Generalized Bayesian Pursuit: A Novel Scheme for Multi-Armed Bernoulli Bandit Problems

Xuan Zhang[1], B. John Oommen[2,1,⋆], and Ole-Christoffer Granmo[1]

[1] Dept. of ICT, University of Agder, Grimstad, Norway
[2] School of Computer Science, Carleton University, Ottawa, Canada

Abstract. In the last decades, a myriad of approaches to the multi-armed bandit problem have appeared in several different fields. The current top performing algorithms from the field of Learning Automata reside in the Pursuit family, while UCB-Tuned and the ε-greedy class of algorithms can be seen as state-of-the-art regret minimizing algorithms. Recently, however, the Bayesian Learning Automaton (BLA) outperformed all of these, and other schemes, in a wide range of experiments. Although seemingly incompatible, in this paper we integrate the foundational learning principles motivating the design of the BLA, with the principles of the so-called Generalized Pursuit algorithm (GPST), leading to the *Generalized Bayesian Pursuit* algorithm (GBPST). As in the BLA, the estimates are truly Bayesian in nature, however, instead of basing exploration upon direct sampling from the estimates, GBPST explores by means of the arm selection probability vector of GPST. Further, as in the GPST, in the interest of higher rates of learning, a *set of arms* that are currently perceived as being optimal is pursued to minimize the probability of pursuing a wrong arm. It turns out that GBPST is superior to GPST and that it even performs better than the BLA by controlling the learning speed of GBPST. We thus believe that GBPST constitutes a new avenue of research, in which the performance benefits of the GPST and the BLA are mutually augmented, opening up for improved performance in a number of applications, currently being tested.

Keywords: Bandit Problems, Estimator Algorithms, Generalized Bayesian Pursuit Algorithm, *Beta* Distribution, Conjugate Priors.

1 Introduction

The multi-armed Bernoulli bandit problem (MABB) is a classical optimization problem that captures the exploration-exploitation dilemma. The MABB setup consists of a gambling machine with multiple arms and an agent that sequentially pulls one of the arms, with each pull resulting in either 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 d_i, and a penalty probability $1 - d_i$. The dilemma is this: Should the arm that so far seems to provide the highest chance of reward be pulled once more, or

⋆ *Chancellor's Professor*; *Fellow: IEEE* and *Fellow: IAPR*. The Author also holds an *Adjunct Professorship* with the Dept. of ICT, University of Agder, Norway.

[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.

should an inferior arm be pulled 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 an inferior arm unnecessarily, postpones the harvest that can be obtained from the optimal arm.

1.1 Existing Solutions to Multi-Armed Bernoulli Bandit Problem

Bandit like problems involve two highly related yet distinct fields: the field of Learning Automata and the field of Bandit Playing Algorithms. A myriad of approaches have been proposed within these two fields. Classical exact solutions for discounted rewards take advantage of Gittins Indices [1,2,3] — by always pulling the arm with the largest Gittins index (measuring the value associated with the state of a stochastic process), the expected amount of discounted rewards obtained is maximized. Calculating Gittins Indices in a computationally efficient manner is far from trivial [4,5], and this problem is currently being pursued [6,7]. Because of the computational difficulties associated with exact solutions based on Gittins Indices, a number of approximate solution techniques has also been proposed. The ε-*greedy* strategy, first described by Watkins [8], represents an early *approximate* solution to the bandit problem, in which the arm so far being perceived as the best is pulled with probability $1-\varepsilon$, and a randomly chosen action is pulled with probability ε. Thus, the expected frequency of exploring a random action is determined by the parameter of ε. A variant of ε-*greedy* strategy is the ε-*decreasing* strategy [9,10], which gradually shifts focus from exploration to exploitation by slowly decreasing ε. Recently, Tokic proposed an *adaptive* ε-greedy strategy based on reward value differences (VDBE) [11]. In this strategy, ε decreases on the basis of changes in the reward value estimates.

Another direction for solving the bandit problem is confidence interval based algorithms. They estimate confidence intervals for the reward probabilities, and identify an "optimistic" reward probability estimate for each arm. The arm with the most optimistic reward probability estimate is then greedily selected [12,13]. Furthermore, Auer et al. has shown that variants of confidence interval based algorithms, the so-called UCB and UCB-Tuned schemes, provide a logarithmic regret bound [10].

There are also algorithms for solving MABB problems that are based on so-called Boltzmann exploration. These introduce the parameter τ as the "temperature" of the exploration. Related schemes include EXP3 [14]. The "Price of Knowledge and Estimated Reward" (POKER) algorithm proposed in [12] takes into consideration pricing of uncertainty, exploiting the arm reward distributions, and the horizon of the problem. The Linear Reward-Inaction (L_{RI}) learning automaton [15] and the Pursuit Scheme based on Linear Reward-Inaction philosophy (PST_{RI}) [16] [17] are known for their ε-optimality. Besides, PST_{RI}, which uses Maximum Likelihood (ML) reward probability estimates to pursue the currently optimal action, is known for its pioneering role in the estimator Learning Automata family. A thorough comparison of several of the above mentioned schemes can be found in [18,12].

There also exists several algorithms based on Bayesian reasoning [19], with [20,21] being a few examples. In general, Bayesian reasoning is in many cases computational intractable for bandit like problems [19]. However, based on the Thompson sampling principle [22], the *Bayesian Learning Automata (BLA)* reported in [23,18] and extended in [24] to deal with non-stationary environments, are inherently Bayesian in nature, yet avoids computational intractability by relying simply on updating the hyper-parameters of sibling conjugate distributions, and on simultaneously sampling randomly from the respective posteriors. As seen in [18] and [24], BLA demonstrate significant performance advantage compared to a number of competing MABB solution schemes.

1.2 Contributions and Paper Organization

In this paper, we propose a new Bayesian algorithm for MABB problems, which we refer to as the Generalized Bayesian Pursuit algorithm (GBPST). GBPST augments the philosophy of BLA with the principles behind the GPST [25] and is able to outperform *both* the GPST as well as the BLA schemes in extensive experiments[2]. This augmentation is achieved as follows: Firstly, as in the BLA, the estimates are truly Bayesian (as opposed to ML in GPST) in nature. However, the arm selection probability vector of GPST is used for exploration purposes. Secondly, as opposed to the ML estimate, which is usually a single value - the one which maximizes the likelihood function - the use of a posterior distribution permits us to choose any one of a *spectrum* of values in the posterior, as the appropriate estimate. In the interest of being concrete, we have chosen a 95% percentile value of the posterior (instead of the mean) to pursue promising arms. Thirdly, as in the GPST, after each arm pull, all arms currently being associated with a higher reward estimate than the currently pulled arm, are pursued. Finally, the pursuit is done using the Linear Reward-Inaction philosophy, leading to the corresponding GBPST$_{RI}$ scheme[3]. To the best of our knowledge, all these contributions are novel to the field of MABB problem, and we thus believe that the GBPST constitutes a new avenue of research, in which the performance benefits of the GPST and the BLA are mutually augmented. We also believe that the theoretical contributions of this paper could lend itself to practical solutions improving performance in a number of applications, some of which are currently being tested.

The paper is organized as follows. In Section 2 we give an overview of GPST and BLA. Then, in Section 3, we present the new Bayesian estimator algorithm – the *Generalized Bayesian Pursuit* algorithm – by incorporating GPST and BLA. In Section 4, we provide extensive experimental results demonstrating that the GBPST is truly superior to GPST. The BLA scheme is also outperformed by appropriately choosing a learning speed parameter for the GBPST$_{RI}$. Finally, in Section 5, we report opportunities for further research, in addition to providing concluding remarks.

2 The Generalized Pursuit Algorithm and Bayesian Learning Automata

We here briefly review the selected schemes upon which the Generalized Bayesian Pursuit scheme builds.

[2] The theoretical results concerning the formal properties of the family of GBPST are currently being compiled.

[3] The pursuit can also be conducted using the Linear Reward-Penalty philosophy (GBPST$_{RP}$), as advocated in [17]. In the interest of brevity, we here report the best performing scheme, which is GBPST$_{RI}$.

Linear Updating Schemes: The more notable and well-used traditional LA approaches include the family of linear updating schemes, with the Linear Reward-Inaction (L_{RI}) automaton being designed for stationary environments [15]. In short, the L_{RI} maintains an arm selection probability vector $\bar{p} = [p_1, p_2, ..., p_r]$, with $\sum_{i=1}^{r} p_i = 1$ and r being the number of arms. 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'_i and p'_j being the resulting updated arm selection probabilities:

$$p'_j = (1-\lambda) \times p_j, 1 \le j \le r, j \neq i$$
$$p'_i = 1 - \sum_{j \neq i} p'_j \text{ if pulling Arm } i \text{ results in a reward.}$$
$$p'_j = p_j, 1 \le j \le r \text{ if pulling Arm } j \text{ results in a penalty.}$$

In the above, the parameter λ ($0 < \lambda < 1$) 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.

Pursuit Schemes: A Pursuit scheme (PST) makes the updating of $\bar{p}$ more goal-directed in the sense that it maintains ML estimates $(\widehat{d_i})$ of the reward probabilities (d_i) associated with each arm. In brief, a Pursuit scheme increases the arm selection probability p_i associated with the currently largest ML estimate $\widehat{d_i}$, instead of the arm actually producing the reward. Thus, unlike L_{RI}, 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 arm is chosen a "sufficiently large number of times". Accordingly, a Pursuit scheme consistently outperforms the L_{RI} in terms of its rate of convergence.

Generalized Pursuit Schemes: The Generalized Pursuit schemes (GPST) generalizes PST by allowing several arms to be pursued at the same time. Instead of only pursuing the arm with the highest reward estimate, the whole *set* of arms that possess higher reward estimates than the arm actually pulled is pursued. In PST, when the arm with the maximum reward estimate is not the one with the highest reward probability, the incorrect arm is pursued, thus potentially derailing the pursuit of the optimal action. The probability of this happening is reduced in GPST.

Bayesian Learning Automata: A unique feature of the *Bayesian Learning Automaton* (BLA) is its computational simplicity, achieved by relying *implicitly* on Bayesian reasoning principles. In essence, at the heart of the BLA we find the *Beta distribution*, which is the conjugate prior for the Bernoulli distribution. Its shape is determined by two positive parameters, denoted by a and b, producing the following probability density function:

$$f(x;a,b) = \frac{x^{a-1}(1-x)^{b-1}}{\int_0^1 u^{a-1}(1-u)^{b-1}du}, \quad x \in [0,1]. \tag{1}$$

Essentially, the BLA uses the *Beta* distribution for two purposes. First of all, it is used to provide a *Bayesian estimate* of the reward probabilities associated with each of the available arms - the latter being valid by virtue of the conjugate prior 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* selection mechanism.

3 The Generalized Bayesian Pursuit Algorithm (GBPST)

Bayesian reasoning is a probabilistic approach to inference which is of significant importance in machine learning because it allows for the *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 [26]. We present here a completely new estimator algorithm that builds upon the GPST framework. However, rather than utilizing ML reward probability estimates, optimistic Bayesian estimates are used to pursue the arms currently perceived to be *potentially* optimal. We thus coin the algorithm *Generalized Bayesian Pursuit* (GBPST).

As in the case of the BLA, the GBPST estimates the reward probability of each arm based on the *Beta* distribution. These Bayesian estimates allow us to accurately calculate an optimistic reward probability x_i that provides a 95% upper bound for the reward probability of arm i, by means of the respective cumulative distribution $F(x_i;a,b)$.

$$F(x_i;a,b) = \frac{\int_0^{x_i} v^{a-1}(1-v)^{b-1}dv}{\int_0^1 u^{a-1}(1-u)^{b-1}du}, \quad x_i \in [0,1]. \tag{2}$$

The following algorithm contains the essence of the GBPST approach.

Algorithm: GBPST$_{RI}$
Parameters:
α: The arm chosen by LA.
p_i: The i^{th} element of the arm selection probability vector P.
λ: The learning speed, where $0 < \lambda < 1$.
a_i, b_i: The two positive parameters of the *Beta* distribution.
x_i: The i^{th} element of the Bayesian estimate vector X, given by the 95% upper bound of the cumulative distribution function of the corresponding *Beta* distribution.
R: The response from the environment, where $R = 0$ (reward) or $R = 1$ (penalty).
Initialization:
1. $p_i(t) = 1/r$, where r is the number of arms.
2. Set $a_i = b_i = 1$. Then repeat Step 1 and Step 2 in "Method" below a small number of times (i.e., in this paper $10 * r$ times) to get initial estimates for a_i and b_i.
Method: For t:=1 to N **Do**

1. Pick $\alpha(t)$ randomly according to arm selection probability vector $P(t)$. Suppose $\alpha(t) = \alpha_i$.

2. Based on the Bayesian nature of the conjugate distributions, update $a_i(t)$ and $b_i(t)$ according to the response from the environment:
   If $R(t) = 0$ **Then** $a_i(t) = a_i(t-1) + 1; b_i(t) = b_i(t-1)$;
   Else $a_i(t) = a_i(t-1); b_i(t) = b_i(t-1) + 1$;
3. Identify the upper 95% reward probability bound of $x_i(t)$ for each arm i as:

$$\frac{\int_0^{x_i(t)} v^{(a_i-1)}(1-v)^{(b_i-1)}dv}{\int_0^1 u^{(a_i-1)}(1-u)^{(b_i-1)}du} = 0.95$$

4. If $M(t)$ is the number of arms with higher upper 95% reward probability bound than the pulled arm at time t, update the arm selection probability vector $P(t+1)$ according to the following rule:
   If $R(t) = 0$ **Then**
   $p_j(t+1) = (1-\lambda)p_j(t) + \frac{\lambda}{M(t)}$, for $\forall j \neq i$ such that $x_j(t) > x_i(t)$;
   $p_j(t+1) = (1-\lambda)p_j(t)$, for $\forall j \neq i$ such that $x_j(t) \leq x_i(t)$;
   $p_i(t+1) = 1 - \sum_{j \neq i} p_j(t+1)$.
   Else
   $P(t+1) = P(t)$.

End Algorithm: GBPST$_{RI}$

Observe that the GBPST is quite similar to the GPST in the sense that both of them pursue the currently perceived *potentially* optimal arms, and update the arm selection probability vector based on a linear updating rule. The difference is that instead of using ML estimates for the reward probabilities, in the GBPST the estimation is Bayesian, allowing the calculation of 95% upper bounds, $\{x_i\}$.

It is crucial that the salient features of the GBPST and the BLA are highlighted. The reader should observe that they both rely on the *Beta distribution* for reward probability estimation. However, the BLA does not perform any Bayesian computations explicitly. Instead, when it comes to arm selection and exploration, the BLA chooses an arm based on sampling directly from the *Beta* distributions, while the GBPST samples the arm selection space based on the arm selection probability vector. Also, by calculating the 95% upper bound, x_i, the GBPST is able to decide which arms are most promising to pursue.

4 Empirical Results

In this section, we evaluate the computational efficiencies of GBPST$_{RI}$ by comparing it with the GPST$_{RI}$ and the BLA. Although we have conducted numerous experiments using various reward distributions, we report here, for the sake of brevity, results based on the experimental configurations listed in Table 1.

In the experiments considered, Configurations 1 and 4 form the simplest environments, possessing a low reward variance and a large difference between the reward probabilities of the arms. This is because by reducing the difference between the arms, we increase the learning difficulty of the environment. Configurations 2 and 5 achieve this task. The challenge of Configurations 3 and 6 is their high variance combined with the small difference between the arms.

For these configurations, an ensemble of 1,000 independent replications with different random number streams was performed to minimize the variance of the reported results. 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.

Since both the two schemes, the GBPST$_{RI}$ and the GPST$_{RI}$ depend on an external parameter λ, we measure performance using a wide range of learning speeds: $\lambda = 0.05$, $\lambda = 0.01$ and $\lambda = 0.005$. We report the best performing learning speeds.

Table 2 reports the average probability of pulling the optimal arm after 10, 100, $1,000$, $10,000$ and $100,000$ rounds of arm pulls, for each configuration. As seen, in Configurations 1, 2 and 4, all three schemes converge to the optimal arm with high accuracy, with the BLA being the fastest scheme. The GBPST$_{RI}$ and the GPST$_{RI}$ perform comparably.

Table 1. Bernoulli distributed rewards used in 4-armed and 10-armed bandit problems

Config./Arm	1	2	3	4	5	6	7	8	9	10
1	0.90	0.60	0.60	0.60	-	-	-	-	-	-
2	0.90	0.80	0.80	0.80	-	-	-	-	-	-
3	0.55	0.45	0.45	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. Probability of pulling the optimal arm after 10, 100, 1000, 10 000, and 100 000 rounds

Configuration	**Algorithm**	**10**	**100**	**1000**	**10000**	**100000**
Conf.1	**GBPST$_{\mathbf{RI}}$ 0.05**	**0.2655**	**0.5949**	**0.9438**	**0.9941**	**0.9993**
	GPST$_{RI}$ 0.05	0.2649	0.5966	0.9437	0.9941	0.9994
	BLA	0.3512	0.7130	0.9540	0.9942	0.9993
Conf.2	**GBPST$_{\mathbf{RI}}$ 0.005**	**0.2514**	**0.2869**	**0.6812**	**0.9645**	**0.9963**
	GPST$_{RI}$ 0.005	0.2507	0.2844	0.6852	0.9652	0.9956
	BLA	0.2852	0.4583	0.8190	0.9712	0.9960
Conf.3	**GBPST$_{\mathbf{RI}}$ 0.01**	**0.2507**	**0.2800**	**0.6312**	**0.9552**	**0.9953**
	GPST$_{RI}$ 0.005	0.2505	0.2647	0.5387	0.9419	0.9933
	BLA	0.2761	0.3856	0.6942	0.9419	0.9915
Conf.4	**GBPST$_{\mathbf{RI}}$ 0.05**	**0.1027**	**0.3136**	**0.8677**	**0.9854**	**0.9985**
	GPST$_{RI}$ 0.05	0.1035	0.3244	0.8735	0.9862	0.9986
	BLA	0.1407	0.4187	0.8707	0.9826	0.9978
Conf.5	**GBPST$_{\mathbf{RI}}$ 0.01**	**0.0998**	**0.1204**	**0.5065**	**0.9368**	**0.9924**
	GPST$_{RI}$ 0.005	0.0996	0.1106	0.4194	0.9187	0.9819
	BLA	0.1103	0.1870	0.5492	0.9163	0.9878
Conf.6	**GBPST$_{\mathbf{RI}}$ 0.005**	**0.0992**	**0.1035**	**0.2357**	**0.8494**	**0.9817**
	GPST$_{RI}$ 0.005	0.0995	0.1041	0.2596	0.8523	0.9581
	BLA	0.1075	0.1493	0.3572	0.8347	0.9752

In Configuration 3, the GBPST$_{RI}$ outperforms the GPST$_{RI}$. The BLA learns faster than GBPST$_{RI}$ scheme at the beginning but was caught up with and surpassed by the GBPST$_{RI}$ in the final $10,000$ to $100,000$ rounds.

Configurations 5 and 6 are the two most challenging experimental set-ups, in which the superiority of the GBPST$_{RI}$ over the GPST$_{RI}$ is more obvious than in previous configurations. As compared with the BLA, the GBPST$_{RI}$ are not as fast as the BLA at the beginning, but again outperforms the BLA from around the time index 10,000.

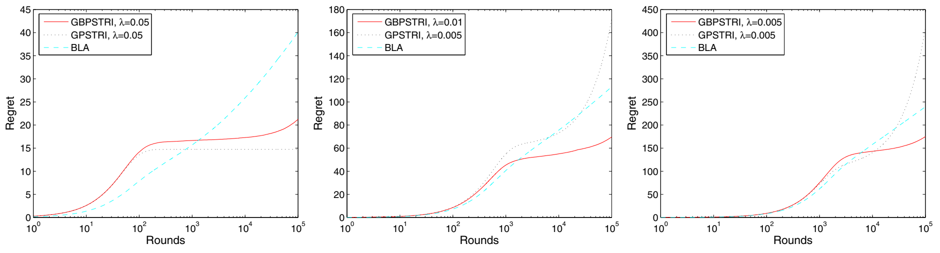
We now consider the so-called *Regret* of the algorithms. The *Regret* is *the difference between the sum of rewards expected after N successive arm pulls, and what would have been obtained by only pulling the optimal arm.* Assuming that a *reward* amounts to the value (utility) of unity (i.e., 1), and that a *penalty* possesses the value 0, the *Regret* can be defined as:

$$d_{opt} \cdot N - \sum_{i=1}^{N} d_i, \tag{3}$$

where d_{opt} is the reward probability of the optimal action and d_i is the reward probability of the selected action i.

The *Regret* offers the advantage that it does not overly emphasize the importance of pulling the best arm. In fact, pulling one of the inferior arms will not necessarily affect the overall amount of rewards obtained in a significant manner if, for instance, the reward probability of the inferior 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, we now consider configuration 4, 5, and 6 only.

The plots in Fig. 1 illustrate the accumulation of the *Regret* of each algorithm with the number of rounds of pulling arms. As seen in Fig. 1(a), early in the learning phase, the BLA is clearly better than the other two schemes, but the GBPST$_{RI}$ and the GPST$_{RI}$ catch up later with the GBPST$_{RI}$ increasing slowly and the GPST$_{RI}$ converging to yield a constant *Regret*. In the more challenging configurations as shown in Fig. 1(b) and Fig. 1(c), none of the schemes converge to yield a constant *Regret* because of their low learning accuracy. However, the *Regret* of the GBPST$_{RI}$ is much lower and increases more slowly than the others, showing its superiority to the other schemes.



(a) *Conf. 4* (b) *Conf. 5* (c) *Conf. 6*

Fig. 1. The *Regret* for experiment conf. 4, conf. 5 and conf. 6

From the above results we draw the following conclusions:

1. The GBPST$_{RI}$ is superior to the GPST$_{RI}$, although the GBPST$_{RI}$ performs *slightly* worse than GPST$_{RI}$ in the simplest configuration. Furthermore, in the other configurations, the GBPST$_{RI}$ provides much better performance than the GPST$_{RI}$, suggesting the former's superiority.
2. By tuning the learning parameter λ, the GBPST$_{RI}$ provides better performance compared to the BLA. On the other hand, in several cases, the BLA initially improves performance faster. This difference in behavior can be explained by their respective distinct strategies for pulling arms. The BLA pulls arms based on the *magnitude* of a random sample drawn from the posterior distribution of the reward estimate, while the GBPST$_{RI}$ chooses arms based on the maintained arm selection probability vector. In fact, with some deeper insight one can see that the initial performance gap can be traced back to the initialization phase of the algorithms, where each arm is pulled to provide initial reward estimates.

5 Conclusion and Further Work

In this paper we have presented the Generalized Bayesian Pursuit Algorithm (GBPST) based on the Reward-Inaction philosophy (GBPST$_{RI}$). The GBPST$_{RI}$ maintains an arm selection probability vector that is used for the selection of the arms. However, it utilizes Bayesian estimates for the reward probabilities associated with each available arm, and adopts a reward-inaction linear updating rule for arm selection probability updating. Also, because we have used the posterior distributions, we are able to utilize a 95% upper bound of the estimates (instead of the mean) to pursue a set of potentially optimal arms. Thus, to the best of our knowledge, the GBPST is the first MABB solution scheme built according to the GPST strategy that also takes advantage of a Bayesian estimation scheme. Our reported extensive experimental results demonstrate the advantages of the GBPST over the GPST scheme. The GBPST also provides better performance than the BLA by choosing λ suitably.

Based on these results, we thus believe that the GBPST forms a new avenue of research, in which the performance benefits of the GPST and the BLA can be combined. In our further work, we intend to investigate how our Generalized Bayesian Pursuit strategy can be extended to the Discretized Pursuit family of schemes. Besides this, we are currently working on convergence proofs, including the results for games of GBPSTs, involving multiple interacting GBPSTs.

References

1. Gittins, J.C.: Bandit processes and dynamic allocation indices. Journal of the Royal Statistical Society. Series B (Methodological) 41(2), 148–177 (1979)
2. Gittins, J.C., Jones, D.M.: A dynamic allocation index for the discounted multiarmed bandit problem. Biometrika 66(3), 561–565 (1979)
3. Whittle, P.: Multi-armed bandits and the gittins index. Journal of the Royal Statistical Society. Series B (Methodological) 42(2), 143–149 (1980)
4. Varaiya, P., Walrand, J., Buyukkoc, C.: Extensions of the multiarmed bandit problem. IEEE Trans. Autom. Control 30, 426–439 (1985)

5. Katehakis, M., Veinott, A.: The multi-armed bandit problem: decomposition and computation. Math. Oper. Res. 12(2), 262–268 (1987)
6. Sonin, I.: A generalized gittins index for a markov chain and its recursive calculation. Statistics and Probability Letters 78, 1526–1533 (2008)
7. Nino-Mora, J.: A (2/3)n3 fast-pivoting algorithm for the gittins index and optimal stopping of a markov chain. INFORMS Journal of Computing 19(4), 596–606 (2007)
8. Watkins, C.J.C.H.: Learning from delayed rewards. Ph.D. thesis. Cambridge University (1989)
9. Cesa-Bianchi, N., Fischer, P.: Finite-time regret bounds for the multiarmed bandit problem. In: ICML1998, Madison, Wisconsin, USA, pp. 100–108 (1998)
10. Auer, P., Cesa-Bianchi, N., Fischer, P.: Finite time analysis of the multiarmed bandit problem. Machine Learning 47, 235–256 (2002)
11. Tokic, M.: Adaptive ε-greedy exploration in reinforcement learning based on value differences. In: Dillmann, R., Beyerer, J., Hanebeck, U.D., Schultz, T. (eds.) KI 2010. LNCS, vol. 6359, pp. 203–210. Springer, Heidelberg (2010)
12. Vermorel, J., Mohri, M.: Multi-armed bandit algorithms and empirical evaluation. In: Gama, J., Camacho, R., Brazdil, P.B., Jorge, A.M., Torgo, L. (eds.) ECML 2005. LNCS (LNAI), vol. 3720, pp. 437–448. Springer, Heidelberg (2005)
13. Kaelbling, L.P.: Learning in embedded systems. PhD thesis, Stanford University (1993)
14. Auer, P., Cesa-Bianchi, N., Freund, Y., Schapire, R.E.: Gambling in a rigged casino: the adversial multi-armed bandit problem. In: the 36th Annual Symposium on Foundations of Computer Science (FOCS 1995), Milwaukee, Wisconsin, pp. 322–331 (1995)
15. Narendra, K.S., Thathachar, M.A.L.: Learning Automat: An Introduction. Prentice Hall, Englewood Cliffs (1989)
16. Thathachar, M., Sastry, P.: Estimator algorithms for learning automata. In: The Platinum Jubilee Conference on Systems and Signal Processing, Bangalore, India, pp. 29–32 (1986)
17. Oommen, B., Agache, M.: Continuous and discretized pursuit learning schemes: various algorithms and their comparison. IEEE Trans. on Systems, Man, and Cybernetics, Part B: Cybernetics 31(3), 277–287 (2001)
18. Norheim, T., Brådland, T., Granmo, O.C., Oommen, B.J.: A generic solution to multi-armed bernoulli bandit problems based on random sampling from sibling conjugate priors. In: Filipe, J., Fred, A., Sharp, B. (eds.) ICAART 2010. CCIS, vol. 129, pp. 36–44. Springer, Heidelberg (2011)
19. Sutton, R.S., Barto, A.G.: Reinforcement learning: An introduction. MIT Press, Cambridge (1998)
20. Wyatt, J.: Exploration and inference in learning from reinforcement. PhD thesis, University of Edinburgh (1997)
21. Dearden, R., Friedman, N., Russell, S.: Bayesian q-learning. In: The 15th National Conf. on Artificial Intelligence, Madison, Wisconsin, pp. 761–768 (1998)
22. Thompson, W.R.: On the likelihood that one unknown probability exceeds another in view of the evidence of two samples. Biometrika 25, 285–294 (1933)
23. Granmo, O.: Solving two-armed bernoulli bandit problems using a bayesian learning automaton. International Journal of Intelligent Computing and Cybernetics 3(2), 207–234 (2010)
24. Granmo, O.C., Berg, S.: Solving Non-Stationary Bandit Problems by Random Sampling from Sibling Kalman Filters. In: García-Pedrajas, N., Herrera, F., Fyfe, C., Benítez, J.M., Ali, M. (eds.) IEA/AIE 2010. LNCS, vol. 6098, pp. 199–208. Springer, Heidelberg (2010)
25. Agache, M., Oommen, B.J.: Generalized pursuit learning schemes: new families of continuous and discretized learning automata. IEEE Trans. on Systems, Man, and Cybernetics, Part B: Cybernetics 32(6), 738–749 (2002)
26. Mitchell, T.M.: Machine Learning. McGraw-Hill, New York (1997)