A PTAS for the Bayesian Thresholding Bandit Problem

Jian Peng
jianpeng@illinois.edu
UIUC

Yue Qin
qinyue@iu.edu
Indiana University

Yadi Wei
weiyadi@iu.edu
Indiana University

Yuan Zhou
yuanz@illinois.edu
UIUC

Abstract

In this paper, we study the Bayesian thresholding bandit problem (BTBP), where the goal is to adaptively make a budget of Q queries to n stochastic arms and determine the label for each arm (whether its mean reward is closer to 0 or 1). We present a polynomial-time approximation scheme for the BTBP with runtime $O(f(\epsilon) + Q)$ that achieves expected labeling accuracy at least $(\mathrm{OPT}(Q) - \epsilon)$, where $f(\cdot)$ is a function that only depends on ϵ and $\mathrm{OPT}(Q)$ is the optimal expected accuracy achieved by any algorithm. For any fixed $\epsilon > 0$, our algorithm runs in time linear with Q. The main algorithmic ideas we use include discretization employed in the PTASs for many dynamic programming algorithms (such as Knapsack), as well as many problem specific techniques such as proving an upper bound on the number of query numbers for any arm made by an almost optimal policy, and establishing the smoothness property of the $\mathrm{OPT}(\cdot)$ curve, etc.

1 Introduction

The multi-armed bandit (MAB) problem is an important model for sequential decision making. In addition to clinical trials [Thompson, 1933], it has been widely used in many real-world applications, such as advertisement placement [Agarwal et al., 2009, Chakrabarti et al., 2009], hyperparameter tuning [Li et al., 2016], crowdsourcing [Zhou et al., 2014, Chen et al., 2015], and portfolio selection [Shen et al., 2015].

The thresholding bandit problem (TBP), a specific case of MAB, considers a bandit setting where the goal is to check for each arm whether its mean is above a threshold (e.g., 0.5) with a fixed budget of Q queries allowed. The TBP has been formulated and studied in a few settings such as anomaly detection [Steinwart et al., 2005] and active learning [Tong and Koller, 2001]. A simple quantitative measure for the TBP goal is to maximize the *success probability*, which is the probability that all arms are correctly classified. In a recent work, Locatelli et al. [2016] proposed an anytime parameter-free algorithm that greedily pulls arms according to the empirical means on previously observed outcomes. Mukherjee et al. [2017] introduced a TBP algorithm that uses both mean and variance estimates to make adaptive pulling strategies. Zhong et al. [2017] studied the problem in the asynchronous distributed computing setting.

Most recently, motivated by applications in crowdsourcing, Tao et al. [2019] proposed to maximize the *expected accuracy*, which is the expected fraction of the correctly classified arms, and showed an algorithm to maximize the expected accuracy using the asymptotically optimal query budget. Indeed, in the crowdsourced binary labeling problem, the learner faces n binary classification tasks, where each task i is associated with a latent true label $z_i \in \{0, 1\}$, and a latent soft-label θ_i. The soft-label θ_i may be used to model the *labeling difficulty/ambiguity* of the task, in the sense that θ_i fraction of the crowd will label task i as 1 and the rest labels task i as 0. The crowd is also assumed to be *reliable*, i.e., $z_i = 0$ if and only if $\theta_i \geq \frac{1}{2}$. The goal of the crowdsourcing problem is to sequentially query a random worker from the large crowd about his/her label on task i for a budget of Q times, and then label the tasks with as high (expected) accuracy as possible. If we treat each of the binary classification task as a Bernoulli arm with mean reward θ_i, then this crowdsourced problem becomes expected accuracy maximization in TBP with $\theta = \frac{1}{2}$. If a few tasks are extremely ambiguous (i.e., $\theta_i \to \frac{1}{2}$), the success probability would trivially approach 0 (i.e., every algorithm would almost always fail to correctly label all tasks). In such cases, however, a good learner

Proceedings of the 23rd International Conference on Artificial Intelligence and Statistics (AISTATS) 2020, Palermo, Italy. PMLR: Volume 108.

may turn to accurately label the less ambiguous tasks and still achieve a meaningful expected accuracy.

A Bayesian version of the thresholding bandit problem (BTBP) was firstly introduced in [Chen et al., 2015], also motivated by crowdsourcing applications. In BTBP, we assume that the mean of each arm is drawn from a prior distribution, such as the Beta distribution; and the goal is to find a policy to maximize the expected fraction of n arms that are correctly classified according to a predefined threshold (or expected accuracy) with Q queries. For simplicity and without loss of generality, we assume that the threshold is 0.5 throughout this paper.

Comparing the TBP and the BTBP for expected accuracy maximization, according to a special case constructed in the discussion in [Chen et al., 2015], there is no optimal policy under the frequentist setting of TBP in general. (Note that this does not contradict with the results of [Tao et al., 2019] where the number of queries used by the algorithm is within a constant factor times the optimal amount of queries needed by any algorithm.) In contrast, in BTBP, since the mean of each arm is assumed to be drawn from a prior distribution, the optimal policy can be properly defined as the policy which results in the highest expected accuracy under the given prior.

However, computing the optimal policy for BTBP is nontrivial. In [Chen et al., 2015], a straightforward dynamic programming algorithm is introduced based on the Bellman equation. The algorithm is computationally intractable since the size of the state space in the dynamic programming is exponential in the size of the problem (please refer to Section 2.1 for details). To make the BTBP tractable for real-world applications, such as crowdsourcing, Chen et al. [2015] proposed a greedy heuristic algorithm to myopically select the next arm to maximize the one-step-ahead expected accuracy. While the algorithm runs in polynomial time, no theoretical guarantee about the optimality of the algorithm was shown in the paper.

In this paper, we present a polynomial-time approximation scheme (PTAS) algorithm for the BTBP with runtime linear in the budget Q. In particular, our main Theorem 1 shows an algorithm to achieve the optimal expected accuracy up to ϵ-additive approximation in time $(\exp(O(\epsilon^{-2}\ln^4\epsilon^{-1})) + O(Q))$. Note that besides the additive term that is exponential in $\mathrm{poly}(1/\epsilon)$, the runtime of the algorithm is linear in Q. In other words, for fixed error parameter ϵ (i.e., treating ϵ as a constant), our algorithm runs in linear time with Q.

While the trivial dynamic programming algorithm runs in time $Q^{\Theta(n)}$, we introduce three algorithmic ideas to construct the linear-time PTAS. First, we restrict the search space by only considering the policies with bounded number of queries made to any single arm. We prove that this restriction does not affect the optimal expected accuracy by much, while substantially reduces the size of the search space. The second idea is to discretize the search space by only allowing a list of geometrically increasing pre-defined numbers of queries to each individual arm. The third idea is to split the n arms into smaller partitions and obtain policies on each partition. The second and third ideas serve to further reduce the search space for an optimal policy. We also need to establish a smoothness property of the optimal curve of the expected accuracy as a function of the budget Q, which in all leads to a proof of the linear-time PTAS for the BTBP.

Our result theoretically confirms the existence of PTAS for the BTBP. However, it remains an interesting further work to further reduce the runtime dependence on the approximation parameter ϵ, to make the algorithm practically efficient. On the other hand, we will also discuss how this theoretical result might help in designing effective heuristics for the problem.

Related Works on Bayesian Multi-Armed Bandit Problems. Thompson [1933] first studied Bayesian model for maximizing the cumulative rewards and minimizing the regret, where the straightforward dynamic programming algorithm is computationally intractable. Assuming product priors, and when the goal is to minimize the geometrically discounted regret, the celebrated Gittins index theorem [Gittins, 1979] gives sufficient condition to dramatically reduce the computational complexity for the optimal strategy. For general (correlated) priors, the highly practical Thompson Sampling (TS) method [Thompson, 1933] performs well in empirical evaluations [Chapelle and Li, 2011]. Under the framework of objective Bayesian analysis, TS (and its variants) have been studied for the frequentist setting (e.g., Agrawal and Goyal [2012] and Kaufmann et al. [2012] provided tight theoretical analysis for regret minimization with fixed time horizon; Russo [2016] showed tight analysis for the best arm identification problem; Agrawal et al. [2017] studied TS for regret minimization in multinomial-logit bandits).

Organisation. In Section 2, we formally define the Bayesian thresholding bandit problem and introduce the naïve dynamic programing approach. In Section 3, we present our algorithmic ideas and theoretical analysis of the linear-time PTAS. In Section 4, we discuss how our algorithmic ideas may help design an effective heuristic algorithm and conclude the paper with a few future directions.

2 Bayesian Thresholding Bandit Problem

We have n arms numbered from $1, 2, 3, \ldots, n$. Each arm i is associated with a given prior distribution $\mathcal{D}_i = \text{Beta}(\alpha_i, \beta_i)$; and its mean θ_i is independently sampled from $\mathcal{D}_i$. Then, for each arm $i \in [n]$, there is a Bernoulli distribution $\mathcal{B}_{\theta_i}$ with probability mass θ on outcome 1 and $(1-\theta)$ mass on outcome 0. A policy is allowed to adaptively make Q sequential queries to the arms in total. If arm i_t is selected for the t-th query, a sample from $\mathcal{B}_{\theta_{i_t}}$ is made and the outcome is revealed to the policy. After all Q queries, the policy will have to make a $\{0,1\}$ guess for each arm based on the revealed outcomes. For each arm i, if $\theta_i > 0.5$, the policy is expected to guess 1; otherwise the expected guess is 0. The goal for an optimal policy is to maximize the expected accuracy (i.e., the fraction of correct guesses out of n arms), where the expectation is taken over the query samples and the Baysian prior $\mathcal{D}$.

2.1 The Straightforward Dynamic Programming Approach

We use a pair (a_i, b_i) to record the query history for arm i, meaning that we have observed a_i 1's and b_i 0's from the past $(a_i + b_i)$ queries on arm i. We use a set $S = \{(a_1, b_1), (a_2, b_2), \ldots, (a_n, b_n)\}$ to record the query history for all arms. Naturally, we use S to be the state of our dynamic programming and let $f(S)$ be the optimal expected number of correct answers we can make with query history S. When S is not a terminal state (i.e., there are less than Q observations in S), we have the following Bellman equation.

$$f(S) = \max_{1 \le i \le n} \{\mathbb{E}[\theta_i|S] f(S^{(a_i \leftarrow a_i+1)}) + (1 - \mathbb{E}[\theta_i|S]) f(S^{(b_i \leftarrow b_i+1)})\}, \quad (1)$$

where $S^{(a_i \leftarrow a_i+1)}$ represents the new query history with one more observation 1 for arm i, and similarly $S^{(b_i \leftarrow b_i+1)}$ represents the new query history with one more observation 0 for arm i. We also note that $\theta_i|S$ follows $\text{Beta}(a_i + \alpha_i, b_i + \beta_i)$ distribution and we can also write $\mathbb{E}[\theta_i|S]$ as $\mathbb{E}\,\text{Beta}(a_i + \alpha_i, b_i + \beta_i)$.

When S is a terminal state, we have to decide our guess for every arm. It is easy to check that the optimal solution is are the majority votes – guess 1 for arm i when $a_i > b_i$ and 0 otherwise. For arm i, we denote the probability of a wrong guess by

$$\text{err}(a_i, b_i) = \min\{\Pr[\text{Beta}(a_i + \alpha_i, b_i + \beta_i) < .5], \Pr[\text{Beta}(a_i + \alpha_i, b_i + \beta_i) > .5]\}. \quad (2)$$

Therefore, we have

$$f(S) = \frac{1}{n} \sum_{i=1}^{n} (1 - \text{err}(a_i, b_i)). \quad (3)$$

There are $Q^{\Theta(n)}$ states in this dynamic programming formulation. Therefore, the straightforward implementation of the algorithm has time complexity $Q^{\Theta(n)}$.

2.2 Notations

For each policy f, let $\mathsf{val}_Q(f)$ be the expected accuracy of f with query budget Q. We omit the subscript Q when it is clear from the context. Let $\text{OPT}(Q)$ be the value of the optimal policy with query budget Q.

3 The Linear-Time PTAS for Thresholding Bandit

In this section, we develop a PTAS for the thresholding bandit problem with runtime linear in Q. In contrast, the straightforward dynamic programming algorithm for the optimal solution takes time $Q^{\Theta(n)}$. Specifically, we prove the following theorem.

Theorem 1 *For $Q \ge n$, and each $\epsilon > 0$ such that $Q = \Omega(\epsilon^{-4} \ln^3 \epsilon^{-1})$, there is an algorithm that runs in time $(\exp(O(\epsilon^{-2} \ln^4 \epsilon^{-1})) + O(Q))$ to find a policy F with query budget Q such that $\mathsf{val}_Q(F) \ge \text{OPT}(Q) - \epsilon$.*

We highlight a few technical ideas used in our linear-time PTAS as follows. Note that discretization idea is used in many PTASs for dynamic programming-based algorithms (such as Knapsack). However, the other algorithmic ideas are very problem specific.

The first algorithmic idea is *query truncation*. We only look for the policies that do not make too many queries to any single arm. In Section 3.1, show that this restriction greatly reduces the search space (and therefore improves the time complexity), however still provides an excellent approximation to the optimal solution. Using the query truncation idea, we are able to get a PTAS for the Bayesian thresholding bandit problem. However, we need a few more ideas to bring the time complexity to be linear in Q.

In Section 3.2, we prove the *smoothness* property of the optimum value function $\text{OPT}(Q)$, which will be useful in the later algorithm analysis.

In Section 3.3, we use the *discretization* idea to further restrict our search space for approximate optimal policies. More specifically, we only look for policies the numbers of queries to individual arms made by which are a few pre-defined discretized values. Together with

the smoothness property, the discretization idea leads to an improvement of the time complexity of the PTAS derived in Section 3.1.

The final algorithmic idea is *partitioning*. In Section 3.4, we show that if we divide the arms to a few equal-sized partitions, and make a (roughly) equal allocation of query budget to each partition beforehand, it is still possible to well approximate the optimal policy. This idea enables us to decompose the relatively large-scale problem with n arms to a few smaller problems, and solve each of the smaller problems with less time complexity.

In Section 3.5, we integrate all technical results discussed above, present a linear-time PTAS for the Bayesian thresholding bandit problem, and prove our main Theorem 1.

For simplicity of exposition, we present our proofs with the assumption that $\alpha_i \equiv \beta_i \equiv 1$ (i.e., the prior distributions $\mathcal{D}_i$ are uniform on $[0,1]$). However, one can generalize the proofs to general Beta priors.

3.1 Bounded Query Policy

In this subsection, we introduce the idea of *query truncation*. As stated before, we will only look for the policies that do not make too many queries to any single arm. We first formally define this restriction by bounded query policies as follows.

Definition 1 *We say a policy f is an* M-Bounded Query Policy (M-BQP) *if it satisfies the following two constraints.*

1. *For each arm i, f never queries it for more than $100M\ln^2 M$ times,*
2. *Suppose the policy has queried a arm (namely i) for a_i+b_i times and seen a_i 1's at some point, where a_i and b_i satisfy $|a_i-b_i| > \sqrt{a_i+b_i}\cdot(3\ln M)$, f never queries in future.*

Definition 1 is made in a way such that the M-BQPs are able to approximate the optimal policy up to $O(1/\sqrt{M})$ additive error (by Lemma 2, which is to be introduced soon). The intuition about the restrictions made in Definition 1 is as follows. First, with very high probability, using at most $O(M\ln^2 M)$ queries to an arm suffices to make a correct guess for the arm with probability $1-O(1/\sqrt{M})$, which corresponds to the first item in the definition. Second, during the query process, if the number of 1's and 0's are very unbalanced, the player also has a very high confidence to make a correct guess for the arm, and does not wish to use more query budget, which corresponds to the second item in the definition.

The intuition mentioned above is substantiated by the following lemma, which is also main lemma of this subsection. The lemma shows that bounded query policies are able to well approximate the global optimal policy.

Lemma 2 *For any policy f and any sufficiently large M, there exists an M-BQP $\tilde{f}$ such that $\mathsf{val}_Q(\tilde{f}) \geq \mathsf{val}_Q(f) - \frac{1}{\sqrt{M}}$.*

On the other hand, we are able to construct a dynamic programming algorithm to compute the optimal bounded query policies.

Lemma 3 *The natural dynamic programming algorithm computes the optimal M-BQP in time $n^{O(M^{1.5}\ln^4 M)}$.*

Proof. Observe that for each arm, there are at most $O(M^{1.5}\ln^4 M)$ possible query records. (This is because for each arm, the only information matters is that the number of queries made to the arm (namely X), and the number of 1's observed from these queries (namely Y). By Definition 1, the number of possible values for X is $O(M\ln^2 M)$, and for each X, the number of possible values for Y is $O(\sqrt{X}\ln M) \leq O(\sqrt{M}\ln^2 M)$. Therefore, the total number of possible values for (X,Y) pair is $O(M^{1.5}\ln^4 M)$.) Due to the symmetry among all arms, two arm history set $\tilde{S}$ and $\tilde{S}'$ are essentially the same if they correspond to the same multi-set. Therefore there are essentially $n^{O(M^{1.5}\ln^4 M)}$ different states for the natural dynamic programming algorithm, and the run time is also $n^{O(M^{1.5}\ln^4 M)}$. □

By setting $M=\epsilon^{-2}$, we get the following corollary of Lemma 2 and Lemma 3, which gives a PTAS for the Bayesian thresholding bandit problem. While the time complexity is not linear in Q, it serves as the first step towards our main theorem.

Corollary 4 *Given $\epsilon \geq 0$ and query budget Q, there is a dynamic programming algorithm that computes a policy with value at least $(\mathrm{OPT}(Q)-\epsilon)$ in time $n^{O(\epsilon^{-3}\ln^4\epsilon^{-1})}$.*

The rest of this subsection is devoted to the proof of Lemma 2.

For any policy f, we construct $\tilde{f}$ as follows. The policy $\tilde{f}$ keeps a query history set $\tilde{S} = \{(a_1,b_1),(a_2,b_2),\ldots,(a_n,b_n)\}$, a set C of *corrupted* arms, where each arm $i\in C$ is associated with a pair $(\tilde{a}_i,\tilde{b}_i)$ and a mean $\tilde{\theta}_i$. Initially we have $a_i=b_i=0$ for all arms $i\in[n]$ for $\tilde{S}$, and $C=\emptyset$.

At each time, $\tilde{f}$ works as follows to decide which arm to query. Suppose $f(\tilde{S})$ chooses to query arm i, if arm

i is not corrupted (i.e., $i \notin C$) and querying it would not violate either of the two constraints in M-BQP, $\tilde{f}$ queries arm i and update $\tilde{S}$ according to the observed bit. Otherwise,

1. If i is not corrupted, we add i to C (i.e., it is now corrupted), set its associated $(\tilde{a}_i, \tilde{b}_i) = (a_i, b_i)$, and draw $\tilde{\theta}_i \sim \mathrm{Beta}(a_i + 1, b_i + 1)$.
2. We draw a bit $\tilde{r} \sim \mathcal{B}_{\tilde{\theta}}$, let $\tilde{r}$ be the pretended observation from arm i, and update $\tilde{S}$ using $\tilde{r}$.

$\tilde{f}$ terminates the query process when $f(\tilde{S})$ becomes a terminal state. For each arm i, if $i \notin C$, it decides based on (a_i, b_i) (i.e., decide 1 when $a_i > b_i$ and 0 otherwise); if $i \in C$, it decides based on $(\tilde{a}_i, \tilde{b}_i)$ using the same majority-vote rule.

It is easy to see that $\tilde{f}$ is an M-BQP by its construction. We only need to prove that $\mathsf{val}(\tilde{f}) \geq \mathsf{val}(f) - \frac{1}{\sqrt{M}}$. For notational convenience, we let S_t be the state after the t-th query made by policy f, and let $\tilde{S}_t$ be the state after the t-th query made by policy $\tilde{f}$.

We prove the following technical lemma in Appendix B.

Lemma 5 *For every state S and S', and every time t, we have* $\Pr[S_{t+1} = S' | S_t = S] = \Pr[\tilde{S}_{t+1} = S' | \tilde{S}_t = S]$.

Corollary 6 *For every query history S and time t, we have* $\Pr[S_t = S] = \Pr[\tilde{S}_t = S]$.

After a run of the policy $\tilde{f}$, for each arm i, we say it is *corrupted* if $i \in C$ at the end of the run. When arm i is corrupted, we further say it is *marked* if it was put in C because of the second constraint in Definition 1. Let K be the set of all marked arms.

The following two technical lemmas are proved in Appendix B.

Lemma 7 *Conditioned on arm i marked, $\tilde{f}$ makes the correct decision for arm i is at least $1 - \frac{1}{2\sqrt{M}}$ (when M is sufficiently large).*

Lemma 8 *The probability of arm i being corrupted but not marked is at most $\frac{1}{2\sqrt{M}}$.*

Now we are ready to prove Lemma 2.

Proof of Lemma 2. We consider a run of $\tilde{f}$ and let $\tilde{S} = \{(a_1, b_1), \ldots, (a_n, b_n)\}$ be the terminal state reached by $\tilde{f}$. For each arm i, we let $(\tilde{a}_i, \tilde{b}_i) = (a_i, b_i)$ if i is not corrupted. Let C and K be the corrupted set and marked set when $\tilde{f}$ reaches the terminal state $\tilde{S}$ (note that $K \subseteq C$). We use $\tilde{\mathbb{E}}$ to be the expectation operator over the distribution defined by this process.

We will also consider a run of f and let $S = \{(a_1, b_1), \ldots, (a_n, b_n)\}$ be the terminate state reached by f. We use the $\mathbb{E}$ to denote the expectation over the this distribution.

We have

$$\mathsf{val}(\tilde{f}) = \tilde{\mathbb{E}}\left[\frac{1}{n}\sum_{i=1}^{n}(1 - \mathrm{err}(\tilde{a}_i, \tilde{b}_i))\right] = \frac{1}{n}\sum_{i=1}^{n}\tilde{\mathbb{E}}[1 - \mathrm{err}(\tilde{a}_i, \tilde{b}_i)]. \quad (4)$$

Then we compute $\tilde{\mathbb{E}}[1 - \mathrm{err}(\tilde{a}_i, \tilde{b}_i)]$. Note that C, $C \setminus K$ and $[n] \setminus C$ partitions the arm set $[n]$, we have

$$\begin{aligned}
&\tilde{\mathbb{E}}[1 - \mathrm{err}(\tilde{a}_i, \tilde{b}_i)] \\
&= \tilde{\mathbb{E}}[1 - \mathrm{err}(\tilde{a}_i, \tilde{b}_i) | i \in K] \Pr[i \in K] \\
&\quad + \tilde{\mathbb{E}}[1 - \mathrm{err}(\tilde{a}_i, \tilde{b}_i) | i \in C \setminus K] \Pr[i \in C \setminus K] \\
&\quad + \tilde{\mathbb{E}}[1 - \mathrm{err}(\tilde{a}_i, \tilde{b}_i) | i \notin C] \Pr[i \notin C] \\
&\geq \tilde{\mathbb{E}}[1 - \mathrm{err}(\tilde{a}_i, \tilde{b}_i) | i \in K] \Pr[i \in K] \\
&\quad + \tilde{\mathbb{E}}[1 - \mathrm{err}(\tilde{a}_i, \tilde{b}_i) | i \notin C] \Pr[i \notin C] \\
&\geq \left(1 - \frac{1}{2\sqrt{M}}\right) \Pr[i \in K] \\
&\quad + \tilde{\mathbb{E}}[1 - \mathrm{err}(\tilde{a}_i, \tilde{b}_i) | i \notin C] \Pr[i \notin C] \\
&\geq \left(1 - \frac{1}{2\sqrt{M}}\right) \Pr[i \in K] \\
&\quad + \tilde{\mathbb{E}}[1 - \mathrm{err}(\tilde{a}_i, \tilde{b}_i) | i \notin C] \Pr[i \notin C] \\
&\quad + \Pr[i \in C \setminus K] - \frac{1}{2\sqrt{M}}, \qquad (5)
\end{aligned}$$

where the second inequality holds because of Lemma 7, and the last inequality holds because of Lemma 8. Since $\mathbb{E}[1 - \mathrm{err}(a_i, b_i)] \leq 1$ and $\Pr[i \in K] \leq 1$, we have

$$\begin{aligned}
(5) &\geq \tilde{\mathbb{E}}[1 - \mathrm{err}(a_i, b_i)](\Pr[i \in K] + \Pr[i \in C \setminus K] \\
&\quad + \Pr[i \notin C]) - \frac{1}{2\sqrt{M}} \Pr[i \in K] - \frac{1}{2\sqrt{M}} \\
&\geq \tilde{\mathbb{E}}[1 - \mathrm{err}(a_i, b_i)] - \frac{1}{\sqrt{M}}. \qquad (6)
\end{aligned}$$

According to Corollary 6, f and $\tilde{f}$ have the same distribution on each terminal state, and therefore we have

$$\tilde{\mathbb{E}}[\mathrm{err}(a_i, b_i)] = \mathbb{E}[\mathrm{err}(a_i, b_i)]. \quad (7)$$

Combining (4), (6), and (7), we have, $\mathsf{val}(\tilde{f}) \geq \frac{1}{n}\sum_{i=1}^{n}\mathbb{E}[1 - \mathrm{err}(a_i, b_i)] - \frac{1}{\sqrt{M}} = \mathsf{val}(f) - \frac{1}{\sqrt{M}}$. ∎

3.2 Smoothness of the $\mathrm{OPT}(\cdot)$ Curve

In this subsection, we show the smoothness property of the $\mathrm{OPT}(Q)$ function, which will be useful to the further improvement of our algorithms. In particular, we prove the following statement.

Lemma 9 *For each $\epsilon \geq 0$, when $Q \geq 1200\epsilon^{-4}\ln^3\epsilon^{-1}$, we have $\mathrm{OPT}((1-\epsilon)Q) \geq \mathrm{OPT}(Q) - 4\epsilon$.*

At a high level, the proof of Lemma 9 first constructs an ϵ^{-2}-BQP $\tilde{f}$ with query budget Q and $\mathsf{val}(\tilde{f}) \geq \mathrm{OPT}(Q) - \epsilon$. This is made possible thanks to 2. Then the proof constructs another policy g by simulating $\tilde{f}$ and skipping the queries to a random $\Omega(\epsilon)$-fraction of the arms, so that the query complexity of g is at most $(1-\epsilon)Q$. Finally, we show that $\mathsf{val}(g) \geq \mathrm{OPT}(Q) - 4\epsilon$ to complete the proof. Due to space constraints, the full proof is deferred to Appendix C.

3.3 Buffered and Bounded Query Policy

In this subsection, we present our *discretization* idea. We first carefully select a set of discretized integer values and introduce a class of more restricted policies (namely Buffered and Bounded Query Policies) that only make the number of queries equal to one of the discretized values to each arm. Building on our results on bounded query policies in Section 3.1, we show that such class of policies also well approximates the global optimal policy, and leads to a PTAS with improved time complexity. All omitted proofs in this subsection can be found in Appendix D.

Given $\gamma > 0$, we define an integer series $\tau_0 = 0, \tau_1 = 1$, and $\tau_i = \lceil (1+\gamma)\tau_{i-1} \rceil$ for all $i = 2, 3, 4, \ldots$. Now we define the class of Buffered and Bounded Query Policies as follows.

Definition 2 *We say a policy f is a (γ, M)-*Buffered and Bounded Query Policy *((γ, M)-BBQP) if*

1. *Whenever f decides to query a arm (namely arm i), if the arm has been queried for τ_j times, the policy f makes a sequence of $\tau_{j+1} - \tau_j$ queries on the arm. In other words, at any stage of the process, the number of queries made to any arm will be a number form the τ_j series.*
2. *f also satisfies the two constraints in Definition 1 for an M-BQP.*

Lemma 10 *For any M-BQP $\tilde{f}$ with query budget Q, and any $\gamma > 0$, there exists a (γ, M)-BBQP $\tilde{g}$ with query budget $(1+\gamma)Q$ such that $\mathsf{val}_Q(\tilde{g}) \geq \mathsf{val}_Q(\tilde{f})$.*

While the proof of Lemma 10 is deferred to Appendix D, the idea is to let $\tilde{g}$ create "buffers" and simulate the M-BQP policy $\tilde{f}$. More specifically, $\tilde{g}$ use a buffer for each arm to store the outcomes of the sequence of queries, and feed an element that is dequeued from the buffer whenever $\tilde{f}$ asks to make a query to the arm. $\tilde{g}$ makes a new sequence of queries whenever the buffer is empty.

The following lemma shows that the optimal BBQP can be found by a dynamic programming algorithm.

Lemma 11 *The natural dynamic programming algorithm computes the optimal (γ, M)-BBQP with query budget Q in time $n^{O(\frac{\sqrt{M}\ln^3 M}{\gamma})}$.*

Proof. Observe that for each arm, there are at most $O(\log_{1+\gamma}(M\ln^2 M)) \cdot \sqrt{M}\ln^2 M = O(\frac{\sqrt{M}\ln^3 M}{\gamma})$ possible query records. Similarly to the proof of Lemma 3, we conclude that the natural dynamic programming algorithm for (γ, M)-BP runs in time $n^{O(\frac{\sqrt{M}\ln^3 M}{\gamma})}$. □

Now we have a faster algorithm to approximate the optimal policy.

Lemma 12 *Given $\epsilon > 0$ and query budget Q such that $Q = \Omega(\epsilon^{-4}\ln^3\epsilon^{-1})$, there is a dynamic programming algorithm that computes a policy with value at least $(\mathrm{OPT}(Q) - \epsilon)$ in time $n^{O(\epsilon^{-2}\ln^3\epsilon^{-1})}$.*

Proof. Set $\gamma = \frac{\epsilon}{5}$ and $M = \frac{25}{\epsilon^2}$. We use Lemma 11 to compute an optimal (γ, M)-BBQP with query budget Q, namely $\tilde{g}$ in time $n^{O(\frac{\sqrt{M}\ln^3 M}{\gamma})} = n^{O(\epsilon^{-2}\ln^3\epsilon^{-1})}$. Let $\tilde{f}$ be the optimal M-BQP with query budget $\frac{Q}{1+\gamma}$. By Lemma 10, we know that $\mathsf{val}(\tilde{g}) \geq \mathsf{val}(\tilde{f})$. By Lemma 2, we know that

$$\mathsf{val}(\tilde{f}) \geq \mathrm{OPT}\left(\frac{Q}{1+\gamma}\right) - \frac{\epsilon}{5}.$$

By Lemma 9, we know that

$$\begin{aligned}&\mathrm{OPT}\left(\frac{Q}{1+\gamma}\right)\\ &\quad\geq \mathrm{OPT}(Q) - 4\left(1 - \frac{1}{1+\gamma}\right) \geq \mathrm{OPT}(Q) - \frac{4\epsilon}{5}.\end{aligned}$$

In total, we have

$$\mathsf{val}(\tilde{g}) \geq \mathrm{OPT}(Q) - \frac{4\epsilon}{5} - \frac{\epsilon}{5} = \mathrm{OPT}(Q) - \epsilon.$$

□

3.4 Partitioned Policy

In this subsection, we introduce the *partitioning* idea. We formally define the class of Partitioned Policies

and show that these policies may well approximate the global optimal policy.

Given an integer P, we say $\mathcal{A} = \{A_1, A_2, \ldots, A_{\lceil n/P \rceil}\}$ is an *P-partition* of the arm set $[n]$ if every arm appears in exactly one group A_i, and $P - 1 \le |A_i| \le P$ for every $A_i \in \mathcal{A}$. Now we define the class of Partitioned Policies as follows.

Definition 3 *We say a policy h is a (γ, P)-*Partitioned Policy *(P-PP) with query budget Q, if h first chooses a uniform random P-partition $\mathcal{A} = \{A_1, A_2, \ldots, A_{\lfloor n/P \rfloor}\}$, and makes sure that the total number of queries made to the arms in every $A_i \in \mathcal{A}$ is at most $\frac{(1+\gamma)Q}{n} \cdot P$.*

The following lemma shows that partitioned policies are powerful enough to approximate the optimal solution.

Lemma 13 *For any M, γ, and P such that $P \ge 300\gamma^{-2}(\ln \gamma^{-1})M \ln^2 M$, given an M-BQP $\tilde{f}$ with query budget $Q \ge n$, there exists a (γ, P)-PP h with query budget Q, such that $\mathsf{val}_Q(h) \ge \mathsf{val}_Q(\tilde{f}) - \gamma$.*

If we set $M = \epsilon^{-2}$ and $\gamma = \epsilon$ in Lemma 13, combining Lemma 2 and Lemma 13, we have

Corollary 14 *When $Q \ge n$, for each $\epsilon > 0$, let $P = 1200\epsilon^{-4} \ln^3 \epsilon^{-1}$, there exists a (ϵ, P)-PP h with query budget Q, such that $\mathsf{val}_Q(h) \ge \mathrm{OPT}(Q) - 2\epsilon$.*

The rest of this subsection is devoted to the proof of Lemma 13.

We first define a policy h similarly as we did in Section 3.1. For any M-BQP $\tilde{f}$, we construct h as follows. The policy g keeps a query history set $\tilde{S} = \{(a_1, b_1), (a_2, b_2), \ldots, (a_n, b_n)\}$, a set C of *corrupted groups*, where for each group $A \in C$ and each arm $i \in A$, arm i is associated with a pair $(\tilde{a}_i, \tilde{b}_i)$ and a mean $\tilde{\theta}_i$. Initially we have $a_i = b_i = 0$ for all arms $i \in [n]$ for $\tilde{S}$, and $C = \emptyset$.

At each time, g works as follows to decide which arm to query. Suppose $f(\tilde{S})$ chooses to query arm i, and A is the group that includes i. If A is not corrupted (i.e., A does not appear in C) and querying arm i would not go over the budget for A, g queries arm i and update $\tilde{S}$ according to the observed bit. Otherwise,

1. If A is not corrupted, we add A to C (i.e., it is now corrupted). For each arm $i' \in A$, set its associated $(\tilde{a}_{i'}, \tilde{b}_{i'}) = (a_{i'}, b_{i'})$, and draw $\tilde{\theta}_{i'} \sim \mathrm{Beta}(a_{i'} + 1, b_{i'} + 1)$.
2. We draw a bit $\tilde{r} \sim \mathcal{B}_{\tilde{\theta}_i}$, let $\tilde{r}$ be the pretended observation from arm i, and update $\tilde{S}$ using $\tilde{r}$.

g terminates the query process when $f(\tilde{S})$ becomes a terminal state. For each arm i that is in a non-corrupted group, we make the decision according to (a_i, b_i). For every arm i that is in a corrupted group, we guess an arbitrary bit.

Similarly as we did for Lemma 5 and Corollary 6, we prove the following statement.

Lemma 15 *For every realization of the P-partition $\mathcal{A}$, and every state S, we have that $\Pr[\tilde{f}$ reaches $S] = \Pr[h$ reaches $S|\mathcal{A}]$. As a corollary, the event that h reaches S is independent from the random variable $\mathcal{A}$.*

Now let S be the terminal state reached by h, by Lemma 15, conditioned on S, the distribution of $\mathcal{A}$ is the same as the unconditioned probability distribution. We prove the following statement.

Lemma 16 *For each group A_i, we have*

$$\Pr[A_i \in C \text{ when } h \text{ terminates}|S] \le \gamma.$$

Proof. Let $q_j = a_j + b_j$ be the number of queries (including the pretended ones) made to arm j when h terminates. Since $\tilde{f}$ is an M-BQP, we have $q_i \le 100M \ln^2 M$. Let $X_j = 1$ when $j \in A_i$ and $X_j = 0$ otherwise. We have that $\mathbb{E}[X_j|S] = \frac{|A_i|}{n} \le \frac{P}{n}$. Also, by Definition 2.1 and Theorem 2.7 in [Joag-Dev and Proschan, 1983], $\{X_1, X_2, \ldots, X_n\}$ (conditioned on S) are a set of negatively associated random variables. Let $Y_j = \frac{X_j q_j}{100M \ln^2 M} \in [0, 1]$. By Property P6 in [Joag-Dev and Proschan, 1983], $\{Y_1, Y_2, \ldots, Y_n\}$ (conditioned on S) are also negatively associated therefore Chernoff Bound also applies. Now note that $\mathbb{E}[\sum_{j=1}^n Y_j|S] \le \frac{QP}{n \cdot 100M \ln^2 M}$. Using Theorem 18, $\Pr[A_i \in C$ when h terminates$|S]$ equals to

$$\Pr\left[\sum_{j=1}^n Y_j \ge \frac{(1+\gamma)QP}{n \cdot 100M \ln^2 M}|S\right],$$

which is at most $\exp\left(-\frac{\gamma^2 QP}{300nM \ln^2 M}\right) \le \gamma$. □

We are now ready to prove Lemma 13.

Proof of Lemma 13. It is straightforward to verify that h is a P-PP with query budget Q. Now let us bound its value. Let $S = \{(a_1, b_1), (a_2, b_2), \ldots, (a_n, b_n)\}$ be the terminal state reached by h. Let Z_j be 1 if arm j is in a group that is corrupted when h terminates, and 0 otherwise. We have

$$\mathsf{val}(h) = \mathop{\mathbb{E}}_{S}\left[\frac{1}{n}\sum_{j=1}^n (1 - \mathrm{err}(a_j, b_j))(1 - Z_j)|S\right]$$

$$\geq \mathop{\mathbb{E}}_{S} \frac{1}{n}\sum_{j=1}^{n}(1-\mathrm{err}(a_j,b_j)) - \frac{1}{n}\sum_{j=1}^{n}\mathop{\mathbb{E}}_{S}[Z_j|S]. \quad (8)$$

By Lemma 15, we have

$$\begin{aligned}&\mathop{\mathbb{E}}_{S} \frac{1}{n}\sum_{j=1}^{n}(1-\mathrm{err}(a_j,b_j))\\ &= \sum_{S} \frac{1}{n}\sum_{j=1}^{n}(1-\mathrm{err}(a_j,b_j))\Pr[\tilde{f} \text{ reaches } S] = \mathsf{val}(\tilde{f}).\end{aligned} \quad (9)$$

On the other hand, by Lemma 16, we have

$$\begin{aligned}&\frac{1}{n}\sum_{j=1}^{n}\mathop{\mathbb{E}}_{S}[Z_j|S]\\ &= \frac{1}{n}\sum_{A_i\in\mathcal{A}}\Pr_{S}[A_i \in C \text{ when } h \text{ terminates}|S]|A_i| \leq \gamma.\end{aligned} \quad (10)$$

Putting (8), (9), and (10) together, we prove that $\mathsf{val}(h) \geq \mathsf{val}(\tilde{f}) - \gamma$. □

3.5 The Linear-Time PTAS

Now we are ready to prove our main theorem on the linear-time PTAS.

Proof of Theorem 1. Let $P = 1200(\epsilon/8)^{-4}\ln^3(\epsilon/8)^{-1}$. For simplicity of exposition, we assume n is divisible by P (while the general case can be deal with by introducing the negligible $O(1/P)$ additive error in the approximation guarantee).

Our algorithm first generates a uniform random P-partition $\mathcal{A} = \{A_1, A_2, \ldots, A_{\lceil n/P\rceil}\}$ of the arm set $[n]$. We use Lemma 11 to find an optimal policy f on P arms (with uniform prior) with query budget QP/n, up to $(\epsilon/4)$-additive approximation error. This can be done in time $\exp(O(\epsilon^{-2}\ln^4\epsilon^{-1}))$. Then as the meta-policy F, we run f on each group A_i of arms, and decide as f decides. It is straightforward to see that F uses at most Q queries. It remains to show the lower bound on $\mathsf{val}(F)$, which is $\mathsf{val}(f)$ (the average value of $\mathsf{val}(f)$ on all groups).

By Corollary 14, there exists an $(\epsilon/8, P)$-PP H with query budget $\frac{Q}{1+\epsilon/8}$, such that $\mathsf{val}(H) \geq \mathrm{OPT}(\frac{Q}{1+\epsilon/8}) - \epsilon/4$. Now for each group A_i, consider the value of f and the number of correct guesses made by H on the arms in A_i. Since H makes at most $\frac{(1+\epsilon/8)QP}{n}$ queries on arms in A_i, let $\mathrm{OPT}_P(Q)$ denote the value of the optimal policy on P arms with query budget Q, we have $\frac{1}{P}\sum_{j\in A_i}\Pr[H \text{ guesses correctly on arm } j|A_i] \leq \mathrm{OPT}_P(Q) \leq \mathsf{val}(f) + \frac{\epsilon}{4}$. Therefore, we have that $\mathsf{val}(F) = \mathsf{val}(f) \geq \frac{1}{P}\sum_{j\in A_i}\Pr[H \text{ guesses correctly on arm } j|A_i] - \frac{\epsilon}{4}$. On the other hand, since

$$\begin{aligned}&\mathsf{val}(H)\\ &= \frac{1}{|\mathcal{A}|}\sum_{A_i\in\mathcal{A}}\frac{\sum_{j\in A_i}\Pr[H \text{ guesses correctly on arm } j|A_i]}{P}\\ &= \sum_{j\in A_i}\Pr[H \text{ guesses correctly on arm } j|A_i],\end{aligned}$$

we have $\mathsf{val}(F) \geq \mathsf{val}(H) - \frac{\epsilon}{4} \geq \mathrm{OPT}\left(\frac{Q}{1+\epsilon/8}\right) - \frac{\epsilon}{4} - \frac{\epsilon}{4} \geq \mathrm{OPT}\,(Q) - \epsilon$, where the last inequality is by Lemma 9. □

4 Conclusion and Future Work

In this paper, we introduced a PTAS algorithm for the Bayesian thresholding bandit problem with linear runtime complexity in a given bugdet Q. There are a few key ideas of our algorithm. First, we use the query truncation idea to focus on the policies without too many queires to any specific single arm. This idea resulted in a first PTAS for the BTBP problem. Then we use the discretization idea to reduce the search space of approximate optimal policies. With the smoothness property, this discretization idea improves the time complexity of the PTAS. Finally, we devise a partitioning idea to decompose the problem into a few smaller problems with each solved with less complexity ourselves, with a reasonable approximation to the optimal policy.

In the future, we would like to explore empirical implications of our results. In addition to the theoretical guarantees, we would like to empirically test the feasibility of these ideas on simulated BTBP problems. To further reduce the state space complexity of our dynamic programming algorithm, we will apply an approximate dynamic programming motivated by the value iteration methods in reinforcement learning. Motivated by the recent success of deep reinforcement learning, we will check if entries in the dynamic programming table can be well approximated by a neural network function that maps the current query history to the highest expected accuracy, and the weights of the neural network model are updated using the soft Q-learning algorithm with the restriction in the first idea. In this way, we can readily apply the second and the third ideas to this framework and test their effectiveness in reducing the practical runtime of the approximate neural dynamic programming without substantial loss of accuracy.

References

Deepak Agarwal, Bee-Chung Chen, and Pradheep Elango. Explore/exploit schemes for web content optimization. In *Data Mining, 2009. ICDM'09. Ninth IEEE International Conference on*, pages 1–10. IEEE, 2009. 1

Shipra Agrawal and Navin Goyal. Analysis of Thompson sampling for the multi-armed bandit problem. In *Conference on Learning Theory*, pages 39–1, 2012. 2

Shipra Agrawal, Vashist Avadhanula, Vineet Goyal, and Assaf Zeevi. Thompson sampling for the mnl-bandit. In *Conference on Learning Theory*, pages 76–78, 2017. 2

Deepayan Chakrabarti, Ravi Kumar, Filip Radlinski, and Eli Upfal. Mortal multi-armed bandits. In *Advances in neural information processing systems*, pages 273–280, 2009. 1

Olivier Chapelle and Lihong Li. An empirical evaluation of thompson sampling. In *Advances in neural information processing systems*, pages 2249–2257, 2011. 2

Xi Chen, Qihang Lin, and Dengyong Zhou. Statistical decision making for optimal budget allocation in crowd labeling. *Journal of Machine Learning Research*, 16:1–46, 2015. 1, 2

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

Kumar Joag-Dev and Frank Proschan. Negative association of random variables with applications. *The Annals of Statistics*, pages 286–295, 1983. 7

Emilie Kaufmann, Nathaniel Korda, and Rémi Munos. Thompson sampling: An asymptotically optimal finite-time analysis. In *International Conference on Algorithmic Learning Theory*, pages 199–213. Springer, 2012. 2

Lisha Li, Kevin Jamieson, Giulia DeSalvo, Afshin Rostamizadeh, and Ameet Talwalkar. Hyperband: A novel bandit-based approach to hyperparameter optimization. *arXiv preprint arXiv:1603.06560*, 2016. 1

Andrea Locatelli, Maurilio Gutzeit, and Alexandra Carpentier. An optimal algorithm for the thresholding bandit problem. In *International Conference on Machine Learning*, pages 1690–1698, 2016. 1

Subhojyoti Mukherjee, Naveen Kolar Purushothama, Nandan Sudarsanam, and Balaraman Ravindran. Thresholding bandits with augmented UCB. In *Proceedings of the 26th International Joint Conference on Artificial Intelligence*, pages 2515–2521. AAAI Press, 2017. 1

Daniel Russo. Simple bayesian algorithms for best arm identification. In *Conference on Learning Theory*, pages 1417–1418, 2016. 2

Weiwei Shen, Jun Wang, Yu-Gang Jiang, and Hongyuan Zha. Portfolio choices with orthogonal bandit learning. In *Twenty-Fourth International Joint Conference on Artificial Intelligence*, 2015. 1

Ingo Steinwart, Don Hush, and Clint Scovel. A classification framework for anomaly detection. *Journal of Machine Learning Research*, 6(Feb):211–232, 2005. 1

Chao Tao, Saùl Blanco, Jian Peng, and Yuan Zhou. Thresholding bandit with optimal aggregate regret. *arXiv preprint arXiv:1905.11046*, 2019. To appear in *NeurIPS 2019*. 1, 2

W.R. Thompson. On the likelihood that one unknown probability exceeds another in view of the evidence of two samples. *Biometrika*, 25:285–294, 1933. 1, 2

Simon Tong and Daphne Koller. Support vector machine active learning with applications to text classification. *Journal of Machine Learning Research*, 2 (Nov):45–66, 2001. 1

Jie Zhong, Yijun Huang, and Ji Liu. Asynchronous parallel empirical variance guided algorithms for the thresholding bandit problem. *arXiv preprint arXiv:1704.04567*, 2017. 1

Yuan Zhou, Xi Chen, and Jian Li. Optimal PAC multiple arm identification with applications to crowdsourcing. In *Proceedings of International Conference on Machine Learning (ICML)*, 2014. 1