Appl Intell
DOI 10.1007/s10489-013-0424-x

On incorporating the paradigms of discretization and Bayesian estimation to create a new family of pursuit learning automata

Xuan Zhang · Ole-Christoffer Granmo · B. John Oommen

Abstract There are currently two fundamental paradigms that have been used to enhance the convergence speed of Learning Automata (LA). The first involves the concept of utilizing the estimates of the reward probabilities, while the second involves discretizing the probability space in which the LA operates. This paper demonstrates how *both* of these can be simultaneously utilized, and in particular, by using the family of Bayesian estimates that have been proven to have distinct advantages over their maximum likelihood counterparts. The success of 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 by incorporating both the above paradigms. The DBPA is implemented by linearly discretizing the action probability space of the Bayesian Pursuit Algorithm (BPA) (Zhang et al. in IEA-AIE 2011, Springer, New York, pp. 608–620, 2011). The key innovation of this paper 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. Apart from the rigorous experimental demonstration of the strength of the DBPA, the paper also briefly records the proofs of why the BPA and the DBPA are ϵ-optimal in stationary environments.

Keywords Learning automata · Pursuit schemes · Bayesian reasoning · Estimator algorithms · Discretized learning · ε-optimality

1 Introduction

Learning Automata (LA) has 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 (EAs), initially pioneered by the Pursuit Algorithm (PA) [3], and more recently by the Bayesian Pursuit Algorithm (BPA) [2]. Both the PA and the BPA pursue the action currently perceived to be the optimal one. The main

A preliminary version of some of the results of this paper was presented at IEAAIE-2012, the 25th International Conference on Industrial, Engineering and Other Applications of Applied Intelligent Systems, Dalian, China, in June 2012 [1].

X. Zhang (✉) · O.-C. Granmo
Department of ICT, University of Agder, Grimstad, Norway
e-mail: xuan.z.jiao@gmail.com

O.-C. Granmo
e-mail: ole.granmo@uia.no

B.J. Oommen
School of Computer Science, Carleton University, Ottawa, Canada, K1S 5B6
e-mail: oommen@scs.carleton.ca

B.J. Oommen
University of Agder, Grimstad, Norway

Published online: 02 February 2013

difference, however, lies in the strategy used to infer *which action* is to be considered the "optimal" one. 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 the learning phenomenon in a relatively "more correct" direction, thus making the learning converge faster.

This is the first of the fundamental paradigms that have been used to enhance the convergence speed of LA, namely, to utilize the estimates of the reward probabilities. The second philosophy which has been reported since the 1980s, involves discretizing the probability space in which the LA operates. This paper demonstrates how *both* of these can be simultaneously utilized, and in particular, by using the abovementioned family of Bayesian estimates that have distinct advantages over their ML counterparts.

In this paper, we present a new algorithm—the Discretized Bayesian Pursuit Algorithm (DBPA), which is implemented by linearly discretizing the action probability space of the BPA. To the best of our knowledge, the DBPA is the fastest reported LA to date, which has been demonstrated here by rigorous experimentation. Besides this, the paper also briefly records the proofs of why the BPA and the DBPA are ϵ-optimal in stationary environments.

1.1 Learning automata: concept 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. By way of example, consider the Goore Game [4–6], which is a voting game that consists of N voters and one referee, with the referee conducting a series of voting rounds. On each round of voting, the voters vote "Yes" or "No" *simultaneously* and *independently* and the referee counts the number of "Yes" votes. The referee has a unimodal performance criterion $g(\theta)$ over the closed interval $[0, 1]$, and is optimized when the fraction of "Yes" votes is exactly $\theta^\star$. At the end of each voting round, the referee gives to every voter, *independently*, a reward with probability $g(\theta)$ and a penalty with probability $1 - g(\theta)$. Each voter, modeled as an LA, adjusts the voting strategy and votes "Yes" or "No", based *only* on its individual rewards and penalties. We emphasize that in doing this, it does not possess *any* prior information on the criterion of the referee, nor does it communicate with the other voters. It can be seen that, amazingly enough, after enough number of trials, the fraction of "Yes" votes within a round converges to the unknown value of $\theta^\star$, which optimizes the reward probability $g(\theta)$ *for the entire set of voters*.

Because of the "magical" nature of *learning*, LA have found numerous applications in a variety of fields. They have been used in game playing [4–10], parameter optimization [11, 12], channel selection in cognitive radio networks [13], assigning capacities in prioritized networks [14], solving knapsack problems [15], optimizing the web polling problem [16, 17], stochastically optimally allocating limited resources [15, 18, 19], service selection in stochastic environments [20], numerical optimization [21], web crawling [22], microassembly path planning [23], multiagent learning [24], and in batch sequencing and sizing in just-in-time manufacturing systems [25]. An asynchronous action-reward learning has been used for nonstationary serial supply chain inventory control [26].

1.2 Outline of the classification of learning automata

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, and most of the learning automata have been reviewed by Lakshmivarahan [27], and by Narendra and Thathachar [5, 28]. In this paper, a rough classification of different LA will be outlined, including these reviewed LA and the most recent LA.

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 [5] are the most notable examples of this type. Later, Variable Structure Stochastic Automata (VSSA) were developed, with either the transition function, decision function, or both functions evolving with time. In fact, VSSA can be completely characterized by the function that updates the probability of choosing the actions [5]. Earlier representatives of this type include the Linear Reward-Penalty (L_{R-P}) scheme, the Linear Reward-Inaction (L_{R-I}) scheme and the Linear Inaction-Penalty (L_{I-P}) scheme [5]. Of these, the L_{R-I} scheme is the most accurate and the fastest, as it favors rewards over penalties.

Thathachar and Sastry [3, 29] were the first to introduce the concept of Pursuit Algorithms (PA), initiating the re-

search on estimator algorithms [30, 31]. As opposed to non-estimator algorithms, where the action probabilities are directly updated based on rewards/penalties, the estimator algorithms combine running estimates of reward probabilities to make the updating more goal-directed. That is, the reward from the chosen action does not necessarily increase its action probability, but the action probability associated with the largest reward probability estimate will be increased. Therefore, the *current* rewards *alone* will not influence the learning progress in the short term, except by modifying the estimate of the reward vector. Because estimator algorithms consider both the short-term responses of the Environment and the long-term reward probability estimates in formulating the action probability updating rules, they significantly outperform non-estimator schemes in terms of accuracy and rate of convergence.

As an estimator algorithm, the PA utilizes ML method for estimating the reward probabilities, and pursues the current best action (the one with the greatest reward probability estimate) by increasing its action probability, in each iteration. More recently, in [32], Granmo presented the Bayesian Learning Automata (BLA), which uses Bayesian reward probability estimation, and subsequently bases its exploration on the Thompson sampling principle [33]. Inspired by the performance benefits of the BLA and the PA, the Bayesian Pursuit Algorithm (BPA) was proposed in [2], 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 the LA were tested, the BPA outperformed its previous competitors.

After Thathachar and Oommen proposed the *Discretized* Reward-Inaction LA [34], the principle of discretizing the action probability space became a general way for improving the rate of convergence of various LA. The concept of discretization is implemented by restricting the probability of choosing an action to only finitely many values from the interval [0,1]. The discrete counterpart of the PA is called the Discretized Pursuit Algorithm (DPA) [35, 36], and the DPA has been shown to be superior to the PA.

In this paper, as alluded to earlier, we shall incorporate this paradigm into the design of the new algorithm i.e., the DBPA.

We now present one more fundamental contribution of this paper. The most difficult part in the design and analysis of LA consists of the formal proofs of their convergence accuracies.[1] The mathematical techniques used for the various families (FSSA, VSSA, Discretized, etc.) are quite distinct. The proof methodology for the family of FSSA is the simplest: it quite simply involves formulating the Markov chain for the LA, computing its equilibrium (or steady state) probabilities, and then computing the asymptotic action selection probabilities. The proofs of convergence for VSSA are more complex and involve the theory of small-step Markov processes, distance diminishing operators, and the theory of Regular functions. The proofs for Discretized LA involve the asymptotic analysis of the Markov chain that represents the LA in the discretized space, whence the total probability of convergence to the various actions is evaluated. However, understandably, the most difficult proofs involve the family of EAs. This is because the convergence involves two intertwined phenomena, namely the convergence of the reward estimates and the convergence of the action probabilities themselves. Ironically, the combination of these vectors in the updating rule is what renders the EA fast. However, if the accuracy of the estimates are poor because of inadequate estimation (i.e., if the sub-optimal actions are not sampled "enough number of times"), the convergence accuracy can be diminished. Hence the dilemma!

This paper also presents a sketch of the formal analysis concerning why the BPA and DBPA are ϵ-optimal.

1.3 Contributions and paper organization

In this paper, we extend the BPA into the domain of discretization, and propose a new Bayesian estimator algorithm, namely, the Discretized Bayesian Pursuit Algorithm (DBPA) [1]. 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 the 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 [2], 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.

One of the key innovations 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

[1] We refer the readers to [5] and the various papers about the families of estimator algorithms to understand the basic fundamentals of the convergence analysis for various LA.

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.

Besides, this paper states that both the BPA and the DBPA are ϵ-optimal in stationary environments. However, since these proofs are extremely lengthy, in the interest of brevity, we provide here only a sketch of the proof.

The paper is organized as follows. In Sect. 2, we give an overview of the PA, DPA and BPA, as they are the most related algorithms from the family of EAs. In Sect. 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 expands on the ϵ-optimality of the BPA and the DBPA, and outlines the proofs of these claims. Extensive experimental results provided in Sect. 5 show the advantages of the DBPA over the BPA, and demonstrate that the BPA is truly superior to the PA under their individual optimal learning rates. Finally, Sect. 6, 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 EAs, namely, the Pursuit Algorithm (PA), the Discretized Pursuit Algorithm (DPA), and the Bayesian Pursuit Algorithm (BPA).

The PA, the DPA and the 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 $P = [p_1, p_2, \ldots, p_r]$, 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 $\hat{D} = [\hat{d}_1, \hat{d}_2, \ldots, \hat{d}_r]$, with $\hat{d}_i$ $(i = 1, 2, \ldots, r)$ 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 [5]. 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:
  $p_j = (1 - \lambda) \times p_j$, $j \neq i$, $\lambda \in (0, 1)$;
  $p_i = 1 - \sum_{j \neq i} p_j$, 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 neighboring permitted probability values to be Δ, 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 largest 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 pair of hyperparameters, denoted as a_i and b_i, are maintained to count the rewards and penalties received by each action. These parameters, in turn, determine the shape of the *Beta Distribution* and produce the following probability density function:

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

A 95 % percentile value of the posterior is chosen as the appropriate estimate. The latter is calculated by means of the respective cumulative distribution $F(x_i; a, b)$:

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

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 generat-

ing real numbers with arbitrary precision [35]. 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 can 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 [37]. 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
Notations of DBPA: Refer to Table 5 in the Appendix.
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 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, and 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 Convergence of the BPA and the DBPA

Most of the analysis of the convergence of LA is centered around their ϵ-optimality. An LA is said to be ϵ-optimal, if given a sufficiently small learning parameter, it is able to converge to the optimal action with an arbitrarily large probability. We state that both the BPA and the DBPA are ϵ-optimal in all stationary environments. The proof of this statement involves many mathematical details, but due to the lack of space, only a sketch of the proof will be given in this section.

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

The formal claims of the ϵ-optimality of the BPA and the DBPA can be described by Theorems 1 and 2 respectively. In both of these theorems, 't' is measured in terms of the number of iterations.

Theorem 1 *Given any $\epsilon > 0$ and $\delta > 0$, there exist a $\lambda^\star > 0$ and a $t_0 < \infty$ such that for all time $t \geq t_0$ and for any positive learning parameter $\lambda < \lambda^\star$,*

$$\Pr\{p_m(t) > 1 - \epsilon\} > 1 - \delta.$$

Theorem 2 *Given any $\epsilon > 0$ and $\delta > 0$, there exist an $N_0 > 0$ and a $t_0 < \infty$ such that for all time $t \geq t_0$ and for any positive learning parameter $N > N_0$,*

$$\Pr\{p_m(t) > 1 - \epsilon\} > 1 - \delta.$$

The details of the proofs consists of the following three steps.

1. Given a sufficiently small/large value for the learning parameter λ/N, we first prove that all actions will be selected enough number of times before a finite time instant, t_0.

   The proof for this step is the same as that for the PA and the DPA. The reader can refer to [3, 36] and [38] to find the details of this proof.
2. The second step of the proof confirms that the sequence of probabilities, $\{p_m(t)_{(t>t_0)}\}$, is a submartingale.

   This step can be proved by showing that after time instant t_0,

   $$|p_m(t)| \leq 1 \quad \text{and}$$
   $$\Delta p_m(t) = E\big[p_m(t+1) - p_m(t) | \bar{G}(t_0)\big] > 0.$$

   In the above equations, $\bar{G}(t_0)$ is the necessary condition for $\{p_m(t)_{(t>t_0)}\}$ to be a submartingale. $\bar{G}(t_0)$ requires that $\forall t > t_0$, $\Pr\{\hat{d}_m(t) > \hat{d}_j(t), \forall j \neq m\} > \delta'$, with probability 1. Briefly speaking, because by the time instant t_0, each action has been selected a sufficiently large number of times, the estimation for each reward probability after t_0 is accurate enough. Consequently, after t_0, the probability of the optimal action being on top of the ranking of actions will almost always be greater than δ'.
3. The final step proves the ϵ-optimality by showing $\Pr\{p_m(\infty) = 1\} \to 1$.

   Based on the conclusion from the second step, that $\{p_m(t)_{(t>t0)}\}$ is a submartingale, according to the submartingale convergence theory [5], $p_m(t)$ converge to either 0 or 1. The goal in this step is to prove that the convergence probability $\Pr\{p_m(\infty) = 1\} \to 1$ as $\lambda/\Delta \to 0$. This result can be proved by using the theory of Regular functions [5].

   The detailed proofs for the second and the third steps are extremely lengthy. They are omitted here due to the lack of space.

5 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 Sect. 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 and Table 2.

5.1 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.

Table 2 Bernoulli distributed rewards used in benchmark configurations

Config./Actions	1	2	3	4	5	6	7	8	9	10
10	0.70	0.50	0.30	0.20	–	–	–	–	–	–
11	0.10	0.45	0.84	0.76	–	–	–	–	–	–
12	0.70	0.50	0.30	0.20	0.40	0.50	0.40	0.30	0.50	0.20
13	0.10	0.45	0.84	0.76	0.20	0.40	0.60	0.70	0.50	0.30

Table 3 The average number of iterations for the PA, DPA, BPA and DBPA to converge in Configurations 1–9

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

Table 4 The average number of iterations for the LA to converge in benchmark configurations

Conf./Alg.	PA	DPA	BPA	DBPA
Conf. 10	654.220	337.974	276.675	239.539
Conf. 11	3155.000	1566.020	1412.750	1085.100
Conf. 12	1876.370	1060.710	826.257	622.820
Conf. 13	7645.190	3854.640	2753.300	2303.740

Configurations 10, 11, 12 and 13 are the environments which have been earlier used as benchmarks in the field of LA.

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 the PA, DPA, BPA and the new algorithm, DBPA, depend on an external learning parameter, the optimal learning parameter has to be found for each of the four algorithms.
3. In each learning experiment, the LA is considered to have converged if the probability of selecting one 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 λ or Δ is the largest/smallest 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 were 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.

5.2 Experimental results

Table 3 and Table 4 represent the average number of iterations that each of the four algorithms require to converge to the best action in different configurations.

As can be seen from the tables, 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

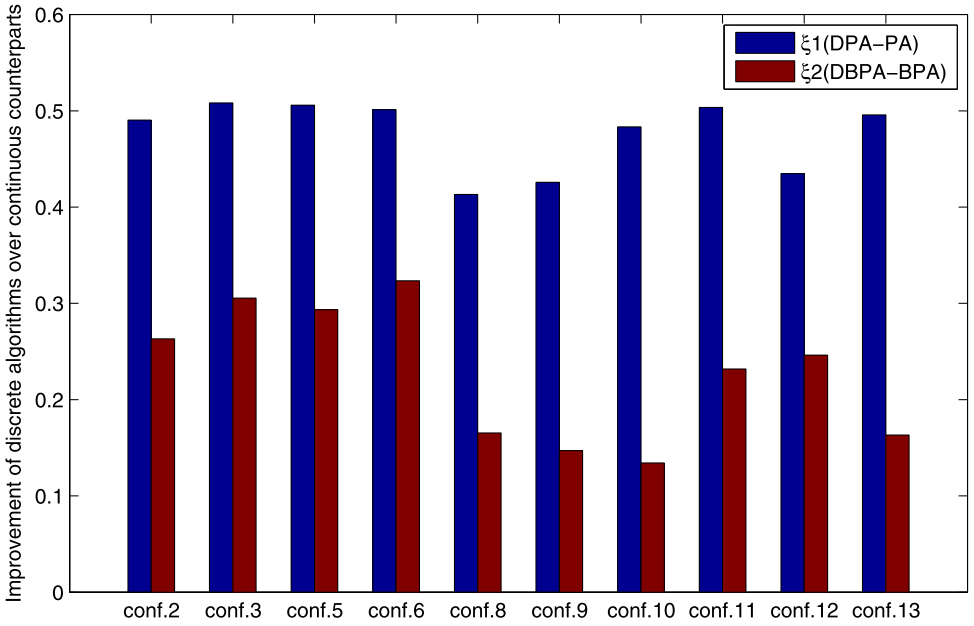

Fig. 1 The improvement of the DPA over the PA versus the improvement of the DBPA over the BPA

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 probabilities in the 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 and the benchmark Configurations 10, 11, 12 and 13. The results in Table 3 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 3 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 results in Table 4 show that the improvement of the DBPA over the BPA in benchmark configurations are 13 % in Configuration 10, 23 % in Configuration 11, 25 % in Configuration 12 and 16 % in Configuration 13. 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 $\xi 1$ and the advantage of the DBPA over the BPA by $\xi 2$, the comparison of $\xi 1$ and $\xi 2$ is shown in Fig. 1.

As the reader will observe, $\xi 1$ is consistently greater than $\xi 2$ in these configurations. The results are consistent with our statement in Sect. 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 are already 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. Figure 2 demonstrates the performance of the al-

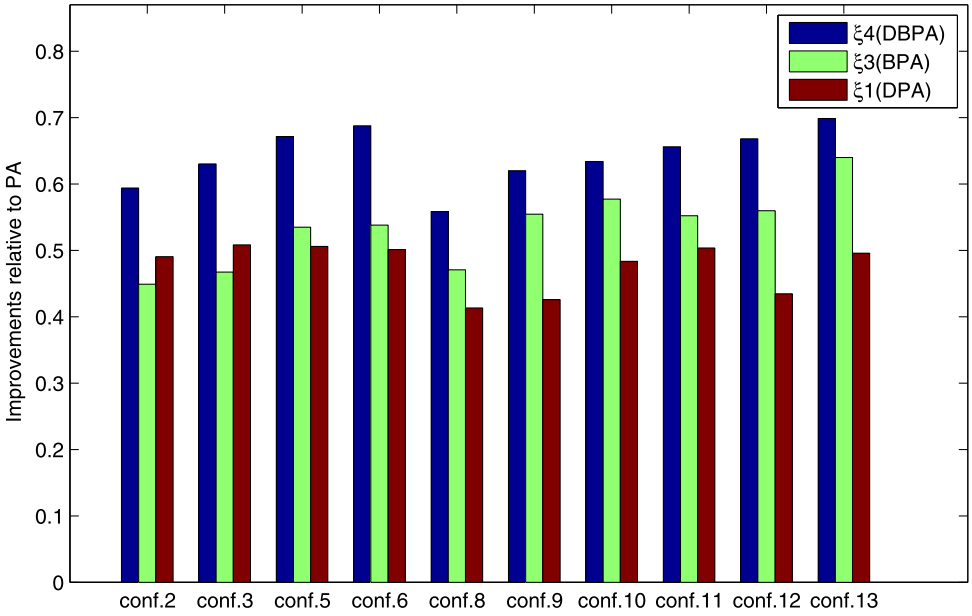

Fig. 2 The improvements of the DPA, the BPA and the DBPA over the PA

gorithms relative to the PA, where $\xi 3$ and $\xi 4$ represent the advantages of the BPA and the DBPA over the PA.

The result of $\xi 3$ 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. 2 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 algorithm that is newly proposed here, the *Discretized Bayesian Pursuit Algorithm* is the best algorithm among the five 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.

6 Conclusions and future work

LA have been studied for decades and a plentitude 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.

Besides, the BPA and the DBPA are ϵ-optimal in all stationary environments. A sketch of the proof has been given in this paper.

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.

Appendix

Table 5 Notations used in DBPA

Parameters	Descriptions
α	The action selected by LA.
p_i	The ith 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 ith 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).
Δ	The minimum step size. $\Delta = \frac{1}{rN}$, with N being a positive integer.

References

1. Zhang X, Granmo O-C, Oommen BJ (2012) Discretized Bayesian pursuit—a new scheme for reinforcement learning. In: IEA-AIE 2012, Dalian, China, Jun 2012, pp 784–793
2. Zhang X, Granmo O-C, Oommen BJ (2011) The Bayesian pursuit algorithm: a new family of estimator learning automata. In: IEA-AIE 2011. Springer, New York, pp 608–620
3. Thathachar M, Sastry P (1986) Estimator algorithms for learning automata. In: The platinum jubilee conference on systems and signal processing, Bangalore, India, Dec 1986, pp 29–32
4. Tsetlin M (1963) Finite automata and the modeling of the simplest forms of behavior. Usp Mat Nauk 8:1–26
5. Narendra KS, Thathachar MAL (1989) Learning automat: an introduction. Prentice Hall, New York
6. Thathachar M, Arvind M (1997) Solution of goore game using models of stochastic learning automata. J Indian Inst Sci 76:47–61
7. Oommen BJ, Granmo O-C, Pedersen A (2006) Empirical verification of a strategy for unbounded resolution in finite player goore games. In: The 19th Australian joint conference on artificial intelligence, Hobart, Tasmania, Dec 2006, pp 1252–1258
8. Oommen BJ, Granmo O-C, Pedersen A (2007) Using stochastic AI techniques to achieve unbounded resolution in finite player goore games and its applications. In: IEEE symposium on computational intelligence and games, Honolulu, HI Apr 2007
9. Granmo O-C, Glimsdal S (2012, to appear) Accelerated Bayesian learning for decentralized two-armed bandit based decision making with applications to the goore game. Appl Intel
10. Granmo O-C, Oommen BJ, Pedersen A (2012) Achieving unbounded resolution in finite player goore games using stochastic automata, and its applications. Seq Anal 31:190–218
11. Narendra MAL, Thathacha KS (1987) Learning automata. Prentice-Hall, Englewood Cliffs
12. Beigy H, Meybodi MR (2000) Adaptation of parameters of BP algorithm using learning automata. In: Sixth Brazilian symposium on neural networks. JR, Brazil, Nov 2000
13. Song Y, Fang Y, Zhang Y (2007) Stochastic channel selection in cognitive radio networks. In: IEEE global telecommunications conference, Washington, DC, USA, Nov 2000, pp 4878–4882
14. Oommen BJ, Roberts TD (2000) Continuous learning automata solutions to the capacity assignment problem. IEEE Trans Comput 49:608–620
15. Granmo O-C, Oommen BJ, Myrer S-A, Olsen MG (2007) Learning automata-based solutions to the nonlinear fractional knapsack problem with applications to optimal resource allocation. IEEE Trans Syst Man Cybern, Part B, Cybern 37(1):166–175
16. Granmo O-C, Oommen BJ, Myrer S-A, Olsen MG (2006) Determining optimal polling frequency using a learning automata-based solution to the fractional knapsack problem. In: The 2006 IEEE international conferences on cybernetics and intelligent systems (CIS) and robotics, automation and mechatronics (RAM), Bangkok, Thailand, Jun 2006, pp 1–7
17. Granmo O-C, Oommen BJ (2011) Learning automata-based solutions to the optimal web polling problem modeled as a nonlinear fractional knapsack problem. Eng Appl Artif Intell 24(7):1238–1251
18. Granmo O-C, Oommen BJ (2006) On allocating limited sampling resources using a learning automata-based solution to the fractional knapsack problem. In: The 2006 international intelligent information processing and web mining conference, advances in soft computing, vol 35. Ustron, Poland, Jun 2006, pp 263–272
19. Granmo O-C, Oommen BJ (2010) Optimal sampling for estimation with constrained resources using a learning automaton-based solution for the nonlinear fractional knapsack problem. Appl Intell 33(1):3–20
20. Yazidi A, Granmo O-C, Oommen BJ (2012) Service selection in stochastic environments: a learning-automaton based solution. Appl Intell 36:617–637
21. Vafashoar R, Meybodi MR, Momeni AAH (2012) CLA-DE: a hybrid model based on cellular learning automata for numerical optimization. Appl Intell 36:735–748
22. Torkestani JA (2012) An adaptive focused web crawling algorithm based on learning automata. Appl Intell 37:586–601
23. Li J, Li Z, Chen J (2011) Microassembly path planning using reinforcement learning for improving positioning accuracy of a 1 cm^3 omni-directional mobile microrobot. Appl Intell 34:211–225
24. Erus G, Polat F (2007) A layered approach to learning coordination knowledge in multiagent environments. Appl Intell 27:249–267
25. Hong J, Prabhu VV (2004) Distributed reinforcement learning control for batch sequencing and sizing in just-in-time manufacturing systems. Appl Intell 20:71–87
26. Kim CO, Kwon I-H, Baek J-G (2008) Asynchronous action-reward learning for nonstationary serial supply chain inventory control. Appl Intell 28:1–16
27. Lakshmivarahan S (1981) Learning algorithms theory and applications. Springer, New York
28. Narendra KS, Thathachar MAL (1974) Learning automata–a survey. IEEE Trans Syst Man Cybern 4:323–334
29. Thathachar MAL, Sastry PS (1985) A class of rapidly converging algorithms for learning automata. IEEE Trans Syst Man Cybern SMC-15:168–175
30. Sastry PS (1985) Systems of learning automata: Estimator algorithms applications. PhD thesis, Dept Elec Eng, Indian Institute of Science
31. Thathachar MAL, Sastry PS (1984) A new approach to designing reinforcement schemes for learning automata. In: IEEE int conf cybern syst, Bombay, India, Jan 1984, pp 1–7
32. Granmo O-C (2010) Solving two-armed Bernoulli bandit problems using a Bayesian learning automaton. Int J Intel Comput Cybern 3(2):207–234
33. Thompson WR (1933) On the likelihood that one unknown probability exceeds another in view of the evidence of two samples. Biometrika 25:285–294
34. Thathachar MAL, Oommen BJ (1979) Discretized reward-inaction learning automata. J Cybern Inf Sci, 24–29

35. Oommen BJ, Lanctot JK (1990) Discretized pursuit learning automata. IEEE Trans Syst Man Cybern 20:931–938
36. Oommen BJ, Agache M (2001) Continuous and discretized pursuit learning schemes: various algorithms and their comparison. IEEE Trans Syst Man Cybern, Part B, Cybern 31(3):277–287
37. Oommen BJ (1990) Absorbing and ergodic discretized two-action learning automata. IEEE Trans Syst Man Cybern SMC-16:282–296
38. Rajaraman K, Sastry PS (1996) Finite time analysis of the pursuit algorithm for learning automata. IEEE Trans Syst Man Cybern, Part B, Cybern 26:590–598

Xuan Zhang obtained her B.Tech. degree from Shandong Univerisity, Jinan, China, in 2008. She obtained her M.E. from Hunan University in Hunan province, China in 2005. She is now working as a Ph.D. research fellow in the University of Agder, Norway. Her research interests include Machine Learning, Learning Automata, and Stochastic Modeling and Optimization.

Ole-Christoffer Granmo was born in Porsgrunn, Norway. He obtained his M.Sc. in 1999 and the Ph.D. degree in 2004, both from the University of Oslo, Norway. He is currently a Professor in the Department of ICT, University of Agder, Norway. His research interests include Intelligent Systems, Stochastic Modelling and Inference, Machine Learning, Pattern Recognition, Learning Automata, Distributed Computing, and Surveillance and Monitoring. He is the author of more than 70 refereed journal and conference publications.

B. John Oommen was born in Coonoor, India on September 9, 1953. He obtained his B.Tech. degree from the Indian Institute of Technology, Madras, India in 1975. He obtained his M.E. from the Indian Institute of Science in Bangalore, India in 1977. He then went on for his M.S. and Ph.D. which he obtained from Purdue University, in West Lafayettte, Indiana in 1979 and 1982 respectively. He joined the School of Computer Science at Carleton University in Ottawa, Canada, in the 1981–1982 academic year. He is still at Carleton and holds the rank of a *Full Professor*. Since July 2006, he has been awarded the honorary rank of *Chancellor's Professor*, which is a lifetime award from Carleton University. His research interests include Automata Learning, Adaptive Data Structures, Statistical and Syntactic Pattern Recognition, Stochastic Algorithms and Partitioning Algorithms. He is the author of more than 380 refereed journal and conference publications, and is a *Fellow of the IEEE* and a *Fellow of the IAPR*. Dr. Oommen has also served on the Editorial Board of the *IEEE Transactions on Systems, Man and Cybernetics*, and *Pattern Recognition*.