# Discretized Bayesian Pursuit – A New Scheme for Reinforcement Learning

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

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

**Abstract.** The success of Learning Automata (LA)-based estimator algorithms over the classical, Linear Reward-Inaction ($L_{RI}$)-like schemes, can be explained by their ability to pursue the actions with the highest reward probability estimates. Without access to reward probability estimates, it makes sense for schemes like the $L_{RI}$ to first make large exploring steps, and then to gradually turn exploration into exploitation by making progressively smaller learning steps. However, this behavior becomes counter-intuitive when pursuing actions based on their estimated reward probabilities. Learning should then ideally proceed in progressively *larger* steps, as the reward probability estimates turn more accurate. This paper introduces a new estimator algorithm, the *Discretized Bayesian Pursuit Algorithm* (DBPA), that achieves this. The DBPA is implemented by linearly discretizing the action probability space of the Bayesian Pursuit Algorithm (BPA) [1]. The key innovation is that the linear discrete updating rules mitigate the counter-intuitive behavior of the corresponding linear continuous updating rules, by augmenting them with the reward probability estimates. Extensive experimental results show the superiority of DBPA over previous estimator algorithms. Indeed, the DBPA is probably the fastest reported LA to date.

**Keywords:** Learning Automata, Pursuit Schemes, Bayesian Reasoning, Estimator Algorithms, Discretized Learning.

## 1 Introduction

Learning Automata (LA) have been widely studied as a "bare bones" model of reinforcement learning. The fastest LA algorithms to date are members of the family of estimator algorithms, initially pioneered by the Pursuit Algorithm (PA) [2], and more recently by the Bayesian Pursuit Algorithm (BPA) [1]. Both PA and BPA pursue the action currently perceived to be the optimal one. The main difference lies in the strategy used in inferring which action is to be considered "optimal". Whereas the PA maintains Maximum Likelihood (ML) estimates to decide which action is currently the best, the BPA utilizes a Bayesian method to achieve a superior estimation. By providing *optimistic* reward probability estimates, the latter estimation approach leads learning in a relatively "more correct" direction, thus making the learning converge faster. In this paper, we present a new algorithm – the Discretized Bayesian Pursuit Algorithm (DBPA),

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

H. Jiang et al. (Eds.): IEA/AIE 2012, LNAI 7345, pp. 784–793, 2012.

which is implemented by linearly discretizing the action probability space of the BPA. To the best of our knowledge, DBPA is the fastest reported LA to date.

### 1.1 Learning Automata and Their Applications

A Learning Automaton (LA) is an adaptive decision-making unit that learns the best action out of a set of actions, offered by an environment. At each iteration, the LA chooses one action, which is either rewarded or penalized by the environment as a response. The response, in turn, affects which action the LA chooses in the next iteration. The optimal action is defined as the one with the highest probability of being rewarded.

The beauty of an LA is that it learns the optimal action through interaction with the environment, without any prior knowledge about it. The environment can be treated as a "black box". Indeed, independent of the intricacies exhibited by the behavior of the environment, the LA, considering only the responses it receives, learns to choose the best action, and adapts itself to the environment.

Learning criteria for evaluating an LA involve two basic aspects. The first of these is its accuracy, i.e., to what degree the LA is able to adapt itself to the optimal action. The other is its rate of convergence, i.e., how much time it takes for the LA to converge. In the pursuit of designing accurate and fast LA, a number of algorithms have been both proposed and studied. Initial LA, which had time invariant transition and decision functions are considered to be Fixed Structure Stochastic Automata (FSSA). The Tsetlin, Krylov and Krinsky automata [3] are the most notable examples of this type. Later, Variable Structure Stochastic Automata (VSSA) were developed, which can be completely characterized by the function that updates the probability of choosing the actions [3]. Earlier representatives of this type include the Linear Reward-Penalty ($L_{R-P}$) scheme, the Linear Reward-Inaction ($L_{R-I}$) scheme and Linear Inaction-Penalty ($L_{I-P}$) scheme [3]. Of these, the $L_{R-I}$ scheme is the most accurate and the fastest, as it favors rewards over penalties. Thathachar and Sastry [2] were the first to introduce the concept of Pursuit Algorithms (PA), initiating the research on estimator algorithms. As an estimator algorithm, the PA utilizes the Maximum Likelihood (ML) method for estimating the reward probabilities. More recently, in [4], Granmo presented the Bayesian Learning Automata (BLA), which uses Bayesian reward probability estimation, and subsequently bases its exploration on the Thompson sampling principle [5]. Inspired by the performance benefits of the BLA and the PA, the Bayesian Pursuit Algorithm (BPA) was proposed in [1], following the concept of pursuing the currently-estimated best action, while utilizing Bayesian principles in the estimation itself. In the majority of environments in which LA were tested, the BPA outperformed its previous competitors.

LA have found applications in a variety of fields. They have been used in game playing [6,7], parameter optimization [8,9], vehicle path control [10], channel selection in cognitive radio networks [11], assigning capacities in prioritized networks [12], and resource allocation [13]. LA have also been used in natural language processing, string taxonomy [14], graph patitioning [15], and map learning [16].

### 1.2 Contributions and Paper Organization

In this paper, we propose a new LA algorithm, namely, the Discretized Bayesian Pursuit Algorithm (DBPA). Firstly, the DBPA maintains an action probability vector for

selecting actions. Secondly, it follows the concept of pursuing the action currently inferred to be the "best" . Thirdly, at any time instant, the DBPA uses Bayesian estimates to infer which action is the best. Finally, the DBPA updates its action probability vector according to linear discretized rules. In DBPA, the combination of Bayesian estimation for reward probabilities augmented with linear discretized updates makes learning converge approximately 20% faster than the previously-recorded algorithms including the BPA, as our extensive experimental results demonstrate.

As opposed to our previous paper on the BPA [1], this present paper also analyzes the performance of the BPA in a stricter and more challenging manner, namely, by comparing the performance of the BPA and the PA under their individual optimal learning rates. The advantage of the BPA over the PA thus becomes more evident and persuasive.

The key innovation of this paper is that it points out the incongruity existing in continuous estimator algorithms. Typically, in estimator algorithms, since the estimation of the reward probability is less accurate initially, large changes in the action probabilities at the beginning should be considered as unwise, and almost reckless. Besides, it is also counter-intuitive and unnecessary to reduce the size of the learning steps as the learning proceeds, since as this happens, the reward probability estimates also get more accurate. Unfortunately, continuous estimator algorithms, utilizing linear continuous updating rules for the action probabilities, operate exactly in this manner. The DBPA, on the other hand, uses linear discretized updating rules to mitigate this incongruity.

The paper is organized as follows. In Section 2, we give an overview of the PA, DPA and BPA, as they are the most related algorithms from the family of estimator algorithms. In Section 3, we analyze the incongruity existing in continuous estimator algorithms, and then present the new LA algorithm - the Discretized Bayesian Pursuit - by discretizing the probability space of the BPA. Section 4 provides extensive experimental results showing the advantages of the DBPA over the BPA, and demonstrates that the BPA is truly superior to PA under their individual optimal learning rates. Finally, Section 5, reports opportunities for further research and submits concluding remarks.

## 2 Related Work

In this section, we briefly review three typical algorithms from the family of estimator algorithms, namely, the Pursuit Algorithm (PA), the Discretized Pursuit Algorithm (DPA), and the Bayesian Pursuit Algorithm (BPA).

The PA, DPA and BPA share a common "pursuit" paradigm of learning – in each iteration, they pursue the action currently perceived to be optimal. Firstly, they maintain an action selection probability vector $P = [p_1, p_2, ..., p_r]$, with $\sum_{i=1}^{r} p_i = 1$ and $r$ being the number of actions. The question of which action is to be selected is decided by sampling from $P$, which is, initially, uniform. Secondly, they maintain a reward probability estimate vector $\hat{D} = [\hat{d}_1, \hat{d}_2, ..., \hat{d}_r]$, with $\hat{d}_i$ $(i = 1, 2, ..., r)$ being the estimate of the reward probability for Action $i$. $\hat{D}$ is updated each time an action is selected and a response of either a reward or a penalty is received. At each iteration, the LA treats the action currently possessing the highest reward probability estimate $\hat{d}$ as the optimal one. Finally, based on the inference of the optimal-action and the response from the environment, these LA update the action probability vector $P$ according to a Linear

Reward-Inaction rule [3]. The result is that the action selection probability of the optimal action increases and the other action probabilities decrease. The updated action selection probabilities, $P$, thus come into effect in the next round of action selection. Subtle differences between the PA, DPA and BPA result in fine performance benefits.

**Pursuit Algorithm:** The PA utilizes the Maximum Likelihood (ML) method to estimate the reward probability of each action. The ML reward probability estimate $\hat{d}_i$ can be calculated as $\hat{d}_i = \frac{W_i}{Z_i}$, with $Z_i$ being the number of times Action $i$ has been selected, and $W_i$ the number of times Action $i$ has been rewarded.

The PA updates the action probabilities according to the linear continuous rules:

- If selecting an action results in a reward:
  $p_j = (1-\lambda) \times p_j$, $j \neq i$
  $p_i = 1 - \sum_{j \neq i} p_j$, where $i$ is the index of the action with the largest element in $\hat{D}$.
- If selecting an action results in a penalty: $\forall j, p_j = p_j$.

**Discretized Pursuit Algorithm:** The DPA is implemented by following the paradigm of the PA with the action probability space being discretized. If we denote the distance between two neighbor states as $\delta$, the DPA updates the $\{p_i\}$ as per the discretized rules:

- If selecting an action results in a reward:
  $p_j = max\{p_j - \delta, 0\}$, $j \neq i$
  $p_i = 1 - \sum_{j \neq i} p_j$, where $i$ is the index of the action with the greatest element in $\hat{D}$.
- If selecting an action results in a penalty: $\forall j, p_j = p_j$.

**Bayesian Pursuit Algorithm:** The BPA also follows the paradigm of a general PA. However, as opposed to using the ML estimates as in the PA, the BPA utilizes Bayesian estimates for reward probability estimation. The Bayesian estimation is based on the *Beta Distribution*, which is the conjugate prior for the Bernoulli distribution. A 95% percentile value of the posterior is chosen as the appropriate estimate. The latter is calculated by means of the respective cumulative distribution $F(x_i; a, b)$:

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

where $a$ and $b$ are two positive parameters determining the shape of *Beta Distribution*.

By virtue of the *Beta Distribution*, the Bayesian estimation is implemented in a computationally simple way.

## 3 Discretized Bayesian Pursuit Algorithm

The initial motivation for discretizing LA was to increase their rate of convergence and to avoid the need for generating real numbers with arbitrary precision [17]. In continuous algorithms, since the action probability is updated by multiplying a constant $1-\lambda$, the probability of selecting the optimal action can *only* be approached asymptotically to unity, but never be *actually* attained. However, in discrete algorithms, a minimum step

size is obtained by discretizing the probability space, and the updating of the action selection probabilities is achieved by subtracting or adding one or a multiple of the step size. If the automaton is close to the end state, the probability of the optimal action will be increased directly to unity with a few more favorable responses.

In this paper, we state another important reason for discretizing *estimator algorithms*. As can be readily understood, in the early learning period, the reward probabilities will be inaccurate due to the small number of samples available. As learning proceeds, however, the number of samples increases, and hence the reward probability estimates become progressively more accurate. Thus, it is unwise to make large action selection probability changes initially, and correspondingly counter-intuitive to update the action selection probabilities with smaller and smaller increments, as more samples are obtained. Unfortunately, in continuous algorithms, action selection probabilities are updated exactly in this non-intuitive manner. Indeed, the size of the update increments varies with the action selection probability vector itself, tending to be greater earlier on, and smaller as the learning progresses. In other words, when the estimation of the reward probabilities cannot reliably discriminate between the actions, "large" changes are made to the action selection probabilities, causing the algorithm to be affected by a "gravitational" pull towards an inferior action. This is tacitly caused by the action selection mechanism, since large action selection probabilities translate into biases towards the corresponding actions. The LA can thus either fail to converge to the best action or invest effort into pursuing an incorrect action, leading to a reduced convergence rate.

Discretized LA are characterized by their discrete action probability space. They are linear if the probability values are spaced equally in the interval $[0,1]$, otherwise, they are called nonlinear [18,19]. In this paper, we study linear discrete algorithms, where the equally divided state space indicates that the change of action selection probabilities is a fixed value. When one compares the "large-to-small" changes in continuous algorithms, the fixed-value of the changes in discrete algorithms are more reasonable as per the accuracy of estimating the reward probabilities.

With the above reasoning in mind, we improve the latest estimator algorithm – the Bayesian Pursuit Algorithm – by discretizing its action probability space. The new algorithm, given below, is the Discretized Bayesian Pursuit Algorithm (DBPA).

**Algorithm:** DBPA
**Parameters:**
$\alpha$: The action selected by LA.
$p_i$: The $i^{th}$ element of the action selection probability vector, $P$.
$a_i, b_i$: The two positive parameters of the *Beta* distribution for Action $i$.
$\hat{d}_i$: The $i^{th}$ element of the Bayesian estimates vector $\hat{D}$, given by the 95% upper bound of the cumulative distribution function of the corresponding *Beta* distribution.
$m$: The index of the maximal component of the reward probability estimates vector $\hat{D}$.
$R$: The response from the environment, where $R = 0$ (reward) or $R = 1$ (penalty).
$\delta$: The minimum step size.

**Initialization:**
1. $p_i(t) = 1/r$, where r is the number of actions.
2. Set $a_i = b_i = 1$. Initialize $a_i$ and $b_i$, by doing Step 1 and Step 2 in "Method" below a small number of times. (i.e., in this paper $10 * r$ times).

**Method:**
**For** t:=1 to N **Do**

1. Pick $\alpha(t)$ randomly as per the action 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 $\hat{d}_i(t)$ for each action $i$ as:
$$\frac{\int_0^{\hat{d}_i(t)} v^{(a_i-1)}(1-v)^{(b_i-1)}dv}{\int_0^1 u^{(a_i-1)}(1-u)^{(b_i-1)}du} = 0.95$$
4. Update the action selection probability vector $P(t+1)$ according to the following linear discretized rule:
   **If** $R(t) = 0$ **Then**
   $p_j(t+1) = max\{p_j(t) - \delta, 0\}$, $j \neq m$
   $p_m(t+1) = 1 - \sum_{j \neq m} p_j(t+1)$.
   **Else**
   $P(t+1) = P(t)$.

**End Algorithm:** DBPA

Briefly speaking, the DBPA maintains an action probability vector $P$ for selecting actions, runs *Bayesian* reward probability estimates to determine the current best action, and updates the action probabilities as per *linear discretized* rules. The combination of Bayesian estimation and linear discretized updating rules allow the LA to converge in a relatively more correct direction, and thus, achieve a higher rate of convergence.

## 4 Experimental Results and Analysis

In this section, we evaluate the computational efficiency of the DBPA by comparing it with the latest estimator algorithm, the BPA, as well as the PA and the DPA mentioned in Section 2. The computational efficiency is characterized by the rate of convergence, i.e., the average number of iterations it takes for an algorithm to converge to the optimal action. Extensive experiments have been conducted based on the experimental configurations listed in Table 1.

### 4.1 Experiment Design

In the experiments considered, Configurations 1, 4 and 7 form the simplest environments, possessing a low reward variance and a large difference between the reward probabilities of the actions. By reducing the differences between the actions, we increase the learning difficulty of the environment. Configurations 2, 5 and 8 achieve this task. The challenge of Configurations 3, 6 and 9 is their high variance combined with the small differences between the actions.

In order to evaluate the algorithms under fair conditions, the experiments were designed by considering the following:

1. To keep the conditions identical, each algorithm sampled all actions ten times each in order to initialize the estimate vector. These extra iterations are also included in the results.
2. As each of the four algorithms depends on an external learning rate, the optimal learning rate has to be found for each algorithm.

3. In each learning experiment, the LA is considered to have converged if the probability of selecting an action is greater than or equal to the threshold 0.999. If the LA converges to the best action, it is considered to have converged correctly.
4. For each algorithm, in each configuration, 750 independent learning experiments were conducted. The optimal learning rate $\lambda$ or $\delta$ is the largest one that achieves 100% accuracy, i.e., all the 750 experiments converge correctly.
5. For these configurations, an ensemble of 100 independent replications with different random number streams was performed to minimize the variance of the optimal learning rate.
6. The algorithms were compared with each other as per their individual optimal learning rates.

**Table 1.** Bernoulli distributed rewards used in 2-action, 4-action and 10-action configurations

| **Config./Actions** | 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 | - | - | - | - | - | - |
| 5 | 0.90 | 0.80 | 0.80 | 0.80 | - | - | - | - | - | - |
| 6 | 0.55 | 0.45 | 0.45 | 0.45 | - | - | - | - | - | - |
| 7 | 0.90 | 0.60 | 0.60 | 0.60 | 0.60 | 0.60 | 0.60 | 0.60 | 0.60 | 0.60 |
| 8 | 0.90 | 0.80 | 0.80 | 0.80 | 0.80 | 0.80 | 0.80 | 0.80 | 0.80 | 0.80 |
| 9 | 0.55 | 0.45 | 0.45 | 0.45 | 0.45 | 0.45 | 0.45 | 0.45 | 0.45 | 0.45 |

### 4.2 Experimental Results

Table 2 represents the average number of iterations that each algorithm required to converge to the best action in 2-action, 4-action and 10-action configurations, respectively.

**Table 2.** The average number of iterations for the LA to converge in different configurations

| **Conf./Alg.** | **PA** | **DPA** | **BPA** | **DBPA** |
|---|---|---|---|---|
| Conf. 1 | 76.6040 | 61.0680 | 52.7465 | 54.0889 |
| Conf. 2 | 891.2050 | 454.1912 | 491.1346 | 361.8910 |
| Conf. 3 | 1890.3317 | 929.5736 | 1007.0200 | 699.3099 |
| Conf. 4 | 165.3456 | 122.4167 | 110.3835 | 98.2777 |
| Conf. 5 | 2110.6593 | 1042.7655 | 981.5411 | 693.5873 |
| Conf. 6 | 4487.0094 | 2238.2541 | 2072.5094 | 1401.9219 |
| Conf. 7 | 479.1508 | 517.0978 | 465.7514 | 510.9999 |
| Conf. 8 | 6138.0661 | 3601.4218 | 3248.8014 | 2711.5099 |
| Conf. 9 | 13154.1980 | 7553.5158 | 5860.3236 | 4998.4137 |

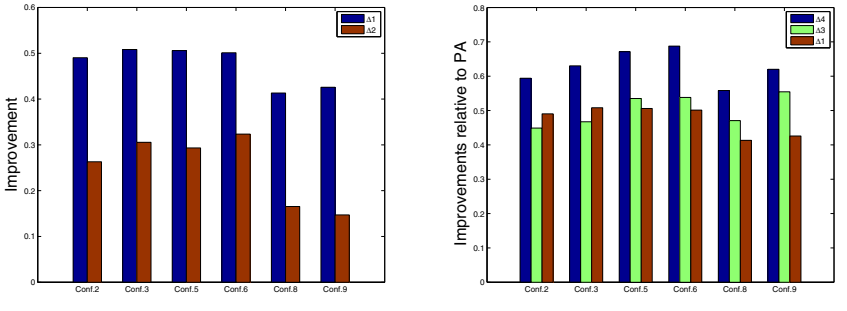
As can be seen from the table, in the simplest configurations of Configurations 1, 4 and 7, the discretized algorithms do not necessarily outperform their continuous counterparts. For example, on the average, the DBPA spent 54 steps to converge to the best action, while the BPA spent only 52 steps. At the same time, it took, on average, 76 steps for the PA to converge correctly, while only 61 steps it took for the DPA to achieve its goal. Besides, discrete algorithms get better results than their continuous counterparts

in 4-action configurations, but in 10-action configurations, continuous algorithms outperform the discrete algorithms. This can be explained by the simplicity of the learning problems associated with the environments. Due to the combination of a low reward variance and a large difference between the reward probabilities of the actions, both ML estimates and Bayesian estimates are able to provide good estimates for the reward probabilities. Therefore, the fixed change in the action probabilities in discrete algorithms yields no evident advantage over the great-to-small changes of action probability in continuous algorithms, and thus the performance of the discrete algorithms is similar to that of their continuous counterparts. Thus, by virtue of the simplicity of the environment, all the four algorithms converge to the best action very quickly.

As a result of the above, our focus is on the relatively more difficult configurations, namely, Configurations 2, 3, 5, 6, 8 and 9. The results in Table 2 show that the DBPA is 26% faster than the BPA in Configuration 2, and 31% faster in Configuration 3. For example, in Configuration 2, the DBPA converges, on the average, in 362 steps, and the BPA needs 491 steps, on average, to converge, which shows an improvement of 26%. In Configuration 3, the DBPA requires, on average, 699 steps for convergence while the BPA requires 1007 steps. The improvement is remarkable, i.e., 31%.

Similarly, the results in Table 2 also present the improvement of the DBPA over the BPA being up to 29% in Configuration 5, 32% in Configuration 6, 17% in Configuration 8 and 15% in Configuration 9. The superiority of the DBPA to its counterpart is clear!

One can also observe that the DPA is obviously faster than the PA in these configurations. If we denote the advantage of the DPA over PA by $\Delta 1$ and the advantage of the DBPA over the BPA by $\Delta 2$, the comparison of $\Delta 1$ and $\Delta 2$ is shown in Fig. 1(a).




(a) Comparison between $\Delta 1$ and $\Delta 2$ (b) Comparison between $\Delta 1$, $\Delta 3$ and $\Delta 4$

**Fig. 1.** The relative improvements of different algorithms

As the reader will observe, $\Delta 1$ is consistently greater than $\Delta 2$ in these configurations. The results are consistent with our statement in Section 3: In estimator algorithms, when the estimation of the reward probabilities is not accurate enough, updating the action probability in a linear discretized manner will be superior than in a linear continuous manner. The reason is as follows. As compared with Bayesian estimates, the PA uses the less accurate ML estimates for the reward probabilities. Therefore, the linear continuous manner of updating action probabilities in the PA is more likely to lead the learning into a wrong direction, resulting in postponed correct convergence. The DPA, on the other

hand, with its linearly discretized action probability space, mitigates the problem caused by linear continuous action probability updating. However, in the BPA, as the Bayesian estimates already are able to provide relatively accurate reward probability estimates, the improvement by discretizing its action probability space becomes less significant.

In order to evaluate the four estimator algorithms in a comprehensive manner, we set the performance of the PA as a benchmark, and compared their individual performance to the PA. Fig. 1(b) demonstrates the performance of the algorithms relative to the PA, where $\Delta 3$ and $\Delta 4$ represent the advantages of the BPA and the DBPA over the PA.

The result of $\Delta 3$ shows clearly that the BPA is superior to the PA, being approximately 50% faster than the latter. The improvement is by virtue of the superiority of Bayesian estimation to ML estimation. The results demonstrated in Fig. 1(b) also show that the DBPA is the best algorithm among the four estimator algorithms, being approximately 60% faster than the PA. The improvement is due to the combination of Bayesian estimation and the linearly discretized action probability space.

Based on the experimental results and analysis, we draw the following conclusions:

1. Considering the average number of steps required to attain the same accuracy of convergence in the listed configurations, the *Discretized Bayesian Pursuit Algorithm* is the best algorithm among the four estimator algorithms.
2. By evaluating the performance of the algorithms under their individual optimal learning rates, the superiority of the Bayesian Pursuit Algorithm over the Pursuit Algorithm is evident and persuasive.
3. In estimator algorithms, updating the action probabilities in a linear discretized manner is more reasonable than in a linear continuous manner. The extensive experimental results warrant this assertion.

## 5 Conclusions and Future Work

LA have been studied for decades and a plenitude of algorithms have been proposed, with the BPA being the most recent and fastest one. Although the BPA is superior to previous estimator algorithms, its action selection probabilities are updated in a linear continuous manner, which is counter-intuitive in the light of the demands for initial caution and later confidence, dictated by the accuracy of the reward probability estimates.

In this paper, we have introduced a new algorithm called the Discretized Bayesian Pursuit Algorithm (DBPA). The DBPA is implemented by discretizing the action selection probabilities of the BPA. Since its estimation for reward probabilities is Bayesian, and since it updates the action selection probabilities in discretized equal-sized steps, more aligned with the accuracy of estimates, the DBPA is able to achieve an even higher rate of convergence than the BPA.

This paper also presented a comprehensive comparison between the four estimator algorithms. Besides, it evaluated the performance of the BPA by comparing it with the PA under fair and reasonable conditions. The results confirm that while the BPA is inferior to the DBPA, it is still superior to the PA. Thus, to the best of our knowledge, the DBPA is the fastest reported LA to date.

A linearly discretized action selection probability space represents an intuitive manner of mitigating the incongruity between linear continuous updating and the mechanisms of reward probability estimation. However, the question of how to mitigate the

incongruity to the largest extent, making updating rules work consistently with the estimation mechanism, remains open. Besides, introducing the state-of-the-art DBPA algorithm to various fields of application also opens promising avenues of research.

## References


1. Zhang, X., Granmo, O.-C., Oommen, B.J.: The Bayesian Pursuit Algorithm: A New Family of Estimator Learning Automata. In: Mehrotra, K.G., Mohan, C.K., Oh, J.C., Varshney, P.K., Ali, M. (eds.) IEA/AIE 2011, Part II. LNCS (LNAI), vol. 6704, pp. 522–531. Springer, Heidelberg (2011)
2. 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)
3. Narendra, K.S., Thathachar, M.A.L.: Learning Automat: An Introduction. Prentice Hall (1989)
4. 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)
5. 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)
6. Lakshmivarahan, S.: Learning Algorithms Theory and Applications. Springer, New York (1981)
7. Oommen, B.J., Granmo, O.C., Asle, P.: Using Stochastic AI Techniques to Achieve Unbounded Resolution in Finite Player Goore Games and its Applications. In: IEEE Symposium on Computational Intelligence and Games 2007, Honolulu, HI (2007)
8. Narendra, K.S., Thathacha, M.A.L.: Learning Automata. Prentice-Hall, Englewood Cliffs (1987)
9. Beigy, H., Meybodi, M.R.: Adaptation of parameters of BP algorithm using learning automata. In: Sixth Brazilian Symposium on Neural Networks 2000, JR, Brazil (2000)
10. Unsal, C., Kachroo, P., Bay, J.S.: Multiple stochastic learning automata for vehicle path control in an automated highway system. IEEE Trans. on Sys., Man, and Cybern., Part A 29, 120–128 (1999)
11. Song, Y., Fang, Y., Zhang, Y.: Stochastic Channel Selection in Cognitive Radio Networks. In: IEEE Global Telecommunications Conference, Washington DC, USA, pp. 4878–4882 (2007)
12. Oommen, B.J., Roberts, T.D.: Continuous learning automata solutions to the capacity assignment problem. IEEE Trans. on Computers 49, 608–620 (2000)
13. Granmo, O., Oommen, B.J., Myrer, S.A., Olsen, M.G.: Learning automata-based solutions to the nonlinear fractional knapsack problem with applications to optimal resource allocation. IEEE Transactions on Systems, Man, and Cybernetics, Part B 37(1), 166–175 (2007)
14. Oommen, B.J., Croix, T.D.S.: String taxonomy using learning automata. IEEE Trans. on Sys., Man, and Cybern. 27, 354–365 (1997)
15. Oommen, B.J., Croix, T.D.S.: Graph partitioning using learning automata. IEEE Trans. on computers 45, 195–208 (1996)
16. Dean, T., Angluin, D., Basye, K., Engelson, S., Aelbling, L., Maron, O.: Inferring finite automata with stochastic output functions and an application to map learning. Maching Learning 18, 81–108 (1995)
17. Oommen, B.J., Lanctot, J.K.: Discretized pursuit learning automata. IEEE Trans. on Sys., Man, and Cybern. 20, 931–938 (1990)
18. Oommen, B.J.: Absorbing and ergodic discretized two-actio learning automata. IEEE Trans. on Sys., Man, and Cybern. SMC-16, 282–296 (1990)
19. Oommen, B.J., Thathachar, M.A.L.: Discretized reward inaction learning atuomata. Journal of Cybernetics and Information Science, 24–29 (1979)