $\mathbb{P}(t \in U_i) = \mathbb{P}(r_t(a_i) = 1) = 1/T$, so $\mathbb{P}(t \in \bigcup_i U_i) = 1 - (1 - 1/T)^N$. Thus $\mathbb{E}[|\bigcup_i U_i|] = T(1 - (1 - 1/T)^N)$ which is $\Theta(N)$.

We now wish to bound $X := |\bigcup_i M_i \setminus \bigcup_i U_i|$. Define Y to be the number of rounds at which exactly one action in $S^\star$ receives a reward, and let ξ be the event $\{\forall a \in \mathscr{A}\ R_T(a) \leqslant T/N\}$. Fixing $i \in [N]$ and $t \in M_i$, the probability that exactly one action in $S^\star$ receives a reward at round t is

$$\mathbb{P}(r_t(a_j^\star) = 0\ \forall j \neq i \mid \xi) = \prod_{j \neq i} \left(1 - \frac{R_T(a_j^\star)}{T}\right) \mid \xi \tag{5.6}$$

$$\geqslant (1 - 1/N)^{N-1} \geqslant \mathrm{e}^{-1}, \tag{5.7}$$

so $\mathbb{E}[Y \mid \xi] \geqslant \mathrm{e}^{-1}\mathbb{E}[\sum_i |M_N|] = \mathrm{e}^{-1}N\mathbb{E}[x_1]$, and by the result from Lemma 5.3 on $\mathrm{Bin}(T, 1/T)$ maxima, $\mathbb{E}[x_1] = \Omega(\log(|\mathscr{A}|/N)/\log\log(|\mathscr{A}|/N))$.

Finally, note that by Hoeffding's inequality

$$\mathbb{P}(\xi) \geqslant 1 - |\mathscr{A}| \cdot \mathbb{P}(R_T(a) \geqslant T/N + 1) \geqslant 1 - \mathrm{e}^{-2T/N^2} = 1 - o(1) \tag{5.8}$$

and $\mathbb{E}[X] \geqslant \mathbb{E}[Y] \geqslant \mathbb{E}[Y \mid \xi]\mathbb{P}(\xi)$ (the first inequality since if a round is marked but only one action in $S^\star$ receives a reward then it cannot also be unmarked), so using that $\mathbb{E}[R_T(S^\star)] = \mathbb{E}[X] + \mathbb{E}[|\bigcup_i U_i|]$ gives the result. □

Chapter 6

Online Submodular Function Maximization

In a closely related work that we mentioned briefly in Section 1.2, Streeter and Golovin [SG08] introduced an online greedy algorithm for maximization of submodular functions (a problem of which the multitasking bandit problem is a special case). In this chapter we first prove a modification to their regret analysis that allows comparison to the regret bounds in this work. Next, we show that our **FPML** algorithm (and multitasking bandit algorithms in general) can be used as a subroutine in an adaptation of theirs to achieve better performance in various cases. In Chapters 7 and 8 we will empirically compare **FPML** with the algorithms discussed in this chapter.

6.1 Background and notation

Consider some preliminary definitions:[1]

Definition 6.1. *Let an action now be an activity-duration pair* $a = (v, \tau) \in \mathcal{V} \times X = \mathcal{A}$ *for some fixed finite set of* activities $\mathcal{V}$ *and some set* $X \subseteq (0, \infty)$ *of allowable durations.*[2]

Define a schedule *to be a finite sequence of actions, and let* $\mathcal{S}$ *be the set of all schedules. The* length $\ell(S)$ *of a schedule* $S \in \mathcal{S}$ *is the sum of the durations of all the actions in* S*. Write* $S_{\langle \tau \rangle}$ *for the prefix of length* τ *of a schedule* S*.*

Finally, define a job *to be a function* $f : \mathcal{S} \longrightarrow [0, 1]$ *such that for any schedules* $S_1, S_2 \in \mathcal{S}$ *and any action* $a \in \mathcal{A}$*:*

1. $f(S_1) \leqslant f(S_1 \oplus S_2)$ *and* $f(S_2) \leqslant f(S_1 \oplus S_2)$ ***(monotonicity)****;*
2. $f(S_1 \oplus S_2 \oplus \langle a \rangle) - f(S_1 \oplus S_2) \leqslant f(S_1 \oplus \langle a \rangle) - f(S_1)$ ***(submodularity)****.*

Streeter and Golovin [SG08] then tackle the following problem:

Definition 6.2 (Online submodular function maximization)**.** *The problem consists of a game with* T *rounds. We are given some fixed budget* $B > 0$ *and at each round* $t \in [T]$ *we must choose*

[1] We use notation based on ours from the rest of this work rather than following that of Streeter and Golovin [SG08].

[2] We will in practice enforce integer durations (i.e. $X = \mathbb{N}$) so that there are only finitely many possible actions to choose from given a duration constraint.

a schedule $S_t \in \mathscr{S}$ with $\mathbb{E}[\ell(S_t)] \leqslant B$ to be evaluated by a job f_t which is only revealed after our choice. The goal is to maximize the cumulative output $\sum_{t=1}^{T} f(S_t)$.

It is worth immediately noting that the full feedback multitasking bandit problem is a special case of this where actions are constrained to be unit-duration (i.e. $\mathscr{A} = \mathscr{V} \times \{1\}$) and each f_t takes the form $\max_{a \in S} r_t(a)$ (in this case the order of actions in a schedule does not matter).

Streeter and Golovin [SG08] use the following regret notion:

Definition 6.3 (Regret). *The $(1 - \mathrm{e}^{-1})$-approximation B-regret of an algorithm solving the above problem is*

$$\mathscr{R}_B^{approx} := \left(1 - \mathrm{e}^{-1}\right) \max_{S \in \mathscr{S} \,:\, \ell(S) \leqslant B} \sum_{t=1}^{T} f_t(S) - \sum_{t=1}^{T} f_t(S_t) \tag{6.1}$$

where $S_1, \ldots, S_T$ are the schedules output by the algorithm.

The authors propose an online greedy algorithm **OG** (Algorithm 8) which uses an experts algorithm (a full feedback classical multi-armed bandit algorithm) such as **WMR** as a subroutine.

Algorithm 8: OG

Input: *Budget $B \in \mathbb{N}$, experts algorithm $\mathscr{E}$, available activities $\mathscr{V}$, job f_t revealed after each round $t \in [T]$.*

Output: *schedules $S_1, \ldots, S_T \in \mathscr{S}$ all with expected length at most B.*

Let $\mathscr{E}_1, \ldots, \mathscr{E}_B$ be separate instances of the experts algorithm $\mathscr{E}$.

For $t \in [T]$:

1. Let $S_t^{(0)} = \langle\,\rangle$ be the empty schedule.
2. For each $i \in [B]$:
   (a) Use $\mathscr{E}_i$ to choose an action $a_t^{(i)} = (v, \tau) \in \mathscr{A}$.
   (b) With probability $1/\tau$ set $S_t^{(i)} := S_t^{(i-1)} \oplus \langle a_t^{(i)} \rangle$, else set $S_t^{(i)} := S_t^{(i-1)}$.
3. Set $S_t := S_t^{(B)}$.
4. Receive the job f_t.
5. For each $i \in [B]$ and each action $a = (v, \tau) \in \mathscr{A}$ feed back the reward

$$r_t^{(i)}(a) := \frac{f_t(S_t^{(i-1)} \oplus \langle a \rangle) - f_t(S_t^{(i-1)})}{\tau} \tag{6.2}$$

   to experts algorithm $\mathscr{E}_i$.

They prove the following regret bound:

Theorem 6.4. *Algorithm **OG** run with **WMR** as the experts algorithm and time allowance B has*

$$\mathbb{E}[\mathscr{R}_B^{\text{approx}}] = \mathscr{O}\left(\sqrt{BT \log |\mathscr{A}|}\right). \tag{6.3}$$

We first look to prove a more general regret bound for the same algorithm.

6.2 A more general regret bound

In this section we generalize the analysis in Streeter and Golovin [SG08] of the algorithm **OG** to the case where the algorithm is competing against time resources N not necessarily equal to its budget B.

We start by showing a key preliminary result, which is a generalization of Theorem 6 in Streeter and Golovin [SG08].

Lemma 6.5. *Let f be any job and let $\bar{G} = \langle \bar{g}_1, \bar{g}_2, \ldots \rangle$ be an infinite 'greedy' schedule satisfying*

$$\frac{f(\bar{G}_j \oplus \bar{g}_j) - f(\bar{G}_j)}{\bar{\tau}_j} \geqslant \max_{(v,\tau)\in\mathscr{V}\times(0,\infty)} \left(\frac{f(\bar{G}_j \oplus \langle (v,\tau) \rangle) - f(\bar{G}_j)}{\tau} \right) - \varepsilon_j, \qquad j \geqslant 1 \tag{6.4}$$

for additive errors $\varepsilon_1, \varepsilon_2, \ldots \geqslant 0$, where $\bar{g}_j = (\bar{v}_j, \bar{\tau}_j)$ and $\bar{G}_j = \langle \bar{g}_1, \ldots, \bar{g}_{j-1} \rangle$ for each $j \geqslant 1$.

Then for any $L, B_0 \in \mathbb{N}$ and for $B' := \sum_{j=1}^{L} \bar{\tau}_j$,

$$f(\bar{G}_{\langle B' \rangle}) > \left(1 - \mathrm{e}^{-B'/B_0}\right) f(S^\star_{B_0}) - \sum_{j=1}^{L} \varepsilon_j \bar{\tau}_j \tag{6.5}$$

where $S^\star_{B_0} := \arg\max_{S\in\mathscr{S}\,:\,\ell(S)=B_0} f(S)$ is the best schedule of length B_0 for f.

Proof. See Appendix A. □

Fix a sequence $f_1, \ldots, f_T$ of jobs we're fed, and define

$$S^\star_N := \arg\max_{S\in\mathscr{S}\,:\,\ell(S)\leqslant N} \sum_{t=1}^{T} f_t(S) \tag{6.6}$$

to be the best-in-hindsight fixed schedule using N time resources or less. Lemma 6.5 motivates the following generalized regret definition:

Definition 6.6. *For any sequence $S_1, \ldots, S_T$ of schedules and any $B, N \in \mathbb{N}$, define*

$$\mathscr{R}^{\text{approx}}_{(B,N)} := \left(1 - \mathrm{e}^{-B/N}\right) \sum_{t=1}^{T} f_t(S^\star_N) - \sum_{t=1}^{T} f_t(S_t) \tag{6.7}$$

(so that $\mathscr{R}^{\text{approx}}_{(B,B)} = \mathscr{R}^{\text{approx}}_{B}$).

OG achieves a nice guarantee on this altered regret:

Proposition 6.7. *For any $N \in \mathbb{N}$ algorithm* **OG** *run with budget $B \in \mathbb{N}$ produces a sequence of schedules (each of length at most B in expectation) satisfying*

$$\mathbb{E}[\mathscr{R}^{\text{approx}}_{(B,N)}] \leqslant \mathbb{E}\left[\sum_{i=1}^{B} \mathscr{R}_1(\mathscr{E}_i)\right], \tag{6.8}$$

where $\mathscr{R}_1(\mathscr{E}_i)$ is the 1-regret incurred (over all T rounds) by the i^{th} subroutine experts algorithm.

Proof. As argued in Streeter and Golovin [SG08], we may view the sequence of actions $a^{(i)}_1, \ldots, a^{(i)}_T$ selected by each experts algorithm $\mathscr{E}_i$ as a single 'meta-action' $\tilde{a}_i \in \mathscr{A}^T$; so the schedules

$S_1, \dots, S_T$ output by **OG** can be viewed as a single 'meta-schedule' $\tilde{S} = \langle \tilde{a}_1, \dots, \tilde{a}_B \rangle$ over $\mathscr{A}^T$ which is a version of the greedy schedule $\bar{G}_{B+1}$ for the job $f = \frac{1}{T}\sum_{t=1}^{T} f_t$, and it may be assumed that each meta-action $\tilde{a}_t$ takes unit time per job. Thus we may write

$$\mathscr{R}^{\text{approx}}_{(B,N)} = T\left[\left(1 - \mathrm{e}^{-B/N}\right) f(S^\star_N) - f(\tilde{S})\right] \tag{6.9}$$

(after extending the domain of f appropriately). Applying Lemma 6.5 with $L = B$, $B_0 = N$, $B' = \sum_{i=1}^{B} \bar{\tau}_i = B$ (by the unit-time assumption) then immediately gives

$$\mathscr{R}^{\text{approx}}_{(B,N)} < T\sum_{i=1}^{B} \bar{\tau}_i \varepsilon_i = T\sum_{i=1}^{B} \varepsilon_i. \tag{6.10}$$

Taking expectations and using that $\mathbb{E}[\varepsilon_i] = \mathbb{E}[\mathscr{R}_1(\mathscr{E}_i)/T]$ as argued in Streeter and Golovin [SG08] gives the result:

$$\mathbb{E}[\mathscr{R}^{\text{approx}}_{(B,N)}] \leqslant T\sum_{i=1}^{B} \mathbb{E}[\varepsilon_i] = T\sum_{i=1}^{B} \mathbb{E}\left[\frac{\mathscr{R}_1(\mathscr{E}_i)}{T}\right]. \tag{6.11}$$

□

In particular, we get the following behaviour:

Definition 6.8. *For any sequence $S_1, \dots, S_T$ of schedules and any $N \in \mathbb{N}$, the N-*regret *is defined as*

$$\mathscr{R}_N := \sum_{t=1}^{T} f_t(S^\star_N) - \sum_{t=1}^{T} f_t(S_t). \tag{6.12}$$

Theorem 6.9. *Let $B = \Omega(N \log T)$ and assume $B \geqslant N$. Then the algorithm* **OG** *produces a sequence of schedules (each of length at most B in expectation) with N-regret satisfying*

$$\mathbb{E}[\mathscr{R}_N] = \mathscr{O}\left(\mathbb{E}\left[\sum_{i=1}^{B} \mathscr{R}_1(\mathscr{E}_i)\right]\right). \tag{6.13}$$

In particular, when run with ***WMR*** *as the subroutine experts algorithm, this is $\mathscr{O}\left(\sqrt{BT \log |\mathscr{A}|}\right)$.*[3]

Proof. This follows quickly from Proposition 6.7: since $B = \Omega(N \log T)$, there must be some constant $c > 0$ such that $B \geqslant cN \ln T$ for all T large enough. So $\mathrm{e}^{-B/N} \leqslant \mathrm{e}^{-c \ln T} = T^{-c}$. Thus

$$\mathscr{R}^{\text{approx}}_{(B,N)} \geqslant \left(1 - T^{-c}\right)\sum_{t=1}^{T} f_t(S^\star_N) - \sum_{t=1}^{T} f_t(S_t) = \mathscr{R}_N - T^{-c}\sum_{t=1}^{T} f_t(S^\star_N). \tag{6.14}$$

It follows that

$$\mathscr{R}_N \leqslant \mathscr{R}^{\text{approx}}_{(B,N)} + T^{-c}\sum_{t=1}^{T} f_t(S^\star_N) \leqslant \mathscr{R}^{\text{approx}}_{(B,N)} + T^{-c} \cdot T. \tag{6.15}$$

[3] Compare this to Theorem 6.4; in other words, allowing time resources that grow logarithmically in the number of rounds (relative to the time constraint we're benchmarking against) is sufficient to be able to perform asymptotically *as well as* the best benchmark schedule, rather than just a fraction $1 - \frac{1}{e}$ times as well.

It remains just to note that, since $B \geqslant N$, we must have $c \geqslant 1$ (otherwise there'd be some T large enough that $1/\ln T < c$ so $B < N$)), and so $\mathscr{R}_N \leqslant \mathscr{R}^{\text{approx}}_{(B,N)} + 1$. The result follows.

The bound $\mathbb{E}\left[\sum_{i=1}^{B} \mathscr{R}_1(\mathscr{E}_i)\right] = \mathscr{O}(\sqrt{BT \log |\mathscr{A}|})$ when using **WMR** was shown in Streeter and Golovin [SG08]. □

6.3 A hybrid algorithm

A generalization of **OG** is possible that uses a *multitasking bandit algorithm* instead of a classical bandit algorithm as the subroutine: each subroutine algorithm gets control over some $B' \in [B]$ of the actions in the final set rather than just one—so the number of subroutines will decrease by a factor of B'—creating a trade-off between the greediness of **OG** and the properties whatever subroutine algorithm is used. In this section we explore how using such a 'hybrid' algorithm can improve the regret bound.

Assume from now on that all actions are of unit duration, so $\mathscr{A} = \mathscr{V} \times \{1\}$. Consider Algorithm 9.

Algorithm 9: $\text{OG}_{\text{hybrid}}$

Input: *Budget $B \in \mathbb{N}$, full feedback multitasking bandit algorithm $\mathscr{B}$, available actions $\mathscr{A}$, batch size B', jobs f_t revealed after each round $t \in [T]$.*

Output: *schedules $S_1, \ldots, S_T$ all of length at most B.*

Assume for simplicity that $B' \mid B$; define $K := B/B'$. Let $\mathscr{B}_1, \ldots, \mathscr{B}_K$ be instances of $\mathscr{B}$, each with budget B'.

For each $t \in [T]$:

1. Let $S_t^{(0)} = \langle\rangle$ be the empty schedule.
2. For each $i \in [K]$:
   (a) Use $\mathscr{B}_i$ to choose B' actions $a_t^{((i-1)B'+1)}, \ldots, a_t^{(iB')}$.
   (b) Set $S_t^{(i)} := S_t^{(i-1)} \oplus \langle a_t^{((i-1)B'+1)}, \ldots a_t^{(iB')} \rangle$.
3. Set $S_t := S_t^{(K)}$.
4. Receive the job f_t.
5. For each $i \in [K]$:
   (a) For each action $a \in \mathscr{A}$ feed back the reward

$$r_t^{(i)}(a) := f_t(\langle a_{t,1}^\star, \ldots, a_{t,i-1}^\star, a\rangle) - f_t(\langle a_{t,i-1}^\star, \ldots, a_{t,i}^\star\rangle) \tag{6.16}$$

   (see def below) to multitasking bandit algorithm $\mathscr{B}_t$.
   (b) Define $a_{t,i}^\star := \arg\max_{j \in [B']} r_t^{(i)}(a_t^{((i-1)B'+j)})$.

Intuitively, each of the K instances of $\mathscr{B}$ selects B' 'good' actions at each round to be added in a batch to the schedule (in some arbitrary order). The overall schedule submitted at each round consists of all $B = KB'$ actions chosen by any of the subroutine algorithms.

Given regret bounds for each instance of $\mathscr{B}$ relative to a single optimal action, we can nicely derive a regret bound for the overall algorithm by arguing it is related to a version of the standard **OG** algorithm with budget K. We will need a small additional assumption on the nature of the

jobs $f_1, \ldots, f_T$:

Assumption 6.10. *In addition to monotonicity and submodularity, each job $f : \mathscr{S} \to [0,1]$ also satisfies the following: for any schedules $S_1, S_2, S_3 \in \mathscr{S}$,*

$$f(S_1 \oplus S_2 \oplus S_3) \geqslant f(S_1 \oplus S_3).^4 \tag{6.17}$$

We can now prove our regret bound:

Proposition 6.11. *Under Assumption 6.10 and for $B = \Omega(NB' \log T)$, algorithm $\mathbf{OG_{hybrid}}$ experiences N-regret $\mathscr{O}(B\mathbb{E}[\mathscr{R}_1(\mathscr{E})]/B')$, where $\mathbb{E}[\mathscr{R}_1(\mathscr{B})]$ is the expected 1-regret incurred by any instance of $\mathscr{B}$.*

Proof outline. The proof proceeds by analysing an instantiation of **OG** with budget K running fictional subroutine experts algorithms $\mathscr{E}_1, \ldots, \mathscr{E}_K$ such that that $\mathscr{E}_i$ at each round t picks the action $a^\star_{t,i}$. It is argued firstly that this imaginary instantiation of **OG** incurs N-regret at least as great as that of the $\mathbf{OG_{hybrid}}$ algorithm of interest, and furthermore that each fictional experts algorithm $\mathscr{E}_i$ incurs the same 1-regret as the corresponding multitasking bandit algorithm $\mathscr{B}_i$. The result from Theorem 6.9 is then appealed to, upper-bounding the former by the sum of the latter in expectation and thus completing the proof. Full details are given in Appendix A. □

In particular, using **FPML** as our subroutine $\mathscr{B}$ gives the following regret bound:

Theorem 6.12. *Under Assumption 6.10 and for $B = \Omega(NB' \log T)$, algorithm $\mathbf{OG_{hybrid}}$ run with subroutine algorithm **FPML** experiences N-regret*

$$\mathbb{E}[\mathscr{R}_N] = \mathscr{O}\left(\frac{BT^{1/(B'+1)}(\log|\mathscr{A}|)^{B'/(B'+1)}}{B'}\right). \tag{6.18}$$

In particular, if $B' = \Omega(\log T)$ (so $B = \Omega(N(\log T)^2)$) this is $\mathscr{O}(B \log|\mathscr{A}|/\log T)$.

Proof. Immediate from Proposition 6.11 and Theorem 3.7. □

6.4 Partial feedback

Streeter and Golovin [SG08] also studied a version of their algorithm **OG** which uses **Exp3** instead of **WMR** as a subroutine to handle the *partially transparent feedback* case where only the values $f_t(S)$ for each *prefix* S of the chosen schedule S_t are revealed after each round, rather than f_t itself. For this version of **OG**, Theorem 6.9 gives an expected N-regret of $\mathscr{O}\left(\sqrt{BT|\mathscr{A}|\log|\mathscr{A}|}\right)$.

$\mathbf{OG_{hybrid}}$cannot so simply be adapted to their partially transparent feedback regime, but the special case of the partial feedback *multitasking bandit problem* considered previously—where the algorithm actually gets strictly *more* information than in this more general partially transparent feedback setting—can be handled nicely by an adaptation of $\mathbf{OG_{hybrid}}$running a partial feedback version of **FPML** as a subroutine. It is these versions of **OG** and $\mathbf{OG_{hybrid}}$that we apply in the next section.

[4] This is a slight extension of monotonicity which holds in most natural applications (and certainly in the multitasking bandit special case).

Chapter 7

Experiments: Online Hyperparameter Search

In this chapter we explore empirically a natural application of multitasking bandit algorithms to *black-box optimization*.

7.1 Background

In machine learning, model hyperparameters are often chosen via a logarithmically-spaced grid search[1]. This quickly becomes infeasible as the space of plausible hyperparameters grows, and is information-inefficient.

Many more sophisticated *black-box optimization* algorithms exist, often based on Bayesian techniques, which aim to minimize a function given only a membership (value) oracle. With a limited number of queries (function evaluations) available these can perform much better than grid search, and are well-suited to the hyperparameter selection problem. However, there is a deluge of such algorithms to choose from, some performing better than others on particular types of task; in a scenario where many hyperparameter selection problems are to be processed (e.g. in a data center or a lifelong learning agent) it may be desirable to learn over time which one of several such optimizers to apply for the most effective results. This situation is modelled by the classical multi-armed bandits problem (with bandit feedback). With a slightly-increased time/compute budget, though—or, in particular, with the availability of parallel processing—one may instead wish to learn while running *several* such optimizers independently on each task, picking the best set of hyperparameters suggested by any one of them and thus further maximizing the expected performance.[2] This is a case of the partial feedback multitasking bandit problem, and we apply our algorithms to this scenario here.

[1] I.e. a well-distributed set of candidates is identified for each hyperparameter and the model is trained with each combination of these, selecting the combination that achieves the lowest training loss.

[2] In practice for the hyperparameter selection problem various less-naïve optimizer ensembling techniques may make better use of the available compute time (e.g. in the parallelized setting by allowing information sharing between the concurrent optimizers); but our scenario is of interest as an instance of the more general situation where a learning agent may have several different learning techniques at its disposal (which in general can *only* be run independently) and may wish to learn over time a limited subset of them to apply on new learning problems which maximizes the chance of success.

7.2 Experimental setup

Recognizing the importance of black-box optimization for machine learning, in 2020 NeurIPS ran a black-box optimization competition for ML hyperparameter selection; each submitted optimizer was evaluated on a cross-section of model architectures and learning problems, with the final ranking determined by an average normalized performance across all of these. We base our approach on theirs, using Uber's Bayesmark package [Ube20], as they did, to construct and execute experiments.[3]

Specifically, for each of 184 classification/regression problems from the Penn Machine Learning Benchmarks dataset[4] we used 9 standard black-box optimization algorithms to tune both a multilayer perceptron (neural network) and a lasso regressor/classifier. On each of these $T = 368$ problems, each optimizer was allowed 20 function evaluations (i.e. training runs).[5]

The best result achieved by each optimizer was used to calculated a normalized performance score in $[0, 1]$ relative to an estimate of the best possible performance achievable and the performance achieved by a random search; our approach is based on the mean scores used by the Bayesmark package.[6] Various multitasking bandit algorithms were then run as follows: each algorithm is given a fixed budget B, the number of different optimizers it is allowed to run on each problem; the ML problems are fed to the algorithm in some particular order, and the normalized scores of the optimizers chosen by the algorithm for the current problem are used as rewards. The algorithm receives these rewards as feedback, and its overall performance is calculated by averaging the maximum reward it receives on each problem over all problems.[7]

The code for our experiments is available at `https://github.com/candidate-1034792/blackboxbandits`.

7.3 Comparison of FPML algorithm variants

We start by comparing four partial feedback versions of **FPML**:

1. **Fixed exploration:** the $\mathbf{FPML_{unif}}$ algorithm from Section 4.2, choosing C actions for exploration uniformly at each round.
2. **Probabilistic exploration:** as above, but now using each of the B action slots for uniform exploration independently with probability γ. In effect, C is now chosen stochastically at each round with distribution $\mathrm{Bin}(B, \gamma)$.
3. **Geometric resampling:** the $\mathbf{FPML_{GR}}$ algorithm from Section 4.3.

[3] The data published from the 2020 BBO competition is insufficient for us to simply use their results.

[4] This is decreased from the full dataset due to dropped datapoints for various reasons.

[5] This was very computationally-intensive, using more than 1500 hours of single-core CPU compute time; while we evaluated far fewer optimizers than in the 2020 competition, we evaluated them on over 6 times more problems.

[6] Formally, the reward for optimizer a at round t is defined as $r_t(a) := \frac{\overline{\texttt{loss}}_t(a) - \texttt{opt}_t}{\overline{\texttt{rand}}_t(a) - \texttt{opt}_t}$, where $\texttt{opt}_t$ is an estimate of the global minimum classification/regression loss achievable (at validation, not test) on the task corresponding to round t, $\overline{\texttt{rand}}_t$ to is the mean performance of a random hyperparameter search on this task (i.e. the smallest loss achieved using any hyperparameter in the random search, averaged over trials), and $\overline{\texttt{loss}}_t(a)$ is the actual averaged minimum loss of the optimizer a on this problem. Conceptually, the reward ranges from 0, when optimizer a performs as badly as a random search, to 1, when it performs as well as is possible on this task.

[7] One detail to note is that each optimizer, on each problem, is actually given two scores: one for its performance on the validation dataset it is optimizing on, and one for the performance of its chosen hyperparameters on an unseen test set. As we are interested in the dynamics of the general online learning setting we've been modelling, the bandit algorithms receive only the validation score as their rewards and we discuss only these scores here; from a practical machine learning perspective, it is likely more desirable for the rewards at each round to be the test scores instead.

Table 7.1: Mean normalized validation scores of **FPML** algorithms over black-box optimizers.

	Number of optimizers in parallel (B)				
	2	**3**	**4**	**5**	**6**
Fixed exploration	0.538	0.684	0.763	0.819	0.857
Probabilistic exploration	0.549	0.683	0.766	0.821	0.859
Resampling	0.654	0.754	0.813	0.854	0.888
Resampling and exploration	0.602	0.721	0.791	0.836	0.870

Table 7.2: Standard deviations of normalized validation scores of **FPML** algorithms over black-box optimizers.

	Number of optimizers in parallel (B)				
	2	**3**	**4**	**5**	**6**
Fixed exploration	0.0200	0.0213	0.0197	0.0164	0.0125
Probabilistic exploration	0.0203	0.0207	0.0169	0.0149	0.0132
Resampling	0.0161	0.0141	0.0099	0.0086	0.0081
Resampling and exploration	0.0284	0.0198	0.0165	0.0145	0.0115

4. **Geometric resampling *and* probabilistic exploration:** this mixes $\mathbf{FPML_{GR}}$ with the γ-probabilistic exploration from the second algorithm, thus both making use of all received information and explicitly exploring some number of actions at each round.

For our initial comparison, we set $C = 1$ and $\gamma = 1/B$ for each B, so that in expectation the same number of actions are explored where relevant. The mean rewards of each algorithm averaged over 100 trials are as in Table 7.1; the sample standard deviations of these are given in Table 7.2. See Section 7.3 for a visual representation.

As expected, the fixed-exploration and probabilistic-exploration variants perform very similarly in all cases; what's interesting is that not only does the version with geometric resampling do much better than either of these, but adding explicit exploration at the to the resampling algorithm is actually worse than using resampling only. In general the variances of the four algorithms are similar.

7.4 Comparison to online greedy algorithms

We next look at how **FPML** compares to best-in-hindsight action sets and partial feedback versions of **OG** and $\mathbf{OG_{hybrid}}$ from Chapter 6 in this black-box optimization setting. For each B we compare the following:

1. The best fixed choice of B optimizers in hindsight.
2. The partial feedback version of **FPML** with geometric resampling and no explicit exploration (i.e. algorithm $\mathbf{FPML_{GR}}$).
3. For each combination $B', K \in \mathbb{N}$ such that $B'K = B$: the algorithm $\mathbf{OG_{hybrid}}$ with the above version of **FPML** as the subroutine, using K internal instances of **FPML** each with an action budget of B'.
4. The original algorithm **OG** from Streeter and Golovin [SG08] using **Exp3** as the subroutine.

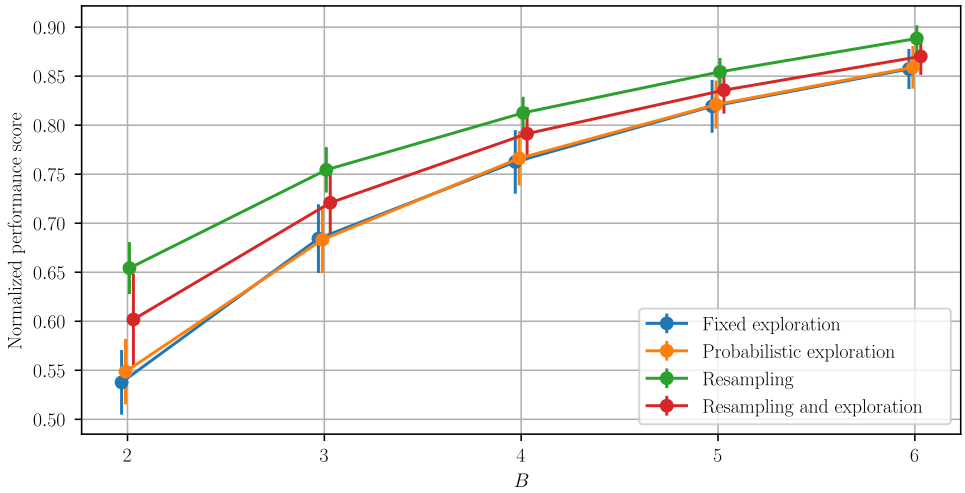

Figure 7.1: Normalized scores for **FPML** algorithms over black-box optimizers. Error bars are 95% confidence intervals for a single observation (i.e. for the underlying distribution, not the sample mean) assuming a Gaussian distribution.

Table 7.3 shows the means and standard deviations over 100 trials comparing the above algorithms for $B \in [6]$; Fig. 7.2 visualizes these results excluding instances of the third algorithm with $B' \neq 1$.

It appears that any introduction of the greediness in the **OG** algorithms harms performance, with our **FPML** algorithm surpassing all others (save the best fixed set of optimizers in hindsight) for every B and with instances of $\mathbf{OG}_{\mathbf{hybrid}}$ doing progressively worse the more internal **FPML** instances there are.[8] The variances of the various algorithms are comparable.

7.5 Problem ordering

In the above experiments, all online algorithms were fed the optimization problems in the following order: first, a multi-layer perceptron (neural network) was to be trained on each dataset, and then a lasso regressor was to be trained on each.

We now re-run the last experiment but using the transposed problem ordering: for each dataset in turn train first an MLP and then a lasso regressor. While the first problem ordering contains one major regime change, this new ordering instead models a situation where incoming tasks are well-mixed.

Fig. 7.3 shows the results corresponding to Fig. 7.2 but for this alternative problem order. The results are very similar; there is a *very* slight improvement in performance on the second ordering, particularly for small B (as might be expected, since the algorithms don't 'mislearn' optimizers that are only good at MLP) but no substantial difference.

[8]The intuition behind this phenomenon will be discussed later.

Table 7.3: Sample means and standard deviations of normalized validation scores of various combinations of **FPML** and online greedy algorithms over black-box optimizers.

(a) $B = 1$

	Mean	StD
Best in hindsight	0.574	0
FPML	0.426	0.0202
Exp3	0.351	0.0194

(b) $B = 2$

	Mean	StD
Best in hindsight	0.710	0
FPML	0.652	0.0194
OG with FPML $((B', K) = (1, 2))$	0.577	0.0187
OG with Exp3	0.519	0.0179

(c) $B = 3$

	Mean	StD
Best in hindsight	0.779	0
FPML	0.751	0.0151
OG with FPML $((B', K) = (1, 3))$	0.657	0.0191
OG with Exp3	0.617	0.0166

(d) $B = 4$

	Mean	StD
Best in hindsight	0.836	0
FPML	0.813	0.0108
OG with FPML $((B', K) = (2, 2))$	0.756	0.0149
OG with FPML $((B', K) = (1, 4))$	0.716	0.0178
OG with Exp3	0.689	0.0151

(e) $B = 5$

	Mean	StD
Best in hindsight	0.874	0
FPML	0.855	0.0094
OG with FPML $((B', K) = (1, 5))$	0.756	0.0150
OG with Exp3	0.734	0.0140

(f) $B = 6$

	Mean	StD
Best in hindsight	0.901	0
FPML	0.888	0.0072
OG with FPML $((B', K) = (3, 2))$	0.836	0.0111
OG with FPML $((B', K) = (2, 3))$	0.814	0.0143
OG with FPML $((B', K) = (1, 6))$	0.785	0.0137
OG with Exp3	0.767	0.0157

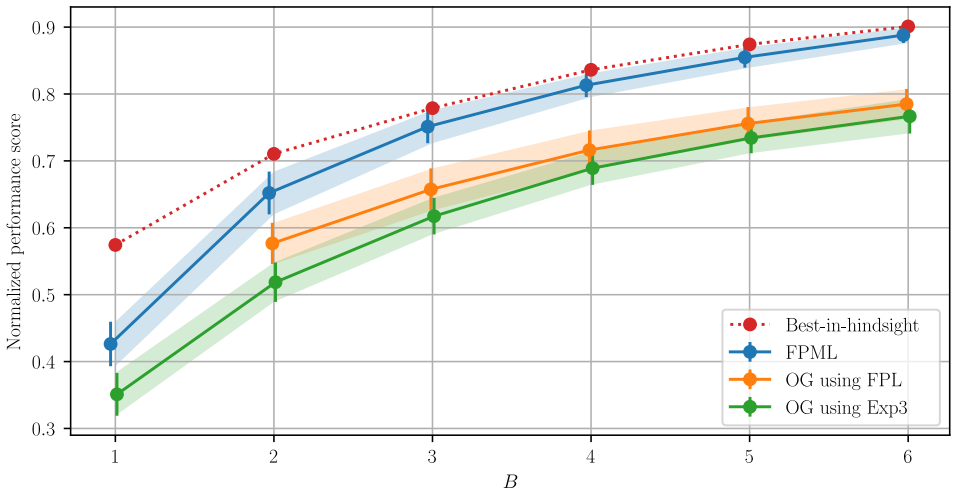

Figure 7.2: Normalized scores for various combinations of **FPML** and online greedy algorithms over black-box optimizers. Error bars are 95% confidence intervals for a single observation (i.e. for the underlying distribution, not the sample mean) assuming a Gaussian distribution.

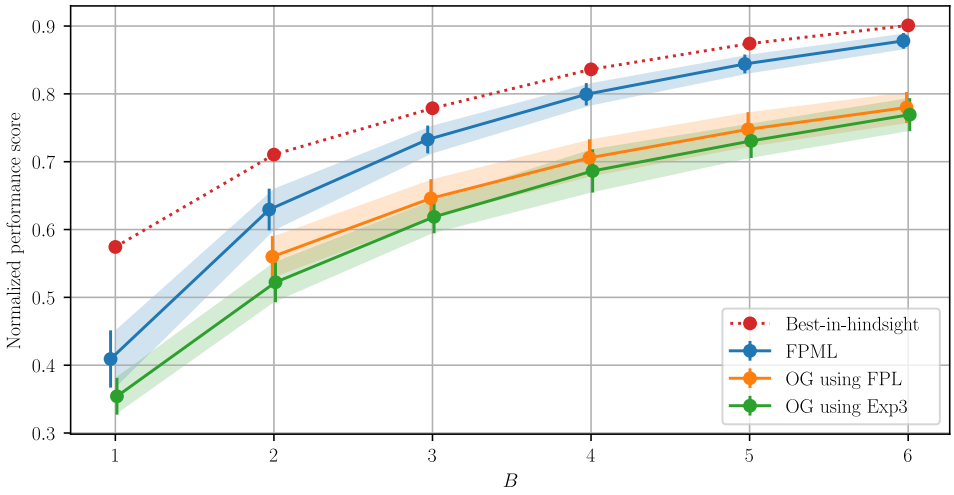

Figure 7.3: Normalized scores for various combinations of **FPML** and online greedy algorithms over black-box optimizers, *on the alternative problem ordering*. Error bars are 95% confidence intervals for a single observation (i.e. for the underlying distribution, not the sample mean) assuming a Gaussian distribution.

7.6 Discussion

So our algorithm seemingly performs much better at the black-box optimization problem than any of the greedy algorithms. Why might this be the case?

As discussed in Chapter 3, **FPML** with B actions achieves asymptotically small regret relative to the set of the top B *individual* actions in hindsight; see Proposition 3.8. On problem instances where the error $\texttt{err}_B$ defined there is small, we can expect the **FPML** algorithm to achieve much better B-regret (and hence better absolute performance) than the online greedy algorithms.[9]

The black-box optimization problem is an example of such a problem instance: Table 7.4 shows the scores of the optimal B-subset and of the set of the top B actions in hindsight for each $B \in [9]$; Table 7.5 shows which optimizers each of these sets consisted of. It can be seen that **(a)** the reward difference $\texttt{err}_B$ between the two is very small for every value of B and **(b)** the underlying sets are very similar, explaining this similarity in reward.

This observation explains the good performance of **FPML** on this problem. In the next chapter we explore synthetic problem instances to shed further light on when we can and can't expect **FPML** to outperform greedy algorithms.

[9]Recall that **OG** only has asymptotically small $(1-e^{-1})$-approximation B-regret, not small B-regret itself. Intuitively, small $\texttt{err}_B$ means greediness is not necessary, as the actions aren't sufficiently anticorrelated for choosing the top B actions in hindsight to be *that* suboptimal, and so any of the greedy algorithms are a waste of computational resources: it pays to focus on the leaderboard and spend your exploration that way.

Table 7.4: Normalized scores of the best fixed sets in hindsight for the BBO problem vs. the tops of the leaderboard.

B	Best B-set	Top B actions	Difference
1	0.574	0.574	0
2	0.710	0.681	0.0298
3	0.779	0.779	0
4	0.836	0.818	0.0186
5	0.874	0.852	0.0226
6	0.901	0.898	0.0025
7	0.923	0.923	0
8	0.935	0.935	0
9	0.935	0.935	0

Table 7.5: The best fixed sets of optimizers in hindsight for the BBO problem vs. the tops of the leaderboard, with differences highlighted in red. Optimizers are labelled A through I.

B	Best B-set	Top B actions
1	H	H
2	G,H	H,I
3	G,H,I	G,H,I
4	B,G,H,I	E,G,H,I
5	B,D,G,H,I	A,E,G,H,I
6	A,B,D,G,H,I	A,B,E,G,H,I
7	A,B,D,E,G,H,I	A,B,D,E,G,H,I
8	A,B,C,D,E,G,H,I	A,B,C,D,E,G,H,I
9	A,B,C,D,E,F,G,H,I	A,B,C,D,E,F,G,H,I

Chapter 8

Experiments: Synthetic Environments

Motivated by the discussion at the end of the last chapter, we seek now to further explore the relative performance of the partial feedback versions of **FPML**, **OG** and $\mathbf{OG_{hybrid}}$[1] on various synthetic problem classes. Let:

- $S^\star$ be the best-in-hindsight set of B actions;
- S_{greedy} be the greedy choice of B actions in hindsight;
- S_{top} be the set of the top B actions in hindsight.

8.1 An anticorrelated environment

The first environment we examine is one where $S^\star = S_{\text{greedy}}$ and this set does better than S_{top}; greediness is better than picking the top B actions. There are $|\mathscr{A}| = 15$ available actions and two types of round, I and II, which occur with equal probability; rewards are distributed within each round according to Table 8.1. So the best fixed action set of any size up to 10 will be split evenly across actions $\{1, 2, 3, 4, 5\}$ and actions $\{11, 12, 13, 14, 15\}$—and will be the greedy choice—but for $B \leqslant 5$ the top B actions will always be in $\{1, 2, 3, 4, 5\}$.[2] We see in Fig. 8.1 that **FPML** does not outperform the greedy algorithms on this task.[3]

[1] All **FPML** instances use geometric resampling without explicit exploration.

[2] The purpose of the intermediate actions $\{6, 7, 8, 9, 10\}$ is to ensure that perturbations to the leaderboard don't result in 'accidentally better' action sets.

[3] It is perhaps surprising that **OG** and $\mathbf{OG_{hybrid}}$ does not clearly outperform **FPML** in this setting.

Table 8.1: Reward distributions for round types I and II in the first synthetic environment; Beta distributions are parameterized by mean and variance, not shape.

Action	I-rounds	II-rounds	Resulting mean
Actions 1 to 5	Beta(0.6, 0.01)	Always 0	0.3
Actions 6 to 10	Beta(0.4, 0.01)	Always 0	0.2
Actions 11 to 15	Always 0	Beta(0.2, 0.01)	0.1

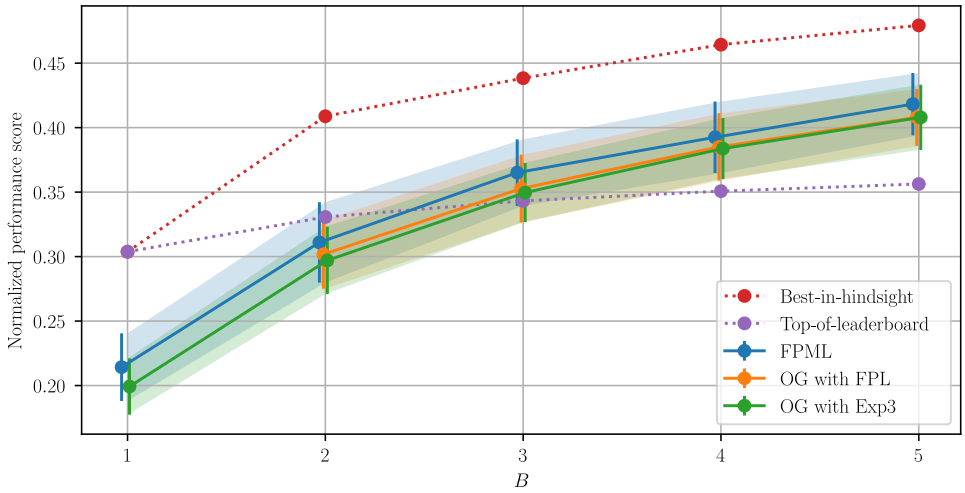

Figure 8.1: Normalized scores for various combinations of **FPML** and online greedy algorithms over 300 rounds from the first first synthetic environment, averaged over 50 trials. Error bars are 95% confidence intervals for a single observation. Note also the large gap in this environment between the performance of the best-in-hindsight action sets and the top B individual actions, as designed.

8.2 An anticorrelated environment with no reward gap

The second environment is one where (approximately) $S^\star = S_{\text{greedy}} = S_{\text{top}}$; greediness is good but no better than picking the top B actions. There are now $|\mathscr{A}| = 10$ available actions and rewards are distributed according to Table 8.2; as before the best action subsets for this environment will be evenly split between $\{1, 2, 3, 4, 5\}$ and $\{6, 7, 8, 9, 10\}$, but now the leaderboard looks like $1, 6, 2, 7, 3, 8, 4, 9, 5, 10$ in expectation, so picking the top B actions will automatically result in a well-balanced set covering both round types. For this reason, although the two groups of actions here are highly anticorrelated, we expect $\texttt{err}_B$ to be small in this example and thus for **FPML** to do well relative to the greedy algorithms.

Section 8.2 shows the results for 50 trials of 300 rounds on this environment, and Section 8.2 shows the same after introducing a regime change halfway through each trial — specifically, the means of the Beta distributions gradually reverse in order[4], so that actions 1 through 5 eventually have means $0.2, \ldots, 0.6$ respectively on I-rounds and similarly for actions 6 through 10 on II-rounds. In both cases, the **FPML** algorithm outperforms the others, particularly for larger B.

8.3 A correlated environment

Another case where $S^\star = S_{\text{greedy}} = S_{\text{top}}$ is when there are no groups of anticorrelated actions, and so the gap between the best set and the top of the leaderboard is trivially small. Suppose now there are $|\mathscr{A}| = 10$ available actions only one round type, with rewards distributed according to Table 8.3; the results on this environment are shown in Fig. 8.4. Predictably, we again see **FPML** outperforming the other algorithms.

[4]This is achieved by deciding which of the two regimes to use at each round by sampling a Bernoulli random variable with mean sigmoidal in the round number.

Table 8.2: Reward distributions for round types I and II in the second synthetic environment.

Action	I-rounds	II-rounds	Resulting mean
1	Beta(0.6, 0.01)	Always 0	0.3
2	Beta(0.5, 0.01)	Always 0	0.25
3	Beta(0.4, 0.01)	Always 0	0.2
4	Beta(0.3, 0.01)	Always 0	0.15
5	Beta(0.2, 0.01)	Always 0	0.1
6	Always 0	Beta(0.6, 0.01)	0.3
7	Always 0	Beta(0.5, 0.01)	0.25
8	Always 0	Beta(0.4, 0.01)	0.2
9	Always 0	Beta(0.3, 0.01)	0.15
10	Always 0	Beta(0.2, 0.01)	0.1

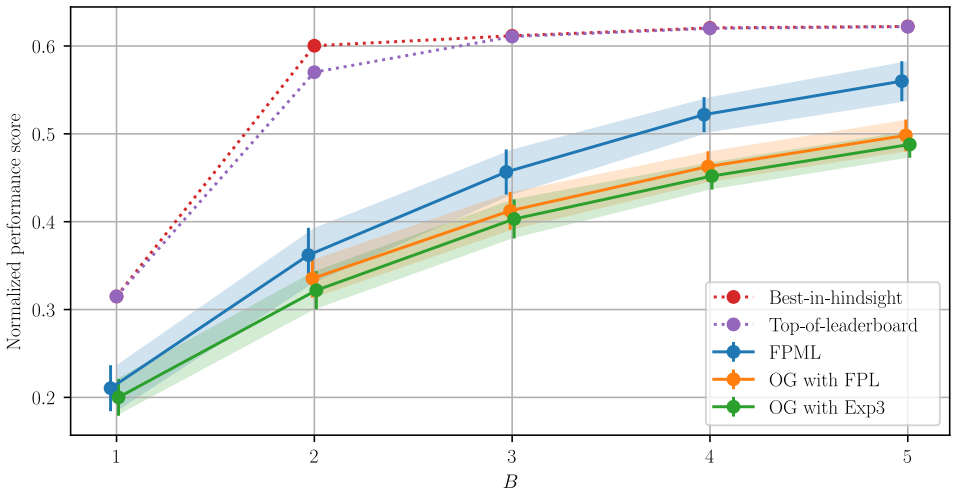

Figure 8.2: Normalized scores for various combinations of **FPML** and online greedy algorithms in the second synthetic environment. Error bars are 95% confidence intervals for a single observation.

Table 8.3: Reward distributions in the third synthetic environment.

Action	Distribution
1	Beta(0.6, 0.01)
2	Beta(0.55, 0.01)
3	Beta(0.5, 0.01)
4	Beta(0.45, 0.01)
5	Beta(0.4, 0.01)
6	Beta(0.35, 0.01)
7	Beta(0.3, 0.01)
8	Beta(0.25, 0.01)
9	Beta(0.2, 0.01)
10	Beta(0.15, 0.01)

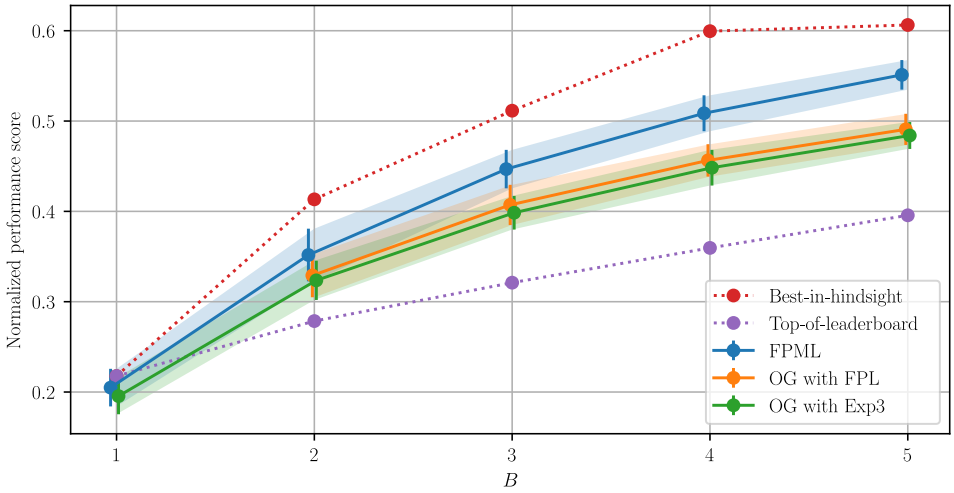

Figure 8.3: Normalized scores for various combinations of **FPML** and online greedy algorithms in the second synthetic environment *with the added regime change halfway through*. Error bars are 95% confidence intervals for a single observation. Note in the this version (with the regime change) the gap between the best action set and the top of the leaderboard *is* large; but within each regime it will be small. This hints at **FPML**'s ability to identify *local* leaderboards better than other algorithms.

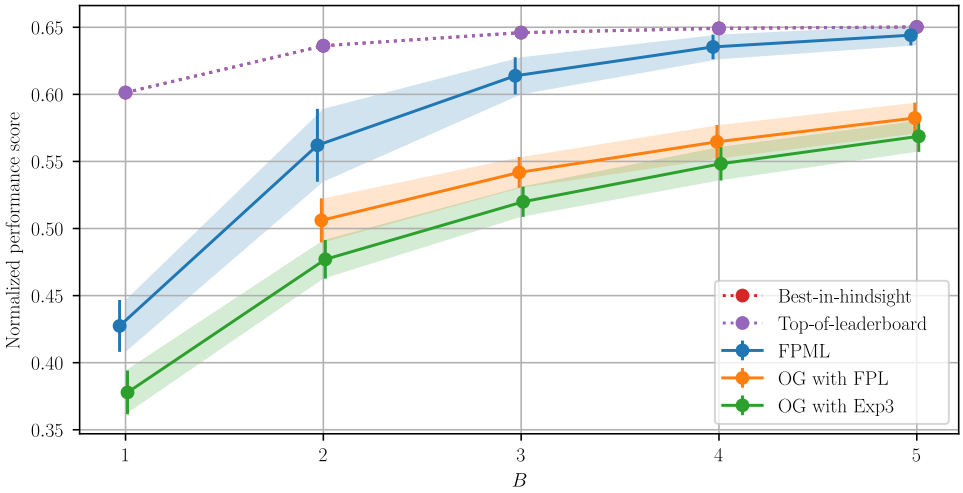

Figure 8.4: Normalized scores for various combinations of **FPML** and online greedy algorithms in the third synthetic environment. Error bars are 95% confidence intervals for a single observation.

Table 8.4: Rewards in the fourth synthetic environment, for some parameter $\delta \in (0, 1/2)$, run for 300 rounds and averaged over 50 trials.

Action	Reward at rounds $i \equiv k \mod 4$ for...				Average reward
	$k = 1$	$k = 2$	$k = 3$	$k = 4$	
1	δ	δ	1	1	$1/2 + \delta/2$
2	$1/2 + \delta$	$1/2 + \delta$	0	0	$1/4 + \delta/2$
3	0	1	0	1	$1/2$
4	1	0	1	0	$1/2$

Table 8.5: Means and standard deviations over 50 trials of rewards for various combinations of **FPML** and online greedy algorithms in the fourth synthetic environment, with $\delta = 0.01$.

Algorithm	Mean	StD
Best-in-hindsight	1.000	0
Top-of-leaderboard	1.000	0
FPML	0.964	0.0145
OG with FPML $((B', K) = 1, 3))$	0.823	0.0200
OG with Exp3	0.799	0.0202

8.4 When is greediness bad?

The fourth environment is one where $S^\star = S_{\text{top}}$ and this set does better better than S_{greedy}; greediness is worse than just picking the top B actions. Suppose there are $|\mathscr{A}| = 4$ available actions and a fixed budget of $B = 3$. Rewards are deterministic and listed in Table 8.4 for some parameter δ which we set to 0.01.[5] The top 3 actions are $S_{\text{top}} = \{1, 3, 4\}$ and this is also the best-in-hindsight set $S^\star$, receiving maximum reward 1 at each round. A quick calculation shows that the greedy choice S_{greedy} is either $\{1, 2, 3\}$ or $\{1, 2, 4\}$, though, and either of these sets receive an average maximum reward of $7/8 + \delta/4$, substantially higher. Our empirical results in Table 8.5 show this gap in practice.

[5] Credit to my supervisor for the example idea.

Chapter 9

Discussion and Future Work

We finish with a short discussion of our results and their relation to the wider literature, followed by some suggestions for future research directions and finally a few personal reflections on this project.

9.1 Conclusions

First a brief recap. The centrepiece of this work has been the new 'Follow the Perturbed Multiple Leaders' algorithm we introduced in Chapter 3. We theoretically bounded its 1-regret using a generalization of the elegant proof technique Kalai and Vempala [KV05] employed on the original **FPL** algorithm, and we bounded the N-regret for problem instances where $\texttt{err}_N$ is small (e.g. the hyperparameter selection application considered in Chapter 7), as well as mentioning an inefficient version of the algorithm that handles large $\texttt{err}_N$. We then generalized **FPML** to handle partial feedback in two main ways—one uniformly exploring a set number of actions at each round and make reward estimates based on these, and one using geometric resampling to make self-exploring reward estimates based on all actions taken—and generalized our earlier theoretical 1-regret bounds to these cases. We proved lower bounds on the 1-regret and N-regret achievable in our problem setting by any algorithm—with the 1-regret bound being matched by a full feedback version of **FPML** asymptotically in $|\mathscr{A}|$—before showing new 1-regret bounds for the online greedy submodular function maximization algorithm **OG** from Streeter and Golovin [SG08] and applying **FPML** to construct a new hybridization $\mathbf{OG_{hybrid}}$of this algorithm that improves on these bounds. Finally, we evaluated **FPML**, **OG** and $\mathbf{OG_{hybrid}}$empirically on an online hyperparameter selection algorithm choice problem inspired by the 2020 NeurIPS BBO Challenge as well as in a number of synthetic environments.

The experiments have shown that our **FPML** algorithm outperforms the existing algorithm **OG** on the hyperparameter selection task (with $\mathbf{OG_{hybrid}}$also providing a modest improvement over **OG**). As discussed, we think this is due to the problem structure, with small $\texttt{err}_B$; in applications like this where greediness presents no advantage **FPML** is the winner. This was confirmed by our synthetic-environment experiments. In practice many real-life problems may have this structure, and **FPML** will be a better choice in these settings than **OG**.

Our work thus provides five main advantages over anything currently in the literature:

1. ours is the first explicit discussion of the multitasking bandit problem in the adversarial setting and its many applications;

2. our analysis of budget-B algorithms (both our novel multitasking bandit algorithms and existing ones in the more general online submodular function maximization setting) using the N-regret for $N < B$ is new and interesting, and better models the trade-off of performance against resources inherent in many applications;
3. we think our **FPML** algorithm is an interesting generalization of **FPL** in its own right and that our proofs generalizing the proof techniques from Kalai and Vempala [KV05] are of independent interest;
4. our **FPML** algorithm achieves better 1-regret than **OG** in the multitasking bandit setting (and our $\mathbf{OG}_{\mathbf{hybrid}}$algorithm does the same in the more general submodular function maximization setting), providing better guarantees on the performance increase to expect when increasing your available resources;
5. the **FPML** algorithm has superior absolute performance to **OG** in many practical settings where greediness isn't important.

Indeed, as well as several of these results being of theoretical interest, we believe **FPML** is an important practical step towards performant ways of managing limited time/computational resources in lifelong learning settings (and others) when executing a stream of tasks with access to a large repertoire of problem-solving techniques.

9.2 Future work

Some possible future directions—both theoretical and practical—are given in Table 9.1, some of which we have played with during the course of this project but not included in the final report.

9.3 Personal reflections

I've very much enjoyed working on this project over the last nine months. Delving into the vast theory of multi-armed bandits has been fascinating, and the theoretical parts of the project have been a good fit for my mathematical background. In particular, I have found it very rewarding to be able to work both on theoretical proofs—using a good selection of techniques/results from probability theory and statistics—and on practical implementations in Python, and to see the two complement each other. It has also been a pleasure to collaborate with my supervisor on the project—to flesh out the details of his ideas and come up with new ones of my own—an opportunity for which I'm very grateful.

The path has not been without obstacles. At times it's been challenging to fit in work on this project alongside the Oxford workload, and in particular to 'refocus' after each extended interruption; relatedly, I've definitely grappled with limiting time spent on stubborn details and forcing myself to move on and keep the big picture in mind. On occasion, spending several days/weeks ironing out a proof that ends up not working (e.g. several ideas for lower bounds) has been frustrating, though of course to be expected. Another more subtle challenge has been forming strong accurate idea-guiding intuitions about the problems and algorithms at hand, which I've found harder than I expected to, as well as maintaining consistent belief in the importance/relevance of the problem at times (for which returning to concrete applications has proven helpful).

Several take-aways come to mind. Firstly, I've learnt not to underestimate the time it can take to work out details for arguments whose veracity you doubt—and that in these situations rigour deserves much less priority than I'm used to affording it, forming more of a 'last step'. Secondly, I

Table 9.1: Possible future research directions.

- The regret analyses in this work for the first time give theoretical expectations on the *increase* in performance gained from increasing your per-round action budget B. A natural next step is to provide algorithms that dynamically select the best action budget B for the problem, e.g. when B represents time to spend on each round. For example, if increasing B provides only a small performance boost then sticking with smaller B is likely optimal, but if a small increase in B causes a great increase in performance then a larger choice is suitable. This could be a periodical update of B or a new choice at each round.
- Related to the above, a regret analysis when executing with arbitrary *different* action budgets per round would be relevant to settings where your available resources are decided for you on the fly at each round.
- Work on tighter lower bounds (in T, especially) will be important to better understanding the dynamics of this problem.
- It would be interesting to develop an efficient approximation algorithm using the 'super-actions' technique mentioned in Chapter 3: this may offer advantages when competing against large fixed action sets or in some practical settings where **FPML** is sub-optimal.
- A closer comparison of the algorithms in this work to the algorithm for the stochastic K-max problem from Chen et al. [Che+16] would be interesting.
- A generalization of **FPML** to the more general adversarial combinatorial bandit setting (with max as the reward function, or potentially something more general) may be possible.
- An extension of **FPML** to the general 'experts' case where we have access to a number of experts who each recommend an action to be taken at each round.

now better appreciate the benefits of **(a)** fully deciding on the target problem and understanding the current literature before trying to solve something, and **(b)** then focussing on one thing at a time, to avoid losing time to 'switching' between mental tree branches. Thirdly, I have slightly recalibrated my future research interests: if anything, this project (along with other work this year) has taught me that I'm more motivated when developing and experimenting with application-motivated new techniques than when finding and proving theoretical guarantees, and my interests have thus shifted a little accordingly.

References

[ABL14] Jean-Yves Audibert, Sébastien Bubeck, and Gábor Lugosi. "Regret in online combinatorial optimization". In: *Mathematics of Operations Research* 39.1 (2014), pp. 31–45.

[AK08] Baruch Awerbuch and Robert Kleinberg. "Online linear optimization and adaptive routing". In: *Journal of Computer and System Sciences* 74.1 (2008), pp. 97–114.

[Aue+02] Peter Auer et al. "The nonstochastic multiarmed bandit problem". In: *SIAM journal on computing* 32.1 (2002), pp. 48–77.

[Ava+21] Vashist Avadhanula et al. "Stochastic bandits for multi-platform budget optimization in online advertising". In: *Proceedings of the Web Conference 2021*. 2021, pp. 2805–2817.

[Bas+21] Hamsa Bastani et al. "Efficient and targeted COVID-19 border testing via reinforcement learning". In: *Nature* 599.7883 (2021), pp. 108–113.

[BCB12] Sébastien Bubeck and Nicolo Cesa-Bianchi. "Regret analysis of stochastic and nonstochastic multi-armed bandit problems". In: *arXiv preprint arXiv:1204.5721* (2012).

[BCBK12] Sébastien Bubeck, Nicolo Cesa-Bianchi, and Sham M Kakade. "Towards minimax policies for online linear optimization with bandit feedback". In: *Conference on Learning Theory*. JMLR Workshop and Conference Proceedings. 2012, pp. 41–1.

[BK18] Lilian Besson and Emilie Kaufmann. "What doubling tricks can and can't do for multi-armed bandits". In: *arXiv preprint arXiv:1803.06971* (2018).

[BRA20] Djallel Bouneffouf, Irina Rish, and Charu Aggarwal. "Survey on applications of multi-armed and contextual bandits". In: *2020 IEEE Congress on Evolutionary Computation (CEC)*. IEEE. 2020, pp. 1–8.

[CBL12] Nicolo Cesa-Bianchi and Gábor Lugosi. "Combinatorial bandits". In: *Journal of Computer and System Sciences* 78.5 (2012), pp. 1404–1422.

[Che+16] Wei Chen et al. "Combinatorial multi-armed bandit with general reward functions". In: *Advances in Neural Information Processing Systems* 29 (2016).

[Che+18] Xi Chen et al. "Dynamic Assortment Selection under the Nested Logit Models". In: *arXiv preprint arXiv:1806.10410* (2018).

[Com+15] Richard Combes et al. "Combinatorial bandits revisited". In: *Advances in neural information processing systems* 28 (2015).

[CWZ18] Xi Chen, Yining Wang, and Yuan Zhou. "An optimal policy for dynamic assortment planning under uncapacitated multinomial logit models". In: *arXiv preprint arXiv:1805.04785* (2018).

[Dur+18] Audrey Durand et al. “Contextual bandits for adapting treatment in a mouse model of de novo carcinogenesis”. In: *Machine learning for healthcare conference*. PMLR. 2018, pp. 67–82.

[Gne43] Boris Gnedenko. “Sur la distribution limite du terme maximum d’une série aléatoire”. In: *Annals of mathematics* (1943), pp. 423–453.

[Han57] James Hannan. “Approximation to Bayes risk in repeated play”. In: *Contributions to the Theory of Games* 3.2 (1957), pp. 97–139.

[HF17] Xiaoguang Huo and Feng Fu. “Risk-aware multi-armed bandit problem with application to portfolio selection”. In: *Royal Society open science* 4.11 (2017), p. 171377.

[HWC21] Yanjun Han, Yining Wang, and Xi Chen. “Adversarial Combinatorial Bandits with General Non-linear Reward Functions”. In: *International Conference on Machine Learning*. PMLR. 2021, pp. 4030–4039.

[Ker+18] Raouf Kerkouche et al. “Node-based optimization of LoRa transmissions with Multi-Armed Bandit algorithms”. In: *2018 25th International Conference on Telecommunications (ICT)*. IEEE. 2018, pp. 521–526.

[KV05] Adam Kalai and Santosh Vempala. “Efficient algorithms for online decision problems”. In: *Journal of Computer and System Sciences* 71.3 (2005), pp. 291–307.

[Kve+15] Branislav Kveton et al. “Tight regret bounds for stochastic combinatorial semi-bandits”. In: *Artificial Intelligence and Statistics*. PMLR. 2015, pp. 535–543.

[Law72] Eugene L Lawler. “A procedure for computing the k best solutions to discrete optimization problems and its application to the shortest path problem”. In: *Management science* 18.7 (1972), pp. 401–405.

[LF17] Romain Laroche and Raphael Feraud. “Reinforcement learning algorithm selection”. In: *arXiv preprint arXiv:1701.08810* (2017).

[LW94] Nick Littlestone and Manfred K Warmuth. “The weighted majority algorithm”. In: *Information and computation* 108.2 (1994), pp. 212–261.

[MB04] H Brendan McMahan and Avrim Blum. “Online geometric optimization in the bandit setting against an adaptive adversary”. In: *International Conference on Computational Learning Theory*. Springer. 2004, pp. 109–123.

[NB13] Gergely Neu and Gábor Bartók. “An efficient algorithm for learning with semi-bandit feedback”. In: *International Conference on Algorithmic Learning Theory*. Springer. 2013, pp. 234–248.

[NWF78] George L Nemhauser, Laurence A Wolsey, and Marshall L Fisher. “An analysis of approximations for maximizing submodular set functions—I”. In: *Mathematical programming* 14.1 (1978), pp. 265–294.

[Pol05] Jan Poland. “FPL analysis for adaptive bandits”. In: *International Symposium on Stochastic Algorithms*. Springer. 2005, pp. 58–69.

[Rob52] Herbert Robbins. “Some aspects of the sequential design of experiments”. In: *Bulletin of the American Mathematical Society* 58.5 (1952), pp. 527–535.

[SG08] Matthew Streeter and Daniel Golovin. “An online algorithm for maximizing submodular functions”. In: *Advances in Neural Information Processing Systems* 21 (2008).

[Sli19] Aleksandrs Slivkins. “Introduction to multi-armed bandits”. In: *arXiv preprint arXiv:1904.07272* (2019).

[TM17] Ambuj Tewari and Susan A Murphy. “From ads to interventions: Contextual bandits in mobile health”. In: *Mobile Health*. Springer, 2017, pp. 495–517.

[Tur+21] Ryan Turner et al. “Bayesian optimization is superior to random search for machine learning hyperparameter tuning: Analysis of the black-box optimization challenge 2020”. In: *NeurIPS 2020 Competition and Demonstration Track*. PMLR. 2021, pp. 3–26.

[Ube20] Uber. *Bayesmark Documentation*. 2020. URL: `https://bayesmark.readthedocs.io/en/latest/index.html`.

Appendix A

Full Proofs

Proof of Lemma 3.4

Proof. Now that $\varepsilon < \infty$, the random exponential permutations $(p_t(a))_{a\in\mathscr{A},t\in[T]}$ are not necessarily zero. As in Kalai and Vempala [KV05], assume w.l.o.g. that $p_1(a) = p_2(a) = \cdots = p_T(a)$ for each $a \in \mathscr{A}$; by linearity of expectation, this does not change the expected regret of the algorithm summed across rounds. Write $p := p_1$. Moreover, again following Kalai and Vempala [KV05], imagine *instead of* adding perturbations that there is a 'round zero' with rewards $r_0(a) := p(a)\ \forall a \in \mathscr{A}$ and that the algorithm uses for its calculations cumulative rewards $R_t^\star(\cdot) := \sum_{s=0}^{t} r_s(\cdot)$ including these 'random initializations'. This is of course numerically equivalent to adding the same perturbation at each round:

$$R_t^\star(a) = \sum_{s=0}^{t} r_t(a) = r_0(a) + R_t(a) = p(a) + R_t(a) = p_{t+1}(a) + R_t(a) = \tilde{R}_t(a). \tag{A.1}$$

We proceed by adapting the proof of Lemma 3.3. Write now $m_t = \max_{a\in\mathscr{A}} R_t^\star(a)$ for each $t \in [T]$, so in particular now $m_T = R_T^\star(\tilde{a}^\star) = R_T(\tilde{a}^\star) + p(\tilde{a}^\star)$, where $\tilde{a}^\star := \arg\max_{a\in\mathscr{A}} R_T^\star(a)$. Including the random initializations when defining $(\mathscr{E})_{t\in[T]}$ and $\mathscr{I}$, the proof of Lemma 3.3 then applies without modification up to showing that

$$\sum_{t=1}^{T} r_t(S_t) \geqslant m_T - m_0 - |\mathscr{I}|. \tag{A.2}$$

With the new definitions of m_t, though, this lower bound now evaluates to

$$R_T(\tilde{a}^\star) + p(\tilde{a}^\star) - \max_{a\in\mathscr{A}} p(a) - |\mathscr{I}| \tag{A.3}$$

and by definition of $\tilde{a}^\star$, then,

$$\sum_{t=1}^{T} r_t(S_t) \geqslant R_T(a^\star) + p(a^\star) - \max_{a\in\mathscr{A}} p(a) - |\mathscr{I}| \tag{A.4}$$

(where $a^\star = \arg\max_{a\in\mathscr{A}} R_T(a)$). This motivates us to upper-bound $\mathbb{E}[\max_{a\in\mathscr{A}} p(a) - p(a^\star)]$. But this is simple: $\mathbb{E}[p(a^\star)] = 1/\varepsilon$ as for any fixed action, and $\max_{a\in\mathscr{A}} p(a)$ is the maximum of $|\mathscr{A}|$ i.i.d. $\mathrm{Exp}(\varepsilon)$ random variables, so has expectation at most $(1 + \ln|\mathscr{A}|)/\varepsilon$ as argued in Kalai

and Vempala [KV05]. Thus

$$\mathbb{E}\left[\sum_{t=1}^{T} r_t(S_t)\right] \geqslant \mathbb{E}[R_T(a^\star)] - \frac{\ln|\mathscr{A}|}{\varepsilon} - \mathbb{E}[|\mathscr{I}|] \tag{A.5}$$

which rearranges to the result. □

Proof of Proposition 3.5

We'll need a pure-probabilistic lemma:

Lemma A.1. *For some finite index set I, let $(X_i)_{i\in I}$ be independent real-valued random variables and define $\mathscr{X} \subseteq I$ to be the indices of the smallest m of them. Then conditional on the random variables $\mathscr{X}$ and $(X_i)_{i\in\mathscr{X}}$, the remaining random variables $(X_j)_{j\in I\setminus\mathscr{X}}$ are independent with marginal distributions*

$$\mu_{X_j}(\cdot \mid \sigma(\mathscr{X}, (X_i)_{i\in\mathscr{X}})) = \mu_{X_j}(\cdot \mid \{X_j \geqslant \max_{i\in\mathscr{X}} X_i\}), \quad j \in I \setminus \mathscr{X}. \tag{A.6}$$

Proof. We prove this in the case that $(X_i)_{i\in I}$ are absolutely continuous, with densities $(f_i)_{i\in I}$ respectively. (Note the lemma still holds in the general case.) Assume w.l.o.g. that $I = [n]$ for $n \in \mathbb{N}$. For $C \subseteq [n]$ of size m, write $C = \{i_1, \dots, i_m\}$ and $C^c = \{i_{m+1}, \dots, i_n\}$; then for any $x_1, \dots, x_n \in \mathbb{R}$ the relevant conditional joint density is

$$\begin{aligned}
&f_{X_{i_{m+1}},\dots,X_n|X_{i_1},\dots,X_{i_n}}(x_{i_{m+1}}, \dots, x_{i_n} \mid x_{i_1}, \dots, x_{i_m}, \{\mathscr{X} = C\}) && \text{(A.7)}\\
&= \frac{f_{X_1,\dots,X_n}(x_1, \dots, x_n \mid \{\mathscr{X} = C\})}{f_{X_{i_1},\dots,X_{i_m}}(x_{i_1}, \dots, x_{i_m} \mid \{\mathscr{X} = C\})} && \text{(A.8)}\\
&\propto f_{X_1,\dots,X_n}(x_1, \dots, x_n \mid \{\mathscr{X} = C\}) && \text{(A.9)}\\
&= f_{X_1,\dots,X_n}(x_1, \dots, x_n \mid \{X_j \geqslant \max_{i\in C} X_i \ \forall j \notin C\}) && \text{(A.10)}\\
&= \frac{f_{X_1,\dots,X_n}(x_1, \dots, x_n)}{\mathbb{P}(X_j \geqslant \max_{i\in C} X_i \ \forall j \notin C)} \mathbb{1}_{x_j \geqslant \max_{i\in C} x_i \ \forall j\notin C)} && \text{(A.11)}\\
&= \frac{f_{X_1,\dots,X_n}(x_1, \dots, x_n)}{\int_x \mathbb{P}(X_j \geqslant x \ \forall j \notin C) f_{\max_{i\in C} X_i}(x)\,\mathrm{d}x} \mathbb{1}_{x_j \geqslant \max_{i\in C} x_i \ \forall j\notin C)} && \text{(A.12)}\\
&= \frac{f_{X_1}(x_1)\cdots f_{X_n}(x_n)}{\int_x \mathbb{P}(X_{i_{m+1}} \geqslant x)\cdots\mathbb{P}(X_{i_n} \geqslant x) f_{\max_{i\in C} X_i}(x)\,\mathrm{d}x} \mathbb{1}_{x_j \geqslant \max_{i\in C} x_i \ \forall j\notin C)} \text{ by independence} && \text{(A.13)}\\
&\propto \left(\frac{f_{X_{i_{m+1}}}(x_{i_{m+1}})}{\mathbb{P}(X_{i_{m+1}} \geqslant \max_{i\in C} X_i)} \mathbb{1}_{x_{i_{m+1}} \geqslant \max_{i\in C} x_i}\right) \cdots \left(\frac{f_{X_{i_n}}(x_{i_n})}{\mathbb{P}(X_{i_n} \geqslant \max_{i\in C} X_i)} \mathbb{1}_{x_{i_n} \geqslant \max_{i\in C} x_i}\right) && \text{(A.14)}\\
&= f_{X_{i_{m+1}}}(x_{i_{m+1}} \mid \{X_{i_{m+1}} \geqslant \max_{i\in C} X_i\}) \cdots f_{X_{i_n}}(x_{i_n} \mid \{X_{i_n} \geqslant \max_{i\in C} X_i\}) && \text{(A.15)}\\
&= \prod_{j\notin C} f_{X_j}(x_j \mid \{X_j \geqslant \max_{i\in C} X_i\}) && \text{(A.16)}
\end{aligned}$$

where proportionality is in $x_{i_{m+1}}, \dots, x_{i_n}$. The result follows. □

Proof of Proposition 3.5

Proof. We will argue that $\mathbb{E}[|\mathscr{I}|] \leqslant T(1 - \mathrm{e}^{-\varepsilon})^B$ (where $\mathscr{I}$ is as defined in Lemma 3.4). As in the proof of Lemma 3.4, assume w.l.o.g. that instead of perturbations there is a 'round zero'

with random i.i.d. Exp(ε) rewards $(p(a))_{a\in\mathscr{A}}$. Fix a round $t \in [T]$ and define $\mathscr{E}_t$ as before; so

$$\mathscr{E}_t = \{\exists a' \in \mathscr{A} \setminus S_t \,:\, \forall a \in S_t,\ a' \text{ overtakes } a \text{ at round } t\} \tag{A.17}$$

$$= \bigcup_{a'\in\mathscr{A}\setminus S_t} \bigcap_{a\in S_t} \{a' \text{ overtakes } a \text{ at round } t\} \tag{A.18}$$

$$\subseteq \bigcap_{a\in S_t} \bigcup_{a'\in\mathscr{A}\setminus S_t} \{a' \text{ overtakes } a \text{ at round } t\} = \bigcap_{a\in S_t} \{a \text{ overtaken by some } a' \notin S_t\}. \tag{A.19}$$

Let V represent the cumulative reward (including round zero) of the $(B+1)^{\text{th}}$ best action; so $V = \max_{a\in\mathscr{A}\setminus S_t} R^\star_{t-1}(a)$. For each action $a \in \mathscr{A}$, then, $a \in S_t$ iff $R^\star_{t-1}(a) > V$.[1] Define the event $E_a := \{R^\star_{t-1}(a) > V + 1\}$; if this holds then then we know for sure not only that $a \in S_t$ but that action a must *remain* ahead of every action $a' \notin S_t$ after this round, since a must have been ahead of every such action by at least 1 already. That is,

$$\{a \text{ overtaken by some } a' \notin S_t\} \subseteq E_a^c. \tag{A.20}$$

Let $\mathscr{G}_t := \sigma(S_t, (R^\star_{t-1}(a))_{a\in S_t})$ be the σ-algebra generated by the random set S_t and the current cumulative rewards of the actions in it. So we have

$$\mathbb{P}\left(\mathscr{E}_t \mid \mathscr{G}_t\right) \leqslant \mathbb{P}\left(\bigcap_{a\in S_t} \{a \text{ overtaken by some } a' \notin S_t\} \mid \mathscr{G}_t\right) \tag{A.21}$$

$$\leqslant \mathbb{P}\left(\bigcap_{a\in C} E_a^c \mid \mathscr{G}_t\right) \tag{A.22}$$

$$= \mathbb{P}\left(\bigcap_{a\in S_t} \{R^\star_{t-1}(a) < V + 1\} \mid \mathscr{G}_t\right). \tag{A.23}$$

But, since $V = \max_{a\in\mathscr{A}\setminus S_t} R^\star_{t-1}(a)$, applying Lemma A.1 with $I = \mathscr{A}$, $\mathscr{X} = S_t$ and $(X_i)_{i\in I} = (R^\star_{t-1}(a))_{a\in\mathscr{A}}$ gives us that

$$\mathbb{P}\left(\bigcap_{a\in S_t} \{R^\star_{t-1}(a) < V + 1\} \mid \mathscr{G}_t\right) = \prod_{a\in S_t} \mathbb{P}\left(R^\star_{t-1}(a) < V + 1 \mid R^\star_{t-1}(a) \geqslant V\right). \tag{A.24}$$

By the memoryless property of the exponential distribution, each term here just becomes

$$1 - \mathbb{P}\left(p(a) \geqslant V - R_{t-1}(a) + 1 \mid p(a) \geqslant V - R_{t-1}(a)\right) \leqslant 1 - \mathbb{P}(p(a) \geqslant 1) = 1 - \mathrm{e}^{-\varepsilon}. \tag{A.25}$$

Thus $\mathbb{P}(\mathscr{E}_t \mid \mathscr{G}_t) \leqslant (1-\mathrm{e}^{-\varepsilon})^B$. Since this expression is deterministic and so trivially independent from the σ-algebra $\mathscr{G}_t$, this immediately implies that $\mathbb{P}(\mathscr{E}_t) \leqslant (1-\mathrm{e}^{-\varepsilon})^B$.

The result then follows from Lemma 3.4 since $\mathbb{E}[|\mathscr{I}|] = \sum_{t=1}^T \mathbb{P}(\mathscr{E}_t) \leqslant T(1-\mathrm{e}^{-\varepsilon})^B$. □

Proof of Proposition 4.1

Proof. We adapt the proofs of Lemma 3.4 and Proposition 3.5. As in those proofs, assume w.l.o.g. that instead of perturbations there is a 'round zero' with i.i.d. Exp(ε) random rewards $(p(a))_{a\in\mathscr{A}}$. Define the σ-algebra $\mathscr{F}_0 := \sigma((p(a))_{a\in\mathscr{A}})$ and extend all subsequent σ-algebras in the filtration $(\mathscr{F}_t)_{t\in[T]}$ to contain $\mathscr{F}_0$.

[1] For the analysis that follows, we take the rewards at every non-zero round to be deterministic, so the only source of randomness is the round zero rewards $(p(a))_{a\in\mathscr{A}}$.

Writing $\hat{R}_t(\cdot) := \sum_{s=1}^{t} \hat{r}_s(\cdot)$ for cumulative estimated reward and $\hat{R}^\star_t(\cdot) := \sum_{s=0}^{t} \hat{r}_s(\cdot) = \hat{R}_t(\cdot) + p(\cdot)$ for the same but including the 'round zero' random initializations, define $m_t := \max_{a\in\mathscr{A}} \hat{R}^\star_t(a)$ for each $t \in [T]$. In particular then

$$\mathbb{E}[m_T \mid \mathscr{F}_0] = \mathbb{E}\left[\max_{a\in\mathscr{A}} \hat{R}^\star_t(a) \mid \mathscr{F}_0\right] \geqslant \max_{a\in\mathscr{A}} \mathbb{E}\left[\hat{R}^\star_t(a) \mid \mathscr{F}_0\right] \tag{A.26}$$

$$= \max_{a\in\mathscr{A}} \left(p(a) + \sum_{t=1}^{T} \mathbb{E}[\hat{r}_t(a) \mid \mathscr{F}_0] \right) \tag{A.27}$$

$$= \max_{a\in\mathscr{A}} \left(p(a) + \sum_{t=1}^{T} r_t(a) \right) = R_T(\tilde{a}^\star) + p(\tilde{a}^\star) \tag{A.28}$$

where $\tilde{a}^\star := \arg\max_{a\in\mathscr{A}} R^\star_T(a)$. The initial step here follows from Jensen's inequality (as the max function is convex in its inputs).

Similarly to in Lemma 3.3, for each t let $a^\star_t := \arg\max_{a\in\mathscr{A}} \hat{R}^\star_{t-1}(a)$ and define the event $\mathscr{E}_t = \{a^\star_{t+1} \notin S_t\}$.

Fix t and assume $\neg\mathscr{E}_t$. Action $a^\star_{t+1}$ has estimated reward

$$\hat{r}_t(a^\star_{t+1}) = \hat{R}_t(a^\star_{t+1}) - \hat{R}_{t-1}(a^\star_{t+1}) = m_t - \hat{R}_{t-1}(a^\star_{t+1}) \geqslant m_t - m_{t-1} \tag{A.29}$$

by definition of m_t, m_{t-1}, so as we're assuming $a^\star_{t+1} \in S_t$ we get that

$$r_t(S_t) = \max_{a\in S_t} r_t(a) \geqslant r_t(a^\star_{t+1}) = \mathbb{E}[\hat{r}_t(a^\star_{t+1}) \mid \mathscr{F}_0] \geqslant \mathbb{E}[m_t - m_{t-1} \mid \mathscr{F}_0]. \tag{A.30}$$

So, writing $\mathscr{I} := \{t \in [T] : \mathscr{E}_t \text{ holds}\}$, the total actual reward of the algorithm is

$$\sum_{t=1}^{T} r_t(S_t) \geqslant \sum_{t\in\mathscr{I}^c} r_t(S_t) \geqslant \sum_{t\in\mathscr{I}^c} \mathbb{E}[m_t - m_{t-1} \mid \mathscr{F}_0] \tag{A.31}$$

$$= \sum_{t=1}^{T} \mathbb{E}[m_t - m_{t-1} \mid \mathscr{F}_0] - \sum_{t\in\mathscr{I}} \mathbb{E}[m_t - m_{t-1} \mid \mathscr{F}_0] \tag{A.32}$$

$$= \mathbb{E}[m_T \mid \mathscr{F}_0] - m_0 - \sum_{t\in\mathscr{I}} \mathbb{E}[m_t - m_{t-1} \mid \mathscr{F}_0]. \tag{A.33}$$

But the increase in the highest estimated (perturbed) reward from one round to the next is at most β, by definition of the estimates; so

$$\sum_{t=1}^{T} r_t(S_t) \geqslant \mathbb{E}[m_T \mid \mathscr{F}_0] - m_0 - \beta|\mathscr{I}| \tag{A.34}$$

$$= R_T(\tilde{a}^\star) + p(\tilde{a}^\star) - \max_{a\in\mathscr{A}} p(a) - \beta|\mathscr{I}| \tag{A.35}$$

$$\geqslant R_T(a^\star) + p(a^\star) - \max_{a\in\mathscr{A}} p(a) - \beta|\mathscr{I}| \tag{A.36}$$

(where $a^\star := \arg\max_{a\in\mathscr{A}} R_T(a)$ is the actual best action).

Taking expectations and upper-bounding $\mathbb{E}[\max_{a\in\mathscr{A}} p(a) - p(a^\star)]$ as in Lemma 3.4, then,

$$\mathbb{E}\left[\sum_{t=1}^{T} r_t(S_t)\right] \geqslant \mathbb{E}[R_T(a^\star)] - \frac{\ln|\mathscr{A}|}{\varepsilon} - \beta\mathbb{E}[|\mathscr{I}|]; \tag{A.37}$$

so the expected 1-regret of the algorithm is $\mathbb{E}[\mathscr{R}_1] \leqslant \frac{\ln|\mathscr{A}|}{\varepsilon} + \beta\mathbb{E}[|\mathscr{I}|]$.

It remains to upper-bound $\mathbb{E}[|\mathscr{I}|]$. We proceed very similarly to as in Proposition 3.5.

Fix $t \in [T]$ and let $V := \max_{a \in \mathscr{A}\setminus S_t} \hat{R}^\star_{t-1}(a)$. So for any a, $\{a \in S_t\} = \{\hat{R}^\star_{t-1}(a) > V\}$. Define $E_a := \{\hat{R}^\star_{t-1}(a) > V + \alpha + \beta\}$; if this holds then a must have been ahead of every action $a' \notin S_t$ by at least $\alpha + \beta$ and therefore *cannot* be overtaken by any such action, since the estimated rewards are all bounded by $[-\alpha, \beta]$. So

$$\{a \text{ overtaken by some } a' \notin S_t\} \subseteq E_a^c. \tag{A.38}$$

The argument in Proposition 3.5 then applies exactly, simply replacing 1 with $\alpha + \beta$ where necessary to conform to the new definition of E_a. We get that $\mathbb{P}(\mathscr{E}_t) \leqslant (1 - \mathrm{e}^{-\varepsilon(\alpha+\beta)})^B$ for each t, and so $\mathbb{E}[|\mathscr{I}|] = \sum_{t=1}^T \mathbb{P}(\mathscr{E}_t) \leqslant T(1 - \mathrm{e}^{-\varepsilon(\alpha+\beta)})^B$. □

Proof of Lemma 4.4

Proof. Fix t and a and write $K = \min(Z_{t,a}, M)$. Define the event

$$E := \{a \in S_t\}; \tag{A.39}$$

then using the definition of $\hat{r}_{t,a}$, we have that

$$\begin{aligned}
\mathbb{E}[1 - \hat{r}_t(a) \mid \mathscr{F}_{t-1}] &= \mathbb{E}[(1 - r_t(a)) \min(Z_{t,a}, M)\mathbb{1}_E + 0\,\mathbb{1}_{E^c} \mid \mathscr{F}_{t-1}] && \text{(A.40)}\\
&= (1 - r_t(a))\mathbb{E}[K\mathbb{1}_E \mid \mathscr{F}_{t-1}] && \text{(A.41)}\\
&= (1 - r_t(a))\mathbb{P}(E \mid \mathscr{F}_{t-1})\mathbb{E}[K \mid \mathscr{F}_{t-1}] && \text{(A.42)}\\
&= (1 - r_t(a))q_{t,a}\mathbb{E}[K \mid \mathscr{F}_{t-1}] && \text{(A.43)}
\end{aligned}$$

where the penultimate line followed from the conditional independence of E and $Z_{t,a}$ given $\mathscr{F}_{t-1}$ (by definition of $Z_{t,a}$). But the inner expectation is just

$$\begin{aligned}
\mathbb{E}[K \mid \mathscr{F}_{t-1}] &= \sum_{k=1}^{M} k\mathbb{P}(K = k \mid \mathscr{F}_{t-1}) && \text{(A.44)}\\
&= \sum_{k=1}^{M-1} k\mathbb{P}(Z_{t,a} = k \mid \mathscr{F}_{t-1}) + M\mathbb{P}(Z_{t,a} \geqslant M \mid \mathscr{F}_{t-1}) && \text{(A.45)}\\
&= \sum_{k=1}^{M-1} k(1 - q_{t,a})^{k-1}q_{t,a} + M(1 - q_{t,a})^{M-1} && \text{(A.46)}
\end{aligned}$$

since conditional on $\mathscr{F}_{t-1}$, $Z_{t,a}$ has distribution $\mathrm{Geom}(q_{t,a})$.

In general sums of the form on the left have value

$$\sum_{k=1}^{n} kr^{k-1}(1-r) = \sum_{k=1}^{n} kr^{k-1} - \sum_{k=1}^{n}(k+1)r^k + \sum_{k=1}^{n} r^k \tag{A.47}$$

$$= \frac{\mathrm{d}}{\mathrm{d}r}\sum_{k=1}^{n} r^k - \frac{\mathrm{d}}{\mathrm{d}r}\sum_{k=1}^{n} r^{k+1} + \sum_{k=1}^{n} r^k \tag{A.48}$$

$$= \frac{\mathrm{d}}{\mathrm{d}r}\frac{r(1-r^n)}{1-r} - \frac{\mathrm{d}}{\mathrm{d}r}\frac{r^2(1-r^n)}{1-r} + \frac{r(1-r^n)}{1-r} \tag{A.49}$$

$$= \frac{nr^{n+1} - (n+1)r^n + 1}{(1-r)^2} \tag{A.50}$$

$$- \frac{(n+1)r^{n+2} - (n+2)r^{n+1} - r^2 + 2r}{(1-r)^2} + \frac{r(1-r^n)}{1-r} \tag{A.51}$$

$$= 1 - (n+1)r^n + \frac{r - r^{n+1}}{1-r} \tag{A.52}$$

Thus we get

$$\mathbb{E}[K \mid \mathscr{F}_{t-1}] = 1 + \frac{(1-q_{t,a}) - (1-q_{t,a})^M}{q_{t,a}} = \frac{1-(1-q_{t,a})^M}{q_{t,a}}. \tag{A.53}$$

Combining this with Eq. (A.43) gives the result. □

Proof of Proposition 4.5

Proof. We closely adapt the proof of Proposition 4.1. Note that the estimated rewards are always *overestimates*, so the argument that $\mathbb{E}[m_T \mid \mathscr{F}_0] \geqslant R_T(\tilde{a}^\star) + p(\tilde{a}^\star)$ still applies. However, the bound on the *algorithm* regret at rounds t where $\neg\mathscr{E}_t$ holds is now slightly different: we can only argue that

$$r_t(S_t) \geqslant r_t(a^\star_{t+1}) = \mathbb{E}[\hat{r}_t(a^\star_{t+1}) \mid \mathscr{F}_0] - (1-q_{t,a^\star_{t+1}})^M(1-r_t(a^\star_{t+1})) \tag{A.54}$$

$$\geqslant \mathbb{E}[m_t - m_{t-1} \mid \mathscr{F}_0] - (1-q_{t,a^\star_{t+1}})^M. \tag{A.55}$$

The last term here satisfies

$$\mathbb{E}[(1-q_{t,a^\star_{t+1}})^M \mid \mathscr{F}_{t-1}] = \sum_{a\in\mathscr{A}} \mathbb{P}(a = a^\star_{t+1} \mid \mathscr{F}_{t-1})(1-q_{t,a})^M \tag{A.56}$$

$$\leqslant \sum_{a\in\mathscr{A}} q_{t,a}(1-q_{t,a})^M \leqslant \sum_{a\in\mathscr{A}} q_{t,a}\mathrm{e}^{-q_{t,a}M}, \tag{A.57}$$

where we used that $\mathbb{P}(a = a^\star_{t+1}) \leqslant \mathbb{P}(a \in S_t) = q_{t,a}$ (since $a^\star_{t+1} \in S_t$) and the approximation $1 - x \leqslant \mathrm{e}^{-x}$.

Noting that the functon $q \mapsto q\mathrm{e}^{-qM}$ is maximized at $q = 1/M$, then,

$$\mathbb{E}[(1-q_{t,a^{(1)}_{t+1}})^M \mid \mathscr{F}_{t-1}] \leqslant \sum_{a\in\mathscr{A}} \frac{1}{M}\mathrm{e}^{-\frac{1}{M}M} = \frac{|\mathscr{A}|}{eM}. \tag{A.58}$$

Following through the proof of Proposition 4.1, then, and noting that the estimates $\hat{r}_i(a)$ are

bounded by $[1-M, 1]$ for all i, a, reveals that

$$\mathbb{E}\left[\sum_{t=1}^{T} r_t(S_t)\right] \geqslant \mathbb{E}[R_T(a^\star)] - \frac{\ln|\mathscr{A}|}{\varepsilon} - T \cdot \frac{|\mathscr{A}|}{eM} - 1\mathbb{E}[|\mathscr{I}|] \tag{A.59}$$

and the result follows quickly from the same argument bounding $\mathbb{E}[|\mathscr{I}|]$ (using $\alpha = M-1, \beta = 1$).

The second part of the theorem follows from the first part using the approximation $1-x \leqslant \mathrm{e}^{-x}$; the values of ε and M were chosen by setting the three terms approximately equal. □

Proof of Lemma 5.3

Proof. Let $X_1, \dots, X_k$ be i.i.d. $\mathrm{Bin}(n, 1/n)$ r.v.s and define $M_k := \max_{i=1,\dots,k} X_i$. For large n, it is a standard result that each of these are well-approximated by a $\mathrm{Po}(1)$ distribution. So consider the limiting case, where the X_i are exactly Poisson-distributed.

Define a continuous extension $F(x) = Q(x, 1)$ of the Poisson distribution function (here Q is the regularized upper incomplete Gamma function); we will assume the X_i follow this continuous distribution, observing that this doesn't change the asymptotic behaviour.

Define A_k such that $F(A_k) = 1 - 1/k$; so $\lim_{k\to\infty} A_k = \infty$. Noting that $\lim_{x\to\infty} \frac{1-F(x+\varepsilon)}{1-F(x)} = 0$ for any $\varepsilon > 0$ (this is easily shown and verifiable numerically),

$$k(1 - F(A_k + \varepsilon)) = \frac{1 - F(A_k + \varepsilon)}{1 - F(A_k)} \longrightarrow 0 \tag{A.60}$$

and

$$k(1 - F(A_k - \varepsilon)) = \frac{1 - F(A_k - \varepsilon)}{1 - F(A_k)} \longrightarrow \infty \tag{A.61}$$

for all $\varepsilon > 0$ as $k \longrightarrow \infty$. Since $1 - F(x) \longrightarrow 0$ as $x \longrightarrow \infty$, asymptotically we have $\log F(A_k \pm \varepsilon) \sim -(1 - F(A_k \pm \varepsilon))$ (by the series expansion of $\log(1-x)$), so it follows from the above that

$$k \log F(A_k + \varepsilon) \longrightarrow 0, \quad k \log F(A_k - \varepsilon) \longrightarrow -\infty \tag{A.62}$$

for all $\varepsilon > 0$. Exponentiating, therefore, $F(A_k + \varepsilon)^k \longrightarrow 1$ and $F(A_k - \varepsilon)^k \longrightarrow 0$; since F^k is the distribution function of M_k, it follows that

$$\mathbb{P}(|M_k - A_k| \leqslant \varepsilon) = F(A_k + \varepsilon)^k - F(A_k - \varepsilon)^k \longrightarrow 1 \tag{A.63}$$

for any ε, i.e. M_k concentrates around A_k. Asymptotically, then, $\mathbb{E}[M_k] = \Theta(A_k)$.

It remains to find A_k. The lower incomplete Gamma function has a first-order approximation $\gamma(a, z) \sim z^a \Gamma(a) \mathrm{e}^{-z} / \Gamma(1+a)$ for large a, so

$$1 - F(x) = 1 - Q(x, 1) = \frac{\gamma(x, 1)}{\Gamma(x)} \sim \frac{1}{\mathrm{e}\Gamma(x+1)} \sim \frac{\mathrm{e}^{x-1}}{\sqrt{2\pi} x^{x+1/2}} = \frac{\mathrm{e}}{\sqrt{2\pi}} \mathrm{e}^{x - (x+1/2)\log x} \tag{A.64}$$

by Stirling's approximation; thus by definition of A_k, $1/k \sim \mathrm{e}^{A_k - (A_k+1/2)\log A_k + \mathrm{const}}$, i.e.

$$A_k \log A_k - A_k = \Theta(\log k). \tag{A.65}$$

When the equality is exact this solves to $A_k = \mathrm{e}^{W(\log k/\mathrm{e})+1}$ where W is the principal branch of the Lambert product logarithm. Since asymptotically $W(x) = \Theta(\log x - \log\log x)$, then, we

have

$$A_k = \Theta(e^{\log(\log k) - \log\log(\log k)}) = \Theta\left(\frac{\log k}{\log\log k}\right). \tag{A.66}$$

□

Proof of Lemma 6.5

Proof. For each $j \in \mathbb{N}$ write $\Delta_j := f(S^\star_{B_0}) - f(\bar{G}_j)$. By Fact 2 from Streeter and Golovin [SG08], for any $j \in \mathbb{N}$, $b > 0$ and $S \in \mathscr{S}$ with $\ell(S) \leqslant b$,

$$f(S) \leqslant f(\bar{G}_j) + b \cdot (s_j + \varepsilon_j), \tag{A.67}$$

where

$$s_j := \max_{(v,\tau)\in\mathscr{V}\times(0,\infty)} \frac{f(\bar{G}_j \oplus \langle (n,\tau)\rangle) - f(\bar{G}_j)}{\tau} = \frac{f(\bar{G}_j \oplus \bar{g}_j) - f(\bar{G}_j)}{\bar{\tau}_j} = \frac{f(\bar{G}_{j+1}) - f(\bar{G}_j)}{\bar{\tau}_j}, \tag{A.68}$$

so in particular for any j

$$f(S^\star_{B_0}) = \max_{S\in\mathscr{S}\,:\,\ell(S)=B_0} f(S) \leqslant f(\hat{G}_j) + B_0 \cdot (s_j + \varepsilon_j) \tag{A.69}$$

$$= f(\hat{G}_j) + B_0\left(\frac{f(\bar{G}_{j+1}) - f(\bar{G}_j)}{\bar{\tau}_j} + \varepsilon_j\right) \tag{A.70}$$

$$= f(\hat{G}_j) + B_0\left(\frac{\Delta_j - \Delta_{j+1}}{\bar{\tau}_j} + \varepsilon_j\right), \tag{A.71}$$

giving $\Delta_j \leqslant B_0\left(\frac{\Delta_j - \Delta_{j+1}}{\bar{\tau}_j} + \varepsilon_j\right)$.

Rearranging gives $\Delta_{j+1} \leqslant \Delta_j\left(1 - \frac{\bar{\tau}_j}{B_0}\right) + \bar{\tau}_j\varepsilon_j$ for each j, and unrolling this inequality and using that $1 - \frac{\bar{\tau}_j}{B_0} < 1\ \forall j$ as in Streeter and Golovin [SG08] gives us

$$\Delta_{L+1} \leqslant \Delta_1\left(\prod_{j=1}^{L} 1 - \frac{\bar{\tau}_j}{B_0}\right) + \sum_{j=1}^{L} \bar{\tau}_j\varepsilon_j. \tag{A.72}$$

By definition $B' = \sum_{j=1}^{L} \bar{\tau}_j\varepsilon_j$, and maximizing the product above subject to this constraint results in $\bar{\tau}_j = \frac{B'}{L}$ for all j. Thus

$$\prod_{j=1}^{L} 1 - \frac{\bar{\tau}_j}{B_0} \leqslant \prod_{j=1}^{L} 1 - \frac{B'/L}{B_0} = \left(1 + \frac{(-B'/B_0)}{L}\right)^L < e^{-B'/B_0} \tag{A.73}$$

and so

$$f(S^\star_{B_0}) - f(\bar{G}_{L+1}) = \Delta_{L+1} < \Delta_1 e^{-T_1/T_0} + \sum_{j=1}^{L} \bar{\tau}_j\varepsilon_j \leqslant f(S^\star_{B_0})e^{-B'/B_0} + \sum_{j=1}^{L} \bar{\tau}_j\varepsilon_j, \tag{A.74}$$

giving $f(\bar{G}_{\langle T_1\rangle}) = f(\bar{G}_{L+1}) > (1 - e^{-B'/B_0})f(S^\star_{B_0}) - \sum_{j=1}^{L} \bar{\tau}_j\varepsilon_j$ as required. □

Proof of Proposition 6.11

A lemma is needed:

Lemma A.2. *Let $f : \mathscr{S} \longrightarrow [0,1]$ be a job satisfying Assumption 6.10. Then for any schedule*

$S \in \mathscr{S}$ and any sub-schedule S' of S (i.e. a schedule $S' \in \mathscr{S}$ whose actions appear in order but not necessarily consecutively in S),

$$f(S) \geqslant f(S'). \tag{A.75}$$

Proof. Immediate from monotonicity, Assumption 6.10 and induction. □

Proof of Proposition 6.11. Suppose for each $i \in [K]$ there is a fictional experts algorithm (classical full feedback multi-armed bandit algorithm) $\mathscr{E}_i$ which picks $a^\star_{t,i}$ at each round t, and consider a hypothetical instance of the standard algorithm **OG** run with time allowance K and these fictional experts algorithms $\mathscr{E}_1, \dots, \mathscr{E}_K$ as subroutines.

Since $K = \Omega(N \log T)$ (by our assumption that $B = \Omega(NB' \log T)$), by Theorem 6.9 the N-regret of our **OG** instance is upper-bounded in expectation by $\sum_{i=1}^K \mathscr{R}_1(\mathscr{E}_i)$.

But the payoff received by this **OG** instance at each round t is $f(\langle a^\star_{t,1}, \dots, a^\star_{t,K} \rangle)$, which by Lemma A.2 is upper-bounded by $f(S_t)$, the payoff of $\mathbf{OG_{hybrid}}$, since the actions $a^\star_{t,1}, \dots, a^\star_{t,K}$ appear in order in S_t. So the N-regret $\mathscr{R}_N$ of $\mathbf{OG_{hybrid}}$ must be at most that of our fictional **OG** instance, giving the upper bound

$$\mathbb{E}[\mathscr{R}_N] \leqslant \sum_{i=1}^K \mathbb{E}[\mathscr{R}_1(\mathscr{E}_i)]. \tag{A.76}$$

It remains to argue how large each of the regret of each of these 'fictional' experts algorithms $\mathscr{E}_i$ is. Writing $a_i^{\star\star} = \arg\max_{a \in \mathscr{A}} \sum_{t=1}^T r_t^{(i)}(a)$ for the best-in-hindsight fixed action under the rewards passed to $\mathscr{E}_i$, the regret incurred by $\mathscr{E}_i$ is therefore

$$\mathscr{R}_1(\mathscr{E}_i) = \sum_{t=1}^T r_t^{(i)}(a_i^{\star\star}) - \sum_{t=1}^T r_t^{(i)}(a^\star_{t,i}) \tag{A.77}$$

$$= \sum_{t=1}^T r_t^{(i)}(a_i^{\star\star}) - \sum_{t=1}^T \max_{j \in [B']} r_t^{(i)}(a_t^{((i-1)B'+j)}) = \mathscr{R}_1(\mathscr{B}_i). \tag{A.78}$$

where $\mathscr{R}_1(\mathscr{B}_i)$ is the 1-regret incurred by multitasking bandit algorithm $\mathscr{B}_i$. So by Eq. (A.76)

$$\mathbb{E}[\mathscr{R}_N] \leqslant \sum_{i=1}^K \mathbb{E}[\mathscr{R}_1(\mathscr{B}_i)] = K\mathbb{E}[\mathscr{R}_1(\mathscr{B})] = \frac{B\mathbb{E}[\mathscr{R}_1(\mathscr{B})]}{B'}. \tag{A.79}$$

□

Appendix B

Further Empirical Results

B.1 Choice of perturbation rate

The experiments in Chapter 7 and Chapter 8 were run with the theoretically-derived values for the perturbation rate ε, which guarantee small asymptotic regret growth but may not always be optimal; here we investigate various other choices of ε in the black-box optimization setting from Chapter 7.

Tables B.1 and B.2 summarize the **FPML**[1] performance scores for a range of values of ε. The perturbation rate makes less difference than might be expected, though for small B in particular high perturbation rates predictably causes greater variance in outcome. A good choice in general for this application seems to be $\varepsilon = 0.3$.

B.2 Choice of exploration rate

We observed in Chapter 7 that adding probabilistic exploration with $\gamma = 0.2$ to the **FPML** algorithm with geometric resampling appeared to *decrease* the performance of the algorithm versus having no explicit exploration at all. Here we examine this more closely, looking at the empirical performance with various fixed choices of γ.

Tables B.3 and B.4 describe the **FPML** performance scores for $\gamma = \{0.0, 0.1, \ldots, 1.0\}$. As one would expect from our previous observation, the performance is always highest with $\gamma = 0$ (no explicit exploration), tapering off towards $\gamma = 1$ (entirely uniformly random action choices at each round). The effect of γ on score variance seems to be more unpredictable, with a slight tendency for values of γ closer to 0.5 to cause higher variance.

[1] In both Appendix B.1 and Appendix B.2 we use the partial feedback variant of **FPML** with geometric resampling and no explicit exploration.

Table B.1: Mean normalized validation scores of **FPML** with resampling and probabilistic exploration over black-box optimizers for various different perturbation rates (50 trials).

	Number of optimizers in parallel (B)					
	1	**2**	**3**	**4**	**5**	**6**
$\varepsilon = 0.001$	0.349	0.536	0.663	0.739	0.799	0.846
$\varepsilon = 0.003$	0.359	0.561	0.674	0.746	0.808	0.850
$\varepsilon = 0.01$	0.376	0.588	0.692	0.773	0.828	0.867
$\varepsilon = 0.03$	0.403	0.617	0.726	0.802	0.844	0.880
$\varepsilon = 0.1$	0.437	0.650	0.754	0.812	0.856	0.886
$\varepsilon = 0.3$	0.440	0.655	0.758	0.816	0.861	0.890
$\varepsilon = 1$	0.438	0.645	0.746	0.813	0.856	0.888
$\varepsilon = 3$	0.416	0.620	0.732	0.800	0.843	0.882
$\varepsilon = 10$	0.416	0.611	0.714	0.786	0.841	0.873
$\varepsilon = 30$	0.411	0.605	0.707	0.785	0.836	0.868
$\varepsilon = 100$	0.434	0.636	0.741	0.805	0.851	0.886
$\varepsilon = 300$	0.423	0.611	0.719	0.789	0.844	0.877
$\varepsilon = 1000$	0.405	0.610	0.710	0.785	0.835	0.869

Table B.2: Standard deviation of normalized validation scores of **FPML** with resampling and probabilistic exploration over black-box optimizers for various different perturbation rates (50 trials).

	Number of optimizers in parallel (B)					
	1	**2**	**3**	**4**	**5**	**6**
$\varepsilon = 0.001$	0.0193	0.0182	0.0158	0.0133	0.0114	0.0115
$\varepsilon = 0.003$	0.0157	0.0219	0.0172	0.0130	0.0119	0.0113
$\varepsilon = 0.01$	0.0177	0.0198	0.0188	0.0153	0.0109	0.0090
$\varepsilon = 0.03$	0.0242	0.0239	0.0211	0.0107	0.0101	0.0072
$\varepsilon = 0.1$	0.0211	0.0213	0.0176	0.0152	0.0099	0.0089
$\varepsilon = 0.3$	0.0288	0.0169	0.0145	0.0134	0.0090	0.0071
$\varepsilon = 1$	0.0267	0.0199	0.0129	0.0115	0.0084	0.0064
$\varepsilon = 3$	0.0246	0.0212	0.0150	0.0129	0.0101	0.0079
$\varepsilon = 10$	0.0217	0.0179	0.0133	0.0107	0.0089	0.0068
$\varepsilon = 30$	0.0177	0.0176	0.0139	0.0130	0.0092	0.0075
$\varepsilon = 100$	0.0205	0.0166	0.0172	0.0124	0.0087	0.0064
$\varepsilon = 300$	0.0248	0.0176	0.0142	0.0112	0.0088	0.0083
$\varepsilon = 1000$	0.0219	0.0189	0.0166	0.0113	0.0072	0.0086

Table B.3: Mean normalized validation scores of **FPML** with resampling and probabilistic exploration over black-box optimizers for various different exploration rates (50 trials).

	Number of optimizers in parallel (B)					
	1	2	3	4	5	6
$\gamma = 0.0$	0.424	0.654	0.754	0.811	0.853	0.888
$\gamma = 0.1$	0.433	0.654	0.752	0.803	0.846	0.883
$\gamma = 0.2$	0.422	0.643	0.739	0.795	0.840	0.867
$\gamma = 0.3$	0.416	0.629	0.730	0.786	0.829	0.856
$\gamma = 0.4$	0.421	0.620	0.713	0.773	0.819	0.851
$\gamma = 0.5$	0.405	0.604	0.708	0.767	0.809	0.843
$\gamma = 0.6$	0.388	0.589	0.693	0.760	0.805	0.837
$\gamma = 0.7$	0.378	0.577	0.681	0.751	0.799	0.836
$\gamma = 0.8$	0.363	0.562	0.674	0.744	0.796	0.836
$\gamma = 0.9$	0.359	0.545	0.667	0.740	0.795	0.838
$\gamma = 1.0$	0.345	0.536	0.657	0.733	0.798	0.842

Table B.4: Standard deviation of normalized validation scores of **FPML** with resampling and probabilistic exploration over black-box optimizers for various different exploration rates (50 trials).

	Number of optimizers in parallel (B)					
	1	2	3	4	5	6
$\gamma = 0.0$	0.0254	0.0191	0.0143	0.0112	0.0092	0.0076
$\gamma = 0.1$	0.0276	0.0202	0.0168	0.0167	0.0162	0.0099
$\gamma = 0.2$	0.0246	0.0235	0.0193	0.0157	0.0113	0.0133
$\gamma = 0.3$	0.0282	0.0287	0.0206	0.0169	0.0136	0.0115
$\gamma = 0.4$	0.0215	0.0238	0.0211	0.0200	0.0153	0.0114
$\gamma = 0.5$	0.0212	0.0264	0.0176	0.0157	0.0167	0.0138
$\gamma = 0.6$	0.0223	0.0229	0.0211	0.0147	0.0124	0.0124
$\gamma = 0.7$	0.0225	0.0220	0.0196	0.0164	0.0168	0.0112
$\gamma = 0.8$	0.0163	0.0230	0.0165	0.0182	0.0150	0.0127
$\gamma = 0.9$	0.0212	0.0189	0.0179	0.0139	0.0133	0.0121
$\gamma = 1.0$	0.0202	0.0197	0.0150	0.0165	0.0122	0.0096