The Bayesian Pursuit Algorithm: A New Family of Estimator Learning Automata

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

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

Abstract. The fastest Learning Automata (LA) algorithms currently available come from the family of *estimator* algorithms. The Pursuit algorithm (PST), a pioneering scheme in the estimator family, obtains its superior learning speed by using Maximum Likelihood (ML) estimates to pursue the action currently perceived as being optimal. Recently, a *Bayesian* LA (BLA) was introduced, and empirical results that demonstrated its advantages over established top performers, including the PST scheme, were reported. The BLA scheme is inherently Bayesian in nature, but it succeeds in avoiding the computational intractability by merely relying on updating the hyper-parameters of sibling conjugate priors, and on random sampling from the resulting posteriors.

In this paper, we integrate the foundational learning principles motivating the design of the BLA, with the principles of the PST. By doing this, we have succeeded in obtaining a completely novel, and rather pioneering, approach to solving LA-like problems, namely, by designing the Bayesian Pursuit algorithm (BPST). As in the BLA, the estimates are truly Bayesian (as opposed to ML) in nature. However, the action selection probability vector of the PST is used for its exploration purposes. Also, unlike the ML estimate, which is usually a single value, the use of a posterior distribution permits us to choose any one of a *spectrum* of values in the posterior, as the appropriate estimate. Thus, in this paper, we have chosen a 95% percentile value of the posterior (instead of the mean) to pursue the most promising actions. Further, as advocated in [7], the pursuit has been done using both the Linear Reward-Penalty and Reward-Inaction philosophies, leading to the corresponding BPST$_{RP}$ and BPST$_{RI}$ schemes respectively.

It turns out that the BPST is superior to the PST, with the BPST$_{RI}$ being even more robust than the BPST$_{RP}$. Moreover, by controlling the learning speed of the BPST, the BPST schemes perform either better or comparable to the BLA. We thus believe that the BPST constitutes a new avenue of research, in which the performance benefits of the PST and the BLA are mutually augmented, opening up for improved performance in a number of applications, currently being tested.

Keywords: Estimator Algorithms, Learning Automata, Pursuit Algorithm, Bayesian Learning Automata, Bayesian Pursuit Algorithm, *Beta* Distribution.

[⋆] *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 Introduction

A Learning Automaton (LA) is an adaptive decision-making unit, operating within a random environment, and that learns the optimal action among a finite (or infinite) set of actions offered by the environment. Each action, when performed or chosen, produces either a reward or a penalty, and the optimal action is defined as the action with the highest probability of producing a reward. Of all the LA that have been proposed, estimator algorithms are among the fastest ones. These were introduced by Thathachar and Sastry [2] with the so-called Pursuit algorithm (PST), which utilizes Maximum Likelihood (ML) reward probability estimates to pursue the action currently perceived to be optimal. The latter method was later enhanced by the authors of [7], as we shall explain presently. Recently, however, in [3], a novel sampling-based *Bayesian* scheme, coined the Bayesian Learning Automata (BLA), was introduced. Being simultaneously based on counting rewards/penalties and on random sampling from a pair of sibling *Beta* distributions (akin to the *Thompson Sampling* [1] principle), the BLA offers significantly better performance than recent LA and related schemes, including the PST [3], especially for bandit-like problems.

Research in the theory of LA has made significant advances in the last decades. LA have, for instance, found novel applications in systems that possess incomplete knowledge about the environment in which they operate. In the interest of brevity, we list a few more-recent applications here. They are routing in wireless sensor networks [8], [9], [10], stochastic traffic engineering in multihop cognitive wireless mesh networks [11], design of cognitive radio systems [12], and the near-optimal solution of NP-hard problems like data fragment allocation in distributed database systems [13].

In this paper, we propose a new class of estimator algorithms, which we refer to as the family of Bayesian Pursuit algorithms (BPSTs). Briefly stated, by augmenting (or "piggy-backing") the method of the BLA with the principles behind the PST, we are able to outperform *both* schemes in extensive experiments[1]. This augmentation is achieved as follows: First of all, as in the BLA, the estimates are truly Bayesian (as opposed to ML) in nature. However, the action selection probability vector of the PST is used for its exploration purposes. Additionally, as opposed to the ML estimate, which is usually a *single* value - i.e., 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 the most promising actions. Finally, as advocated in [7], the pursuit can be done using both the Linear Reward-Penalty and Reward-Inaction philosophies. This too has been accomplished here, leading to the corresponding BPST$_{RP}$ and BPST$_{RI}$ schemes respectively. To the best of our knowledge, all these contributions are novel to the field of LA, and we thus believe that the BPST constitutes a new avenue of research, in which the performance benefits of the PST 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.

[1] The theoretical results proving the formal properties of the family of BPSTs are currently being compiled.

The paper is organized as follows. In Section 3, we present the new branch of estimator algorithms – the *Bayesian Pursuit* algorithms – including those incorporating the two linear updating schemes, the $BPST_{RP}$ and $BPST_{RI}$. Then, in Section 4, we provide extensive experimental results demonstrating that the BPSTs are truly superior to the PST – with the BPST$_{RI}$ performing in an even more robust manner than the BPST$_{RP}$. In comparison to the BLA, by appropriately choosing a learning speed parameter for the BPST schemes, one can obtain comparable or even superior performances. Finally, in Section 5, we report opportunities for further research, in addition to providing concluding remarks.

2 The Pursuit Algorithm (PST) and Bayesian Learning Automata

A myriad of approaches have been proposed within the field of LA, and we refer the reader to [5, 6] for an overview and comparison of schemes. However, in this work, we will briefly review[2] the LA upon which the Bayesian Pursuit scheme builds.

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_{R-I}) automaton being designed for stationary environments [5]. In short, the L_{R-I} maintains an action probability selection vector $\bar{p} = [p_1, p_2]$, with $p_2 = 1 - p_1$. The question of which action is to be chosen 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 action selection probabilities:

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

In the above, the parameter λ ($0 < \lambda < 1$) governs the learning speed. As seen, after action i has been chosen, 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 $(\hat{d}_1, \hat{d}_2)$ of the reward probabilities (d_1, d_2) associated with each action. 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 action selection probability, p_i, is to be increased. In brief, a Pursuit scheme increases the action selection probability p_i associated with the currently-largest ML estimate $\hat{d}_i$, instead of the action actually producing the reward.Thus, unlike the L_{R-I}, in which the reward from an inferior action 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

[2] Throughout this review, we only consider the 2-action LA. The r-action LA are found in [5,6].

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.

Bayesian Learning Automaton: The *Bayesian Learning Automata* (BLA) inherently builds upon the Bayesian reasoning framework. A unique feature of the 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 actions - 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 Bayesian Pursuit Algorithm (BPST)

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 [14]. We present here a new branch of estimator algorithms that builds upon the PST framework. However, rather than utilizing ML reward probability estimates, optimistic Bayesian estimates are used to pursue the action currently perceived to be optimal. We coin the algorithm as the *Bayesian Pursuit* algorithm (BPST).

As in the case of the BLA, the BPST estimates the reward probabilities of all the actions based on the *Beta* distribution. These Bayesian estimates allow us to accurately calculate an optimistic estimate of the reward probability, x_i, that provides a 95% upper bound for d_i, by means of the respective cumulative distribution $F(x_i;a,b)$ as below, and the subsequent algorithm contains the essence of the BPST approach.

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

Algorithm: BPST$_{RP}$
Parameters:
α: The action chosen by LA.
p_i: The i^{th} element of the action probability vector P.
λ: The learning parameter, where $0 < \lambda < 1$.

m: The index of the maximal component of the Bayesian estimate vector X.
e_m: The unit vector with '1' in the m^{th} coordinate.
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 actions.
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 action 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 action 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. Update the action probability vector $P(t)$:
   $P(t) = (1-\lambda)P(t-1) + \lambda e_m$, with $m = \text{argmax}_i[x_i(t)]$;

End Algorithm: BPST$_{RP}$

Note that the above algorithm implements the Reward-Penalty strategy ($BPST_{RP}$), in which both reward and penalty responses are processed. In an analogous manner, one can design a BPST scheme based on the Reward-Inaction strategy, i.e., one in which penalties are ignored ($BPST_{RI}$). Also observe that the BPST is quite similar to the PST in the sense that both of them pursue the currently-perceived optimal action, and update the action probability vector based on a linear updating rule. The difference is that instead of using ML estimates for the reward probabilities, in the BPST, the estimation is Bayesian, allowing the calculation of 95% upper bounds, $\{x_i\}$.

It is crucial that the salient features of the BPST 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 action selection and exploration, the BLA chooses an action based on sampling directly from the *Beta* distributions, while the BPST samples the action space based on the action probability vector. Also, by calculating the 95% upper bound, x_i, the BPST is able to decide which action is most promising to pursue.

4 Empirical Results

In this section, we evaluate the computational efficiencies of both the BPST$_{RP}$ and the BPST$_{RI}$ by comparing them with the PST$_{RI}$ and the BLA. Although a more detailed comparison against the family of PST schemes is also currently being compiled, we report here, in all brevity, that both the BPST$_{RP}$ and the BPST$_{RI}$ are superior to the other PST schemes reported in [7]. Further, although we have conducted numerous experiments using various reward distributions, we report here, for the sake of brevity, results for 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 actions. This is because, by reducing the difference between the actions, 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 actions.

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, $100,000$ actions selection tasks were conducted in order to examine both the short term and the limiting performance of the evaluated algorithms.

Since all the three schemes, the BPST$_{RP}$, BPST$_{RI}$, and the PST$_{RI}$ depend on an external learning parameter λ (the learning rate), we measured the performance using a wide range of parameters: $\lambda = 0.05$, $\lambda = 0.01$ and $\lambda = 0.005$. We report the best-performing learning rates.

Table 2 reports the average probability of selecting the optimal action after 10, 100, $1,000$, $10,000$ and $100,000$ rounds of action selections, for each configuration. As seen, in Configurations 1 and 2, all four schemes converge to the optimal action with high accuracy, with the BLA being the fastest scheme and the BPST$_{RP}$ being the second fastest. The BPST$_{RI}$ and PST$_{RI}$ come third, providing almost identical performance.

In Configuration 3, the BPST schemes outperform PST$_{RI}$. The BLA learns faster than the BPSTs at the beginning, but was caught up to and surpassed by the latter in the final $10,000$ to $100,000$ rounds.

The BPST schemes are also superior to the PST$_{RI}$ in Configuration 4. Indeed, the BPST schemes also yield more accurate results than the BLA from the rounds whose indices are from $1,000$ to $100,000$, although it learns slightly slower initially.

Table 1. Reward distributions used in 2-action and 10-action problems with Uniformly distributed rewards

Config./Action	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. Detailed overview of probability of choosing the optimal action after 10, 100, 1000, 10 000, and 100 000 rounds

Configuration	**Algorithm**	**10**	**100**	**1000**	**10000**	**100000**
Conf.1	$BPST_{RI}$ 0.05	0.5227	0.8101	0.9774	0.9977	0.9997
	$BPST_{RP}$ 0.05	0.5323	0.8280	0.9795	0.9979	0.9998
	PST_{RI} 0.05	0.5232	0.8097	0.9774	0.9977	0.9998
	BLA	0.6266	0.8839	0.9839	0.9980	0.9997
Conf.2	$BPST_{RI}$ 0.01	0.5050	0.5986	0.9134	0.9909	0.9990
	$BPST_{RP}$ 0.01	0.5022	0.6080	0.9209	0.9916	0.9991
	PST_{RI} 0.01	0.5036	0.5957	0.9138	0.9912	0.9991
	BLA	0.5644	0.7458	0.9365	0.9902	0.9986
Conf.3	$BPST_{RI}$ 0.01	0.5000	0.5437	0.8460	0.9831	0.9982
	$BPST_{RP}$ 0.01	0.5019	0.5753	0.8816	0.9865	0.9986
	PST_{RI} 0.005	0.5015	0.5277	0.7810	0.9759	0.9976
	BLA	0.5459	0.6766	0.8833	0.9799	0.9971
Conf.4	$BPST_{RI}$ 0.05	0.1069	0.3800	0.8816	0.9870	0.9986
	$BPST_{RP}$ 0.05	0.1114	0.4118	0.8873	0.9876	0.9987
	PST_{RI} 0.01	0.1012	0.1970	0.7957	0.9777	0.9978
	BLA	0.1407	0.4187	0.8707	0.9826	0.9978
Conf.5	$BPST_{RI}$ 0.005	0.1003	0.1182	0.4690	0.9289	0.9923
	$BPST_{RP}$ 0.005	0.1005	0.1246	0.4903	0.9326	0.9927
	PST_{RI} 0.005	0.1001	0.1169	0.5123	0.9215	0.9748
	BLA	0.1103	0.1870	0.5492	0.9163	0.9878
Conf.6	$BPST_{RI}$ 0.005	0.1006	0.1088	0.2868	0.8634	0.9843
	$BPST_{RP}$ 0.005	0.1000	0.1154	0.3363	0.8816	0.9820
	PST_{RI} 0.005	0.1000	0.1082	0.3236	0.8490	0.9334
	BLA	0.1075	0.1493	0.3572	0.8347	0.9752

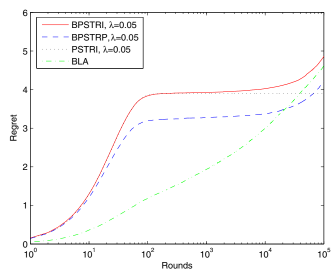
Configurations 5 and 6 are the two most challenging experimental set-ups, in which the superiority of the BPST schemes over the PST$_{RI}$ is more obvious than in previous configurations. As compared with the BLA, the BPST schemes are not as fast as the BLA at the beginning, but again outperforms the 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 action choices, and what would have been obtained by only choosing the optimal action*. 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:

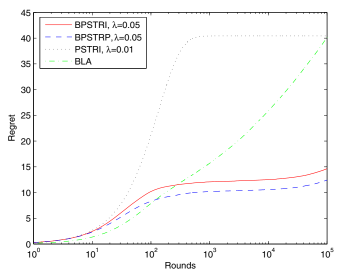
$$d_{opt} \cdot N - \sum_{i=1}^{N} d_i, \tag{3}$$

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

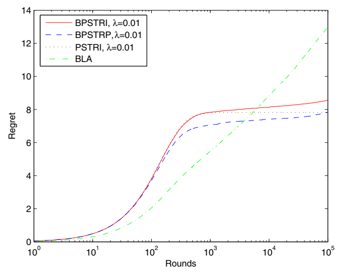
The *Regret* offers the advantage that it does not overly emphasize the importance of choosing the best action. In fact, choosing one of the non-optimal actions will not necessarily affect the overall amount of rewards obtained in a significant manner if,



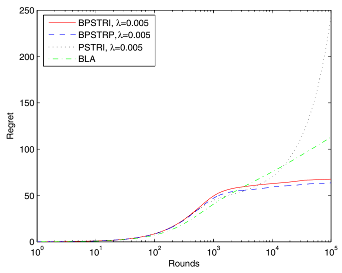
(a) Conf. 1



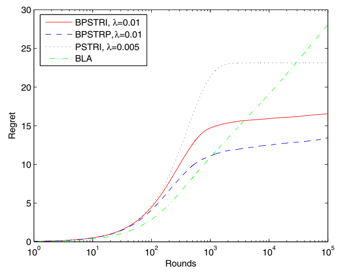
(d) Conf. 4



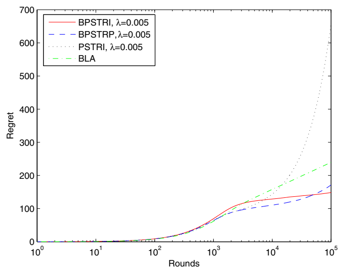
(b) Conf. 2



(e) Conf. 5



(c) Conf. 3



(f) Conf. 6

Fig. 1. The *Regret* of the various LA studied in different configurations

for instance, the reward probability of the non-optimal action is relatively close to the optimal reward probability. The plots in Fig. 1 illustrate the accumulation of the *Regret* of each algorithm with the number of rounds of selecting the various actions.

The left three plots in Fig. 1 display the *Regret* of the algorithms in 2-action experimental configurations. In Configuration 1, the BLA is clearly better than the other three schemes in the early phase of learning. However, by virtue of the simplicity of this configuration, the *Regret*s of all the four schemes are on a low level. In Configurations 2 and 3, as the learning difficulty of the environment increases, the early-phase advantages of the BLA over the other schemes get reduced. Furthermore, the BPST schemes and the PST$_{RI}$ later surpass the BLA with a "flatter" *Regret*. Among the three pursuit schemes, the advantages of the BPST schemes over the PST$_{RI}$ become obvious as the difficulty of the environment increases.

The three plots on the right in Fig. 1 display the *Regret* of the algorithms in the 10-action configurations. As seen, the BPST schemes outperform the PST$_{RI}$. When compared to the BLA, even though it has a slightly higher *Regret* in the early phase, the BPST schemes outperform the BLA with a much "flatter" *Regret* trend later.

It is worth mentioning that the BPST$_{RP}$ performs consistently better than the BPST$_{RI}$ in all the configurations except in the most challenging configuration, i.e., Configuration 6, as seen in both Table 2 and Fig. 1. A possible explanation of the latter observation is the fact that BPST$_{RP}$ updates the action probabilities both upon reward and penalty, which makes learning converge faster than the BPST$_{RI}$, which only updates the action probabilities on being rewarded. Accordingly, when the learning difficulty of Configuration 6 reduces the accuracy of the estimation of the reward probabilities, the performance of the BPST$_{RP}$ will be impacted more than BPST$_{RI}$.

From the above results we draw the following conclusions:

1. The BPST schemes are superior to the PST$_{RI}$, providing better performance in most of the experimental configurations. This indicates that in the field of LA, Bayesian estimates, in general, prove to yield better values than ML estimates.
2. Although the BPST$_{RP}$ performs consistently better than BPST$_{RI}$ in most of the experimental configurations, the BPST$_{RI}$, simultaneously, provides an almost-similar performance. Furthermore, in the most challenging configurations, the BPST$_{RI}$ has a more robust performance than the BPST$_{RP}$, suggesting the former's superiority.
3. By tuning the learning parameter λ, the BPST provides either better or comparable performance compared to the BLA. On the other hand, in several cases, the BLA initially yields a faster performance. This difference in behavior can be explained by their respective distinct strategies for selecting actions. The BLA chooses actions based on the *magnitude* of a random sample drawn from the posterior distribution of the reward estimate, while the BPST chooses actions based on the maintained action 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 the algorithms, where each action is chosen to provide initial reward estimates.

5 Conclusion and Further Work

In this paper we have presented the Bayesian Pursuit Algorithms (BPST), including the *BPST$_{RP}$* and the *BPST$_{RI}$*. The BPST maintains an action probability vector that is used for the selection of the actions. However, it utilizes Bayesian estimates for the reward probabilities associated with each available action, and adopts a linear updating rule for updating the action probability vector. 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 the most promising actions. Thus, to the best of our knowledge, the BPST is the first LA built according to the PST strategy that also takes advantage of a Bayesian estimation scheme. Our reported extensive experimental results demonstrate the advantages of the BPST over the reported best PST scheme. The BPST also provides either comparable or better performance than the BLA by choosing λ suitably.

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

References

1. 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)
2. Thathachar, M.A.L., Sastry, P.S.: Estimator Algorithms for Learning Automata. In: Proc. of the Platinum Jubilee Conference on Systems and Signal Processing, Bangalore, India (December 1986)
3. Granmo, O.-C.: Solving Two-Armed Bernoulli Bandit Problems Using a Bayesian Learning Automaton. International Journal of Intelligent Computing and Cybernetics (IJICC) 3(2), 207–234 (2010)
4. 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: Proc. of International Conference on Agents and Artificial Intelligence, Valencia, Spain, pp. 36–44 (January 2010)
5. Narendra, K.S., Thathachar, M.A.L.: Learning Automata: An Introduction. Prentice-Hall, Englewood Cliffs (1989)
6. Thathachar, M.A.L., Sastry, P.S.: Networks of Learning Automata: Techniques for Online Stochastic Optimization. Kluwer Academic Publishers, Dordrecht (2004)
7. Oommen, B.J., Agache, M.: Continuous and Discretized Pursuit Learning Schemes: Various Algorithms and Their Comparison. IEEE Transactions on Systems, Man, and Cybernetics - Part B: Cybernetics 31(3), 277–287 (2001)
8. Jamei, S.M.: Proposing a New Energy Efficient Routing Protocol for Wireless Sensor Networks. International Journal of Computer Science and Network Security 10(2), 241–245 (2010)
9. Navid, A.H.F.: Scalable and Energy-Aware Learning Automata-Based Routing Protocols for Wireless Sensor Networks. In: Proc. of IEEE International Conference on Sensor Technologies and Applications, Venice/Mestre, Italy, pp. 570–576 (July 2010)
10. Navid, A.H.F., Javadi, H.H.S.: ICLEAR: Energy aware Routing Protocol for WSN using Irregular Cellular Learning Automata. In: Proc. of IEEE Symposium on Industrial Electronics and Applications, Kuala Lumpur, pp. 463–468 (October 2009)
11. Song, Y., Zhang, C., Fang, Y.: Stochastic Traffic Engineering in Multihop Cognitive Wireless Mesh Networks. IEEE Transaction on Mobile Computing 9(3), 305–316 (2010)
12. Gao, F., Zhang, R., Liang, Y.-C., Wang, X.: Design of Learning-Based MIMO Cognitive Radio Systems. IEEE Transaction on Vehicular Technology 59(4), 1707–1720 (2010)
13. Mamaghani, A.S., Mahi, M., Meybodi, M.R.: A Learning Automaton Based Approach for Data Fragments Allocation in Distributed Database Systems. In: Proc. of IEEE International Conference on Computer and Information Technology, Bradford, pp. 8–12 (September 2010)
14. Mitchell, T.M.: Machine Learning. McGraw-Hill, New York (1997)