# A GENERIC SOLUTION TO *MULTI*-ARMED BERNOULLI BANDIT PROBLEMS BASED ON RANDOM SAMPLING FROM SIBLING CONJUGATE PRIORS

Thomas Norheim, Terje Brådland, Ole-Christoffer Granmo
*Department of ICT, University of Agder, Grimstad, Norway*

B. John Oommen
*School of Computer Science, Carleton University, Ottawa, Canada*

Keywords: Bandit problems, Conjugate priors, Sampling, Bayesian learning.

Abstract: The Multi-Armed Bernoulli Bandit (MABB) problem is a classical optimization problem where an agent sequentially pulls one of multiple arms attached to a gambling machine, with each pull resulting in a random reward. The reward distributions are unknown, and thus, one must balance between exploiting existing knowledge about the arms, and obtaining new information. Although poised in an abstract framework, the applications of the MABB are numerous (Gelly and Wang, 2006; Kocsis and Szepesvari, 2006; Granmo et al., 2007; Granmo and Bouhmala, 2007) . On the other hand, while Bayesian methods are generally computationally intractable, they have been shown to provide a standard for optimal decision making. This paper proposes a novel MABB solution scheme that is inherently Bayesian in nature, and which yet avoids the computational intractability by relying simply on updating the hyper-*parameters* of the sibling conjugate distributions, and on simultaneously sampling randomly from the respective posteriors. Although, in principle, our solution is generic, to be concise, we present here the strategy for Bernoulli distributed rewards. Extensive experiments demonstrate that our scheme outperforms recently proposed bandit playing algorithms. We thus believe that our methodology opens avenues for obtaining improved novel solutions.

## 1 INTRODUCTION

The conflict between exploration and exploitation is a well-known problem in Reinforcement Learning (RL), and other areas of artificial intelligence. The Multi-Armed Bernoulli Bandit (MABB) problem captures the essence of this conflict, and has thus occupied researchers for over fifty years (Wyatt, 1997). In (Granmo, 2009) a new family of *Bayesian* techniques for solving the classical Two-Armed Bernoulli Bandit (TABB) problem was introduced, and empirical results that demonstrated its advantages over established top performers were reported. In this present paper, we address the *Multi*-Armed Bernoulli Bandit (MABB). Observe that a TABB scheme can solve any MABB problem by incorporating either a parallel or serial philosophy by considering the arms in a pairwise manner. If operating in parallel, since the pairwise solutions are themselves uncorrelated, the overall MABB solution would require the solution of $\binom{r}{2}$ TABB problems (where $r$ is the number of bandit arms). Alternatively, if the solutions are invoked serially, it is easy to see that $r-1$ TABB solutions suffice, namely by each solution leading to the elimination of an inferior bandit arm - after convergence. The solution that we propose here is inherently distinct, and does not require any primitive TABB solution strategy. Rather, we propose a general scheme which considers *all* the $r$ arms in a *single* sequential "game". Thus, we believe that the paper presents a novel solution that searches for the optimal arm by evaluating arms simultaneously, and yet, with a complexity that grows *linearly* with the number of arms. We are not aware of any Bayesian sampling-based solution to the MABB, and thus add that, to the best of our knowledge, this paper is of a pioneering sort.


Norheim T., Brådland T., Granmo O. and John Oommen B. (2010).
A GENERIC SOLUTION TO MULTI-ARMED BERNOULLI BANDIT PROBLEMS BASED ON RANDOM SAMPLING FROM SIBLING CONJUGATE PRIORS.
In *Proceedings of the 2nd International Conference on Agents and Artificial Intelligence - Artificial Intelligence*, pages 36-44
DOI: 10.5220/0002712500360044

## 1.1 The Multi-Armed Bernoulli Bandit (MABB) Problem

The MABB problem is a classical optimization problem that explores the trade off between exploitation and exploration in reinforcement learning. The problem consists of an agent that sequentially pulls one of multiple arms attached to a gambling machine, with each pull resulting either in a *reward* or a *penalty*[1]. The sequence of rewards/penalties obtained from each arm $i$ forms a Bernoulli process with an *unknown* reward probability $r_i$, and a penalty probability $1 - r_i$. This leaves the agent with the following dilemma: Should the arm that so far seems to provide the highest chance of reward be pulled once more, or should the inferior arm be pulled in order to learn more about *its* reward probability? Sticking prematurely with the arm that is presently considered to be the best one, may lead to not discovering which arm is truly optimal. On the other hand, lingering with the inferior arm unnecessarily, postpones the harvest that can be obtained from the optimal arm.

With the above in mind, we intend to evaluate an agent's arm selection strategy in terms of the so-called *Regret*, and in terms of the *probability of selecting the optimal arm*[2]. The *Regret* measure is non-trivial, and in all brevity, can be perceived to be *the difference between the sum of rewards expected after N successive arm pulls, and what would have been obtained by only pulling the optimal arm*. To clarify issues, assume that a *reward* amounts to the value (utility) of unity (i.e., 1), and that a *penalty* possesses the value 0. We then observe that the expected returns for pulling Arm $i$ is $r_i$. Thus, if the optimal arm is Arm 1, the *Regret* after $N$ plays would become:

$$r_1 N - \sum_{i=1}^{N} \hat{r}_i, \qquad (1)$$

with $\hat{r}_n$ being the expected reward at Arm pull $i$, given the agent's arm-selection strategy. In other words, as will be clear in the following, we consider the case where rewards are *undiscounted*, as discussed in (Auer et al., 2002).

In the last decades, several computationally efficient algorithms for tackling the MABB Problem have emerged. From a theoretical point of view, LA are known for their ε-optimality. From the field of Bandit Playing Algorithms, *confidence interval based* algorithms are known for logarithmically growing *Regret*.

## 1.2 Applications

Solution schemes for bandit problems have formed the basis for dealing with a number of applications. For instance, a UCB-TUNED scheme (Auer et al., 2002) is used for move exploration in *MoGo*, a top-level Computer-Go program on $9 \times 9$ *Go* boards (Gelly and Wang, 2006). Furthermore, the so-called UBC1 scheme has formed the basis for guiding Monte-Carlo planning, and improving planning efficiency significantly in several domains (Kocsis and Szepesvari, 2006).

The applications of LA are many – in the interest of brevity, we list a few more-recent ones. LA have been used to allocate polling resources optimally in web monitoring, and for allocating limited sampling resources in binomial estimation problems (Granmo et al., 2007) . LA have also been applied for solving NP-complete SAT problems (Granmo and Bouhmala, 2007) .

## 1.3 Contributions and Paper Organization

The contributions of this paper can be summarized as follows. In Sect. 2 we briefly review a selection of the main MABB solution approaches, including LA and confidence interval-based schemes. Then, in Sect. 3 we present the Bayesian Learning Automaton (BLA). In contrast to the latter reviewed schemes, the BLA is inherently Bayesian in nature, even though it only relies on simple counting and random sampling. Thus, to the best of our knowledge, BLA is the first MABB algorithm that takes advantage of the Bayesian perspective in a computationally efficient manner. In Sect. 4 we provide extensive experimental results that demonstrate that, in contrast to typical LA schemes as well as some Bandit Playing Algorithms, BLA does not rely on external learning speed/accuracy control. The BLA also outperforms established top performers from the field of Bandit Playing Algorithms[3]. Accordingly, from the above perspective, it is our belief that the BLA represents the current state-of-the-art and a new avenue of research. Finally, in Sect. 5 we list open BLA-related research problems, in addition to providing concluding remarks.

[1] A *penalty* may also be perceived as the absence of a *reward*. However, we choose to use the term *penalty* as is customary in the LA and RL literature.

[2] Using *Regrets* as a performance measure is typical in the literature on Bandit Playing Algorithms, while using the "arm selection probability" is typical in the LA literature. In this paper, we will use both these concepts in the interest of comprehensiveness.

[3] A comparison of Bandit Playing Algorithms can be found in (Vermorel and Mohri, 2005), with the UCB-TUNED distinguishing itself in (Auer et al., 2002).

## 2 RELATED WORK

The MABB problem has been studied in a disparate range of research fields. From a machine learning point of view, Sutton et. al placed an emphasis on computationally efficient solution techniques that are suitable for RL. While there are algorithms for computing the optimal Bayes strategy to balance exploration and exploitation, these are computationally intractable for the general case(Sutton and Barto, 1998), mainly because of the magnitude of the state space associated with typical bandit problems.

From a broader point of view, one can distinguish two distinct fields that address bandit like problems, namely, the field of Learning Automata and the field of Bandit Playing Algorithms. A myriad of approaches have been proposed within these two fields, and we refer the reader to (Narendra and Thathachar, 1989; Thathachar and Sastry, 2004) and (Vermorel and Mohri, 2005) for an overview and comparison of schemes. Although these fields are quite related, research spanning them both is surprisingly sparse. In this paper, however, we will include the established top performers from both of the two fields. These are reviewed here in some detail in order to cast light on the distinguishing properties of BLA, both from an LA perspective and from the perspective of Bandit Playing Algorithms.

### 2.1 Learning Automata (LA) — The $L_{R-I}$ and Pursuit Schemes

LA have been used to model biological systems (Tsetlin, 1973; Narendra and Thathachar, 1989; Thathachar and Sastry, 2004) and have attracted considerable interest in the last decade because they can learn the optimal action when operating in (or interacting with) unknown stochastic environments. Furthermore, they combine rapid and accurate convergence with low computational complexity. For the sake of conceptual simplicity, note that we in this subsection, we assume that we are dealing with a bandit associated with two arms.

More notable approaches include the family of linear updating schemes, with the Linear Reward-Inaction ($L_{R-I}$) automaton being designed for stationary environments (Narendra and Thathachar, 1989). In short, the $L_{R-I}$ maintains an Arm probability selection vector $\bar{p} = [p_1, p_2]$, with $p_2 = 1 - p_1$. The question of which Arm is to be pulled is decided randomly by sampling from $\bar{p}$. Initially, $\bar{p}$ is uniform. The following linear updating rules summarize how rewards and penalties affect $\bar{p}$ with $p'_1$ and $1 - p'_1$ being the resulting updated Arm selection probabilities:

$$\begin{aligned} p'_1 &= p_1 + (1-a) \times (1-p_1) \\ &\quad \text{if pulling Arm 1 results in a reward} \\ p'_1 &= a \times p_1 \\ &\quad \text{if pulling Arm 2 results in a reward} \\ p'_1 &= p_1 \\ &\quad \text{if pulling Arm 1 or Arm 2 results in a penalty.} \end{aligned}$$

In the above, the parameter $a$ ($0 \ll a < 1$) governs the learning speed. As seen, after Arm $i$ has been pulled, the associated probability $p_i$ is increased using the linear updating rule upon receiving a reward, with $p_j (j \neq i)$ being decreased correspondingly. Note that $\bar{p}$ is left unchanged upon a penalty.

A distinguishing feature of the $L_{R-I}$ scheme, and indeed the best LA within the field of LA, is its ε-optimality(Narendra and Thathachar, 1989): *By a suitable choice of some parameter of the LA, the expected reward probability obtained from each arm pull can be made arbitrarily close to the optimal reward probability, as the number of arm pulls tends to infinity.*

A *Pursuit scheme* (P-scheme) makes the updating of $\bar{p}$ more goal-directed in the sense that it maintains Maximum Likelihood (ML) estimates $(\hat{r_1}, \hat{r_2})$ of the reward probabilities $(r_1, r_2)$ associated with each Arm. Instead of using the rewards/penalties that are received to update $\bar{p}$ directly, they are rather used to update the ML estimates. The ML estimates, in turn, are used to decide which Arm selection probability $p_i$ is to be increased. In brief, a Pursuit scheme increases the Arm selection probability $p_i$ associated with the currently largest ML estimate $\hat{r}_i$, instead of the Arm actually producing the reward. Thus, unlike the $L_{R-I}$, in which the reward from an inferior Arm can cause unsuitable probability updates, in the Pursuit scheme, these rewards will not influence the learning progress in the short term, except by modifying the estimate of the reward vector. This, of course, assumes that the ranking of the ML estimates are correct, which is what it will be if each action is chosen a "sufficiently large number of times". Accordingly, a Pursuit scheme consistently outperforms the $L_{R-I}$ in terms of its rate of convergence.

Discretized and Continuous variants of the Pursuit scheme has been proposed (Agache and Oommen, 2002) , with slightly superior performances. But, in general, any Pursuit scheme can be seen to be representative of this entire family.

## 2.2 The ε-Greedy and $\varepsilon_n$-Greedy Policies

The ε-greedy rule is a well-known strategy for the bandit problem (Sutton and Barto, 1998). In short, the Arm with the presently highest average reward is pulled with probability $1-\varepsilon$, while a randomly chosen Arm is pulled with probability ε. In other words, the balancing of exploration and exploitation is controlled by the ε-parameter. Note that the ε-greedy strategy persistently explores the available Arms with constant effort, which clearly is sub-optimal for the MABB problem (unless the reward probabilities are changing with time).

As a remedy for the above problem, ε can be slowly decreased, leading to the $\varepsilon_n$-greedy strategy described in (Auer et al., 2002). The purpose is to gradually shift focus from exploration to exploitation. The latter work focuses on algorithms that minimizes the so-called *Regret* formally described above. It turns out that the $\varepsilon_n$-greedy strategy asymptotically provides a *logarithmically* increasing *Regret*. Indeed, it has been proved that logarithmically increasing *Regret* is the best possible (Auer et al., 2002) strategy.

## 2.3 Confidence Interval Based Algorithms

A promising line of thought is the interval estimation methods, where a confidence interval for the reward probability of each Arm is estimated, and an "optimistic reward probability estimate" is identified for each Arm. The Arm with the most optimistic reward probability estimate is then greedily selected (Vermorel and Mohri, 2005; **?**).

In (Auer et al., 2002), several confidence interval based algorithms are analysed. These algorithms also provide logarithmically increasing *Regret*, with *UCB-TUNED* – a variant of the well-known UBC1 algorithm — outperforming both UBC1,UCB2, as well as the $\varepsilon_n$-greedy strategy. In brief, in UCB-TUNED, the following optimistic estimates are used for each Arm *i*:

$$\mu_i + \sqrt{\frac{\ln n}{n_i} \min\{1/4, \sigma_i^2 + \sqrt{\frac{2\ln n}{n_i}}\}} \qquad (2)$$

where $\mu_i$ and $\sigma_i^2$ are the sample mean and variance of the rewards that have been obtained from Arm *i*, *n* is the number of Arms pulled in total, and $n_i$ is the number of times Arm *i* has been pulled. Thus, the quantity added to the sample average of a specific Arm *i* is steadily reduced as the Arm is pulled, and uncertainty about the reward probability is reduced. As a result, by always selecting the Arm with the highest optimistic reward estimate, UCB-TUNED gradually shifts from exploration to exploitation.

## 2.4 Bayesian Approaches

The use of Bayesian methods in inference problems of this nature has also been reported. The authors of (Wyatt, 1997) have proposed the use of such a philosophy in their probability matching algorithms. By using conjugate priors, they have resorted to a Bayesian analysis to obtain a closed form expression for the probability that each arm is optimal given the prior observed rewards/penalties. Informally, the method proposes a policy which consists of calculating the probability of each arm being optimal before an arm pull, and then randomly selecting the arm to be pulled next using these probabilities. Unfortunately, for the case of two arms in which the rewards are Bernoulli-distributed, the computation time becomes unbounded, and it increases with the number of arm pulls. Furthermore, it turns out that for the multi-armed case, the resulting integrations have no analytical solution. Similar problems surface when the probability of each arm being optimal is computed for the case when the rewards are normally distributed[4]. The authors of (Dearden et al., 1998) take advantage of a Bayesian strategy in a related domain, i.e., in Q-learning. They show that for normally distributed rewards, in which the parameters have a prior normal-gamma distribution, the posteriors also have a normal-gamma distribution, rendering the computation efficient. They then integrate this into a framework for Bayesian Q-learning by maintaining and propagating probability distributions over the Q-values, and suggest that a non-approximate solution can be obtained by means of random sampling for the normal distribution case. It would be interesting to investigate the applicability of these results for the MABB.

## 2.5 Boltzmann Exploration and POKER

One class of algorithms for solving MABB problems is based on so-called Boltzmann exploration. In brief, an arm *i* is pulled with probability $p_k = \frac{e^{\hat{\mu}_i/\tau}}{\sum_{j=1}^{n}}$ where $\hat{\mu}_i$ is the sample mean and τ is defined as the temperature of the exploration. A high temperature τ leads

[4]It turns out that in the latter case, the approximate Bayesian solution reported by (Wyatt, 1997) is computationally efficient

to increased exploration since each arm will have approximately the same probability of being pulled. A low temperature, on the other hand, leads to arms being pulled proportionally to the size of the rewards that can be expected. Typically, the temperature is set to be high initially, and then is gradually reduced in order to shift from exploration to exploitation. Note that the EXP3 scheme, proposed and detailed in (Auer et al., 1995), is a more complicated variant of Boltzmann exploration. In brief, this scheme calculates the arm selection probabilities $p_k$ based on dividing the rewards obtained with the probability of pulling the arm that produced the rewards (Vermorel and Mohri, 2005).

The "Price of Knowledge and Estimated Reward" (POKER) algorithm proposed in (Vermorel and Mohri, 2005) attempts to combine the following three principles: (1) Reducing uncertainty about the arm reward probabilities should grant a bonus to stimulate exploration; (2) Information obtained from pulling arms should be used to estimate the properties of arms that have not yet been pulled; and (3) Knowledge about the number of rounds that remains (the horizon) should be used to plan the exploitation and exploration of arms. We refer the reader to (Vermorel and Mohri, 2005) for the specific algorithm that incorporates these three principles.

# 3 THE BAYESIAN LEARNING AUTOMATON (BLA)

Bayesian reasoning is a probabilistic approach to inference which is of significant importance in machine learning because it allows quantitative weighting of evidence supporting alternative hypotheses, with the purpose of allowing optimal decisions to be made. Furthermore, it provides a framework for analyzing learning algorithms (Mitchell, 1997).

We here present a scheme for solving the MABB problem that inherently builds upon the Bayesian reasoning framework. We coin the scheme *Bayesian Learning Automaton* (BLA) since it can be modelled as a state machine with each state associated with unique Arm selection probabilities, in an LA manner.

A unique feature of the BLA is its computational simplicity, achieved by relying *implicitly* on Bayesian reasoning principles. In essence, at the heart of BLA we find the *Beta distribution*. Its shape is determined by two positive parameters, usually denoted by $\alpha$ and $\beta$, producing the following probability density function:

$$f(x;\alpha,\beta) = \frac{x^{\alpha-1}(1-x)^{\beta-1}}{\int_0^1 u^{\alpha-1}(1-u)^{\beta-1}\,du}\ , \quad x \in [0,1] \quad (3)$$

and the corresponding cumulative distribution function:

$$F(x;\alpha,\beta) = \frac{\int_0^x t^{\alpha-1}(1-t)^{\beta-1}\,dt}{\int_0^1 u^{\alpha-1}(1-u)^{\beta-1}\,du}\ , \quad x \in [0,1]. \quad (4)$$

Essentially, the BLA uses the *Beta* distribution for two purposes. First of all, the *Beta* distribution is used to provide a *Bayesian estimate* of the reward probabilities associated with each of the available bandit Arms - the latter being valid by virtue of the Conjugate Prior (Duda et al., 2000) nature of the Binomial parameter. Secondly, a novel feature of the BLA is that it uses the *Beta* distribution as the basis for an *Order-of-Statistics*-based *randomized* Arm selection mechanism.

The following algorithm contains the essence of the BLA approach.

**Algorithm: BLA-MABB**
**Input:** Number of bandit Arms $r$.
**Initialization:** $\alpha_1^1 = \beta_1^1 = \alpha_2^1 = \beta_2^1 = \ldots = \alpha_r^1 = \beta_r^1 = 1$.
**Method:**
**For** $N = 1, 2, \ldots$ **Do**

1. For each *Arm* $j \in \{1,\ldots,r\}$, draw a value $x_j$ randomly from the associated *Beta* distribution $f(x_j;\alpha_j^N,\beta_j^N)$ with the parameters $\alpha_j^N,\beta_j^N$.
2. Pull the *Arm i* whose drawn value $x_i$ is the largest one of the randomly drawn values:
$$i = \underset{j\in\{1,\ldots,r\}}{\text{argmax}}\ x_j.$$
3. Receive either *Reward* or *Penalty* as a result of pulling *Arm i*, and update parameters as follows:
   - Upon *Reward*: $\alpha_i^{N+1} = \alpha_i^N + 1$; $\beta_i^{N+1} = \beta_i^N$; and $\alpha_j^{N+1} = \alpha_j^N$, $\beta_j^{N+1} = \beta_j^N$ for $j \neq i$.
   - Upon *Penalty*: $\alpha_i^{N+1} = \alpha_i^N$; $\beta_i^{N+1} = \beta_i^N + 1$; and $\alpha_j^{N+1} = \alpha_j^N$, $\beta_j^{N+1} = \beta_j^N$ for $j \neq i$.

**End Algorithm: BLA-MABB**

As seen from the above BLA algorithm, $N$ is a discrete time index and the parameters $\phi^N = \langle \alpha_1^N, \beta_1^N, \alpha_2^N, \beta_2^N, \ldots, \alpha_r^N, \beta_r^N \rangle$ form an infinite discrete $2 \times r$-dimensional state space, which we will denote with $\Phi$. Within $\Phi$ the BLA navigates by iteratively adding 1 to either $\alpha_1^N$, $\beta_1^N$, $\alpha_2^N$, $\beta_2^N, \ldots,$ $\alpha_r^N$ or $\beta_r^N$.

Since the state space of BLA is both discrete and infinite, BLA is quite different from both the *Variable Structure*- and the *Fixed Structure* LA families (Thathachar and Sastry, 2004), traditionally referred to as *Learning Automata*. In all brevity, the novel aspects of the BLA are listed below:

1. In traditional LA, the action chosen (i.e, Arm pulled) is based on the so-called action probability vector. The BLA does not maintain such a vector, but chooses the arm based on the *distribution* of the components of the *Estimate* vector.
2. The second difference is that we have not chosen the arm based on the *a posteriori* distribution of the estimate. Rather, it has been chosen based on the magnitude of a *random sample* drawn from the *a posteriori* distribution, and thus it is more appropriate to state that the arm is chosen based on the *order of statistics* of instances of these variables[5].
3. The third significant aspect is that we can now consider the design of Pursuit LA in which the estimate used is not of the ML family, but on a Bayesian updating scheme. As far as we know, such a mechanism is also unreported in the literature.
4. The final significant aspect is that we can now devise solutions to the Multi-Armed Bandit problem even for cases when the Reward/Penalty distribution is not Bernoulli distributed. Indeed, we advocate the use of a Bayesian methodology with the appropriate Conjugate Prior (Duda et al., 2000).

In the interest of notational simplicity, let *Arm* 1 be the Arm under investigation. Then, for any parameter configuration $\phi^N \in \Phi$ we can state, using a generic notation[6], that the probability of selecting *Arm* 1 is equal to the probability $P(X_1^N > X_2^N \wedge X_1^N > X_3^N \wedge \cdots \wedge X_1^N > X_r^N | \phi^N)$ — the probability that a randomly drawn value $x_1 \in X_1^N$ is greater than all of the other randomly drawn values $x_j \in X_j^N, j \neq i$, at time step $N$, when the associated stochastic variables $X_1^N, X_2^N, \ldots, X_r^N$ are *Beta* distributed, with parameters $\alpha_1^N, \beta_1^N, \alpha_2^N, \beta_2^N, \ldots, \alpha_r^N, \beta_r^N$ respectively. In the following, we will let $p_1^{\phi^N}$ denote this latter probability.

The probability $p_1^{\phi^N}$ can also be interpreted as the probability that *Arm* 1 is the optimal one, given the observations $\phi^N$. The formal result that we derive in the unabridged paper shows that the BLA will gradually shift its Arm selection focus towards the Arm which most likely is the optimal one, as the observations are received.

Finally, observe that the BLA does not rely on any external parameters that must be configured to optimize performance for specific problem instances. This is in contrast to the traditional LA family of algorithms, where a "learning speed/accuracy" parameter is inherent in $\varepsilon$-optimal schemes.

# 4 EMPIRICAL RESULTS

In this section we evaluate the BLA by comparing it with the best performing algorithms from (Auer et al., 2002; Vermorel and Mohri, 2005), as well as the $L_{R-I}$ and Pursuit schemes, which can be seen as established top performers in the LA field. Based on our comparison with these "reference" algorithms, it should be quite straightforward to also relate the BLA performance results to the performance of other similar algorithms.

For the sake of fairness, we base our comparison on the experimental setup for the MABB found in (Auer et al., 2002). Although we have conducted numerous experiments using various reward distributions, we here report, for the sake of brevity, results for the experiment configurations enumerated in Table 1.

Experiment configuration 1 and 4 forms the simplest environment, with low reward variance and a large difference between the reward probabilities of the arms. By reducing the difference between the arms, we increase the difficulty of the MABB problem. Configuration 2 and 5 fulfill this purpose. The challenge of configuration 3 and 6 is their high variance combined with the small difference between the available arms.

For these experiment configurations, an ensemble of 1000 independent replications with different random number streams was performed to minimize the variance of the reported results[7]. In each replication, 100 000 arm pulls were conducted in order to examine both the short term and the limiting performance of the evaluated algorithms.

Note that real-world instantiations of the bandit problem, such as Resource Allocation in Web Polling (Granmo et al., 2007) , may exhibit any reward probability in the interval $[0,1]$. Hence, a solution scheme designed to tackle bandit problems in general, should perform well across the complete space of reward probabilities.

[5] To the best of our knowledge, the concept of having automata choose actions based on the *order of statistics* of instances of estimate distributions, has been unreported in the literature

[6] By this we mean that $P$ is not a fixed function. Rather, it denotes the probability function for a random variable, given as an argument to $P$.

[7] Some of the tested algorithms were unstable for certain reward distributions, producing a high variance compared to the mean regret. This confirms the observations from (Audibert et al., 2007) where the high variance of e.g. UCB-TUNED was first reported. Thus, in our experience 100 replications were too few to unveil the "true" performance of these algorithms.

Table 1: Reward distributions used in 2-armed and 10-armed Bandit problems with Bernoulli distributed rewards.

| **Config./Arm** | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 |
|---|---|---|---|---|---|---|---|---|---|---|
| 1 | 0.90 | 0.60 | - | - | - | - | - | - | - | - |
| 2 | 0.90 | 0.80 | - | - | - | - | - | - | - | - |
| 3 | 0.55 | 0.45 | - | - | - | - | - | - | - | - |
| 4 | 0.90 | 0.60 | 0.60 | 0.60 | 0.60 | 0.60 | 0.60 | 0.60 | 0.60 | 0.60 |
| 5 | 0.90 | 0.80 | 0.80 | 0.80 | 0.80 | 0.80 | 0.80 | 0.80 | 0.80 | 0.80 |
| 6 | 0.55 | 0.45 | 0.45 | 0.45 | 0.45 | 0.45 | 0.45 | 0.45 | 0.45 | 0.45 |

Table 2: Results on 2-armed and 10-armed Bandit problem with Bernoulli distributed rewards.

| **Algorithm/Config.** | **1** | **2** | **3** | **4** | **5** | **6** |
|---|---|---|---|---|---|---|
| BLA Bernoulli | 1.000 | 0.999 | 0.997 | 0.998 | 0.988 | 0.975 |
| $\varepsilon_n$ − GREEDY $c$ =0.05 † | 0.981 | 0.992 | 0.965 | 0.996 | 0.961 | 0.893 |
| $\varepsilon_n$ − GREEDY $c$ =0.15 † | 1.000 | 0.999 | 0.991 | 0.990 | 0.988 | 0.957 |
| $\varepsilon_n$ − GREEDY $c$ =0.30 † | 1.000 | 0.997 | 0.997 | 0.982 | 0.981 | 0.977 |
| $L_{R-I}$ 0.05 | 0.999 | 0.918 | 0.985 | 0.832 | 0.378 | 0.526 |
| $L_{R-I}$ 0.01 | 0.998 | 0.993 | 0.993 | 0.992 | 0.885 | 0.958 |
| $L_{R-I}$ 0.005 | 0.995 | 0.986 | 0.986 | 0.984 | 0.940 | 0.951 |
| Pursuit 0.05 | 1.000 | 0.970 | 0.932 | 0.912 | 0.699 | 0.608 |
| Pursuit 0.01 | 0.999 | 0.998 | 0.998 | 0.998 | 0.875 | 0.848 |
| Pursuit 0.005 | 0.999 | 0.999 | 0.998 | 0.997 | 0.960 | 0.924 |
| UCB1 | 0.998 | 0.982 | 0.983 | 0.979 | 0.848 | 0.848 |
| UCB-TUNED | 1.000 | 0.997 | 0.997 | 0.997 | 0.977 | 0.978 |
| Exp3 $\gamma$ = 0.01 | 0.990 | 0.978 | 0.980 | 0.913 | 0.736 | 0.749 |
| POKER | 0.995 | 0.991 | 0.876 | 0.982 | 0.916 | 0.812 |
| INTESTIM 0.01 | 0.961 | 0.949 | 0.796 | 0.920 | 0.905 | 0.577 |

† Parameter $d$ is set to be the difference in reward probability between the best arm and the second best arm

For all of the experiment configurations in the table, we compared the performance of both the BLA, $\varepsilon_n$-GREEDY, $L_{R-I}$, Pursuit, UCB-1, UCB-TUNED, EXP3, POKER, and INTESTIM. In Table 2 we report the average probability of pulling the best arm over 100 000 arm pulls. By taking the average probability over all the arm selections, a low learning pace is penalized, however, long term performance is emphasized. As seen in the table, BLA provides either equal or better performance than any of the compared algorithms, expect for experiment configuration 6 where UCB-TUNED provides slightly better performance than BLA. Also note that the $\varepsilon_n$-GREEDY algorithm is given the difference between the best arm and the second best arm, thus giving it an unfair advantage.

Both learning accuracy and learning speed governs the performance of bandit playing algorithms in practice. Table 3 reports the average probability of selecting the best arms after 10, 100, 1000, 10 000, and 100 000 arm pulls for experiment configuration 5.

As seen from the table, INTESTIM provides the best performance after 10 arm pulls, being slightly better than BLA. After 100 arm pulls, however, BLA provides the best performance. Then, after 1000 arm pulls, one of the parameter configurations of $\varepsilon_n$-GREEDY as well as the Pursuit scheme provide slightly better performance than BLA, with BLA being clearly superior after 10 000 and 100 000 arm pulls.

We now consider the *Regret* of the algorithms. *Regret* offers the advantage that it does not overly emphasize the importance of pulling the best arm. Indeed, pulling one of the non-optimal arms will not necessarily affect the overall amount of rewards obtained in a significant manner if for instance the reward probability of the non-optimal arm is relatively close to the optimal reward probability. For *Regret* it turns out that the performance characteristics of the algorithms are mainly decided by the reward distributions, and not by the number of arms. Thus, in Fig. 1 we now consider configuration 4, 5, and 6 only. The plots in the figure show the accumulation of regret with the number of arm pulls. Because of the logarithmically scaled x- and y-axes, it is clear from the plots that both BLA and UCB-TUNED attain a logarithmi-

Table 3: Detailed overview of the 10-armed problem with optimal arm $p = 0.9$ and $p = 0.8$ on the rest.

| **Algorithm/#Arm Pulls** | **10** | **100** | **1000** | **10000** | **100000** |
|---|---|---|---|---|---|
| BLA Bernoulli | 0.112 | 0.197 | 0.549 | 0.916 | 0.988 |
| $\varepsilon_n$ − GREEDY $c$ =0.05 $d$ =0.10 | 0.101 | 0.124 | 0.630 | 0.898 | 0.961 |
| $\varepsilon_n$ − GREEDY $c$ =0.15 $d$ =0.10 | 0.105 | 0.100 | 0.511 | 0.911 | 0.988 |
| $\varepsilon_n$ − GREEDY $c$ =0.30 $d$ =0.10 | 0.099 | 0.099 | 0.359 | 0.872 | 0.981 |
| $L_{R-I}$ 0.05 | 0.103 | 0.119 | 0.273 | 0.368 | 0.378 |
| $L_{R-I}$ 0.01 | 0.104 | 0.105 | 0.156 | 0.672 | 0.885 |
| $L_{R-I}$ 0.005 | 0.102 | 0.102 | 0.126 | 0.518 | 0.940 |
| Pursuit 0.05 | 0.100 | 0.157 | 0.567 | 0.682 | 0.699 |
| Pursuit 0.01 | 0.098 | 0.116 | 0.550 | 0.840 | 0.875 |
| Pursuit 0.005 | 0.101 | 0.108 | 0.488 | 0.910 | 0.960 |
| UCB1 | 0.100 | 0.119 | 0.166 | 0.406 | 0.848 |
| UCB-TUNED | 0.100 | 0.164 | 0.425 | 0.841 | 0.977 |
| Exp3 $\gamma = 0.01$ | 0.097 | 0.099 | 0.104 | 0.156 | 0.736 |
| POKER | 0.105 | 0.180 | 0.444 | 0.751 | 0.916 |
| INTESTIM 0.01 | 0.126 | 0.194 | 0.519 | 0.857 | 0.905 |

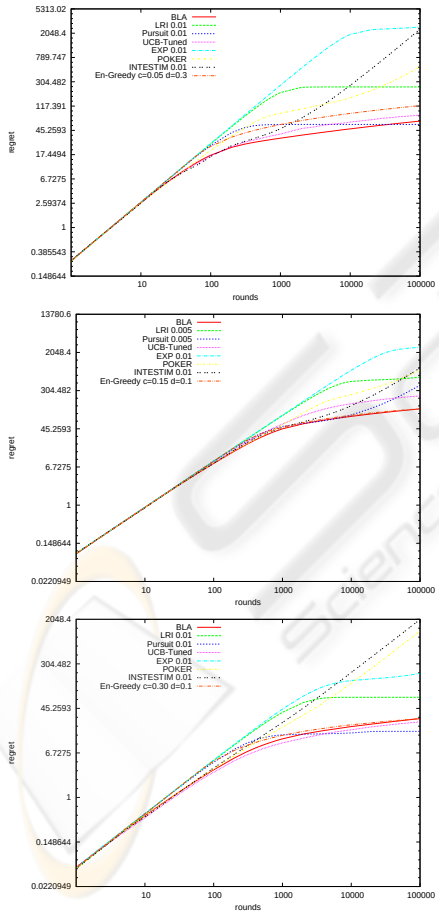


Figure 1: Regret for experiment conf. 4 (top left), conf. 5 (top right), and conf. 6 (bottom).

cally growing regret. Moreover, for configuration 4, the performance of BLA is significantly better than that of the other algorithms, with the Pursuit scheme catching up from the final 10 000 to 100 000 rounds. Note that if the learning speed of the Pursuit scheme is increased to match that of BLA, the accuracy of the Pursuit schemes becomes significantly lower than that of BLA. Surprisingly, both of the LA schemes converge to constant regret. This can be explained by their ε-optimality and the relatively low learning speed parameter used ($a = 0.01$). In brief, the LA converged to only selecting the optimal arm in all of the 1000 replications.

For experiment configuration 5, however, it turns out that the applied learning accuracy of the LA is too low to always converge to only selecting the optimal arm ($a = 0.005$). In some of the replications, the LA also converges to selecting the inferior arm only, and this leads to linearly growing regret. Note that the LA can achieve constant regret in this latter experiment too, by increasing learning accuracy. However, this reduces learning speed, which for the present setting already is worse than that of BLA and UCB-TUNED. As also seen in the plots, the BLA continues to provide the best performance.

Finally, we observe that the high variance of configuration 3 and 6 reduces the performance gap between BLA and UCB-TUNED, leaving UCB-TUNED with slightly lower regret compared to BLA. Also, notice that the Pursuit scheme in this case too is able to achieve more or less constant regret, at the cost of somewhat reduced learning speed.

From the above results, we conclude that BLA is the superior choice for MABB problems in general, providing significantly better performance in most of

the experiment configurations. Only in two of the experiment configurations does it provide *slightly* lower performance than the second best algorithm for *those* configurations. Finally, BLA does not rely on fine-tuning some learning parameter to achieve this performance.

## 5 CONCLUSIONS AND FURTHER WORK

In this paper we presented the Bayesian Learning Automaton (BLA) for tackling the classical MABB problem. In contrast to previous LA and regret minimizing approaches, BLA is inherently Bayesian in nature. Still, it relies simply on counting of rewards/penalties and random sampling from a set of sibling beta distributions. Thus, to the best of our knowledge, BLA is the first MABB algorithm that takes advantage of Bayesian estimation in a computationally efficient manner. Furthermore, extensive experiments demonstrates that our scheme outperforms recently proposed bandit playing algorithms.

Accordingly, in the above perspective, it is our belief that the BLA represents a new promising avenue of research. E.g., incorporating other reward distributions, such as Gaussian and multinomial distributions, into our scheme is of interest. Secondly, we believe that our scheme can be modified to tackle bandit problems that are non-stationary, i.e., where the reward probabilities are changing with time. Finally, systems of BLA can be studied from a game theory point of view, where multiple BLAs interact forming the basis for multi-agent systems.

## REFERENCES