Value Directed Exploration in Multi-Armed Bandits with Structured Priors

Bence Cserna **Marek Petrik** **Reazul Hasan Russel** **Wheeler Ruml**
University of New Hampshire
Durham, NH 03824 USA
bence, mpetrik, rrussel, ruml at cs.unh.edu

Abstract

Multi-armed bandits are a quintessential machine learning problem requiring balancing exploration with exploitation. While there has been progress in developing algorithms with strong theoretical guarantees, there has been less focus on practical near-optimal finite-time performance. In this paper, we propose an algorithm for Bayesian multi-armed bandits that utilizes approximate value functions. Building on previous work on UCB and Gittins index, we introduce *linearly-separable value functions* that capture the benefit of exploration when choosing the next arm to pull. Our algorithm enjoys a sub-linear performance guarantee and our simulation results confirm its strength in problems with structured priors. The simplicity and generality of our approach makes it a strong candidate for use in more complex multi-armed bandit problems.

1 INTRODUCTION

In the multi-armed bandit setup, a decision-maker repeatedly chooses from a finite set of actions. A reward is generated independently from a probability distribution associated with the action. The underlying reward distribution for each action is unknown to the decision-maker, but each action-reward pair can further inform future choices. Strong performance in this setup critically depends on the balance between exploring less well-understood actions and exploiting actions thought to provide high reward. In this sense, the problem captures the quintessence of the interplay between learning and decision-making.

Many approaches to the multi-armed bandit problem have achieved impressive theoretical and empirical results [Bubeck and Cesa-Bianchi, 2012; Kuleshov and Precup, 2014]. There is, however, growing recognition that more wide-spread practical use will require algorithms that can better exploit structured prior information [Russo and Roy, 2014; Lattimore and Szepesvári, 2016]. For example, consider a bandit problem in which the arms represent different levels of customer discounts such as 5% or 10%. The conversion probabilities for the discounts are not known before the promotion starts, but one can safely assume that more customers choose to buy the product when offered a 10% rather than a 5% discount. Such prior information should ideally be used to increase the efficiency of exploration in particular when a large number of discounts is considered. While in some problems such prior information can be captured using parametric models, like GLM-UCB [Filippi *et al.*, 2010], such models make many additional assumptions that are difficult to verify when little or no data is available.

In this paper, we propose a Bayesian bandit algorithm designed to use structured prior information in order to achieve good short-term performance with a small number of arm pulls. We take an approach based on online planning, using lookahead search to consider states to which each possible sequence of actions might lead. Relying on lookahead search makes it possible to easily exploit structured prior information when it is available. Because the state space grows exponentially with the number of arms, it is impossible to enumerate all reachable states to the problem horizon. Thus, this formulation of the problem relies crucially on having a value function that can be applied at a modest depth cut-off in lieu of further state search.

Many methods for computing approximate value functions have been developed in reinforcement learning and some have been used to solve some multi-armed bandit problems [Whittle, 1988; Adelman and Mersereau, 2008]. But they can be computationally intensive and cumbersome to use. As our main contribution, we de-

rive in Section 3 a new method for computing *linearly-separable value functions* that, when used in concert with lookahead search, performs as well as state-of-the-art-algorithms. The method computes value functions by exploiting existing algorithms and the weakly-coupled property of multi-armed bandit problems. It also enjoys sublinear regret as we show in Section 4. Our algorithm is simple to implement and, as we demonstrate in Section 5, performs well in bandit problems with structured prior information.

Given the fundamental simplicity of our approach and its empirical success, we are optimistic that it may provide a basis for addressing more complex problems, such as contextual bandits, in the future. Our approach also opens the door to bandit algorithms that can yield improved performance when additional computation time is available.

2 BACKGROUND

We begin by describing the Bayesian multi-armed bandit problem in more detail. We focus on the case of Bernoulli bandits, deferring discussion of more complex models to Section 5.2. We then briefly review previously proposed algorithms before turning to our new method.

2.1 PROBLEM FORMULATION

The decision-maker in the standard multi-armed bandit problem aims to maximize the cumulative return by repeatedly choosing one of N arms: $\mathcal{A} = \{a_1, \ldots, a_N\}$. Choosing an arm a_i results in receiving a reward $R_i \in \{0, 1\}$ distributed according to a Bernoulli distribution with a mean μ_i. The mean μ_i is not known in advance. To achieve the maximal cumulative return over a horizon of T steps, the decision-maker must balance exploration to learn about the expected returns of arms with exploitation in order to learn which arms are more likely to provide high rewards.

In the Bayesian variant of the problem, the decision-maker has access to a prior distribution over the expected reward $\mu_1, \mu_2, \ldots, \mu_N$ for each arm $a_1, a_2, \ldots, a_N$. We use $\mu = (\mu_1, \ldots, \mu_N)$ to represent the prior parameters of the bandit; each μ_i is distributed according to a Beta distribution. As in most machine learning settings, the Bayesian approach has both advantages and disadvantages —a proper discussion is beyond the scope of this paper and we refer to Kaufmann *et al.* [2012a]; Russo and Van Roy [2014]; Kim and Lim [2015] and the references therein for details.

The Bayesian multi-armed bandit problem can be modeled as a Markov Decision Process (MDP). The state in this MDP represents the sufficient statistic of the history of the observed rewards for each arm. We denote the state space of the MDP that represents the multi-armed bandit problem as $\mathcal{S} = \mathcal{S}_1 \cup \mathcal{S}_2 \cup \ldots \cup \mathcal{S}_T$, where $\mathcal{S}_t$ is the set of states at time t. The actions in this MDP are simply the pulls of arms of the bandit. Fig. 1 shows a fragment of the MDP for a two-arm bandit, illustrating the transition from one state to the four possible successors.

In order for the Markov property to hold, each state of the MDP must represent the posterior distribution of μ_i given the history of rewards for every arm a_i. The posterior estimate $\tilde{\mu}_i$ of μ_i is distributed according to the Beta distribution $\mathrm{Beta}(\alpha, \beta)$ with some parameters α, β since it is the conjugate prior to Bernoulli distribution (e.g., [Gittins *et al.*, 2011]). Any state $s \in \mathcal{S}$ can, therefore, be represented as

$$s = \Big((\alpha_1, \beta_1), (\alpha_2, \beta_2), \ldots, (\alpha_N, \beta_N) \Big) ,$$

where α_i, β_i represent the Beta distribution parameters for each arm i. We use α_i^0, β_i^0 to denote the parameters of the prior Beta distributions and generally assume that $\alpha_i^0 = \beta_i^0 = 1$ which corresponds to the uniform prior. The parameters α, β of the Beta distribution have a convenient interpretation: after observing n_s successes (value 1) and n_f failures (value 0) for R_i then $\tilde{\mu}_i \sim \mathrm{Beta}(\alpha_i^0 + n_s, \beta_i^0 + n_f)$. Thus the transition, after pulling an arm a_i, consists of merely adding one to the appropriate α_i or β_i based on the observed reward.

When the bandit is in state s_t and the decision maker chooses arm a_i, the subsequent state is represented by a random variable $S_{t+1}(s_t, a_i)$. When $s_t = (\ldots, (\alpha_i, \beta_i), \ldots)$, the random variable $S_{t+1}(s_t, a_i)$ is distributed as

$$\mathbb{P}\left[S_{t+1} = (\ldots, (\alpha_i + 1, \beta_i), \ldots) \right] = \frac{\alpha_i}{\alpha_i + \beta_i} , \quad (1)$$

$$\mathbb{P}\left[S_{t+1} = (\ldots, (\alpha_i, \beta_i + 1), \ldots) \right] = \frac{\beta_i}{\alpha_i + \beta_i} , \quad (2)$$

where the transition probabilities follow from the definition of the mean of the Beta distribution. To reduce clutter, we omit s_t and a_i when they are obvious from the context. The rewards received in transitions (1) and (2) are 1 and 0 respectively.

By specifying which arm to pull at any given state, a multi-armed bandit algorithm defines a policy for this bandit MDP. The decision rule at time t is denoted as $\pi_t : \mathcal{S}_t \to \mathcal{A}$ and the policy is the collection of the decision rules $\pi = \{\pi_t \mid t = 1 \ldots T\}$ for each time step t. The celebrated Gittins index defines the optimal policy

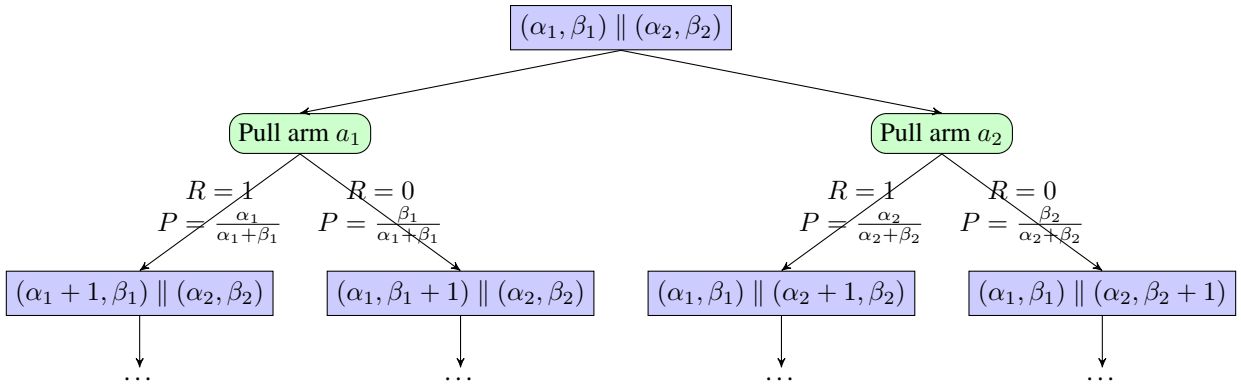

Figure 1: A single transition of the Bernoulli multi-armed bandit problem.

for the discounted infinite-horizon version of the bandit problem, but that method is not based on directly solving the MDP. In most practical problems, it is impossible to compute the optimal policy because the number of states grows exponentially with the number of arms. Unfortunately, while Gittins index may be optimal, it provably does not generalize to most other bandit problems [Gittins *et al.*, 2011].

The established performance measure for classic bandit algorithms is the regret, sometimes referred to as pseudo-regret [Bubeck and Cesa-Bianchi, 2012], which is defined for a particular realization of the bandit parameters μ and policy π as

$$\mathrm{Regret}(\pi, T, \mu) = \sum_{t=1}^{T} \Big(\max_{i=1\ldots N} \mu_i - \mathbb{E}\left[R_{\pi(S_t)} \mid \mu \right] \Big) ,$$

where S_t is a random variable that represents the state of the bandit process. In Bayesian bandits, it is natural to instead evaluate Bayesian regret:

$$\mathrm{BayesRegret}(\pi, T) = \mathbb{E}_\mu \left[\mathrm{Regret}(\pi, T, \mu) \right] .$$

We aim to minimize Bayesian regret with a particular focus on the regret in the first few steps. While the guarantees provided by a small bound on the Bayesian regret are somewhat weaker than that of regular regret, it is a very reasonable measure in most circumstances.

2.2 PREVIOUS WORK

The literature on bandit problems is enormous, so we will focus only on the most relevant algorithms. The UCB family of algorithms [Auer *et al.*, 2002] use the problem structure to derive tight optimistic upper bounds. While these algorithms are simple and have been used in various applications with success, they lack the ability to incorporate structured prior information such as arm dependency or different reward policies without requiring complex and difficult re-analysis of the upper bounds. Kaufmann *et al.* [2012a] propose Bayes-UCB, a Bayesian index policy that improves on UCB in Bayesian bandits by taking advantage of the prior distribution.

Russo and Roy [2014] describe a method that improves on the optimistic approach taken by UCB algorithms. Their method considers the information gain from taking an action to optimize the exploration-exploitation trade-off. They provide a strong Bayesian regret bound that applies for a general class of models. Our new method is based on a similar principle but uses additive value functions to estimate the information gain.

Thompson sampling works by choosing an arm based on its probability of being the best arm. Specifically, the method draws a sample from the decision maker's current belief distribution for each arm and then chooses the arm that yielded the highest sample. The performance of Thompson sampling has been proved to be near optimal, and it is simple and efficient to implement. Thompson sampling can easily be adapted to a wide range of problem structures and prior distributions [Agrawal and Goyal, 2012; Leike *et al.*, 2016; Kaufmann *et al.*, 2012b]. For example, one can reject sets of samples that contradict contextual information. However, the simplicity of the method makes it also difficult to improve its performance.

Gittins indices exploit the weak dependence between actions to compute the optimal action in time that is linear in the number of arms [Gittins, 1979; Chakravorty and Mahajan, 2014]. Gittins indices, however, are guaranteed to be optimal only for the basic multi-armed bandit problem, require a discounted infinite-horizon objective, and provably cannot be extended to most interesting practical problems which involve correlations between arms or an additional context [Gittins *et al.*, 2011].

3 SEPARABLE VALUE FUNCTIONS

We now turn to our new approach, which we call "ELSV" (Exploration via Linearly Separable Value Functions). As described above, our main goal is a method that is flexible and takes advantage of the complex problem structure or prior knowledge in order to reduce the regret. We achieve this by taking a state-space search-based approach and by leveraging the exploration-exploitation trade-off behavior of existing algorithms to build good value functions.

The ELSV algorithm, as described in this section, does not improve the performance of existing methods when applied to Bernoulli bandits. In Section 5, we describe how it can be extended easily to settings with prior information in which it significantly outperforms state-of-the-art methods.

Algorithm 1: One-step lookahead algorithm

Input: Current time step t, current state s_t, and value function $v_t : \mathcal{S}_t \rightarrow \mathbb{R}$
Output: Arm to pull at time step t

for $a \in \mathcal{A}$ **do**
  `// ` $q_t(s_t, a)$ `is the expected value for action` a `in state` s_t
  $q_t(s_t, a) \leftarrow \mathbb{E}\left[r(s_t, a, S_{t+1}) + v_{t+1}(S_{t+1}) \right]$;
return $\arg\max_{a \in \mathcal{A}} q_t(s_t, a)$;

Our general approach is based on an n-step lookahead guided by a specific value function. This is an instance of receding horizon control, a common approach to solving online planning and reinforcement learning problems [Sutton and Barto, 1998]. The state space is enumerated to depth n, at which point a value function is evaluated at the frontier states to avoid further expansion. Note that, because there are multiple ways of reaching a state in the MDP, the state space forms a graph and it is important to detect and merge duplicate states. The values are backed up to the current state at the root and the best-looking action is chosen. After the outcome of pulling the arm is observed, the cycle repeats again with a fresh lookahead. A simplified version of the algorithm, depicted in Algorithm 1, estimates the value of each action by computing the expected value for the next step.

Since Algorithm 1 does not rely on any complex confidence bounds, one would expect that it can easily generalize to many different problems. Choosing a longer lookahead horizon also offers the promise of trading off computational time for more efficient exploration. The quality of this algorithm will clearly depend on whether it is supplied with a good value function v.

There has been little previous work that considered a value function-driven approach to bandit problems, with Adelman and Mersereau [2008] being one notable exception. This is perhaps because UCB is simple and efficient, while computing value functions via approximate dynamic programming requires complex computation and can be unreliable. We show, however, that it is possible to *efficiently* construct good value functions directly from UCB and other popular bandit algorithms. Surprisingly, such value functions are simple and linearly separate over the individual arms.

Before describing how we construct the value function, consider what it is supposed to represent. Consider, for example, a state $s = ((2, 3), (10, 10))$ in a two-arm bandit problem with 10 steps remaining until the end of the horizon T is reached. The expected returns of the two arms are $2/5 \cdot 10 = 4$ and $1/2 \cdot 10 = 5$ respectively. One could simply assign $v_{10}(s) = \max\{4, 5\} = 5$, but this would not be precise. The first arm, while apparently having a lower expected mean, is far less certain than the second arm (because of a smaller number of pulls). In order to achieve good results, and in particular a sub-linear regret guarantee, the value function must consider not only the expected return but also the confidence of the estimates. Another way to put it is that the value function must model both the expected return (exploitation) and the benefit of exploration.

Algorithm 2: Index Policy

Input: Current time step t, current state s_t, and index function $z_t : \mathcal{S}_t \times \mathcal{A} \rightarrow \mathbb{R}$
Output: Arm to pull at time step t

return $\arg\max_{a \in \mathcal{A}} z_t(s_t^i, a_i)$;

We seek to take advantage of the implicit value of exploration that is encoded by existing multi-armed bandit index algorithm. Algorithm 2 shows a canonical example of an index-based algorithm. UCB, Gittins index, and many other methods fit this basic mold. Note that the index $z_t(s_t^i, a_i)$ is computed for each arm separately. The notation s_t^i denotes the component of state s_t that corresponds to arm a_i, that is $s_t^i = (\alpha_i, \beta_i)$. For example, if $s_t = ((5, 2), (4, 6))$ then $s_t^2 = (4, 6)$. An important property of the index function $z_t(s_t^i, a_i)$ is that it is completely independent of the states of other arms and can thus be computed efficiently.

The challenge when constructing an index algorithm is how to define the index value z_t. If we use the Bayesian expectation of the immediate return in place of the standard frequentist one, then the value of the index for the

α-UCB algorithm for each arm $a \in \mathcal{A}$ is

$$\begin{aligned} z_t^{\text{UCB}}(s_t^i, a_i) &= r(s_t^i, a_i) + \sqrt{\frac{\alpha \log t}{T_i(s_t^i)}} \\ &= \underbrace{r(s_t^i, a_i)}_{\text{Immediate reward}} + \underbrace{b_t^{\text{UCB}}(s_t^i, a_i)}_{\text{Exploration bonus}} \end{aligned} \tag{3}$$

where $r(s_t^i, a_i) = \mathbb{E}\left[r(s_t^i, a_i, S_{t+1}^i) \right] = {}^{\alpha_i}/_{\alpha_i+\beta_i}$ is the expected reward after pulling arm a_i, $T_i(s_t^i) = \alpha_i + \beta_i - 2$ is the number of times the arm has been pulled (recall that the initial states is $\alpha_i^0 = \beta_i^0 = 1$), and b_t^{UCB} is the exploration bonus.

The classic UCB algorithm uses $\alpha = 2$, but sub-linear regret can be in fact shown with $\alpha > 1$ [Bubeck and Cesa-Bianchi, 2012]. Lower values of α typically lead to better empirical performance but also make it more difficult to bound the regret. We use $\alpha = 1$ unless otherwise specified.

Another celebrated example of an index policy is the Gittins index, see for example [Gittins *et al.*, 2011]. While UCB is only *asymptotically* optimal (up to a constant factor), following the Gittins index is truly optimal in some simple bandit settings. For example, Gittins indices are optimal for an infinite-horizon discounted Bernoulli bandit problem. We generally use discount $\gamma = 0.99$ and the horizon of 1000 when approximating the infinite horizon.

Unlike UCB, Gittins index does not have a closed-form expression, but it must be precomputed. Since the index is computed for each arm independently, it can be computed and used efficiently regardless of the number of arms in the bandit problem [Niño-Mora, 2011]. We use $z_t^{\text{Gitt}}(s_t^i, a)$ to denote the value of the Gittins index and $b_t^{\text{Gitt}}(s_t^i, a_i) = z_t^{\text{Gitt}}(s_t^i, a_i) - r(s_t^i, a_i)$ to denote the exploration benefit that it assigns to the arms. Many other multi-armed bandit methods have been proposed, some examples are Bayes-UCB [Kaufmann *et al.*, 2012a] or UCB-V [Audibert *et al.*, 2009] to name a few.

We are now ready to describe ELSV, the new method for constructing linearly-separable value functions. A linearly-separable value function for components $\upsilon_t^i : \mathcal{S}_t^i \to \mathbb{R}$ is such that, for each $t \in \mathcal{T}$ and for each state $s_t \in \mathcal{S}_t$,

$$v_t(s_t) = \sum_{a_i \in \mathcal{A}} \upsilon_t^i(s_t^i) \,. \tag{4}$$

Linearly separable value functions are attractive due to their simplicity and have been used widely in reinforcement learning and approximate dynamic programming [Powell, 2008; Powell *et al.*, 2004; Rust, 1996] and previously for approximating the value function in bandit problems [Adelman and Mersereau, 2008].

We begin with an arbitrary bandit index function z_t^i and the corresponding exploration bonus function b_t^i. Each component υ_t^i must satisfy the following condition for every a_i, t, and $\tau \le t$:

$$\begin{aligned} \upsilon_{t+1}^i(s_\tau^i) &= \mathbb{E}\left[\upsilon_{t+1}^i(S_{\tau+1}^i) \right] + r(s_\tau^i, a_i) - z_t(s_\tau^i, a_i) \\ &= \mathbb{E}\left[\upsilon_{t+1}^i(S_{\tau+1}^i) \right] - b_t(s_\tau^i, a_i) \,, \end{aligned} \tag{5}$$

where $S_{\tau+1}^i$ is short for $S_{\tau+1}^i(s_\tau, a_i)$, the random variable that represents the state following the pull of arm a_i. It is important to note that all υ's involved use the same time index $t+1$ while the states are over two time steps τ and $\tau+1$.

To understand the requirement in (5) more intuitively, we can rewrite it as

$$\mathbb{E}\left[\upsilon_{t+1}^i(S_{\tau+1}^i) \right] - \upsilon_{t+1}^i(s_\tau^i) = b_t(s_\tau^i, a_i) \,.$$

The term $\mathbb{E}\left[\upsilon_{t+1}^i(S_{\tau+1}^i) \right]$ represents the expected value of the state at time $t+1$ after pulling the arm a_i; $S_{\tau+1}$ is a random variable. In contrast, the term $\upsilon_{t+1}^i(s_\tau^i)$ represents the expected value of the state s_τ^i at time $t+1$ when arm a_i is not pulled. The difference between these two terms is the change in the value of the current state, or in other words, how much we have learned about the arm after pulling it. And this increase in information about the arm should be equal to the exploration bonus assigned by the index.

Algorithm 3: ELSV: Computing linearly separable value functions that satisfy (5).

Input: Arm a_i, time step $t \in 1, \ldots, T$, and exploration bonus function $b_t : \mathcal{S}_t \times \mathcal{A} \to \mathbb{R}$
Output: Value function υ_t^i for arm a_i at time t

$\mathcal{S}_t \leftarrow \left\{ (\alpha, \beta) \in \mathbb{N}_{>0}^2 \mid \alpha + \beta - 2 \le t - 1 \right\}$;
$\upsilon_t^i(s) \leftarrow 0 \quad \forall s \in \mathcal{S}_t$;
`// τ = num. of pulls of a_i so far`
for $\tau = t - 2$ **to** 1 **do**
  `// Iterate possible histories`
  foreach $(\alpha, \beta) \in \{ s \in \mathcal{S}_\tau \mid \alpha + \beta - 2 = \tau \}$ **do**
    $p \leftarrow \frac{\alpha}{\alpha+\beta}, \quad q \leftarrow \frac{\beta}{\alpha+\beta}$;
    $\upsilon_t^i(\alpha, \beta) \leftarrow$
      $p \cdot \upsilon_t^i(\alpha+1, \beta) + q \cdot \upsilon_t^i(\alpha, \beta+1) - b_t(\alpha, \beta)$;
return υ_t^i

Theorem 3.1. *Let π_I represent Algorithm 2 with index function $\hat{z}_t$. Suppose a policy π_V represents Algorithm 1 with value function $\hat{v}_t$ as defined in* (4)–(5) *using an index function $\hat{z}_t^i$. Then $\pi_V(s_t) = \pi_I(s_t)$ for each $s_t \in \mathcal{S}_t$.*

We defer the proof of Theorem 3.1 to Appendix A.1. It follows by comparing the value of pulling an arm with the value of a hypothetical state which would have resulted from not pulling any arm. The argument relies on

the fact that a policy is not affected by adding or subtracting a constant from the value function for all states in $\mathcal{S}_t$ for any t.

The simplest arm index would simply ignore the exploration bonus by setting it to $b_t(s_t^i, a_i) = 0$. The value function for this setting will be a constant and the 1-step lookahead policy will simply be greedy. Having no exploration bonus, therefore, leads to no exploration and pure exploitation.

The value function in Theorem 3.1 induces the same policy as the index, but it is still an approximation of the true value of each state. Therefore, a value function v constructed from the Gittins index will lead to the optimal policy (for the discounted infinite-horizon bandit) but it is not the optimal value function.

Algorithm 3 describes a dynamic programming method that can be used to compute a value function that satisfies (5). The following proposition, which can be shown readily by algebraic manipulation, states the complexity of the algorithm.

Proposition 3.1. *The linearly separable value function v_t^i can be computed using dynamic programming with computational complexity $O(t^3)$.*

Since the values v are computed independently for each t they do not have to be pre-computed ahead of time but can be computed on as needed basis and only for the states relevant to choosing an action. It is also important to note that the complexity in Proposition 3.1 is independent of the number of arms and the complexity of the 1-step lookahead in Algorithm 1 is linear in the number of arms. ELSV can scale to a large number of arms with no significant difficulties.

Fig. 2 depicts value functions computed by ELSV for Gittins index at $t = 10$ and $t = 200$ for each state. The states in $\mathcal{S}$ are mapped to a 2-dimensional space and the contours indicate the value function at that state. The number of arm pulls on the vertical axis can be smaller than t since the arm may not be pulled in every time step. Fig. 3 shows the value function computed by ELSV for UCB index for comparison. As noted above, the constant offset of the value functions is irrelevant to the quality of the policy. The value functions in the plots are offset to satisfy,

$$v_t^i(s_t) \geq \max\left\{r(s_t^i, a_i) + \mathbb{E}\left[\, v_{t+1}^i(S_{t+1}^i)\,\right], v_{t+1}^i(s_t^i)\right\} ,$$

which leads to more reasonable value estimates without influencing the policy.

The value function of UCB is notably simpler than the one for Gittins index. As expected, the UCB value function is *concave* and increasing but independent of the expected success probability. This indicates that exploration in UCB is really driven just by the immediate reward and the certainty in it—the potential long-term benefits of the arm are ignored. On the other hand, the value function for the Gittins index value exhibits a curious structure: it increases toward both low and high probabilities. This is counterintuitive as one would expect the value function to monotonically increase with the success probability. In a multi-armed bandit, however, it is also valuable to learn that an arm is not good which reduces the need for further exploration. Arms with medium success probabilities do not provide high rewards and yet require significant exploration. Notice also that this property is much more exaggerated at $t = 200$.

In the remainder of the paper, we discuss regret bounds ELSV and evaluate its performance experimentally on a bandit problem with additional problem structure.

4 ANALYSIS OF REGRET

In this section, we prove that ELSV with the UCB value function has sublinear regret. The sublinear bound on the regret is not surprising; Theorem 4.1 shows that ELSV with such a value function behaves identically to UCB which enjoys sublinear regret bounds. Instead, the main goal of this section is to establish a new methodology that can be used to analyze regret of multi-armed bandit algorithms driven by value functions.

Our goal is, in particular, to derive regret bounds that depend on some property of the value function used by ELSV. We need additional notation to describe such a property concisely. Let $q_t(s, a)$ stand for the expected value after pulling an action a in state s_t:

$$q_t(s_t, a) = r(s_t, a) + \mathbb{E}\left[\, v_{t+1}\Big(S_{t+1}(s_t, a)\Big)\,\right] .$$

As we show below, to bound regret it is sufficient to establish an upper bound on:

$$\varphi_t(\mu, s_t, a_i) = q_t(s_t, a_i) - (\mu_i + v_{t+1}(s_t)) . \qquad (6)$$

This value φ_t compares the estimate of the expected value $q_t(s_t, a_i)$ with a more precise estimate of the same value $(\mu_i + v_{t+1}(s_t))$. The more precise estimate uses the unknown parameter μ. One could also interpret φ_t as a finite-horizon form of the Bellman residual used in bounds on performance loss in reinforcement learning (e.g. [Petrik and Zilberstein, 2011]).

The following lemma plays a key role in establishing the regret bounds. It shows how the regret of ELSV can be decomposed into two components: one is independent of the algorithm and the other one is independent of the optimal action.

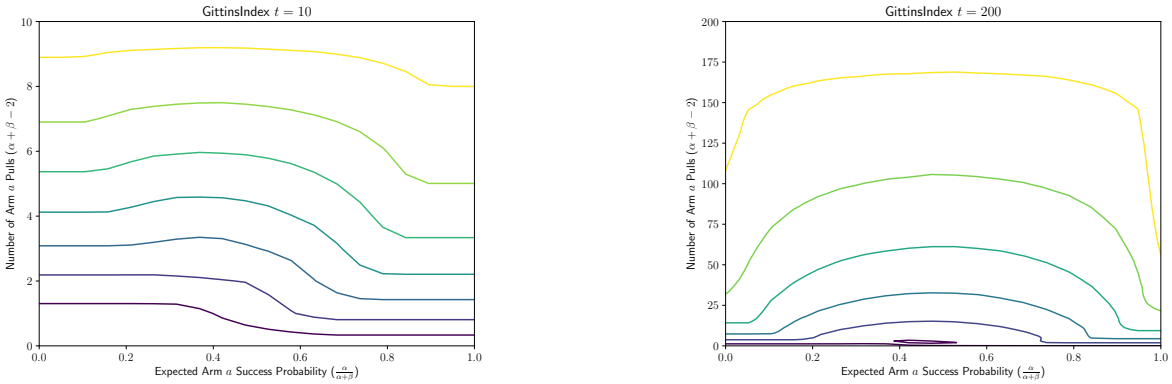

Figure 2: Value functions for Gittins index at $t = 10$ and $t = 200$ (after 10 and 200 pulls of some arm).

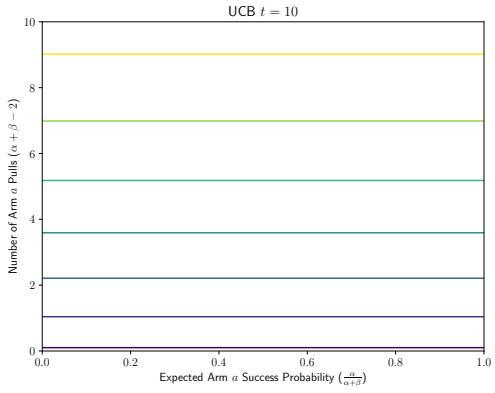

Figure 3: Value function for UCB index at $t = 10$ (after 10 pulls of some arm).

Lemma 4.1. *The regret for any policy π_t computed using 1-step lookahead with respect to value function v_t is upper bounded as follows:*

$$\mathrm{Regret}(\pi, T, \mu) \leq \sum_{t=1}^{T} \mathbb{E}_{S_t} \left[\varphi_t(\mu, S_t, \pi(S_t)) \right] - \\ - \sum_{t=1}^{T} \mathbb{E}_{S_t} \left[\varphi_t(\mu, S_t, a_{i^\star_\mu}) \right] ,$$

where S_t is the random variable representing the state at time t under policy π and $i^\star_\mu = \arg\max_{i \in \mathcal{A}} \mu_i$ is the optimal action for the unknown parameter μ.

The proof of the lemma, deferred to Appendix A.2, follows readily from the fact that ELSV chooses actions that maximize q_t.

The actual regret bound depends, of course, on the value function that is used. We now use Lemma 4.1 to bound the regret of the policy that uses UCB derived value function v^{UCB} as described in (5).

Theorem 4.1. *The regret of policy π_U of Algorithm 1 that uses $\hat{v}^{UCB}$ (with $\alpha > 1$) is bounded as:*

$$\mathrm{Regret}(\pi_U, T, \mu) \leq O(\sqrt{T \log(T)}) \, .$$

The proof of the theorem can be found in Appendix A.3. Note that this bound is not tighter than existing bounds on UCB algorithms, but it does establish sublinear regret of ELSV.

To establish the bound in Theorem 4.1, it is necessary that the difference of the residuals (6) for the arm chosen by ELSV and the optimal arm are not only small but must also decrease at least quickly $1/\sqrt{t}$. In other words, it is not sufficient for the errors in (6) to be small, they also must decrease as more information about the returns of the arms becomes available.

5 EXPERIMENTAL RESULTS

In this section, we compare the performance of ELSV to that of UCB, Bayes-UCB, Thompson sampling, and Gittins indices in simulation. We first analyze in Section 5.1 the impact of the lookahead horizon on the performance in the plain Bernoulli bandit setting. Then, in Section 5.2, we describe the application to a problem in which structured prior information is available.

Unlike UCB and Thompson sampling, Gittins index must be pre-computed in advance. It is also optimal only for infinite-horizon discounted problems. The index that we use in our experiments was computed with a discount factor $\gamma = 0.99$ and horizon of 1000 to approximate the infinite-horizon value. We computed the index using the calibration method with the step size of

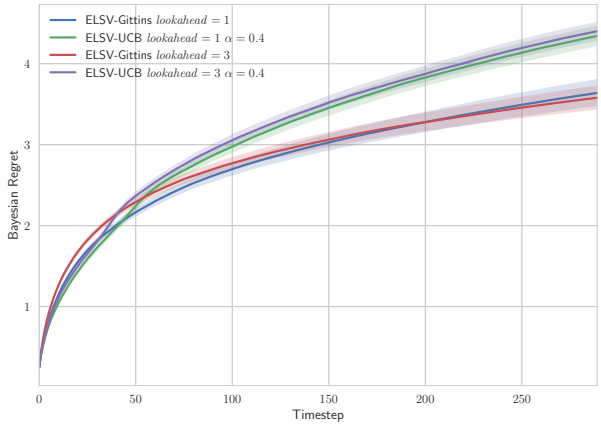

Figure 4: Bernoulli bandit regret with lookahead of 1 and 3 steps.

0.001 as described, for example, in Niño-Mora [2011]. Other approximations for computing the index have been proposed recently [Gutin and Farias, 2016; Lattimore, 2016].

5.1 BERNOULLI BANDITS

Our first set of experiments is in the standard Bernoulli bandit setting as described above in Section 2.1. As shown in Theorem 3.1, ELSV's performance will be identical to the index algorithm it is based upon. And our experiments indeed confirm this fact.

The main purpose of the experimental evaluation in this section is to understand the effect of the size of the lookahead on the performance of the algorithm. A reasonable assumption in online planning algorithms is that their performance generally improves with an increasing horizon size. Fig. 4 compares the Bayesian regret of 1-step and 3-step lookaheads in a 3-armed problem with value functions computed by ELSV. We use two value functions, one computed from the Gittins index and another one from α-UCB with $\alpha = 0.4$ (this α is unrelated to the α value used in each state). The results are averaged over 5200 problem instances with arm success probabilities drawn from the uniform Beta distribution and the shaded areas around the curves show 95% confidence intervals.

Fig. 4 highlights a surprising finding: longer lookahead does not reduce the regret. We observed virtually no improvement in the regret up to a horizon 10 at which point the search becomes computationally intractable. We hypothesize that a more careful, focused, and deeper search would be more likely to yield improvements.

5.2 CONSTRAINED BANDITS

Algorithm 4: Constrained single step lookahead algorithm using rejection sampling

Input: Current time step t, current state s_t, and value function $v_t : \mathcal{S}_t \to \mathbb{R}$
Output: Arm to pull at time step t

$\mathcal{X} \leftarrow \emptyset$
$(\alpha, \beta) \leftarrow s_t$
for $k \in 1 \ldots \mathrm{SampleCount}$ **do**
  Sample $\theta_i \sim \mathrm{Beta}(\alpha_i, \beta_i)$ for each $a_i \in \mathcal{A}$
  if $\theta_1 \geq \theta_2 \geq \ldots \geq \theta_n$ **then**
    $\mathcal{X} \leftarrow \mathcal{X} \cup \{(\theta_1, \theta_2, \ldots, \theta_N)\}$
// compute average
$\tilde{\mu}_i \leftarrow \frac{1}{|\mathcal{X}|} \cdot \sum_{\theta \in \mathcal{X}} \theta_i$;
for $i \in 1 \ldots N$ **do**
  // value of success
  $\mathrm{sc} \leftarrow r(s_t, a_i, (\alpha_i + 1, \beta_i)) + v^i_{t+1}(\alpha_i + 1, \beta_i)$;
  // value of failure
  $\mathrm{fl} \leftarrow v^i_{t+1}(\alpha_i, \beta_i + 1)$;
  // action value
  $q_t(s_t, a_i) \leftarrow \tilde{\mu}_i \cdot \mathrm{sc} + (1 - \tilde{\mu}_i) \cdot \mathrm{sc}$;
return $\arg\max_{a \in \mathcal{A}} q_t(s_t, a)$;

The constrained Bernoulli bandit problem represents a more challenging case that is not handled well by existing algorithms. As described in the introduction, this problem is motivated by an application when trying to optimizing the level of *personalized* discount offers to customers in an e-commerce setting. It has been studied extensively in operations research using customer choice models [Train, 2003]. Although such choice models can be combined with UCB methods, their application with no historical data is often problematic. For the purpose of these experiments, we simply assume that the success probabilities do not increase with a *decreasing* discount percentage: $\mu_1 \geq \mu_2 \geq \ldots \geq \mu_N$. Arm $i + 1$ represents a *smaller* discount than arm i: the experimental results use discount levels of $20\%, 10\%, 0\%$ for arms $1, 2, 3$ respectively. Our approach can be used also with a much more complex set of possible prior assumptions.

Unlike in regular Bernoulli bandits, the rewards r_i depend on arm i. That is, after pulling an arm (choosing a discount level) we received reward r_i if the customer decides to purchase the product and 0 otherwise. Since the arm discount decreases with the index i, the rewards satisfy: $r_1 < r_2 < \ldots < r_N$.

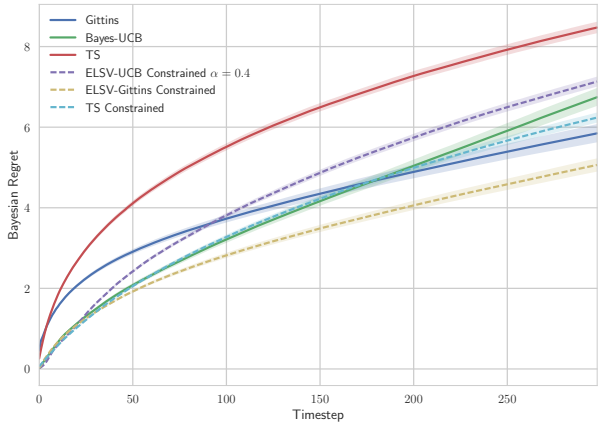

Figure 5: Bayesian Regret of constrained Bernoulli bandit.

Adapting Algorithm 1 to this constrained setting is relatively straightforward. We use the linearly separable value function computed by ELSV and only modify the lookahead to respect the constraints on μ. In particular, we use *rejection sampling* to appropriately adjust the transition probabilities when updating the values in the lookahead. As Algorithm 4 shows, we essentially compute the conditional probability distribution for each μ_i given the observations for that arm as well as the observations for other arms. This probability distribution must be estimated empirically as it does not have a closed form. The algorithm is only a heuristic in this setting and we have no regret bounds yet.

Fig. 5 shows the regret of ELSV (with 1-step lookahead) compared with several state-of-the-art algorithms on the constrained bandit problem with 3 arms and averaged over 5300 problem instances and with 95% confidence intervals. ELSV-UCB and ELSV-Gittins use a value function computed from UCB and Gittins index respectively. We omit the regret of UCB from the plot because its regret was much higher than that of the other algorithms. TS stands for regular Thompson sampling that ignores the constraints on μ, while "TS Constrained" samples from the constrained posterior distribution using rejection sampling similarly to Algorithm 4.

Our results show that ELSV-Gittins outperforms all other algorithms even in a problem with 3 arms. The magnitude of improvement in ELSV-Gittins over Gittins index grows as the number of arms in the problem increases since the constraint becomes more important and more informative. ELSV-UCB performs a bit worse, it but still represents a very significant improvement over plain UCB.

6 CONCLUSION

We have proposed a new approach to bandit problems focused on good short-term performance in problems with structured prior information. The approach is based on a new kind of linearly separable value function that incorporates the value of exploration and can be used in concert with online planning methods. Our method, ELSV, performs close to optimal on basic Bernoulli bandits and can significantly outperform existing methods in problems with prior information. The results on simple bandit problems are promising and we hope to extend the approach also to contextual bandits. We also believe that ELSV is a good first step in developing more sophisticated value-directed methods in the future.

ACKNOWLEDGMENTS

We thank the anonymous reviewers for detailed comments that helped to improve this paper significantly. This work was in part supported by the National Science Foundation under Grant No. IIS-1150068.

REFERENCES

Daniel Adelman and AJ Mersereau. Relaxations of weakly coupled stochastic dynamic programs. *Operations Research*, 56(3):712–727, 2008.

Shipra Agrawal and Navin Goyal. Analysis of Thompson sampling for the multi-armed bandit problem. In *Annual Conference on Learning Theory (COLT)*, pages 39.1–39.26, 2012.

Jean Yves Audibert, Rémi Munos, and Csaba Szepesvári. Exploration-exploitation tradeoff using variance estimates in multi-armed bandits. *Theoretical Computer Science*, 410(19):1876–1902, 2009.

Peter Auer, Nicolo Cesa-Bianchi, and Paul Fischer. Finite time analysis of the multiarmed bandit problem. *Machine Learning*, 47(2-3):235–256, 2002.

Sébastien Bubeck and Nicolò Cesa-Bianchi. Regret Analysis of Stochastic and Nonstochastic Multi-armed Bandit Problems. *Foundations and Trends in Machine Learning*, 5(1):1–122, 2012.

Jhelum Chakravorty and Aditya Mahajan. Multi-Armed Bandits, Gittins Index, and Its Calculation. In *Methods and Applications of Statistics in Clinical Trials*, pages 416–435. 2014.

Sarah Filippi, Olivier Cappe, Aurelien Garivier, and Csaba Szepesvári. Parametric Bandits: The Gener-

alized Linear Case. In *Neural Information Processing Systems (NIPS)*, 2010.

John Gittins, Kevin Glazerbrook, and Richard Weber. *Multi-Armed Bandit Allocation Indices*. John Wiley & Sons, 2nd edition, 2011.

John Gittins. Bandit processes and dynamic allocation indices. *Journal of the Royal Statistical Society. Series B*, 41(2):148–177, 1979.

Eli Gutin and Vivek F. Farias. Optimistic Gittins Indices. In *Conference on Neural Information Processing Systems (NIPS)*, 2016.

Emilie Kaufmann, Olivier Cappé, and Aurélien Garivier. On Bayesian upper confidence bounds for bandit problems. *International Conference on Artificial Intelligence and Statistics*, pages 592–600, 2012.

Emilie Kaufmann, Nathaniel Korda, and Rémi Munos. Thompson Sampling: An Asymptotically Optimal Finite Time Analysis. *Algorithmic Learning Theory*, (1):15, 2012.

Michael Jong Kim and Andrew E.B. Lim. Robust Multiarmed Bandit Problems. *Management Science*, 62(1):264–285, 2015.

Volodymyr Kuleshov and Doina Precup. Algorithms for multi-armed bandit problems. Technical report, 2014.

Tor Lattimore and Csaba Szepesvári. The End of Optimism? An Asymptotic Analysis of Finite-Armed Linear Bandits. Technical report, 2016.

Tor Lattimore. Regret Analysis of the Finite-Horizon Gittins Index Strategy for Multi-Armed Bandits. In *Annual Conference on Learning Theory (COLT)*. arXiv, 2016.

Jan Leike, Tor Lattimore, Laurent Orseau, and Marcus Hutter. Thompson Sampling is Asymptotically Optimal in General Environments. In *Uncertainty in Artificial Intelligence (UAI)*, 2016.

José Niño-Mora. Computing a classic index for finite-horizon bandits. *INFORMS Journal on Computing*, 23(2):254–267, 2011.

Marek Petrik and Shlomo Zilberstein. Robust approximate bilinear programming for value function approximation. *Journal of Machine Learning Research*, 12(1):3027–3063, 2011.

Warren Powell, Andrzej Ruszczynski, and Huseyin Topaloglu. Learning Algorithms for Separable Approximations of Discrete Stochastic Optimization Problems. *Mathematics of Operations Research*, 29(4):814–836, 2004.

Warren B Powell. Value Function Approximation using Multiple Aggregation for Multiattribute Resource Management. *Journal of Machine Learning Research*, 9:2079–2111, 2008.

Daniel Russo and Benjamin Van Roy. Learning to Optimize Via Information-Directed Sampling. pages 1–34, 2014.

Daniel Russo and Benjamin Van Roy. Learning to Optimize via Posterior Sampling. *Mathematics of Operations Research*, 39(4):1221–1243, 2014.

John Rust. Numerical dynamic programming in economics. *Handbook of computational economics*, (November), 1996.

Richard S Sutton and Andrew Barto. *Reinforcement learning*. 1998.

Kenneth E. Train. *Discrete Choice Methods with Simulation*. 2003.

P. Whittle. Restless Bandits: Activity Allocation in a Changing World. *Journal of Applied Probability*, 25(1988):287, 1988.