Empirical Evaluation of the Bayesian Learning Automaton Family

By
Terje Brådland and Thomas Norheim

Thesis submitted in Partial Fulfillment of the
Requirements for the Degree Master of Technology in
Information and Communication Technology

Faculty of Engineering and Science
University of Agder

Grimstad
May 2009

Abstract

The two-armed bandit problem is a classical optimization problem where a player sequentially selects and pulls one of two arms attached to a gambling machine, and each arm pull results in either a *reward* or *penalty* to the player. Each arm is associated with a certain reward probability which is unknown to the player, and the player needs to sequentially select and play an arm and receive a reward or a penalty in order to discover its true reward probability. The overall goal for the player is reward maximization, and the player needs to balance between exploiting existing knowledge or obtaining new knowledge by trying different arms. In the long run it may be beneficial to risk short term loss to gain greater certainty about the reward probability associated with each arm.

The problem as described above is the simplest case of the more general k-armed bandit problem and is concerned with the exploration vs. exploitation dilemma. A wide range of schemes has been proposed to solve the problem, and in this thesis we focus our attention on a series of schemes collectively referred to as the *Bayesian Learning Automaton family*, originally introduced by O-C. Granmo with the Bayesian Learning Automaton deigned for Bernoulli distributed reward.

The Bayesian Learning Automata rely upon Bayesian statistics and the use of conjugate prior distributions with sets of updating rules of hyperparameters associated with these conjugate prior distributions. These updating rules make it possible to apply Bayesian statistics in an automaton in a simple and highly computationally efficient manner.

In this thesis we extend the Bayesian Learning Automaton family with three new members designed for Poisson and normally distributed reward.

We empirically evaluate both the Bayesian Learning Automaton family and competing schemes in the general k-armed bandit problem with Bernoulli, Poisson and normally distributed reward. We also empirically evaluate these players in the Goore game, iterative prisoners' dilemma and two-player zero-sum game from game theory.

Through extensive experiments we show that the Bayesian Learning Automaton family overall outperforms all other comparable learning schemes in the k-armed bandit problem, and the Bayesian Learning Automata are among the top performers in all the the games they are introduced to in this thesis. Thus, we believe that the Bayesian Learning Automaton family is an important addition to the field of bandit playing algorithms and is an important area for further research.

Preface

This thesis is submitted in partial fulfillment of the requirements for the degree master of technology in Information and Communication Technology at the University of Agder, Faculty of Engineering and Science, under the supervision of associate professor Ole-Christoffer Granmo.

We wish to thank Ole-Christoffer Granmo for great support during our work with this master thesis. Granmo has been a great inspiration to us during this period, and his knowledge in the field of learning automata has been invaluable.

Grimstad, May 2009

Terje Brådland Thomas Norheim

Contents

List of Figures

List of Tables

List of Algorithms

Chapter 1

Introduction

In the following chapter we will briefly introduce the k-armed bandit problem and its importance. Furthermore, we present the area of focus in thesis and the contributions of this thesis to the field of reinforcement learning.

We start by introducing the importance of research in Section 1.1, where we state the importance of the problem and the importance of our contributions. Section 1.2 contains the full thesis definition. In Section 1.3 we state our research questions and briefly comment their problems and importance, and follow up with a presentation of our contributions in Section 1.4. We end this chapter with a section with the intended target audience of this thesis in Section 1.5, and present an overview of the thesis in Section 1.6.

1.1 Importance of research

In reinforcement learning one faces the exploration vs. exploitation dilemma. This dilemma involves searching for a balance between discovering new information and exploiting existing knowledge. The k-armed bandit problem is a well-known and highly researched example of this problem.

The simplest case of the k-armed bandit problem is the two-armed Bernoulli bandit problem, which may be described as follows. A player sequentially pulls one of two arms attached to a gambling machine, and each arm pull results in either a reward or a penalty. The two arms are associated with two distinct, and to the player unknown, reward probabilities. The overall goal for the player is reward maximization, and the player then needs to balance between exploiting existing knowledge or obtaining new knowledge by trying different arms. In the long run it may be beneficial to risk a short term loss in order to build greater certainty about the unknown reward probabilities.

The k-armed bandit problem is a highly applicable problem found in everyday life. For instance, in computer-networks and routing there may exist several paths between nodes, and a path is ordinary selected by using traditional routing algorithms which may take delay, bandwidth, cost and other parameters into consideration in the selection process. In [1] learning schemes are applied to routing in MPLS networks with a new proposed learning routing algorithm. The performance of this algorithm was experimentally shown to be better than the most important routing algorithms in the literature. In relation to the bandit problem, each path may be regarded as an arm with initially unknown reward probability.

Several solutions to the bandit problem have been proposed, among them several confidence interval and learning automata schemes, which have proven to be effective with regards to machine learning in several important areas [2, 3, 1, 4].

In this master thesis we investigate the performance of a new Bayesian approach to the k-armed bandit problem, the Bayesian Learning Automaton (BLA). In [5] Granmo shows through experiments that the two-armed version of the BLA has some advantages over confidence interval based algorithms and traditional learning automata.

The BLA as proposed by Granmo is the main starting point for this thesis. The experiments performed in [5] seem promising, but extensive experiments are only performed with the 2-armed bandit problem with Bernoulli distributed reward. These experiments need to be extended in order to get a more complete picture of the performance and properties of the BLA.

We also propose new members to the Bayesian Learning Automaton family suitable for use in environments with Poisson distributed reward, as well as normally distributed reward with both known and unknown variance. These distributions are widely used within statistics, and the normal distribution especially has a general applicability, making it suitable in a wide range of applications.

The results of our experiments provide insight into advantages, disadvantages and properties of the new proposed Bayesian Learning Automata family. Such knowledge is a contribution to the field of reinforcement learning, and is valuable knowledge regardless of the performance of the BLA. This may result in additional research on the BLA, and possibly lead to improved performance in a wide range of real world applications.

1.2 Thesis definition

The thesis definition is as follows:

> *The k-armed bandit problem is a well studied example of the exploration vs. exploitation dilemma. This dilemma involves the balance between exploring new information to identify profitable actions and exploiting existing knowledge, while the overall goal is reward maximization. In the long run it may be beneficial to sacrifice short term reward to gain greater certainty about the environment. In the paper* ***The Bayesian Learning Automaton - Empirical Evaluation with Two-Armed Bernoulli Bandit Problems*** *Granmo proposes a new Bayesian approach, the Bayesian Learning Automata (BLA). Preliminary results presented in Granmo's paper seem very promising, but extensive experiments have only been conducted with the Two-Armed Bernoulli Bandit. The purpose of this master thesis is to investigate whether the performance benefits offered by the Bayesian Learning Automaton scheme extend beyond the simple Two-Armed Bernoulli Bandit Problem. As an option, we also intend to extend the Bayesian Learning Automaton family with new mechanisms where appropriate.*

1.3 Research questions

In this thesis we will answer the following research questions:

- **How does the BLA proposed by Granmo in [5] perform compared to other learning schemes in the k-armed bandit problems**

  The k-armed bandit problem involves finding the optimal balance between exploration and exploitation in an unknown environment, and captures the essence of a problem often encountered in optimization problems. We extend on the experiments already performed by Granmo in [5], and perform extensive empirical evaluation of the BLA in the k-armed bandit problem. In this thesis we introduce three new members to the BLA family, which also are empirically evaluated in the k-armed bandit problem with appropriate reward distributions. We also introduce several "state-of-the-art" players along with other traditional and well-known players as points of reference in the evaluation of the performance of the BLA family.

- **Is a BLA able to solve a multiplayer cooperative game in an efficient manner? How does it perform compared to other players?** We want to investigate how a BLA performs in multiplayer cooperation games, where several players need to cooperate to achieve a common goal without any communication between the players. Studies on such problems have led to improved performance in several real life problems, such as QoS control in sensor networks [6].

  Due to its distributed and cooperative nature, the Goore game is suitable to investigate the performance of the BLA compared to other players in these types of problems.

- **Is a BLA able to find a good solution in a mixed strategy game? If so, does it find a mixed strategy solution or does it converge to a single strategy?**

  In a mixed strategy game it is suboptimal to play a single strategy, and the optimal strategy consists of randomly select among several different single strategies according to a probability distribution.

  For instance, consider the simple and well-known game *Rock-paper-scissors*. In this game it is easy to beat a player that always selects *rock*, as it is beaten by *paper* every time. Therefore, by simply adding some unpredictability in a player's choices, it becomes a lot harder to beat an opponent.

  Mixed strategy is relevant in many important areas in real life as for instance economics, biology, warfare and games [7]. We therefore want to investigate if a BLA is able to identify and play a mixed strategy in a mixed strategy game.

- **How do two BLA players play the iterative prisoner's dilemma? Furthermore, how will the BLA play against other players in this game?**

  The prisoners' dilemma is a game where two opposing players individual pursuit of the optimal reward results in a suboptimal outcome for both players. For instance, in the classical description of the prisoners' dilemma this involves whether or not a prisoner should confess to a crime to receive a reduced sentence at the expense of an accomplice. However, in this dilemma they will both be worse off if they both decide to confess than if both decide not to confess.

  This is a game that has been extensive studied in social science, and the same dilemma is encountered in many situations in social life [7]. In this thesis we will investigate how the BLA players are able to cope with this dilemma compared to other competing learning schemes.

- **Do there exist other potential BLA players than the one already proposed by Granmo?**

  The beta distribution is the conjugate prior for the Bernoulli distribution and is an important part of the BLA designed for Bernoulli distributed reward as presented in [5]. There may exist other convenient conjugate prior distributions for use in BLA players designed for other reward distributions. We therefore investigate the background of the proposed BLA and use the same principles to derive three new members to the BLA family.

1.4 Contributions

In this thesis we extend the Bayesian Learning Automaton (BLA) family originally introduced by Granmo in [5], and introduce three new automata applicable in bandit problems with Poisson and normally distributed reward.

We derive the algorithms of the Bayesian Learning Automata from Bayesian statistics, both for the original automaton proposed by Granmo and the three automata proposed in this thesis.

We perform an extensive empirical evaluation of the BLA family and compare the performance of the BLA family with “state-of-the-art” and other well-known bandit players in the k-armed bandit problem. We also introduce the BLA in selected games from game theory, such as the Goore game, iterative prisoners’ dilemma and two-player zero-sum game, and perform an extensive empirical evaluation of the BLA in these games.

We present the importance and strength of a carefully selected initial state of a Bayesian Learning Automaton, and demonstrate the strength of such a carefully selected state through our empirical evaluation of the BLA family.

Finally, we propose areas for further research with regards to the Bayesian Learning Automaton family in order to be able to extend it with new solutions applicable in new domains. We propose further work both to ease the use and simplify the initialization phase of the BLA, and also propose further work with more complex systems of Bayesian Learning Automata.

1.5 Target audience

The target audience of this thesis is anyone interested in reinforcement learning and the learning automata field. The thesis requires that the reader is familiar with basic concepts from statistics and game theory, but the thesis is written such that is should be possible for any interested reader to follow. A brief introduction to the most important concepts and theory is given, but especially with regards to statistics and probability distributions we assume some background knowledge.

1.6 Report outline

We start Chapter 2 by introducing some general information about reinforcement learning, learning automata, players and environments. In Chapter 3 we introduce selected topics from Bayesian statistics that are crucial to a good understanding of the BLA family. We briefly present some competing learning schemes in Chapter 4, where we provide a brief description of the theoretical background and give an algorithmic description of each scheme.

In Chapter 5 the Bayesian Learning Automaton family is introduced, and we explain how Bayesian statistics is used in the automata to govern action selection. We present the original BLA player designed for Bernoulli distributed reward, and extend the BLA family with three new players designed for Poisson and normally distributed reward.

Important results from the experiments are listed and commented in Chapter 6, with additional data from the experiments available in the appendix. We then provide a discussion and summary of the results in Chapter 7, and end the thesis with a conclusion and suggestions for further work in Chapter 8.

Chapter 2

Reinforcement learning and automata theory

Chapter 1 briefly introduced the k-armed bandit problem and briefly mentioned that the reward from the gambling machine either is Bernoulli, Poisson or normally distributed. In this chapter we therefore present a more general description of the k-armed bandit problem and introduce the term *feedback* instead of *reward*, as a more general term which is less dependent on the characteristics of the arm distributions.

In Section 2.1 we introduce the essence of the k-armed bandit problem, namely that this problem explores the trade-off between exploration and exploitation found in reinforcement learning. In this context we also give a definition of *learning* and introduce *machine learning*, which reinforcement learning is a sub-area of.

With a fundamental understanding of learning in mind, we give a more precise description of a player in the k-armed bandit problem in Sections 2.2 and 2.3, and also introduce *environment* as a more general term instead of the more specific term gambling machine.

In Section 2.4 we present a branch of the machine learning field, called *Learning Automata* (LA), which the Bayesian Learning Automata are based upon. Game theory regards an environment as a game, and in Section 2.5 we introduce selected concepts from game theory, which provide a powerful framework to analyze outcomes of learning problems.

We end this chapter with a more detailed description of the k-armed bandit problem and introduce games that bare resemblance to this problem.

2.1 Learning

The idea of creating intelligent machines has been around for a long time, and gone from just being a distant dream to becoming a reality in the last decades. There has been a substantial amount of development in the field of machine learning [8], but still machines are far from being as intelligent as human beings.

2.1.1 Concepts of learning

Learning has long been considered an important aspect of intelligent behavior and has been studied extensively by psychologists during the last decades. One avenue of research involving psychologists and biologists have conducted learning experiments where subjects are placed in controlled environments, which are manipulated in order to observe possible changes in the behavior of the subjects. By conducting such experiments psychologists and biologists try to create models of learning.

From an engineering perspective, the principles of learning are used to create intelligent machines. Intelligent machines are able to improve their performance by the means of a learning process, and the basic features of the learning process in psychology are applied in this domain. In this context, learning is defined as follows [9]:

> *A machine or system is said to learn if it improves its performance through experience gained over a period of time without complete information about the environment in which it operates.*

A machine, also referred to as a learner in this context, is improving its performance by the means of a learning process. The learning process is a result of interactions between a learner and an external teacher, often called the environment. The learner selects one appropriate action out of several available actions, which is evaluated by the teacher in the form of a feedback to the learner. The learner then, based upon the feedback, updates its knowledge about the environment, which affects its future action selection procedure. This is repeated until some condition is met, for instance until a satisfactory result is achieved.

The way the environment evaluates the action performed by the learner, that is the type of feedback a learner receives, is used to distinguish three sub-areas within machine learning; supervised, unsupervised and reinforcement learning. In this thesis we focus on reinforcement learning, which is the most general of these three in that the learner is not told what to do, but must learn from the feedback it receives [10]. The interested reader is referred to [10] and [8] for more information about machine learning and supervised and unsupervised learning.

2.1.2 Reinforcement learning

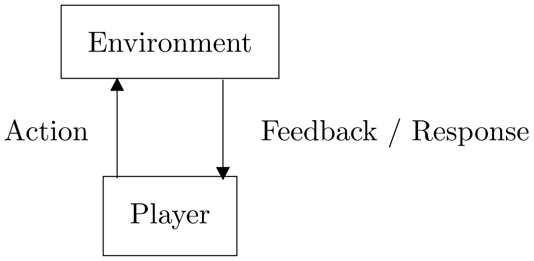
In reinforcement learning the learner is not told what the optimal action is, but only receives a scalar feedback to the selected action and must therefore learn from the feedback, also referred to as reinforcement, obtained from the environment. Thus, in this way the learner is learning how the environment works [10].

High feedback values are more desirable, and the goal of the learner is to maximize its received feedback, thus it is important that it is able to learn and identify the optimal action by performing a sequence of trials. The learner must, therefore, try several different actions in order to identify the optimal action. In addition, the response from the environment may be a stochastic variable yielding a stochastic environment, which means that all actions must be tried a number of times in order to evaluate the mean feedback associated with each action such that it is able to identify the optimal action. In a stochastic environment the feedback will typically be distributed in accordance to a probability distribution, where the parameters of the probability distribution are unknown to the learner. To obtain knowledge about these parameters, the learner performs a sequence of trials, and throughout the learning process the learner will perceive a view of the probability distribution of the feedback and its parameters. The learner will then be able to exploit the obtained knowledge by choosing actions that on average yield high feedback values, and thus it is able to improve its performance.

A common problem in reinforcement learning is the trade-off between exploration, where the learner tries different actions to gain knowledge about them, and exploitation, where the learner selects actions which it believes give favorable feedback. The trade-off between exploration and exploitation is crucial, as a suboptimal trade-off will affect the results, and hence the learning process. Neither pure exploration or pure exploitation is desirable. Pure exploration will not be beneficial as the obtained knowledge is not used and hence not exploited, while pure exploitation will typically result in getting stuck on a suboptimal action [10]. Furthermore, focusing too much on exploitation could lead to premature conclusions concerning the perceived optimal action, which in turn could lead to a loss in favorable future feedback. Too much exploration on the other hand is just as bad, as a lot of action selections and possible favorable feedback may be wasted even though the true optimal action has been discovered. The difficulty of reinforcement learning is typically concerned with this trade-off.

In relation to the exploration vs. exploitation dilemma an optimal exploration policy is desirable. Optimal exploration policies have been extensively studied in statistical decision theory in the so called k-armed bandit problem, which later also have been fundamental in the field of artificial intelligence [10]. Finding an optimal exploration scheme is difficult, but it is possible to find a reasonable scheme that eventually leads to an optimal behavior of the learner [10]. We discuss this further in Section 2.6.1.

Figure 2.1: Interaction between a Player and an Environment



2.2 Bandit player

In the above we used the term *learner* to denote a machine which is able to learn by experience. This definition is quite general and therefore applies to a lot of different techniques and areas within machine learning, and in the following we introduce a term that is more suitable in the context of this thesis.

In this thesis the focus is on evaluation of learning algorithms that are applicable in the *k-armed bandit problem*. Hence the term *bandit playing algorithms* is suitable for such algorithms. Labeling the algorithm does not necessarily cause any problems, but in relation to learning, the algorithm itself is only one part of what we refer to as a *learner*. In the literature commonly used terms for a learning machine are *agent* as used in [10] and *learning automaton* as described in [11]. Although these and others terms capture the essence of learning, they are still not completely compatible with each other mostly due to historical reasons. In this thesis we therefore use the more general term *bandit player*, or *player* for short, to denote a learning machine that is applicable in the k-armed bandit problem.

2.3 Formal model of a bandit player and its environment

In the following section we make a formal model of a bandit player and the environment it operates in. We refer the reader to Section 2.6.1 for a concrete example of a player and an environment.

We start by considering a player that is put in an environment and must learn how to successfully operate in the environment. From the point of view of the player the environment is unknown and player has no prior knowledge of the environment. In order to gain knowledge about the environment the player interacts with the environment as shown in Figure 2.1.

We observe from the model depicted in Figure 2.1 that the player applies actions on the environment, and the environment responds with feedback to the player. Note that this is the only possible interaction between a player and an environment.

This simple model is used throughout the thesis, where the player is represented by many different learning algorithms, and different applications of the k-armed bandit problem represent the environment. In the following subsections we make a more formal definition of both the player and the environment, inspired by the definitions of environment and learning automata from [11].

2.3.1 Definition of Player

A Player is defined by the quintuple $\{\underline{\Phi}, \underline{\alpha}, \underline{\beta}, A, B\}$, where $\underline{\Phi}$, $\underline{\alpha}$ and $\underline{\beta}$ denote sets, and A and B denote functions.

$\underline{\Phi} = \{\phi_1, \phi_2, \ldots, \phi_r\}$ represents the set of internal states of a Player. The set can either be finite or infinite. The state of the Player at instant n is denoted $\phi(n)$, where $\phi(n) \in \underline{\Phi}$.

$\underline{\alpha} = \{\alpha_1, \alpha_2, \ldots, \alpha_s\}$ is the set of possible actions a Player may apply on the Environment, and in this thesis the number of possible actions will always be finite, hence $s < \infty$. The action at instant n is denoted by $\alpha(n)$, where $\alpha(n) \in \underline{\alpha}$.

$\underline{\beta} = \{\beta_1, \beta_2, \ldots, \beta_t\}$ is the set possible feedback from the Environment and might be finite or infinite depending on the distribution of the Environment. Using the same notation as above, $\beta(n)$ denotes the feedback at instant n, where $\beta(n) \in \underline{\beta}$.

A is a function that brings the Player from one state to the next and is defined as:

$$\phi(n+1) = A[\phi(n), \beta(n)] \tag{2.1}$$

Finally, B is a function that determines the output from the Player, defined as

$$\alpha(n) = B[\phi(n)] \tag{2.2}$$

2.3.2 Definition of Environment

We define an Environment by the triple $\{\underline{\alpha}, F, \underline{\beta}\}$, where $\underline{\alpha}$ and $\underline{\beta}$ denote sets, and F is a function.

$\underline{\alpha} = \{\alpha_1, \alpha_2, \ldots, \alpha_s\}$ is a finite input set, which represents possible actions a Player may apply on the Environment.

$\underline{\beta} = \{\beta_1, \beta_2, \ldots, \beta_t\}$ is an output set of possible feedback from the Environment, and may either be finite or infinite depending on the learning problem.

Finally, F is a function that given input produces an output at instant n and may be described as:

$$\beta(n) = F[\alpha(n)] \tag{2.3}$$

However, in some learning problems several players may be operating simultaneously in the Environment, where the combination of the actions from all players may influence the output to the players, such that the Environment may be described as:

$$\{\beta^1(n), \beta^2(n), \ldots, \beta^p(n)\} = F[\alpha^1(n), \alpha^2(n), \ldots, \alpha^p(n)] \tag{2.4}$$

where p denotes the number of players operating in the Environment.

From a Player's point of view the Environment still may be perceived as $\beta(n) = F[\alpha(n)]$, as the Player independently applies actions on the Environment and receives feedback, and is therefore oblivious about the presence of any opposing players. The Player only assumes that there is some relation between the actions it performs and the feedback it receives.

Note that the sets $\underline{\alpha}$ and $\underline{\beta}$ are the same in both the definition of Player and Environment.

Stationary vs. non-stationary Environment

When the distribution of the feedback is fixed, we refer to the Environment as a *stationary* Environment. However, if the distribution of the feedback is able to change, we refer to the Environment as a *non-stationary* Environment.

For instance, if the feedback from the Environment is given in accordance to a probability distribution we refer to it as a stationary Environment if the parameters of the distribution are kept constant. On the other hand, if the parameters of the probability distribution are able to change we refer to it as as non-stationary Environment.

2.4 Learning automata

The term *learning automata* (LA) was popularized in a survey paper by Thathachar and Sastry [12], and is a branch of the machine learning field which is based upon automata models of learning systems. Initially this field was built upon simple fixed structure state machines such as the Tsetlin automaton, but has later been extended in numerous directions [13].

In this thesis we have included several players from the LA field, which the Bayesian Learning Automaton family shares some principles with. We therefore include a brief overview of learning automata theory, and for a more detailed description of the LA field we refer the reader to [11] and [9].

As mentioned, the definitions of Player and Environment in Sections 2.3.1 and 2.3.2 are inspired by the definition of environment and learning automata as given in [11], and are therefore similar to the definitions given in the following sections.

2.4.1 Definition of finite state-output automaton

A learning automaton is defined by the quintuple $\{\underline{\Phi}, \underline{\alpha}, \underline{\beta}, \mathcal{F}(\cdot,\cdot), \mathcal{H}(\cdot,\cdot)\}$, where $\underline{\Phi}$, $\underline{\alpha}$ and $\underline{\beta}$ are sets, while $\mathcal{F}(\cdot,\cdot)$ and $\mathcal{H}(\cdot,\cdot)$ denote functions.

$\underline{\Phi} = \{\phi_1, \phi_2, \ldots, \phi_s\}$ is a finite set of the internal states of the automaton. The state of the automaton at instant n, where n is a discrete time step, is denoted $\phi(n)$, which is an element in the set $\underline{\Phi}$.

$\underline{\beta} = \{\beta_1, \beta_2, \ldots, \beta_m\}$ denotes the set of possible inputs, also called feedback, the automaton may receive from the environment. The input at instant n is denoted $\beta(n)$, where $\beta(n) \in \underline{\beta}$. This set can either be finite or infinite and depends on the environment.

$\underline{\alpha} = \{\alpha_1, \alpha_2, \ldots, \alpha_r\}$ is a finite set of outputs, also called actions, that the automaton may perform on the environment. The action performed by the automaton at instant n, is denoted $\alpha(n)$, where $\alpha(n) \in \underline{\alpha}$.

$\mathcal{F}(\cdot, \cdot)$ is a transition function that maps from the current state and feedback received from the environment into the next state and can be deterministic or stochastic. The transition function is defined mathematically as [11]:

$$\mathcal{F}(\cdot, \cdot) : \Phi \times \beta \rightarrow \Phi \tag{2.5}$$

That is, the function determines the state at instant $n + 1$, based upon the state and input at instant n [11]:

$$\phi(n + 1) = \mathcal{F}[\phi(n), \beta(n)] \tag{2.6}$$

$\mathcal{H}(\cdot, \cdot)$ denotes either a deterministic or stochastic output function that maps from the current state and response into the output of the automaton. That is, the output of the automaton depends on the current state it resides in together with the feedback it received from the environment. The function $\mathcal{H}(\cdot, \cdot)$ is defined as [11]:

$$\mathcal{H}(\cdot, \cdot) : \Phi \times \beta \rightarrow \alpha \tag{2.7}$$

When the action selection only depends on the state the automaton resides in, the automaton is referred to as a state-output automaton. The output function $\mathcal{H}(\cdot, \cdot)$ is then replaced by the function $\mathcal{G}(\cdot)$, which is defined as [11]:

$$\mathcal{G}(\cdot) : \Phi \rightarrow \alpha \tag{2.8}$$

The action selected by the automaton at instant n is determined in following way [11]:

$$\alpha(n) = \mathcal{G}[\phi(n)] \tag{2.9}$$

The automaton is then defined as the quintuple $\{\underline{\Phi}, \underline{\alpha}, \underline{\beta}, \mathcal{F}(\cdot, \cdot), \mathcal{G}(\cdot)\}$, and in the following we make use of this definition.

2.4.2 Deterministic automaton

A learning automaton is deterministic if both the functions $\mathcal{F}$ and $\mathcal{G}$ are deterministic, which means that the functions uniquely specify a state and an action given the current state and input.

The transition function $\mathcal{F}$ may be represented by m matrices $F(\beta_m)$ of size $s \times s$, where m is the number of feedback elements in the set $\underline{\beta}$ and s denotes the number of states in the set $\underline{\Phi}$. An entry f_{ij}^{β} is defined as [11]:

$$f_{ij}^{\beta} = \begin{cases} 1 & \text{if } \phi_i \rightarrow \phi_j \text{ for input } \beta \\ 0 & \text{otherwise} \end{cases} \tag{2.10}$$

In a similar way the output function $\mathcal{G}$ may be defined in terms of a matrix G of size $s \times r$, where s denotes the number of states and r the number of actions. An entry g_{ij} in the matrix G is defined as [11]:

$$g_{ij} = \begin{cases} 1 & \text{if } G(\phi_i) = \alpha_j \\ 0 & \text{otherwise} \end{cases} \tag{2.11}$$

As we see from the definition of the entries above, each entry in the matrices is either 0 or 1, which is in accordance with the definition of a deterministic automaton. These entries may be regarded as probabilities, and in the case of the state transition function an entry denotes the probability of moving from state ϕ_i to ϕ_j given input β. In the case of the output function an entry denotes the probability that state ϕ_i corresponds to action α_j.

2.4.3 Stochastic automaton

An automaton is stochastic if the state transition function $\mathcal{F}$ or the output function $\mathcal{G}$ is stochastic, which means that the functions do not uniquely specify a state and an action given the current state and input, but instead several states and actions are possible.

If the transition function $\mathcal{F}$ is stochastic, the function gives the probability of reaching the different states. Like a deterministic automaton, a stochastic automaton can be represented by m matrices $F(\beta_m)$ of size $s \times s$, where m is the number of feedback elements in $\underline{\beta}$ and s denotes the number of states. However, in contrast with the deterministic matrices, where an entry is either 1 or 0, an entry in this case is a value in the interval [0,1], representing the probability of moving from state ϕ_i to ϕ_j given input β is defined as [11]:

$$f_{ij}^{\beta} = Pr\{\phi(n+1) = \phi_j | \phi(n) = \phi_i, \beta(n) = \beta\} \tag{2.12}$$

If the output function $\mathcal{G}$ is stochastic, it can be represented by a probability matrix of size $s \times r$, where s denotes the number of states and r the number of actions. Similar as above, an entry is a value in the interval [0,1], representing the probability that state ϕ_i corresponds to action α_j [11]:

$$g_{ij} = Pr\{\alpha(n) = \alpha_j | \phi(n) = \phi_i\} \tag{2.13}$$

The stochastic automaton may be further categorized, depending on the characteristics of the the transition probabilities f_{ij}^{β} and the output probabilities g_{ij}. If f_{ij}^{β} and g_{ij} are assumed to be constant, that is independent of n and the input, the stochastic automaton is called *fixed structure stochastic automaton.* When the transition probabilities change and are updated at each instant n based on the input, the automaton is called a *variable structure stochastic automaton.* The *variable structure stochastic automaton* is explained further in the following section.

2.4.4 Variable structure stochastic automaton

Later it was proposed to use more general stochastic automaton where action and transition probabilities are updated, instead of a more fixed structure state automaton with fixed transition probabilities. A general stochastic automaton is described by the quintuple $\{\underline{\phi}, \underline{\alpha}, \underline{\beta}, A, G\}$ [11]. Similarly, as in the definition of a Player in Section 2.3.1, $\underline{\phi}, \underline{\alpha}$ and $\underline{\beta}$ denote sets, and A and G denote functions.

The definition above may be simplified by just considering the action probabilities, and an automaton is then described by the triple $\{\underline{\alpha}, \underline{\beta}, A\}$ with $r = s < \infty$ and G as an identity mapping between $\underline{\phi}$ and $\underline{\alpha}$ [11]. The terms states and actions may therefore be used interchangeably, and the set $\underline{\alpha}$ contains the state or action probabilities and constitutes a state or an action probability vector. A more detailed description of this simplification is available in [11].

L_{R-P}, L_{R-I} and Pursuit are examples of variable structure stochastic automata and are described in more detail in Chapter 4. The Bayesian Learning Automaton family, presented in Chapter 5, may also be perceived as a variable structure stochastic automaton by replacing the state probabilities $\underline{\alpha}$ with probability distributions and their associated parameters, where the state selection probabilities are dependent on the relative differences between these distributions. This is further explained and discussed in Sections 5.1.2 and 7.5.

2.5 Selected concepts from game theory

Game theory is a branch of economics where an environment with multiple players is regarded as a game [10]. In relation to reinforcement learning such games typically include turn based games, where two players alternate in applying actions on the environment.

In the following we give a short introduction of some concepts from game theory, which are used to describe important characteristics of some learning problems we investigate in this thesis. For a more thorough overview of these concepts and game theory in general we refer the reader to [7] and [14].

2.5.1 Rationality

A player may base his strategy on expectations of what the opponent is most likely to do. For instance, when humans play games, we often assume that the opponent is acting rationally, thus we have a clear view of what we might expect from our opponents. In game theory such assumptions are referred to as common knowledge of rationality (CKR) [7].

Different orders of CKR exist, from the more specific zero-order CKR and first-order CKR to the more general *nth*-order CKR. Zero-order CKR denotes the situation where the players act rational, but do not know anything about the rationality of its opponents. First order CKR means that the players act rationally and assume that their opponents act rationally as well. A more general definition of CKR is the *nth*-order CKR, where n denotes the level of rationality a player assumes in its reasoning. Consider the following example with two players A and B. With a *nth*-order CKR, we have the following rationality reasoning: A believes that B believes that A believes that B believes . . . . Thus, one way to regard *nth*-order CKR is the number of times *believes* is used in the rationality reasoning.

2.5.2 Dominance

A player might be able to depict the actions of a rational opponent by identifying dominant strategies in the game. In this context a strategy may be defined as a dominant- or dominated strategy. These are defined as following in [7]:

> *A strategy is **dominated** if it is not a best response strategy whatever the strategy choice of the opposition. Conversely, a strategy is **dominant** if it is a best strategy (i.e. it maximises a player's utility pay-off) regardless of the opposition's choice of strategy.*

Assuming a two-player game, a rational player will always choose a dominant strategy, regardless of what the player believes that the opponent will do. Thus, from the CKR definition above, this constitutes zero-order CKR. However, in the absence of a dominant strategy a rational player will typically choose an action based upon what he believes the other player will choose, thus using first-order CKR.

2.5.3 Pure strategy

A pure strategy is a single strategy in a set of several possible (pure) strategies. In relation to the definition of Player and Environment in Section 2.3, a pure strategy denotes a single action in the set $\underline{\alpha}$. A solution to a game is often said to be in pure strategies when there exists a single best strategy for all players taken into account all opposing players' strategies.

Consider the game *Rock-paper-scissors*, where *rock*, *paper* and *scissor* constitute the set of strategies $\underline{\alpha}$. For instance, if a player always, no matter what, applies the action *rock*, we refer to the strategy *rock* as a pure strategy.

2.5.4 Mixed strategy

In a mixed strategy game it is suboptimal to play a single strategy, and the optimal strategy consists of randomly select among several different single strategies according to a probability distribution.

For instance, consider the simple and well-known game *Rock-paper-scissors.* In this game it is easy to beat a player if it plays a pure strategy by for instance always selecting *rock*, as it would be beaten by *paper* every time. Therefore, by simply adding some unpredictability by randomly select one of the three possibilities it becomes a lot harder to beat by an opponent.

More formally we describe mixed strategy as follows. For each player k, there is a probability distribution $p^{(k)} = \{p_1^{(k)}, \ldots, p_{N_k}^{(k)}\}$ over the action set $\underline{\alpha}$, and a single pure strategy $\alpha(n)^{(k)}$ is randomly selected from the set of strategies according to the distribution $p^{(k)}$ [14].

2.6 Selected learning problems

It is often useful to construct models of complex problems which incorporate the essential features of the problems. In the following sections we present selected problems of interest in this thesis.

In Section 2.6.1 we provide a more extensive introduction to the general k-armed bandit problem and the feedback distributions used in this thesis.

We then present selected games from game theory which from the players' point of view may be treated as a bandit problem, as each player sequentially selects one of several available actions while the overall goal is to maximize the received feedback. These games consist of the Goore game presented in Section 2.6.2, prisoners' dilemma presented in Section 2.6.3 and the two-player zero-sum game presented in Section 2.6.4.

2.6.1 *K*-armed bandit problem

The k-armed bandit problem, is a well known problem in reinforcement learning and was described by Robbins in [15]. The problem is a formal model of the exploration vs. exploitation problem found in many areas [10].

In the literature the problem is often modelled as one machine with several levers, but may equivalent be modelled as several distinct machines, where each machine represents a lever.

The bandit problem can be described in the following way. Consider a room with k slot machines, where each slot machine is assigned a certain fixed, but to the player, unknown probability distribution. By playing a machine the player receives feedback that is determined by the probability distribution associated with the machine.

The probability distributions differ from machine to machine, and hence the feedback generated by the machines differ, which means that playing different machines may result in a different average feedback. Therefore, the selection strategy of the player is of great interest. A player should sequentially select and play the machines and collect feedback, where the overall goal is to maximize cumulative feedback. As noted above, the player has no prior knowledge about the probability distributions associated with these machines and will only gain knowledge of a machine by playing it and collecting feedback. There is no gain in knowledge about the machines that are not played.

A player needs to exploit the best slot machine as much as possible, but identifying a suboptimal slot machine as the optimal machine can be costly in the long run. Hence, a player needs to exploit the best machine as much as possible, but at the same time make sure that the perceived optimal slot machine is in fact the optimal machine. In the long run it may be beneficial to risk a short term loss by exploring new machines which might identify a better machine.

As noted above, each machine represents an arm on a slot machine, hence the name k-armed bandit problem. In this thesis we model the problem as one machine with several arms. In relation to the definition of Player and Environment in Section 2.3, arm is equivalent with action.

A wide range of solutions to the problem has been proposed, among them statistics and confidence interval, genetic algorithms and many others [16, 10]. In this thesis we investigate a very promising approach based on Bayesian statistics proposed by Granmo in [5].

We present several selection strategies in Chapter 4 and 5.

Feedback distributions

In this thesis we consider the k-armed bandit with several different fixed feedback distributions, where the feedback is either Bernoulli distributed, Poisson distributed, normally distributed or a set of discrete values.

We first consider the k-armed bandit problem with Bernoulli distributed feedback. In this case the environment gives a favorable response, or reward, with probability p and an unfavorable response, or penalty, with probability $1 - p$. Typically 1 and 0 are used to denote a reward and penalty, respectively. Thus, when the feedback is Bernoulli distributed the feedback set is defined as follows: $\underline{\beta} = \{0, 1\}$.

When the feedback is Poisson distributed, the feedback set constitutes a finite set $\underline{\beta}$, where the elements are integer values in the interval $[0, \infty)$. Thus, we can define the feedback set as: $\underline{\beta} = \{\beta_1, \beta_2, \ldots, \beta_m\}$, where $m < \infty$ and each element is an integer in the interval $[0, \infty)$.

With normally distributed feedback, the feedback set constitutes an infinite set, where the elements are any value in the interval $(-\infty, \infty)$. Thus, we can define the feedback set as $\underline{\beta} = \{\ldots, \beta_{m-1}, \beta_m, \beta_{m+1}, \ldots\}$, where each element in the set is a real number in the interval $(-\infty, \infty)$.

The feedback set may also be a set of discrete values, where each element is any value in the interval [0,1]. Thus, we can define the feedback set as $\underline{\beta} = \{\beta_1, \beta_2, \ldots, \beta_m\}$, where $m < \infty$ and each element is a real number in the interval $[0, 1]$.

From the above we see that the term feedback is the most general term, however in the case of Bernoulli distributed feedback, it is also applicable to use the terms reward and penalty. Therefore, when dealing with Bernoulli distributed feedback, we consequently use reward and penalty in this thesis.

2.6.2 Goore game

The Goore game, also referred to as the Gur Game, was introduced by M.L Tsetlin in [17]. The key concept of the game is the ability of players to reach a common goal without communicating with each other. Thus, the game exhibits decentralized decision making, where the players cooperate to reach a common goal that is of every player's best interest.

The Goore game can be illustrated in the following way [11]. Consider a room with N cubicles and a raised platform. One player sits in each cubicle, while a referee stands on the platform. During the game the referee issues several voting iterations. In each voting iteration, each player votes, independently of each other and simultaneously, either *yes* or *no*. The casted votes are collected by the referee, which counts the number of players that voted *yes*, denoted θ. The referee has a unimodal reward function, f, which has a maximum value when the number of yes votes is exactly θ^*. The voting iteration ends with the referee either giving a reward with probability $f(\theta)$ or penalty with probability 1-$f(\theta)$ to each player independently, regardless of its actual vote. After the feedback, the voting procedure starts all over again.

As an example of a unimodal reward function $f(\theta)$ we consider the function used in [6]:

$$f(\theta) = 0.2 + 0.8 * e^{(-0.002(\theta-35)^2)} \tag{2.14}$$

The function above gives values in the interval [0,1], and when $\theta = 35$ the function yields 1, which is the highest value in the specified interval. We refer to 35 as the optimal number of yes votes, denoted θ^* above. When the number of yes votes is 35, the function gives a reward with probability 1, and as the number of yes votes gets closer to the optimal value 35, the function yields higher probability for getting a reward.

The reward function is depicted in Figure 2.2, where the dashed line denotes the optimal number of yes votes. Note that the optimal number 35 in Equation 2.14 may be replaced by any desired number of yes votes.

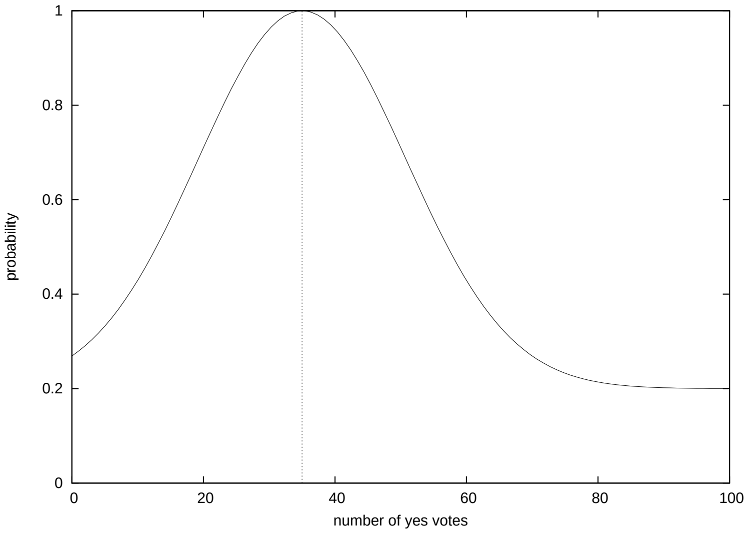


Figure 2.2: Unimodal reward function f(θ) in the Goore game

The Goore game may be regarded as a bandit problem in the sense that each player has to choose between *yes* or *no*, which represent two distinct arms or actions in the bandit problem. Furthermore, from the above we see that the feedback from the referee is Bernoulli distributed. From the definition of non-stationary environments in Section 2.3.2 we see that the Goore game constitutes a non-stationary environment, as the probability of getting a reward change in accordance with the number of players that votes *yes*.

The goal of each player is to maximize its own reward. However, despite each player acting in a rather selfish way, some bandit players are able, without communicating with each other, to cooperate to reach a common goal. For instance, it has been proved that with a team of Tsetlin automata it is possible to get exactly θ^* to vote *yes* with sufficient voting iterations [11].

The principle of the Goore Game has found its uses in several areas. In [18] the paradigm of the Goore game is used to achieve distributed control in distributed systems, and in [6] it is applied to sensor networks to achieve QoS (Quality of Service) control.

2.6.3 Prisoners' dilemma

The prisoners' dilemma is a game where two rational players, selfishly and individually, attempt to maximize their feedback which paradoxically results in a suboptimal outcome for both players.

The game can be described as follows [7]. Two criminals are picked up by the police for a crime they both played an equal part in. The criminals are placed in their own cells, and both criminals are given the same offer: If one of the criminals confesses and the other does not, the confessing criminal will go free while the other is given a sentence of five years. If they both confess they will get a small reduction for their cooperation, and they will both be given a sentence of three years. However, if they both cooperate by not confessing, they will both receive a sentence of one year because of limited proofs. Each prisoner has to decide to confess or not on their own without information about the decision of the other prisoner.

Even if the prisoners somehow would be able to communicate with each other, it does not really change the dilemma, as each prisoner instead has to decide if the other prisoner can be trusted, or if the prisoner itself should break an agreement for a reduced sentence [7].

The game can be described by a game matrix as shown in Figure 2.3, where player A and player B in the matrix denote the two criminals. Player A selects a row, while player B selects a column. The numbers in each entry represent the sentence in years the players will receive, where the value to the left of the comma represents the sentence player A will receive, while the value to the right represents the sentence player B will receive.

Player B

		Not confess	Confess
Player A	Not confess	1,1	5,0
	Confess	0,5	3,3

Figure 2.3: Game matrix in prisoners' dilemma

From the matrix in Figure 2.3, we see that the best outcome for each player is to confess to the crime under the assumption the other player does not confess, since the player that confesses goes free (0), while the other player receives a sentence of 5 years. If both players pursuit this strategy this yields a paradox result by making each player worse off than if they had both decided to cooperate by both not confessing to the crime.

For a more thorough history and description of prisoners' dilemma we refer the reader to [7].

Iterative prisoners' dilemma

In the iterative prisoners' dilemma the game, as described above, is repeated several times. A player is then able to use information about the outcome of previous iterations during decision making, where the overall goal of the players is to minimize their cumulative sentence.

There exist two types of the iterative prisoners' dilemma, and these are distinguished from each other based upon whether or not the players know the number of iterations in the game in advance, which may affect the strategy of the players. If the players know the number of iterations in game, the players may try to use this information to find some strategic points towards the end to confess. This could for instance be to always confess in the last iteration.

The iterative prisoners' dilemma may, from the players point of view, be regarded as a bandit problem with two arms, representing *not confess* and *confess* respectively. The feedback set of the Environment is in this case a set of discrete values, and since the feedback to each player is dependent on the action of the other player the iterative prisoners' dilemma constitutes a non-stationary environment.

Tit-for-Tat and Axelrod's tournament

Robert Axelrod invited several professional game theorists to submit players for playing in a tournament version of the iterative prisoners' dilemma. Each player was matched against each competitor once, and the whole tournament was repeated several times to get reliable results. Axelrod got responses from 14 different people [7].

Quite surprisingly, the player *Tit-for-Tat* submitted by Anatol Rapoport won the tournament. This player simply starts by cooperation, and from then on simply repeats what the other player did last turn. It was one of the simplest submissions and proved superior to several more complex and advanced algorithms.

A second tournament was arranged with some slightly modified rules, and this time 62 submissions was received. Also this time *Tit-for-Tat* was the winner [7].

It seems like *Tit-for-Tat*'s willingness to cooperate and fast response on the other player's decision is a very efficient strategy. It is however, not a perfect player. To be able to get the best response a player needs to know the other player's strategy in advance, and then exploit its weaknesses. For instance, *Tit-for-Tat* will always be beaten by a player that always confesses. The confessing strategy would then win the first round, and from then on they would both confess until the end. This, however, is clearly a suboptimal strategy if the overall goal is to minimize the sentence.

In contrast to the other players we present in Chapters 4 and 5, *Tit-for-Tat* is not a reinforcement scheme, as it is not able to improve its performance through experience, but merely repeats the opposing player's "moves". However, as it is a very good player in prisoners' dilemma we have chosen to include it as a reference point for the bandit players in this dilemma.

2.6.4 Two-player zero-sum game

The two-player zero-sum game is an extensively studied game in game theory [14]. The game consists of two opposing players who both attempt to maximize their own reward. The Environment evaluates the actions applied by the players and responds with zero-sum feedback, such that if a player receives feedback x, the opposing player receives feedback $-x$. Thus, a player can only gain at the expense of the other player, a key feature of a zero-sum game.

The two-player zero-sum game encountered in this thesis can be described as follows. The two players both try to maximize their reward by playing the game through a series of iterations, and the players are able to use information from previous iterations in the decision making in the current iteration. In each iteration both players choose an action independently of each other and without information about the choice of the opposing player. Each player has a certain probability to win based on their combined chosen actions, and the probability of winning (or losing) is determined by a game matrix as shown in Figure 2.4.

$$\begin{bmatrix} 0.6 & 0.8 \\ 0.35 & 0.65 \end{bmatrix}$$

Figure 2.4: Game matrix in a two-player zero-sum game

In the matrix in Figure 2.4 we consider two players A and B, where player A selects a row and player B selects a column. Each entry in the matrix denotes the probability for a victory for player A, and the probabilities for a victory for player B are then easily calculated by $1 - m_{ij}$, where m_{ij} denotes an entry in the matrix. Alternatively, we may describe each entry in terms of the probability that player B looses, where the probabilities that player A looses are then calculated with the same formula as above.

For instance, if player A selects the first row and player B selects the second column, then the probability for a victory for player A is 0.8 and 1 - 0.8 = 0.2 for player B. Alternatively, the probability that player B looses is 0.8 and 1 - 0.8 = 0.2 for player A.

From a player's point of view a two-player zero-sum game may be considered as a bandit problem, where the number of possible choices of a player, i.e the number rows or columns in the game matrix, represents the arms. Also, the feedback given to the players is either a reward or a penalty, which means that the feedback is Bernoulli distributed. Furthermore, as the feedback to both players is dependent on the other player's action, the two-player zero-sum game constitutes a non-stationary environment.

The optimal solution to a two-player zero-sum game can both be in pure and mixed strategies and is determined by the combination of the values in the game matrix.

Chapter 3

Bayesian inference

The Bayesian Learning Automaton family presented in Chapter 5 relies heavily on Bayesian statistics, and we therefore devote this chapter to present selected concepts from Bayesian statistics.

We start this chapter by defining a *random variable* in Section 3.1, and in the context of a random variable we give a more precise definition of *distribution* as we have already encountered in Chapter 2 in terms of the feedback distribution of the environment.

We then introduce statistical inference and the main approaches to statistical inference, the frequentist and Bayesian approach in Section 3.2.

After a brief introduction to Bayesian inference we present the key to Bayesian inference, namely Bayes' theorem in Section 3.3. By applying Bayes' theorem we may update our beliefs about parameters of a probability distribution we want to infer about as new evidence is obtained. Section 3.4 explores important characteristics of the prior distributions, which represent a prior belief in the parameters associated with a probability distribution.

Then in Section 3.4.3 we present an important property of prior distributions that belong to a class of probability distributions called the *exponential family*, namely *conjugacy*. Conjugacy is an important property in Bayesian statistics as it makes it possible to avoid complex mathematical calculations in Bayes' theorem, and instead rely on simple updating rules of the parameters of conjugate prior distributions in the light of new evidence. As briefly mentioned in Chapter 1, these rules play an essential part of the Bayesian Learning Automaton family, and we therefore explore this property in detail in Section 3.4.3.

3.1 Random variables

A random variable describes the outcome of an experiment [19], where the values of the variable occur according to a probability distribution. We distinguish between discrete random variables, where the outcomes are distinct values separated from each other, and continuous random variables, where there are uncountable numbers of values.

In the following we present the definitions of discrete and continuous distribution of random variables, and refer the reader to [19] and [20] for a more thorough overview.

3.1.1 Distribution of a discrete random variable

We let Y be a discrete random variable that only takes on countable values from a set of possible values. The set either consists of a finite number of elements such that $\{y_1, y_2, \ldots, y_k\}$, or a countable infinite number of elements, yielding $\{y_1, y_2, \ldots\}$ [19].

A discrete random variable Y takes on the values y_i, where $i \in \{1, 2, \ldots\}$, with probability p_i. Furthermore, Y takes on the values defined by the set with probability 1, and values not defined by set with probability 0. Therefore we have the following [21]:

$$\sum_{i=1}^{k} p_i = 1 \text{ for a finite number of values } y_i$$

or

$$\sum_{i=1}^{\infty} p_i = 1 \text{ for a infinite number of values } y_i$$

The probability that Y takes on the value y_i is defined as [20]

$$p(Y = y_i) = p_i \; i \in \{1, 2, \ldots\} \tag{3.1}$$

where $p_i \geq 0$.

p_i from Equation 3.1 is called the *discrete probability distribution* for the discrete random variable Y, and $p(Y = y_i)$ is called the *probability mass function.*

In Section 2.6.1 we briefly presented the Bernoulli distribution which is a discrete probability distribution.

3.1.2 Distribution of a continuous random variable

We let Y be a continuous random variable that takes on values in the interval (-∞,∞). Due to the uncountable number of real numbers in this interval the probability of observing a particular value is zero.

For continuous random variables we have a *probability density function*, which measures the density of probability at each point in the interval [19]. The probability that the random variable lies in the interval (a, b) is given by integrating the probability density function $p(y)$ as defined by [19]

$$p(a < Y < b) = \int_a^b p(y)\, dy \tag{3.2}$$

with the following conditions met [20]:

$$p(y) \geq 0$$

and

$$\int_{-\infty}^{\infty} p(y)\, dy = 1$$

$p(y)$ is also referred to as the *continuous probability distribution* for the random variable Y.

In Section 2.6.1 we encountered the normal distribution which is a continuous probability distribution.

3.1.3 Random numbers and random sampling

Random numbers are values of a random variable that are distributed in accordance to a specified distribution, where a large set of these values reconstructs the underlying distribution.

When applied in statistical experiments we often use the term *random sampling* when we randomly select values or samples from a probability distribution or population. Random sampling is useful when it is not practical to examine all possible cases in an experiment, but instead make use of a set of samples from a population to reach conclusions about the population as a whole. We will return to this in Section 3.2.

In this thesis random numbers are applied in the feedback from an environment to a bandit player, presented in Section 2.6.1, where the feedback is generated by chance according to the probability distribution of the environment. Furthermore, the Bayesian Learning Automaton family makes use of random sampling of values as a part of the action selection procedure, which we present in Section 5.1.2.

3.2 Statistical inference

Statistical inference involves making inferences about unknown parameters of a population based upon statistics calculated from data obtained by random sampling from the population. Moreover, we seek, with the aid of statistical inference, the distribution of the population, and the unknown parameters associated with the distribution. The probability distribution governs the observations encountered in experiments, and it is therefore crucial in statistical inference that the observations are representative of the population as a whole, such that the inferred distribution of the samples is indeed similar to the distribution of the population.

The data from the sampling is modelled as observed values of random variables, and we denote the observations as $y = (y_1, y_2, y_3, \ldots y_n)$, where y is a vector of data sampled from the population. Based on these samples we want to infer about the distribution of the population as a whole, and the unknown parameters of the probability distribution of the population constitute a vector $\theta = (\theta_1, \theta_2, \ldots, \theta_n)$. Thus, on the basis of the observed data y we assess some aspect of the unobservable θ.

The notation introduced in this chapter will be used throughout the thesis, where θ constitutes a vector of parameters, and y constitutes a vector of samples from a population.

3.2.1 Approaches to statistical inference

There are two broad main approaches to statistical inference, the Bayesian and the frequentist approach. The differences between these approaches relate to the interpretation of probability, together with the objectives of statistical inference [22]. Below we point out some important differences between these two approaches, and for a more detailed description of these approaches the reader is referred to [19] and [22].

The key concept of the Bayesian approach is that the unknown parameters θ should be treated like random variables. Therefore, in the Bayesian view, a prior probability distribution on θ is specified [22]. The specification of a prior can either be done in an objective or subjective manner. A subjective prior includes the statisticians' own prior belief, which has been a common criticism with subjective priors [22]. In the context of the Bayesian view, inference is related to how the prior distribution changes to the posterior distribution as new evidence in the form of the data y is obtained. This is achieved through the use of Bayes' theorem as described in Section 3.3. Hence, in Bayesian inference probabilities are regarded as degree of belief about a parameter θ_i.

The frequentist approach interprets probability as the relative frequency an event occurs as the number of trials tends to infinity. In contrast to the Bayesian view, the unknown parameters are considered to be unknown fixed constants, hence they are not treated as random variables with a probability distribution. Essential to the frequentist approach is that inferences on the data y should be based on an infinite number of hypothetical repetitions of an experiment. Hence, the frequentist statistical inference is based upon all possible occurring data for the fixed parameters θ, not only on the occurring data as in Bayesian inference. Therefore, given the actual data, there is no random characteristics of the parameters θ left, and therefore in accordance with the frequentist way, one can only make confidence statements about the parameters θ, based on all possible observations [19].

3.3 Bayes' theorem and parameter estimation

Bayes' theorem is the key to Bayesian inference. With the aid of Bayes' theorem the prior distribution of the unknown parameters θ is updated, by observing the data y. This yields a posterior distribution which is conditional on only the occurred data, not all possible occurring data as in the frequentist view.

Bayesian inference seeks to make inferences about the parameters θ given the observed data y. As noted in Section 3.2.1, in Bayesian inference both θ and y are regarded as random variables, where the joint probability distribution $p(\theta, y)$ is defined as the product of the prior distribution, denoted $p(\theta)$, and the sampling distribution, also referred to as the likelihood function, $p(y|\theta)$ [23]:

$$p(\theta, y) = p(\theta)p(y|\theta) \tag{3.3}$$

The posterior distribution of θ given the value of the data y, $p(\theta|y)$, is defined according to the definition of the conditional probability [19]:

$$p(\theta|y) = \frac{p(\theta, y)}{p(y)} \tag{3.4}$$

Combining Equation 3.3 with Equation 3.4 yields Bayes' theorem [23]:

$$p(\theta|y) = \frac{p(\theta)p(y|\theta)}{p(y)} \tag{3.5}$$

where

$$p(y) = \sum_{\theta} p(\theta)p(y|\theta) \text{ for discrete values of } \theta \tag{3.6}$$

or

$$p(y) = \int p(\theta)p(y|\theta)\, d\theta \text{ for continuous values of } \theta \tag{3.7}$$

where $p(y)$ is the probability of y over all possible values of θ. In Equation 3.5 $p(y)$ acts as a normalization constant, and as we see is not dependent on θ. Thus, this term is often omitted from Equation 3.5, and this yields the un-normalized posterior distribution in Equation 3.8 [23]:

$$p(\theta|y) \propto p(\theta)p(y|\theta) \tag{3.8}$$

where $\propto$ denotes proportionality.

Equation 3.8 is often written in words as [19]:

$$\text{posterior} \propto \text{ prior } \times \text{ likelihood} \tag{3.9}$$

3.4 The role of prior distributions in Bayesian statistics

The prior distribution can be any probability distribution, but the general idea is that it should reflect ones knowledge and belief about the parameters of interest that are to be estimated. As noted, θ is a vector with the unknown parameters of the population, and depending on the distribution the vector may consist of one or several elements. The use and choice of priors has long been subject for debates, and a thorough overview of prior distributions is given in [24].

However, a distinction about prior distributions can be made, whether they are so called informative or non-informative.

3.4.1 Non-informative prior distribution

The definition of a non-informative prior distribution is based on the so-called insufficient reasoning principle, also known as the Bayes-Laplace postulate. This principle states that in the absence of prior knowledge of the parameter θ_i, each possible value of the parameter, constituting the parameter space Θ, should be equally weighted [25]. This justifies the use of uniform prior distributions, since there is no reason to believe that one value of parameter θ_i is more likely than any other value. This is often referred to as *letting the data speak for themselves*, as such a prior has minimal affect on the posterior distribution.

Consider an unknown parameter θ_i, that can only take a finite number n values. The parameter space Θ is then a finite set with n elements, where each element is given a probability of $\frac{1}{n}$. A non-informative prior would in this case be a discrete uniform distribution defined as [23]:

$$p(\theta_i) = \left\{ \frac{1}{n}^{(1)}, \frac{1}{n}^{(2)}, \dots, \frac{1}{n}^{(n)} \right\} \tag{3.10}$$

Continuous uniform distributions may also be used, however these often results in improper priors, because they can not be normalized, i.e the integral of $p(\theta_i)$ is infinity. However, the posterior distribution obtained from improper prior distributions may still be proper. In general, a great deal of care should be taken when interpreting a posterior distribution obtained from an improper prior distribution [23].

The motivation for using non-informative priors is to achieve objectivity, i.e no prior belief about the parameters θ exists.

3.4.2 Informative prior distribution

Often some prior knowledge or belief about the parameters θ exists which can be expressed through an informative prior distribution. The prior distribution should represent an initial belief about the population, and should include all plausible values of θ_i. However, these values do not need to be realistically centered around the true values, as the observed data in most cases far outweigh the initial belief.

By using an informative prior distribution a certain degree of subjectivity is introduced. The introduction of subjectivity in priors is, as mentioned in Section 3.2.1, heavily debated, but may prove to be invaluable information in some domains [23]. For instance, in clinical trials of treatments where trials on a large population of subjects are both impossible and unethical, initial belief may provide crucial information and added precision. Clinical trials typically include control groups, and since a single subject can only be part of one group, there will always be some difference between the groups. Because of this difference all reasonable prior information may be important in order to be able to distinguish groups from each other, and therefore be able to evaluate the results of such trials [23].

The main argument for using informative priors is that if some reasonable correct initial belief exists, it would be wasteful not to include it. In most complex real world problems a lot of unknowns quantities will be present, which most likely will affect the results in some way. For instance, in clinical trials, there are a lot of unknowns which affect the outcome of each subject. Such unknowns are for instance each subject's initial health, which may be impossible to evaluate completely objective. The presence of unknowns makes results more difficult to interpret, which argues for that all reasonable prior information should be included to ease the interpretation of the results [23].

3.4.3 Conjugate priors

Conjugacy is a property of a prior distribution over θ that leads to a posterior distribution of the same parametric form as the prior itself. The same parametric form means that the two distributions are defined by the same parameters.

The conjugacy is an important property, because by using conjugate priors there is no need to perform numerical integrations in order to to find the normalization constant $p(y)$, which often requires complex and tedious calculations [22].

Conjugate prior distributions are the heart of the Bayesian Learning Automata, thus the foundation presented in this section is essential to the derivation of the various BLA variants presented in Chapter 5.

A definition of conjugate prior distributions is given in [23]:

> *If $\mathcal{F}$ is a class of sampling distributions $p(y|\theta)$, and $\mathcal{P}$ a class of prior distributions for θ, then the class $\mathcal{P}$ is said to be a conjugate for $\mathcal{F}$ if*
>
> $$p(\theta|y) \in \mathcal{P} \text{ for all } p(\cdot|\theta) \in \mathcal{F} \text{ and } p(\cdot) \in \mathcal{P} \tag{3.11}$$

As noted in [23] such a definition is vague, which when the class $\mathcal{P}$ is composed of all distributions, leads to $\mathcal{P}$ always being conjugate despite the class of sampling distributions. However, by limiting the class $\mathcal{P}$ to only consist of distributions with the same functional form as the likelihood, we get the so called natural conjugate prior families.

Every probability distribution that belongs to the exponential family has a natural conjugate prior distribution. Using the notation from above, class $\mathcal{F}$ is an exponential family if all the distributions have the following form [23]:

$$p(y_i|\theta) = f(y_i)g(\theta)e^{(\phi(\theta)^{\mathcal{T}}u(y_i))} \tag{3.12}$$

where y_i and parameters θ are vectors and $g(\theta)$ and $u(y_i)$ are vectors with the same dimension as θ. Furthermore, $\phi(\theta)$ is called natural parameters of the family $\mathcal{F}$.

Let $y = \{y_1, y_2, \dots y_n\}$ be a set of identically distributed data, where the likelihood function of y belongs to the exponential family and is defined as [23]:

$$p(y|\theta) = \left[\prod_{i=1}^{n} f(y_i)\right] g(\theta)^n e^{\left(\phi(\theta)^{\mathcal{T}} \sum_{i=1}^{n} u(y_i)\right)} \tag{3.13}$$

As a function of the parameters θ, the likelihood, for all n and y, will have the fixed form

$$p(y|\theta) \propto g(\theta)^n e^{(\phi(\theta)^{\mathcal{T}} t(y))} \tag{3.14}$$

where

$$t(y) = \sum_{i=1}^{n} u(y_i)$$

since

$$\left[\prod_{i=1}^{n} f(y_i)\right]$$

does not affect the form of the function [23].

As we see from Equation 3.14, the likelihood function is only dependent on the actual data y through $t(y) = \sum_{i=1}^{n} u(y_i)$, which is referred to as *sufficient statistics* for θ [23]. Sufficient statistics contain all necessary information that is needed to infer on the parameters θ, and by simply updating $t(y)$ with new sample data y we obtain the sufficient statistics for θ. This property is convenient in mathematical manipulations of the posterior distribution and the likelihood function. Thus, we consider a prior with the same functional form, and let the prior distribution be specified as [23]

$$p(\theta) \propto g(\theta)^{\eta} e^{(\phi(\theta)^{\mathcal{T}} \nu)} \tag{3.15}$$

where ν and η are called hyperparameters. The parameters of the prior are called hyperparameters in order to distinguish them from the parameters of the likelihood function. According to Bayes' theorem, the posterior distribution is obtained by multiplying the prior distribution with the likelihood function, yielding:

$$\begin{aligned} p(\theta|y) &\propto p(\theta)p(y|\theta) \\ &\propto g(\theta)^{\eta} e^{(\phi(\theta)^{\mathcal{T}} \nu)} g(\theta)^{n} e^{(\phi(\theta)^{\mathcal{T}} t(y))} \\ &\propto g(\theta)^{\eta+n} e^{(\phi(\theta)^{\mathcal{T}} (\nu + t(y)))} \end{aligned} \tag{3.16}$$

Comparing the posterior distribution in Equation 3.12 with the exponential form of the prior distribution in Equation 3.15, we see that they have indeed the same parametric form. Thus, the conjugacy has been confirmed.

It has been shown that, except for some special cases, the members of the exponential family are the only class of distributions that have natural conjugate prior distributions, due to that these distributions are the only ones with a fixed number of sufficient statistics [23].

By starting with a prior distribution we end up with a posterior distribution with the same parametric form, where the hyperparameters are updated by the sample information:

$$\begin{aligned} \eta_n &= \eta_{n-1} + n \\ \nu_n &= \nu_{n-1} + t(y) \end{aligned}$$

When using a conjugate prior the statistical inference procedure is simplified since we only need to update the hyperparameters of the prior distribution in order to obtain the posterior distribution. Furthermore, by examining Equation 3.16 and considering the update rules above, we see that if an experiment involves several number of trials, the posterior distribution from trial i becomes the prior distribution at trial $i + 1$.

Table 3.1: Likelihood and conjugate priors encountered in this thesis

Likelihood function $p(y\|\theta)$	**Conjugate prior** $p(\theta)$
Bernoulli	Beta
Poisson	Gamma
Normal unknown μ and known σ^2	Normal
Normal unknown μ and σ^2	Normal-Scaled inverse-χ^2

The conjugate priors and the corresponding likelihoods that are used in this thesis are shown in Table 3.1. We will investigate the updating procedure of the hyperparameters of the individual prior distributions in more detail in Chapter 5, which are at the heart of the Bayesian Learning Automaton family.

The unknown parameters θ of a population depend on the probability distribution of a population. For instance, if the population is normally distributed, we might want to infer about the unknown mean μ and the variance σ^2 of the population, which means that $\theta = (\theta_1, \theta_2)$, where $\theta_1 = \mu$ and $\theta_2 = \sigma^2$. Alternatively, we might want to only infer about one of these parameters.

Furthermore, when the population is Poisson distributed, we want to infer about the rate parameter λ, thus $\theta = (\theta_1)$, where $\theta_1 = \lambda$. With a Bernoulli distributed population we want to infer about the probability p of the population that governs the number of successes and failures in a sequence of trials. Thus, $\theta = (\theta_1)$, where $\theta_1 = p$.

The reader is referred to [25], [23] and [24] for a more detailed introduction of conjugate priors and likelihood functions.

As noted in the introduction to this section, conjugate prior distributions are essential to the BLA family, and especially Equation 3.15 and 3.16 are used extensively in Chapter 5.

3.5 Estimation of the parameters of the likelihood function with a conjugate prior distribution

In Section 3.4.3 we identified the conjugate priors for the Bernoulli, Poisson and normal distribution. In this section we make a clearer connection between the unknown parameters we make inferences about and the conjugate priors.

Due to the exponential properties explored in Section 3.4.3, the mean value of the posterior distribution will approach the real value of the parameter we infer about as the number of trials tends to ∞. This feature is crucial to the understanding of parameter estimation with the use of conjugate priors.

The mean of the beta, gamma and normal distribution are shown in Table 3.2. Note that we use m to denote the mean of the normal prior distribution in order to distinguish it from the mean μ of the likelihood function.

Table 3.2: Mean values of the conjugate prior distributions encountered in this thesis

Conjugate prior $p(\theta)$	**Mean**
Beta	$\frac{\alpha}{\alpha+\beta}$
Gamma	$\frac{\alpha}{\beta}$
Normal	m
Normal-Scaled inverse-χ^2	m

For instance, when we infer about the parameter p of the Bernoulli distribution, and use the beta distribution as the conjugate prior for the Bernoulli distribution, the objective with the inference is to get the mean of the beta distribution to approach the true value of parameter p as the number of trials tends to ∞. Furthermore, by using the gamma distribution as the conjugate prior for the Poisson distribution, the mean of the gamma distribution should approach the true value of the parameter λ of the Poisson distribution. Likewise, the mean m of the normal distribution should approach the true value of μ of the normal likelihood function.

Thus, from the above, we get the following

$$\begin{aligned} \frac{\alpha}{\alpha+\beta} &= p \\ \frac{\alpha}{\beta} &= \lambda \\ m &= \mu \end{aligned}$$

when the number of trials tends to ∞.

In Section 5.1.1 we show an example of inference about the parameter p of the Bernoulli distribution, where the mean value of the beta distribution approaches the true value of p as the number of trials increases.

Chapter 4

Non-Bayesian bandit players

In Chapter 2 we presented a formal and precise description of a Player in relation to the k-armed bandit problem. In this chapter and in the following chapter we introduce several bandit players and create two broad categories called *non-Bayesian bandit players* and *Bayesian bandit players.*

In this chapter we present the non-Bayesian bandit players. We further distinguish these players based upon principles and techniques used in the selection strategy and define several subcategories. For each non-Bayesian player we present the background and an algorithmic description.

For the sake of brevity we have omitted some details in the algorithmic descriptions that are in common to several algorithms described in this chapter, such as calculation of mean, variance and other trivial updating procedures.

4.1 Fixed structure deterministic learning automata

4.1.1 Tsetlin $L_{2N,2}$

The Tsetlin $L_{2N,2}$ was proposed by M.L Tsetlin in [26] and is an extension of the more simple $L_{2,2}$ automaton. The $L_{2N,2}$ is a fixed structure deterministic automaton with two available actions that operates in environments where the feedback is Bernoulli distributed.

N is called the memory that is associated with each action, and the total memory of the automaton is $2N$. The memory is represented with states, and these states are used to keep track of the automaton's previous behaviour and received feedback. The value of N determines the learning speed of the automaton [11].

The $2N$ states in the automaton are denoted $\phi_1, \phi_2, \ldots, \phi_{2N}$, and the two actions α_1 and α_2. For each action selection the automaton selects an action based upon the state it resides in. If the automaton is in the states $\phi_1, \phi_2, \ldots, \phi_N$ it selects action α_1, while if it is in states $\phi_{N+1}, \ldots, \phi_{2N}$ it selects action α_2.

Whenever the automaton receives a reward from the environment it moves deeper into the memory of the current selected action, and conversely, upon a receipt of a penalty, it moves outwards of the memory of the current selected action. For instance, if the automaton is in state ϕ_N it selects action α_1. If it receives a reward it moves to state ϕ_{N-1}, and if it receives a penalty it moves to state ϕ_{N+1}.

The algorithm for the Tsetlin automaton is listed in Algorithm 1.

Algorithm 1 Tsetlin $L_{2N,2}$

```
Require: Select current state $\phi_c$ from the interval $[N, N+1]$
loop
  if $\phi_c <= N$ then
    $i \leftarrow 1$
  else
    $i \leftarrow 2$
  end if
  Play action $i$ and receive either reward or penalty
  if reward then
    if $\phi_c <= N$ and $\phi_c > 1$ then
      $\phi_c \leftarrow \phi_c - 1$
    else if $\phi_c > N$ and $\phi_c < 2N$ then
      $\phi_c \leftarrow \phi_c + 1$
    end if
  else
    if $\phi_c <= N$ then
      $\phi_c \leftarrow \phi_c + 1$
    else if $\phi_c > N$ then
      $\phi_c \leftarrow \phi_c - 1$
    end if
  end if
end loop
```

4.2 Variable structure stochastic learning automata

Variable structure learning automata have been proved to be simple and effective learning schemes in several areas. These schemes have been researched extensively, and a detailed overview of this work can be found in [11].

In a variable structure learning automaton, each action is assigned a selection probability, and all of these selection probabilities constitute an action probability vector p that sums to unity. Initially, with no prior information about the actions, each of the actions is assigned an equal probability, where each probability is simply set to $\frac{1}{K}$, where K denotes the total number of actions.

The action selection procedure of the algorithms presented in the following sections may be compared to a roulette wheel, where each action is assigned a slice of the wheel in proportion to the probability of selecting the action. Hence, an action with a large associated probability will have a greater chance of being selected. However, due to the trade-off between exploration and exploitation, it is important that other actions besides the current perceived optimal action are explored as well. A balance between exploration and exploitation is achieved by performing a roulette wheel selection. A threshold value is randomly chosen from the interval [0,1], and all the actions' probability values are summed until this threshold is reached. When the threshold is reached, the action that corresponds to the latest added probability is chosen.

The roulette wheel selection is described in Algorithm 2 and is used in all the learning schemes presented in this section to select an action in accordance to the probability vector.

Algorithm 2 Roulette wheel selection

```
$i \leftarrow 1$
$sum \leftarrow p_1$
$rnd \leftarrow$ Random $\mathbb{R} \in [0,1]$
while $rnd > sum$ do
    $i \leftarrow i + 1$
    $sum \leftarrow sum + p_i$
end while
Select action $i$
```

4.2.1 Linear Reward-Penalty (L_{R-P})

The linear reward-penalty scheme is one of the earliest schemes considered in mathematical psychology (1958) and was studied further in the 1960s and 1970s [11].

The L_{R-P} updates its action probabilities upon both reward and penalty from the environment. If the automaton receives a reward from the environment, the corresponding probability of the selected action is increased, while the other action probabilities are decreased. Upon a penalty from the environment the automaton decreases the selection probability of the corresponding action and increases the action probabilities of the other actions.

The algorithm of the L_{R-P} scheme is shown in Algorithm 3.

Algorithm 3 L_{R-P}

```
Parameters: Real $\alpha, \beta \in (0,1)$
Initialization: $p_j \leftarrow \frac{1}{K}$ for $j \leftarrow 1$ to K
loop
  Draw randomly an action $i$ according to probabilities $p_0, \ldots, p_K$
  Receive either $reward$ or $penalty$
  if $reward$ then
    for $j \leftarrow 1$ to K do
      if $j <> i$ then
        $p_j \leftarrow (1-\alpha)p_j$
      else
        $p_i \leftarrow p_i + \alpha(1-p_i)$
      end if
    end for
  else
    for $j \leftarrow 1$ to K do
      if $j <> i$ then
        $p_j \leftarrow \frac{\beta}{K-1} + (1-\beta)p_j$
      else
        $p_i \leftarrow (1-\beta)p_i$
      end if
    end for
  end if
end loop
```

The scheme makes use of two parameters, α and β, which governs the learning speed, and are referred to as learning parameters. The α parameter is used during updating of the action selection probabilities upon a reward from the environment, while the β parameter is used during the updating of the action probabilities in the case of a penalty.

The purpose of the learning parameters is to control the learning speed of the automaton, and by setting these values sufficiently small the automaton will converge in distribution, and not to a single action [11].

In L_{R-P}, the parameters have equal values, that is $\alpha = \beta$, but it is also possible to use different values, where $\beta = \epsilon\alpha$. The scheme is then referred to as $L_{R-\epsilon P}$, and its properties are in between L_{R-P}, and L_{R-I} [11]. L_{R-I} is described in Section 4.2.2.

This L_{R-P} is the most general automaton of a group of reinforcements schemes called linear updating schemes, which in addition to L_{R-P} consists of L_{R-I} and L_{I-P}. From the L_{R-P} automaton we can obtain the other schemes, where the Linear Reward-Inaction, L_{R-I}, is obtained by setting $\beta = 0$, and Linear Inaction-Penalty, L_{I-P}, not described in this thesis, is obtained by setting $\alpha = 0$.

4.2.2 Linear Reward-Inaction (L_{R-I})

The Linear Reward-Inaction, L_{R-I}, is obtained from the L_{R-P} scheme by setting $\beta = 0$. As the name implies the action probabilities are only updated upon the receipt of a reward from the environment. Upon the receipt of a penalty the automaton keeps the action probabilities unchanged, hence the term inaction since upon receipt of a penalty no updating is performed. Updating of the action probability vector upon the receipt of a reward is performed in a similar way as in the L_{R-P} scheme.

An important property of the L_{R-I} is that the automaton converges against the optimal action with a probability as close to unity as desired in a stationary environment, when the learning parameter α is sufficiently small and the number of trials tends to infinity [11].

The L_{R-I} scheme is also applicable in environments where the feedback is defined in the interval $[0, 1]$. This requires a slightly change of the updating procedure as presented in the L_{R-P} scheme, where α is multiplied with the feedback value r. With Bernoulli distributed feedback r is either 1 or 0.

The algorithm of the L_{R-I} scheme is shown in Algorithm 4.

Algorithm 4 L_{R-I}

```
Parameters: Real $\alpha \in (0, 1)$
Initialization:$p_j \leftarrow \frac{1}{K}$ for $j \leftarrow 1$ to K
loop
  Draw randomly an action $i$ according to probabilities $p_0, \ldots, p_K$
  Receive feedback $r$
  for $j \leftarrow 1$ to K do
    if $j <> i$ then
      $p_j \leftarrow (1 - \alpha r)p_j$
    else
      $p_i \leftarrow p_i + \alpha r(1 - p_i)$
    end if
  end for
end loop
```

4.2.3 Pursuit

The Pursuit scheme computes estimated average feedback of the actions by using the history of previous selected actions and the corresponding received feedback. As with the previous variable structure automata presented in this section, the pursuit scheme uses a learning parameter α that is used to govern learning speed. The algorithm of the Pursuit scheme is shown in Algorithm 5.

Upon a feedback from the environment the pursuit scheme updates a mean estimate for the selected action. The scheme identifies the action which has received the highest average feedback from the environment, and increases the respective action's selection probability. Hence, the selection probability of the action that lead to the feedback from the environment is not necessarily increased. An important consequence of this is that if a non-optimal action should give a favorable feedback to the automaton, this would not affect the automaton in the same matter as it would in the L_{R-I} scheme.

The name of the Pursuit scheme is due to the reason that the vector of action probabilities *pursuits* the estimated optimal vector, i.e it pursuits the vector which leads to that the optimal action is chosen with probability 1.

The Pursuit scheme is shown to be faster than the L_{R-I} in terms of the convergence rate, and as the L_{R-I} scheme it also converges towards the optimal action in a stationary environment when the parameter α is set to a sufficiently small value and the number of trials tends to infinity [9].

As shown in Algorithm 5, the feedback from the environment, do not need to be constrained to the interval $[0, 1]$, since the actual updating of the probability vector is independent of the actual response, thus, making the Pursuit scheme suitable for a wide range of applications.

Algorithm 5 Pursuit

```
Parameters: Real $\alpha \in (0, 1)$
Initialization: $p_j \leftarrow \frac{1}{K}$ for $j \leftarrow 1$ to K
loop
    Draw randomly an action $i$ according to probabilities $p_0, \ldots, p_K$
    $n_i \leftarrow n_i + 1$
    Play action $i$ and receive $feedback$
    $r_i \leftarrow r_i + feedback$
    $i^* =$ argmax $\frac{r_j}{n_j}$, where $j \in \{1, \ldots, K\}$
    for $j \leftarrow 1$ to K do
        $p_j \leftarrow (1 - \alpha)p_j$
    end for
    $p_{i^*} \leftarrow p_{i^*} + \alpha$
end loop
```

4.3 Confidence interval based schemes

In Section 3.2.1 we briefly presented the frequentist approach to inference and introduced confidence statements. Confidence interval based schemes follow the frequentist approach and assign an *optimistic mean estimate* within a confidence interval to each action. The confidence interval is then used to greedily select the action with the highest optimistic estimate. The optimistic part of the mean estimate is used to achieve exploration of actions, and is gradually decreased and approaches the true value when the number of action selections is sufficiently large.

In the following we present four confidence interval based schemes, UCB1, UCB1−TUNED, UCB1−NORMAL and IntEstim.

4.3.1 UCB1

The UCB-1 scheme is based on an index-based policy by Agrawal (1995) [16], and is applicable in environments where the feedback is Bernoulli distributed. In this scheme each action is assigned an upper confidence index based upon the past reward history of the action. The upper confidence index is based on the average reward received, and the size of the one-sided confidence interval of the average reward. A reward expectation for each action is estimated from the index, and the action with the highest mean estimate is then selected [16].

The scheme must keep track of the total number of plays, denoted n, the average reward received for each action, denoted m_j, and the number of selections of an action, denoted n_j. A brief overview of the action selection procedure is shown in Algorithm 6.

Algorithm 6 UCB1

```
Initialization: Play each action once.
loop
    Play action $i$ that maximizes $m_j + \sqrt{\frac{2\ln n}{n_j}}$ for each $j \in \{1, \ldots, K\}$
    Update $m_i, n_i$ and $n$
end loop
```

We refer the reader to [16] for a more detailed description of the algorithm.

4.3.2 UCB1−TUNED

Auer et al. proved in [16] that the UCB1 can be more finely tuned, and proposed a new variant of the UCB1 scheme called UCB1-TUNED which offers improved performance compared to the UCB1 scheme.

The UCB1-TUNED is similar to the UCB1, but the $\sqrt{\frac{2\ln n}{n_j}}$ term from Algorithm 6 is replaced with $\sqrt{\frac{\ln n}{n_j}\min(1/4, V_j(n_j))}$, where $V_j(n_j)$ is defined as in Equation 4.1 [16]:

$$V_j(n_j) \overset{\text{def}}{=} \left(\frac{1}{n_j}\sum_{\tau=1}^{n_j} X_j^2, \tau\right) - \bar{X}_{j,n_j}^2 + \sqrt{\frac{2\ln n}{n_j}} \tag{4.1}$$

A brief overview of the algorithm is shown in Algorithm 7.

Algorithm 7 UCB1-TUNED

```
Initialization: Play each action once.
loop
    Play action $i$ that maximizes $m_j + \sqrt{\frac{\ln n}{n_j}\min(1/4, V_j(n_j))}$ for each $j \in \{1, \ldots, K\}$
    Update parameters $m_i, n_i$ and $n$
end loop
```

We again refer the reader to [16] for a more detailed description of the algorithm.

4.3.3 UCB1-NORMAL

The UCB1-NORMAL scheme, proposed in by Auer et al. in [16], is suitable for environments with normally distributed feedback, where the true feedback mean and variance are unknown.

Like the previous presented confidence interval schemes, the UCB1-NORMAL scheme uses the one-sided confidence interval for the average feedback received from the environment. However, in this case the feedback are known to be normally distributed, and the sample variance is used as an estimate for the unknown variance.

The algorithm of UCB1-NORMAL is listed in Algorithm 8, where the total number of action selections are denoted n, and the average feedback obtained for action j is denoted m_j. The total number of selections of an action is denoted n_j, while q_j denotes the sum of squares.

Algorithm 8 UCB1-NORMAL

```
Initialization: $n \leftarrow 1$
loop
  for $j \leftarrow 1$ to $K$ do
    if $n_j < 8 \log n$ then
      $i \leftarrow j$
      break
    else
      $x_j \leftarrow \bar{m}_j + \sqrt{16 \cdot \frac{q_j - n_j \bar{m}_j^2}{n_j - 1} \cdot \frac{\ln(n-1)}{n_j}}$
      if $x_{max} < x_j$ then
        $x_{max} \leftarrow x_j$
        $i \leftarrow j$
      end if
    end if
  end for
  Select action $i$
  Update $m_i, q_i, n_i$
end loop
```

A more detailed description of the algorithm is available in [16].

4.3.4 IntEstim

IntEstim, short for *Interval Estimation*, also uses an *optimistic mean estimate* for each action to achieve exploration, and the best action is greedily selected at each round. Many variants of IntEstim scheme have been proposed, but the variant described here is the same as used by Vogel et al. in [27].

In the variant described in [27], the feedback, like in the UCB1-NORMAL scheme, is normally distributed, and the upper bound is estimated based on this assumption. The choice of distribution may seem arbitrarily, however as argued in [27], this may be considered reasonable, when no other knowledge of the reward distributions is present. The assumption may further be supported by the so-called central limit theorem, which states that the sum of a large number of independent variables is approximately normal distributed [19].

The algorithm of IntEstim is shown in Algorithm 9. Each action is assigned a reward mean upper bound $100 \cdot (1 - \alpha)$, where α is a tuning parameter in the interval $(0, 1)$. The α parameter defines the exploration rate, where a small α-value leads to more exploration and a high value to less exploration.

For each action, denoted j, the scheme needs to keep track of the reward mean, m_j, the variance, s_j, and the number of selections of that specific action, n_j.

In Algorithm 9 the function $C(m_j, s_j, n_j)$ utilizes the inverse cumulative normal distribution function and is defined as:

$$C(m_j, s_j, n_j) \overset{\text{def}}{=} \text{Inv-normal-cdf}\left(1-\alpha, m_j, \frac{s_j}{\sqrt{n_j}}\right) \tag{4.2}$$

From Equation 4.2 we can recognize $(1-\alpha)$ from above which determines the upper bound on the reward mean, while m_j and s_j are the sample mean and variance for the normal distribution associated with action j in the algorithm.

Algorithm 9 IntEstim

```
Parameters: Real $\alpha \in (0, 1)$
Initialization: Select a random action at least twice and update corresponding $m_j, s_j$ and $n_j$
loop
  for $j \leftarrow 1$ to $K$ do
    if $n_j > 0$ then
      if $n_j > 1$ then
        $price \leftarrow C(m_j, s_j, n_j)$
      else
        $price \leftarrow C\left(m_j, \frac{s_j}{\text{number of actions played twice}}, 1\right)$
      end if
      if $price > price_{max}$ then
        $price_{max} \leftarrow price$
        $i \leftarrow j$
      end if
    end if
  end for
  if actions selected < K then
    $uPrice \leftarrow C\left(\frac{\sum_{i=1}^{K} m_i}{\text{number of actions played}}, \frac{\sum_{i=1}^{K} s_i}{\text{number of actions played twice}}, 1\right)$
    if $uPrice > Price$ then
      $price_{max} \leftarrow uPrice$
      $index \leftarrow$ random $\mathbb{Z} \in$ [0, K - observed actions - 1]
      for $j \leftarrow 1$ to $K$ do
        if $n_j = 0$ then
          if $uCount = index$ then
            $i \leftarrow j$
          else
            $uCount \leftarrow uCount + 1$
          end if
        end if
      end for
    end if
  end if
  Select action $i$
  Update $m_i, s_i$ and $n_i$ for action $i$ with the received feedback
end loop
```

4.4 Exponential weight schemes

4.4.1 Exp3

Exp3, short for *Exponential-weight algorithm for Exploration and Exploitation*, is a variant of an algorithm called *Hedge* [28]. The Exp3 scheme was originally proposed by Auer et al. in [28], along with several other related schemes with varying bounds on the achieved regret. The scheme assumes that the feedback is in the range $[0, 1]$.

The algorithmic description of the Exp3 scheme is shown in Algorithm 10. The probability of selecting an action j at time instant t is defined by [28]:

$$p_j(t) = (1-\alpha)\frac{w_j(t)}{\sum_{i=1}^{K} w_i(t)} + \frac{\alpha}{K} \tag{4.3}$$

The total number of actions is denoted K, α is a tuning parameter defined in the interval $(0, 1]$ and w_j is a weight value associated with each action.

The probability distribution in Equation 4.3, is a mixture of the probability mass exponential in the estimated cumulative feedback of an action and the uniform distribution, and is used to ensure that all actions are played [28].

Algorithm 10 Exp3

```
**Parameters:** Real $\alpha \in (0, 1]$
**Initialization:** $w_j \leftarrow 1$ **for** $j \leftarrow 1$ **to** $K$
**loop**
  **for** $j \leftarrow 1$ **to** $K$ **do**
    $p_j \leftarrow (1-\alpha)\frac{w_j}{\sum_{x=1}^{K} w_x} + \frac{\alpha}{K}$
  **end for**
  Draw randomly an action $i$ according to probabilities $p_0, \ldots, p_K$
  Select action $i$, and receive feedback $x_i$
  **for** $j \leftarrow 1$ **to** $K$ **do**
    **if** $j = i$ **then**
      $\hat{x}_j \leftarrow \frac{x_i}{p_j}$
    **else**
      $\hat{x}_j \leftarrow 0$
    **end if**
    $w_j \leftarrow w_j \exp\left(\frac{\alpha \hat{x}_j}{K}\right)$
  **end for**
**end loop**
```

4.5 Pricing schemes

4.5.1 Poker

Poker, which is an abbreviation for *Price of Knowledge and Estimated Reward*, was proposed by Vogel et al. in [27]. Poker is based upon the assumption that the received feedback is normally distributed. The scheme relies on three principles: pricing of knowledge, estimation of unselected actions and utilizing of the horizon in the selection procedure, i.e utilize the number of trials left.

A price is assigned to the knowledge gained by selecting a particular action. This is used to focus exploitation on actions, where the potentially most valuable information may be gained. For instance, to focus exploitation on actions with few selections, or on actions that seem to be very close to the currently perceived optimal action. In this way a balance between the exploration vs. exploitation is achieved.

Unselected actions could potentially, to some extent, be estimated from the gained information from actions that have already been selected. As argued in [27] this is useful when there are fewer trials than there are number of actions. However, in this thesis this is not an important feature, as the number of trials is large compared to the number of actions.

Utilizing of the horizon in the selection procedure is feasible when there are few trials left in order to focus on exploiting the already obtained knowledge, as new information has less value at this point [27].

The algorithm of the Poker scheme is listed in Algorithm 11 [27]. The total feedback received from the environment for an action j is denoted r_j, the sum of the squared feedback is denoted r_j^2, while the number of selections of each action is denoted n_j.

The term $\hat{E}_{k,n[k]>0[\mu[k]]}$ denotes the empirical feedback mean over the selected actions, while $\hat{E}_{k,n[k]>0[\sigma[k]]}$ denotes the empirical variance over selected actions.

Algorithm 11 POKER

```
**Initializing:** $n_j \leftarrow r_j \leftarrow r_j^2 \leftarrow 0$ **for** $j \leftarrow 1$ **to** $K$
**Initializing:** Select two random actions at least twice, and update the corresponding $r_j, r_j^2$ and $n_j$
**for** $t \leftarrow 1$ **to** $T$ **do**
    $q \leftarrow |\{i, r_i > 0\}|$
    $i_0 \leftarrow \text{argmax}(u_i)$
    $i_1 \leftarrow j$ such that $|i, u_i > u_j| = \sqrt{q}$
    $\delta_\mu \leftarrow (\mu_{i_0} - \mu_{i_1})/\sqrt{q}$
    $\mu^* \leftarrow \text{argmax}(\mu_i)$
    $p_{max} \leftarrow -\infty$
    $i \leftarrow$ UNDEFINED
    **for** $j \leftarrow 1$ **to** $K$ **do**
        **if** $n_j > 0$ **then** $\mu \leftarrow u_j$ **else** $\mu \leftarrow \hat{E}_{k,n[k]>0[\mu[k]]}$ **endif**
        **if** $n_j > 1$ **then** $\sigma \leftarrow \sigma_j$ **else** $\sigma \leftarrow \hat{E}_{k,n[k]>0[\sigma[k]]}$ **endif**
        $p \leftarrow \mu + \delta_\mu(T-t)\int_{\mu*+\delta_\mu}^{\infty} N\left(x, u_j, \frac{\sigma_j}{\sqrt{n_j}}\right) dx$
        **if** $p > p_{max}$ **then** $p_{max} \leftarrow p, i \leftarrow j$ **endif**
    **end for**
    Select action $i_{max}$ and receive reward $x_i$
    $r_i \leftarrow r_{i_{max}} + x_i$
    $r_i^2 \leftarrow r_{i_{max}}^2 + x_i^2$
    $n_i \leftarrow n_i + 1$
**end for**
```

We refer the reader to [27] for a more detailed description of the algorithm.

4.6 Greedy schemes

The classical ε-GREEDY rule is a simple and well-known strategy for the k-armed bandit problem, first described by Watkins in [29]. In the ε-GREEDY strategy the action with the highest feedback probability is selected with probability $1 - \varepsilon$, and a random arm is selected with probability ε.

There are two weaknesses with this scheme when applied in stationary environments. The first is that all actions are explored with the same probability, which is not optimal when a certain amount of knowledge has been obtained from each action. It would probably make more sense to focus the exploration on the actions which seem to be close to the perceived optimal action with regards to the distributions associated with the actions.

The second weakness is that the exploration is made with an constant effort ε, which in a stationary environment is not optimal when a certain level of knowledge has been obtained about the available actions. The obvious solution to this problem would be to slowly decrease the exploration rate as more confidence in the optimal action is built.

Several variants of the ε-GREEDY rule exist, but in the following we present the ε_n-GREEDY as proposed by Auer et al. in [16].

4.6.1 ε_n-GREEDY

Auer et al. in [16] proposed the ε_n-GREEDY scheme where the exploration rate is decreasing as the number of trials is increasing, and the rate of exploration is calculated as follows [16]:

$$\varepsilon_n(n) \overset{\text{def}}{=} \min\left(1, \frac{cK}{d^2 n}\right) \tag{4.4}$$

This function requires the parameters d, which denotes the difference between the best and the second best arm, and c which is used for tuning the learning rate of the algorithm.

The algorithm of ε_n-GREEDY is shown in Algorithm 12. In each action selection the best action is selected with probability $1-\varepsilon_n$ and a random action with probability ε_n. For each action j the algorithm needs to keep track of the sum of received feedback r_j, and the number of selections n_j, in addition to the total number of selections across all actions, denoted n.

Algorithm 12 ε_n-GREEDY

```
Initialization: $r_j \leftarrow 0, n_j \leftarrow 0$ for $j \leftarrow 1$ to $K$
loop
    Let $k$ be the action with the highest feedback mean
    Select action $k$ with probability $1 - \varepsilon_n$, and a random action with probability $\varepsilon_n$
    Update $r_j$ and $n_j$ with received feedback for selected action
end loop
```

Chapter 5

Bayesian bandit players - The Bayesian Learning Automaton family

In this chapter we introduce the Bayesian bandit players from the Bayesian Learning Automaton family, originally proposed by Granmo in the paper *The Bayesian Learning Automaton - Empirical Evaluation with Two-Armed Bernoulli Bandit Problems* [5], where the first member of the family was introduced. We both present the BLA proposed by Granmo and extend the BLA family with three new members designed for Poisson and normally distributed feedback.

In Section 5.1 we illustrate by an example how a conjugate prior distribution is used in a Bayesian Learning Automaton to infer on the unknown parameters θ, and how the posterior distribution is applied in a Bayesian Learning Automaton during the action selection procedure.

The concepts introduced in Sections 3.3 and 3.4 are used extensively in this chapter, and in Sections 5.2 through 5.5 we use these concepts to derive the algorithms of the members of the Bayesian Learning Automaton family.

5.1 Bayesian inference in the general k-armed bandit problem

In the following sections we describe how the conjugate prior distributions are applied in the Bayesian Learning Automata. We illustrate this by examples, where we assume Bernoulli distributed feedback from the environment. In Section 5.1.1 we investigate the development of a simplified inference process of the parameter p of the Bernoulli distribution. We then in Section 5.1.2 see how the posterior beta distribution is applied in the action selection procedure.

5.1.1 Bayesian inference about the unknown parameters θ

A Bayesian Learning Automaton associates a conjugate prior distribution to each action it may apply on the environment. More specifically, this means that each action is associated with the hyperparameters of the conjugate prior distribution.

In accordance to the terminology presented in Section 3.2, the distribution of the environment constitutes the likelihood function, also referred to as the sampling function of the feedback, and the prior distribution infer on the unknown parameter θ of the likelihood function. From Table 3.1 we see that the beta distribution is the conjugate prior for the Bernoulli distribution.

As noted earlier, with Bernoulli distributed feedback, the environment will respond with a reward with probability p and a penalty with probability $1 - p$. Upon receipt of a response the automaton updates its beliefs about the action that led to the response, and hence the beta distribution associated with that action. We defer the details of how the updating of the prior distribution is performed to Section 5.2.

In the following we investigate the development of a simplified inference process with Bernoulli distributed feedback, which is depicted in Figure 5.1, where the reward probability p is set to 0.7.

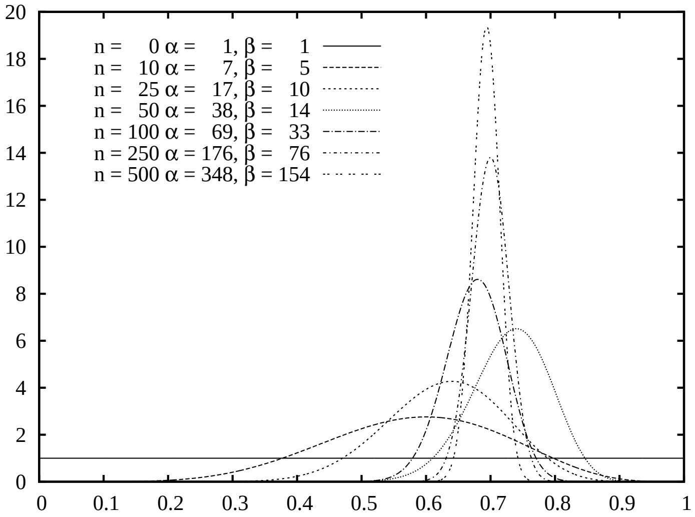


Figure 5.1: Bayesian inference about Bernoulli parameter p

In Figure 5.1 we see how the shape of a beta distribution changes as the number of trials increases and new evidence is obtained. The objective of the inference process is to identify the value of parameter p of the likelihood function. As mentioned in Section 3.5 the expected mean value of a beta distribution will approach the true value of the p parameter of the Bernoulli distribution as the number of trials increases.

At the start of the experiment, $n = 0$, there does not exist any prior belief about the parameter p, and a uniform non-informative beta prior distribution is chosen with parameters $\alpha = 1$ and $\beta = 1$. Such a prior weights every possible value of the parameter p equally, which means that according to the prior distribution at $n = 0$, any value in the range [0,1] is equally likely for the parameter p.

As the number of trails increases we observe how the belief about the unknown parameter p is concentrated to a specific range, and slowly is focused to a specific value as the number of trials n increases. We can also observe how the expected mean value, $(\frac{\alpha}{\alpha+\beta})$, initially shifts back and forth, but gradually is focused towards the true unknown value of the Bernoulli parameter p, namely 0.7.

The principles illustrated in the simplified inference process showed in this section apply to all the automata in the Bayesian Learning Automaton family. Also note, that in relation with the k-armed bandit problem we typically have several actions, where the environment associates a Bernoulli distribution with each action. Hence, the automata perform inferences about several p parameters, and the inference process of a parameter of a given action is performed when the action is applied and a feedback is received. We explore the action selection procedure in the following section.

5.1.2 Action selection in the Bayesian Learning Automaton

The conjugate prior distribution serves two purposes in the BLA. It describes the belief of the unknown parameters θ, and it is used during action selection in the automaton. Action selection is performed by sampling a value x from the distribution associated with each action, and the action that returns the highest sample value is selected.

In the following example we again consider an environment with Bernoulli distributed feedback and two available actions. A uniform non-informative prior beta distribution with initial parameters $\alpha = \beta = 1$ is associated with each action. This means that no prior belief about the value of parameter p exists, or equivalent that the initial belief is that p can take any value in the interval $[0, 1]$. The overall goal is to identify the action that on average returns the highest feedback, i.e we want to identify the action with the highest associated p value.

The beta distributions associated with the two actions a_1 and a_2 are used to make inference about the parameters p_1 and p_2, respectively. Figure 5.2 shows the development of the inferred beta distributions on the parameters $p_1 = 0.7$ and $p_2 = 0.5$, plotted after $n = 50$ and $n = 200$ trials on action a_1 and a_2, respectively.

At $n = 50$ we can observe how the probability distribution for p_1 and p_2 overlap, and even if the distribution associated with a_1 will return the highest random samples in most cases, there is still a certain probability of selecting action a_2, which at this point most likely is inferior. Because of this, actions which yield higher expected values from their associated beta distributions are selected more than those that yield lower expected values. Also, an inferior action is still selected with a certain probability as long as there is a certain probability that this action actually is the optimal action.

From Figure 5.2 we observe that at $n = 100$ the overlap is significantly smaller than it was at $n = 50$, which means that at this point more certainty about the true values of the parameters has been achieved, and less effort is put on exploring the perceived inferior action. This pattern develops further as action selections are performed and feedback from the environment is received. Furthermore, at a certain point there will have been gathered enough samples such that these distributions no longer overlap, and at this point only the optimal action will be selected.

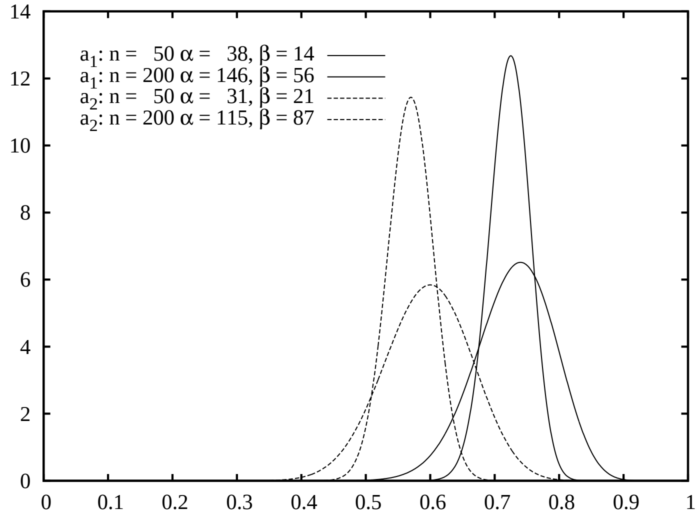


Figure 5.2: Bayesian inference and action selection with Bernoulli distributed feedback

As demonstrated with the example above, by sampling from the inferred probability distribution over each action's true parameter p the Bayesian Learning Automaton is able to balance its exploration vs. exploitation effort.

Note that Figure 5.2 does not show a realistic development of the inferred distributions on p_1 and p_2, but merely serves the purpose as an example of how arm selection is performed in a BLA. The reason for this is that since actions with higher expected values are selected more than those with lower, the distributions associated with actions that on average yields higher feedback develop faster. Therefore, typically an action that yields lower average feedback would develop a distribution that is lower and wider than an action that on average yields a higher feedback.

The principles of the action selection procedure presented in this section are common to all the members of the Bayesian Learning Automaton family.

5.2 BLA Bernoulli

The Bayesian Learning Automaton (BLA) for Bernoulli distributed feedback was presented by Granmo in [5] and marked the beginning of a new family of learning automata. In this thesis we refer to this automaton as BLA Bernoulli in order to distinguish the members of the Bayesian Learning Automaton family from each other.

As reported in Table 3.1, the beta distribution is the conjugate prior for the Bernoulli distribution. In this section we identify the beta distribution as the conjugate prior for the Bernoulli distribution by using the theory from Section 3.4.3 and derive the updating rules of the hyperparameters of the beta distribution. These rules are an essential part of the algorithm of BLA Bernoulli which we present in Section 5.2.2.

5.2.1 Theoretical background

The Bernoulli distribution is a discrete probability distribution and is defined as [25]:

$$p(y|\theta) = \theta^y(1-\theta)^{1-y} \tag{5.1}$$

$$\begin{aligned} y &= \{0,1\} \\ \theta &\in [0,1] \end{aligned}$$

From the definition in Equation 5.1 it implies that y obtains the value 1 with probability θ and 0 with probability $1-\theta$.

The Bernoulli distribution is part of the exponential family, and therefore we can express the Bernoulli distribution in exponential form. Thus, relating Equation 5.1 to Equation 3.14, we obtain the following equalities [25]:

$$\begin{aligned} f(y) &= 1 \\ g(\theta) &= 1-\theta \\ u(y) &= y \\ \phi(\theta) &= \log\frac{\theta}{1-\theta} \end{aligned}$$

Using this, we express the Bernoulli distribution in exponential family form as [25]:

$$p(y|\theta) \propto (1-\theta)e^{y\log\frac{\theta}{1-\theta}} \tag{5.2}$$

The general exponential form of a prior distribution was shown in Equation 3.15. The conjugate prior of the Bernoulli distribution can thus be expressed in exponential family form as [19]:

$$\begin{aligned} p(\theta) &\propto g(\theta)^{\eta} e^{(\phi(\theta)^{\mathcal{T}} \nu)} \\ &\propto (1-\theta)^{\eta} e^{\nu \log \frac{\theta}{1-\theta}} \end{aligned} \tag{5.3}$$

Having identified the natural conjugate prior of the Bernoulli distribution, expressed in exponential family form, we can re-write the prior from Equation 5.3 into the functional form of the Bernoulli distribution. The natural conjugate prior of the Bernoulli distribution is defined as:

$$p(\theta) \propto \theta^{\nu}(1-\theta)^{\eta-\nu} \tag{5.4}$$

By examining the conjugate prior in Equation 5.4 we see that it has the same functional form (shape) as the beta distribution [19]:

$$Beta(\theta|\alpha,\beta) = c\ \theta^{\alpha-1}(1-\theta)^{\beta-1} \tag{5.5}$$

where,

$$\begin{aligned} c &= \frac{\Gamma(\alpha+\beta)}{\Gamma(\alpha)\Gamma(\beta)} \\ \alpha &> 0 \\ \beta &> 0 \end{aligned}$$

By comparing the beta distribution defined in Equation 5.5 to the conjugate prior defined in Equation 5.4 we get the following connections:

$$\begin{aligned} \nu &= \alpha - 1 \\ \eta - \nu &= \beta - 1 \end{aligned}$$

We can therefore use the beta distribution as conjugate prior for the Bernoulli distribution, and state the following:

$$p(\theta) = Beta(\theta|\alpha,\beta) \tag{5.6}$$

Multiplying the conjugate prior with the likelihood function yields the posterior distribution, according to Bayes' theorem [19]:

$$\begin{aligned} p(\theta|y) \quad &\propto p(\theta) \times p(y|\theta) \\ &\propto \theta^{\alpha-1}(1-\theta)^{\beta-1}\theta^{y}(1-\theta)^{1-y} \\ &\propto \theta^{\alpha-1+y}(1-\theta)^{\beta-1+1-y} \end{aligned} \tag{5.7}$$

Thus, as seen from Equation 5.7, the posterior distribution has the same parametric form as the prior. From Equation 5.7 we see that the hyperparameters of the posterior distribution are updated in the following way [19]:

$$\begin{aligned} \alpha_n &= \alpha_{n-1} + y \\ \beta_n &= \beta_{n-1} + 1 - y \end{aligned} \tag{5.8}$$

From the updating rules above we see that when $y = 1$, the α parameter is incremented by one, while the β parameter stays the same. When $y = 0$, we observe the opposite, the α parameter stays the same, while β is incremented by one.

5.2.2 Algorithm description

The algorithm of BLA Bernoulli is listed in Algorithm 13. The parameters α_j and β_j are associated with each action $j \in \{1, \ldots, K\}$ and by choosing a non-informative Beta distribution these are initialized $\alpha_j = \beta_j = 1$.

Algorithm 13 BLA Bernoulli

```
1:  Initialization: $\alpha_j \leftarrow 1$, $\beta_j \leftarrow 1$, for each $j \in \{1, \ldots, K\}$
2:  loop
3:      Sample $x_j$ from $Beta(\alpha_j, \beta_j)$, for each $j \in \{1, \ldots, K\}$
4:      $i =$ argmax $x_j$, where $j \in \{1, \ldots, K\}$
5:      Select action $i$
6:      Receive reward or penalty
7:      if reward then
8:          $\alpha_i \leftarrow \alpha_i + 1$
9:      end if
10:     if penalty then
11:         $\beta_i \leftarrow \beta_i + 1$
12:     end if
13: end loop
```

Action selection is performed at lines 3-5, and consists of sampling x_j from the beta distribution for each action j, and selecting action i which returns the highest sample value.

When an action i has been selected, and either a reward or a penalty has been received, the hyperparameters are updated at lines 7-12. This consists of incrementing α_i by 1 in the case of a reward and incrementing β_i in the case of a penalty.

5.3 BLA Poisson

BLA Poisson is designed for environments where the feedback is Poisson distributed. As shown in Table 3.1, the gamma distribution is the conjugate prior for the Poisson distribution. Below we identify the gamma distribution as the conjugate prior for the Poisson distribution by using the theory presented in Section 3.4.3 and derive the updating rules of the hyperparameters. These rules are an essential part of the algorithm presented in Section 5.3.2.

5.3.1 Theoretical background

The Poisson distribution is a discrete probability distribution, which is used to model the number of events that occurs within a time-interval when the event occurs with a certain average rate, and each event is independent of the previous event. If a single datapoint y is Poisson distributed with rate λ, then the probability of observing y is [23]:

$$\begin{aligned} p(y|\lambda) &= \frac{\lambda^y e^{-\lambda}}{y!} \\ p(y|\lambda) &\propto \lambda^y e^{-\lambda} \qquad (5.9) \\ \\ \lambda &\in \mathbb{R}^+ \end{aligned}$$

The Poisson distribution belongs to the exponential family and relating the likelihood from Equation 5.9 to the exponential family form given in Equation 3.14, gives the following equalities [25]:

$$\begin{aligned} f(y) &= \frac{1}{y!} \\ g(\lambda) &= e^{-\lambda} \\ u(y) &= y \\ \phi(\lambda) &= \log \lambda \end{aligned}$$

Using this, we can rewrite the Poisson likelihood in exponential family form as [23]:

$$p(y|\lambda) \propto e^{-\lambda} e^{y \log \lambda} \qquad (5.10)$$

From Equation 3.15 we can define the natural conjugate prior distribution for the Poisson distribution. Using the equalities from above, the conjugate prior for the Poisson distribution can be expressed in exponential family form as [23]:

$$\begin{aligned} p(\lambda) &\propto g(\lambda)^{\eta} e^{(\phi(\lambda)^{\mathcal{T}} \nu)} \\ &\propto e^{(-\lambda)\eta} e^{\nu \log \lambda} \qquad (5.11) \end{aligned}$$

We can re-write the conjugate prior in Equation 5.11 into the functional form of the Poisson distribution given in Equation 5.9, which yields [19]:

$$p(\lambda) \propto \lambda^{\nu} e^{-\eta\lambda} \tag{5.12}$$

By examining the conjugate prior in Equation 5.12 closer, we see that it has the same functional form (shape) as the gamma distribution, which is defined as [23]:

$$Gamma(\lambda|\alpha, \beta) = c\lambda^{\alpha-1} e^{-\beta\lambda} \tag{5.13}$$

where,

$$\begin{aligned} c &= \frac{\beta^{\alpha}}{\Gamma(\alpha)} \\ \alpha &> 0 \\ \beta &> 0 \end{aligned}$$

The α parameter is called the shape parameter, and the β parameter is referred to as the inverse scale parameter. Sometimes the scale parameter, denoted θ here for clarity, is used instead of the inverse scale parameter in the definition of the gamma distribution. The inverse scale parameter is then naturally defined as $\beta = \frac{1}{\theta}$ [24].

Comparing Equation 5.12 and Equation 5.13, we find the following:

$$\begin{aligned} \nu &= \alpha - 1 \\ \eta &= \beta \end{aligned}$$

Thus, we can then state the following:

$$p(\lambda) = Gamma(\lambda|\alpha, \beta) \tag{5.14}$$

By multiplying the conjugate prior with the likelihood function we obtain the posterior distribution, according to Bayes' theorem [19]:

$$\begin{aligned} p(\lambda|y) \quad & \propto p(\lambda) \times p(y|\lambda) \\ & \propto \lambda^{\alpha-1} e^{-\beta\lambda} \lambda^{y} e^{-\lambda} \\ & \propto \lambda^{\alpha-1+y} e^{-\lambda(\beta+1)} \end{aligned} \tag{5.15}$$

Comparing the posterior distribution in Equation 5.15 to the prior distribution in Equation 5.13, we see that the posterior distribution has the same parametric form as the the prior distribution, where the hyperparameters of the posterior distribution are updated from the prior distribution in the following way [19]:

$$\alpha_n = \alpha_{n-1} + y$$
$$\beta_n = \beta_{n-1} + 1$$

By examining the equations, we see that the α parameter denotes the sum of all random samples sampled from the Poisson distribution, while the β parameter denotes the number of observations.

5.3.2 Algorithm description

The algorithm of BLA Poisson is shown in Algorithm 14. Each action j is associated with the hyperparameters, α_j and β_j, of the gamma distribution.

Algorithm 14 BLA Poisson

```
1: Initialization: Set $\alpha_j$ and $\beta_j$ according to prior belief about $\lambda_j$
2: loop
3:     Sample $x_j$ from $Gamma(\alpha_j, \beta_j)$, for each $j \in \{1, \dots, K\}$
4:     $i$ = argmax $x_j$ where $j \in \{1, \dots, K\}$
5:     Select action $i$ and receive feedback $r$
6:     $\alpha_i \leftarrow \alpha_i + r$
7:     $\beta_i \leftarrow \beta_i + 1$
8: end loop
```

The initialization phase consists of initializing the hyperparameters α_j and β_j associated with each action in accordance with the prior belief about the unknown rate parameter λ.

Similarly as in the case with BLA Bernoulli, action selection is performed by sampling x_j from the inferred distribution about the parameter λ for each action j, and the action with the highest sample value is selected.

Finally, the hyperparameters α_i and β_i are updated in accordance with the received feedback r at the lines 6-7.

5.4 BLA Normal known σ^2

BLA Normal with unknown μ and known σ^2 is designed for environments with normally distributed feedback. From Table 3.1 we see that the conjugate prior for a normal distribution, where μ is unknown, but σ^2 is known, is the normal distribution itself. Below we only make a rough identification of the conjugate prior and derivation of the updating rules of the hyperparameters. The algorithmic description of BLA Normal is presented in Section 5.4.2.

5.4.1 Theoretical background

We again consider a single datapoint y that is normal distributed with mean μ and variance σ^2. The parameter we are interested in estimating in this case is the mean μ, where the variance σ^2 is assumed known. The likelihood function is defined as [19]:

$$p(y|\mu) = \frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{1}{2\sigma^2}(y-\mu)^2} \tag{5.16}$$

Since $\frac{1}{\sqrt{2\pi}\sigma}$ does not influence the shape of $p(y|\mu)$ it may be regarded as a constant, and we therefore rewrite Equation 5.16 as [19]:

$$p(y|\mu) \propto e^{-\frac{1}{2\sigma^2}(y-\mu)^2} \tag{5.17}$$

The Normal distribution is part of the exponential family, and for any member of the exponential family we have already defined in Equation 3.15 the form of a conjugate prior. Thus, we seek a prior distribution that is a conjugate prior for the normal distribution. We omit a detailed derivation for identifying the conjugate prior, and only state that the conjugate prior is the normal distribution with mean m and variance s^2 [19]:

$$p(\mu) \propto e^{-\frac{1}{2s^2}(\mu-m)^2} \tag{5.18}$$

By multiplying the conjugate prior with the likelihood function, we obtain the posterior distribution, according to Bayes' theorem [19]:

$$
\begin{aligned}
p(\mu|y) &\propto p(\mu) \times p(y|\mu) \\
&\propto e^{-\frac{1}{2s^2}(\mu-m)^2} \times e^{-\frac{1}{2\sigma^2}(y-\mu)^2} \\
&\propto e^{-\frac{1}{2}\left[\frac{(\mu-m)^2}{s^2}+\frac{(y-\mu)^2}{\sigma^2}\right]} \\
&\propto e^{-\frac{1}{2}\left[\frac{\sigma^2(\mu^2-2\mu m+m^2)+s^2(y^2-2y\mu+\mu^2)}{\sigma^2 s^2}\right]} \\
&\propto e^{-\frac{1}{2}\left[\frac{(\sigma^2+s^2)\mu^2-2(\sigma^2 m+s^2 y)\mu+m^2\sigma^2+y^2 s^2}{\sigma^2 s^2}\right]} \\
&\propto e^{-\frac{1}{\frac{2\sigma^2 s^2}{\sigma^2+s^2}}\left[\mu^2-2\frac{(\sigma^2 m+s^2 y)}{\sigma^2+s^2}\mu+\left(\frac{(\sigma^2 m+s^2 y)}{\sigma^2+s^2}\right)^2\right]} \\
&\propto e^{-\frac{1}{2\frac{\sigma^2 s^2}{\sigma^2+s^2}}\left[\mu-\frac{\sigma^2 m+s^2 y}{\sigma^2+s^2}\right]^2}
\end{aligned}
\tag{5.19}
$$

Comparing the posterior distribution in Equation 5.19 with the prior distribution in Equation 5.18 we see that the posterior distribution has the same parametric form as the prior distribution, where the hyperparameters of the posterior distribution are updated from the prior distribution in the following way [19]:

$$
\begin{aligned}
m_n &= \frac{(\sigma^2 m_{n-1} + s_{n-1}^2 y)}{\sigma^2 + {s_{n-1}}^2} \\
(s_n)^2 &= \frac{\sigma^2 s_{n-1}^2}{(\sigma^2 + s_{n-1}^2)}
\end{aligned}
$$

5.4.2 Algorithm description

A description of the Bayesian Learning Automaton designed for normally distributed feedback with known variance is listed in Algorithm 15.

Algorithm 15 BLA Normal known σ^2

```
1: **Initialization:** Set $m_j$ and $s_j^2$ according to prior belief of $\mu_j$
2: **loop**
3:     Sample $x_j$ from $N(m_j, s_j^2)$, for each $j \in \{1, \ldots, K\}$
4:     $i = \text{argmax } x_j$, where $j \in \{1, \ldots, K\}$
5:     Select action $i$ and receive feedback $r$
6:     $m_i \leftarrow (\sigma^2 m_i + s_i^2 r)/(\sigma^2 + s_i^2)$
7:     ${s_i}^2 \leftarrow (\sigma^2 s_i^2)/(\sigma^2 + s_i^2)$
8: **end loop**
```

For each action $j \in \{1, \ldots, K\}$ the automaton needs to keep track of the hyperparameters m_j and s_j^2, which represent the mean and variance, respectively. Initialization of the hyperparameters m_j and s_j^2 for each action $j \in \{1, \ldots, K\}$ is done according to the prior belief about the unknown parameter m_j.

A value x_j is randomly drawn according from a normal distribution $N(m_j, s_j^2)$ associated with j for each action $j \in \{1, \ldots, K\}$, and the action associated with the highest drawn x_j value is selected.

Finally, the hyperparameters are updated with the received feedback r as listed at lines 6-7, and the procedure with sampling and updating of hyperparameters is repeated.

5.5 BLA Normal unknown σ^2

BLA Normal with unknown μ and σ^2 is designed for environments with normally distributed feedback, but in contrast to the BLA presented in Section 5.4 the σ^2 parameter of the feedback distribution is unknown. From Table 3.1 we see that the conjugate prior is in this case the normal-scaled-inverse-χ^2. The algorithmic description of BLA Normal with unknown μ and σ^2 is shown in Section 5.5.2.

5.5.1 Theoretical background

In this case we have two unknown parameters, the mean μ and the variance σ^2, in contrast to the other scenarios presented earlier where there is only one unknown parameter. Despite having two unknown parameters we only want to draw conclusions based upon the mean μ. The unknown parameter σ^2 still plays a part in our estimate, but we are not interested in its actual value, thus is referred to as a nuisance parameter.

The likelihood of observing a sample y is defined as [23]:

$$\begin{aligned} p(y|\mu,\sigma^2) &\propto \sigma^{-1}e^{-\frac{1}{2\sigma^2}(y-\mu)^2} \\ &= (\sigma^2)^{-\frac{1}{2}}e^{-\frac{1}{2\sigma^2}(y-\mu)^2} \end{aligned} \tag{5.20}$$

By examining the likelihood in Equation 5.20 we see that the conjugate prior distribution must have the form $p(\mu,\sigma^2)$, which is the product of the marginal probability of $p(\sigma^2)$ and the conditional probability of μ given σ^2, $p(\mu|\sigma^2)$. Thus, we are here seeking a joint prior distribution of the form:

$$p(\mu,\sigma^2) \propto p(\sigma^2)p(\mu|\sigma^2) \tag{5.21}$$

The marginal distribution of σ^2 is the scaled inverse-χ^2 distribution[23]:

$$\begin{aligned} p(\sigma^2) = Inv - \chi^2(\sigma^2|\nu_0,\sigma_0^2) &= \frac{(\frac{\nu_0}{2})^{\frac{\nu_0}{2}}}{\Gamma(\frac{\nu_0}{2})}s_0^{\nu}\sigma^{\frac{-\nu_0}{2}+1}e^{\frac{-\nu_0\sigma_0^2}{2\sigma^2}} \\ &\propto \sigma^{\frac{-\nu_0}{2}+1}e^{\frac{-\nu_0\sigma_0^2}{2\sigma^2}} \end{aligned} \tag{5.22}$$

where σ_0^2 is the scale parameter and ν_0 is the degrees of freedom.

The scaled inverse-χ^2 distribution is a special case of the inverted-gamma distribution, when $\alpha = \frac{\nu_0}{2}$ and $\beta = \frac{\nu_0}{2}\sigma_0^2$.

The conditional distribution of μ given σ^2 is normally distributed and defined as [23]:

$$p(\mu|\sigma^2) = N(\mu_0, \frac{\sigma^2}{\kappa_0}) \propto \sigma^{-1}e^{-\frac{1}{2\sigma^2}\kappa_0(\mu-\mu_0)^2} \tag{5.23}$$

Using these definitions we obtain the joint prior distribution according to Equation 5.21:

$$
\begin{aligned}
p(\mu, \sigma^2) &\propto p(\sigma^2)p(\mu|\sigma^2) \\
&\propto \sigma^{-\frac{\nu_0}{2}+1} e^{\frac{-\nu_0\sigma_0^2}{2\sigma^2}} \sigma^{-1} e^{-\frac{1}{2\sigma^2}\kappa_0(\mu-\mu_0)^2} \\
&\propto \sigma^{-1}(\sigma^2)^{-(\frac{\nu_0}{2}+1)} e^{-\frac{1}{2\sigma^2}[\nu_0\sigma_0^2+\kappa_0(\mu-\mu_0)^2]}
\end{aligned}
\tag{5.24}
$$

In order to obtain the joint posterior distribution we must multiply the likelihood from Equation 5.20 with the join prior distribution from Equation 5.24 [23]:

$$
\begin{aligned}
p(\mu, \sigma^2|y) &\propto p(\mu, \sigma^2)p(y|\mu, \sigma^2) \\
&\propto \sigma^{-1}(\sigma^2)^{-(\frac{\nu_0}{2}+1)} e^{-\frac{1}{2\sigma^2}[\nu_0+\sigma_0^2+\kappa_0(\mu-\mu_0)^2]} \sigma^{-1} e^{-\frac{1}{2\sigma^2}(y-\mu)^2} \\
&\propto \sigma^{-1}(\sigma^2)^{-(\frac{(\nu_0+1)}{2}+1)} e^{\left[\frac{\nu_0\sigma_0^2+\frac{\kappa_0(y-\mu_0)^2}{1+\kappa_0}+(1+\kappa_0)(\mu-\frac{\mu_0\kappa_0+y}{1+\kappa_0})^2}{2\sigma^2}\right]}
\end{aligned}
\tag{5.25}
$$

We let κ_0 denote the number of observations prior to the current update, i.e in relation to current observation κ_n we let κ_0 be denoted κ_{n-1}. Furthermore we let ν_0, σ_0^2 and μ_0 denote the prior values, thus we rewrite them as ν_{n-1}, σ_{n-1}^2 and μ_{n-1}. Using this and comparing the joint prior distribution from Equation 5.24 to the joint posterior distribution in Equation 5.25 we see that the hyperparameters are updated in the following way [23]:

$$
\begin{aligned}
\kappa_n &= \kappa_{n-1} + 1 \\
\nu_n &= \nu_{n-1} + 1 \\
\mu_n &= \frac{\kappa_{n-1}}{\kappa_{n-1}+1}\mu_{n-1} + \frac{y}{\kappa_{n-1}+1} \\
\nu_n\sigma_n^2 &= \nu_{n-1}\sigma_{n-1}^2 + \frac{\kappa_{n-1}}{\kappa_{n-1}+1}(y-\mu_{n-1})^2
\end{aligned}
$$

Despite having two unknown parameters, we are only interesting in reaching a conclusion about the unknown mean parameter μ of the population distribution. In order to do this we need a marginal posterior distribution of the parameters that are of interest, in this case the mean μ.

We are then able to sample from the joint posterior distribution by first sampling σ^2 from the marginal posterior distribution [23]:

$$
p(\sigma^2|y) = Inv - \chi^2(\sigma^2|\nu_n, \sigma_n^2) \tag{5.26}
$$

and then μ from the conditional posterior distribution using σ^2 [23]:

$$
p(\mu|\sigma^2, y) = N(\mu|\mu_n, \frac{\sigma^2}{\kappa_n}) \tag{5.27}
$$

5.5.2 Algorithm description

The algorithm of BLA Normal with unknown μ and unknown σ^2 is shown in Algorithm 16. For each action $j \in \{1, \ldots, K\}$ the automaton needs to keep track of the following hyperparameters: The number of selections of an action denoted as κ_j, sample mean denoted as μ_j, number of degrees of freedom denoted as ν_j, and $\nu_j \sigma_j^2$ which contains the sample variance σ_j^2 and is used during variance sampling [23].

Algorithm 16 BLA Normal unknown σ^2

```
1: Initialization: Set hyperparameters $\mu_j, \nu_j, \nu_j \sigma_j^2$ and $\kappa_j$ for each action $j$ in accordance to
   the prior belief about the true mean $\mu_j$
2: loop
3:     Draw $x_j$ from $N(\mu_j, S^2/\kappa_j)$, where $S^2$ is drawn from scaled-inverse$-\chi^2(\nu_j, \nu_j \sigma_j^2/\nu_j)$
4:     $i = \text{argmax } x_j$, where $j \in \{1, \ldots, K\}$
5:     Select action $i$ and receive feedback $r$
6:     $\nu_i \sigma_i^2 \leftarrow \nu_i \sigma_i^2 + (\kappa_i/(\kappa_i + 1))(r - \mu_i)^2$
7:     $\mu_i \leftarrow (\kappa_i/(\kappa_i + 1))\mu_i + r/(\kappa_i + 1)$
8:     $\nu_i \leftarrow \nu_i + 1$
9:     $\kappa_i \leftarrow \kappa_i + 1$
10: end loop
```

The initialization of the hyperparameters, omitted in the algorithm, is dependent on potential prior belief about the parameter μ.

The sampling procedure is performed by first sample the variance S_j^2 from the scaled-inverse-χ^2 distribution, and then use the sample variance in order to sample the mean x_j from $N\left(\mu_j, \frac{S_j^2}{\kappa_j}\right)$.

The action with the highest mean x_j is selected, and a feedback r is received from the environment. The feedback r is then used to update the hyperparameters associated with the action at lines 6-9, in accordance with the updating rules presented above.

Chapter 6

Experiments and results

In this chapter we compare the performance of the Bayesian Learning Automaton family against other well-known bandit players in the general k-armed bandit problem, and in games that from the players' point of view may be treated like a k-armed bandit problem.

As noted in Chapters 4 and 5 the bandit players are designed for various feedback distributions, which determine applicable domains for the players. In the experiments encountered in this thesis we apply bandit players in applicable environments, and refer the reader to Chapters 4 and 5 for details about the players and their applicable feedback.

In the following we present the most interesting results of the experiments, with additional results available in the appendix.

6.1 Initial values of hyperparameters

Some of the members of the BLA family are introduced in several of the experiments described in this chapter, especially BLA Bernoulli and BLA Normal unknown σ^2. We therefore present the initial values of the hyperparameters of the conjugate prior distributions used in the experiments in this thesis.

6.1.1 Initial values of hyperparameters in BLA Bernoulli

The beta distribution is easy with regards to initial values of the hyperparameters, as the initial values

$$\alpha = 1$$
$$\beta = 1$$

yield a uniform non-informative proper prior distribution over the unknown Bernoulli parameter p. We use these initialization values in all experiments involving BLA Bernoulli.

6.1.2 Initial values of hyperparameters in BLA Poisson

With BLA Poisson the choice of initial hyperparameters of the conjugate prior distribution may be difficult as Poisson distributed feedback is defined in the interval $[0, \infty)$. It is therefore important that the true values of the λ parameters are not excluded when choosing the initial values of the hyperparameters of the gamma distribution. Hence, initial values of the hyperparameters are highly dependent on the range of the λ parameter.

In the experiments involving BLA Poisson the parameter λ is set to be a real number in the interval [0,10], and with this in mind, we use the following initial values of the hyperparameters of the gamma distribution in the k-armed bandit problem with Poisson distributed feedback:

$$\begin{aligned}\alpha &= 10\\ \beta &= 1\end{aligned}$$

6.1.3 Initial values of hyperparameters in BLA Normal known σ^2

In the experiments involving with BLA Normal known σ^2 we introduced an initialization phase to the algorithm to find initial values of the hyperparameters m_j and s_j. This initialization phase consists of selecting each action once, and then use the received feedback as the initial value for the mean hyperparameter m_j.

For convenience we choose to use the variance of the action with the highest variance as the known variance for each action $j \in \{1, \ldots, K\}$ in the automaton. This means that the variance of the action with the highest variance is both used as the initial value of the variance hyperparameter s_j and during the updating procedure in the automaton.

The initialization phase in combination with the use of the highest true variance should ensure that the initial values of the hyperparameters m_j and s_j do not exclude any true value of parameter μ_j.

This results in the following hyperparameters after one trial on each action:

$$\begin{aligned}s_j^2 &= \text{variance of the action with the highest variance}\\ m_j &= \text{value of feedback from first attempt on action } j\end{aligned}$$

6.1.4 Initial values of hyperparameters in BLA Normal unknown σ^2

With BLA Normal unknown σ^2 the choice of initial values of the hyperparameters may be problematic. The reason for this is that normally distributed feedback is defined in the interval $(-\infty, \infty)$.

In most practical cases there exists some information about the environment that we may use to construct a reasonable prior distribution on the unknown mean μ. To demonstrate the flexibility of this automaton we have chosen an extremely wide and vague prior distribution with the following initial hyperparameters:

$$\begin{aligned} \mu &= 0 \\ \kappa &= 0.01 \\ \nu &= 0.1 \\ \nu\sigma^2 &= 0.1 \end{aligned}$$

With these hyperparameters the initial mean samples from the automaton are typically in the range -1×10^{21} to 1×10^{21}, which is very wide considering the true value of the mean μ in the experiments is always within the range $[0, 1]$.

6.2 K-armed bandit experiment configurations

In the experiments conducted in the general k-armed bandit problem, all configurations have been replicated 1000 times, where each replication consists of 100 000 iterations.

To get a picture of how the optimal arm selection probability develops as the number of iterations increases, we present the probability of selecting the optimal arm both as tables and figures. The tables present the overall probability of selecting the optimal arm in the entire interval 0 to 10, 100, 1000, 10 000 and 100 000, while in the figures we show the probability of selecting the optimal arm at each instant.

We also consider the *regret* of the bandit players, which expresses the expected loss a player experience since it does not always select the optimal arm [16]. As the number of iterations increases long term performance is emphasized and low learning pace is penalized. By considering the regret of the players we avoid too much emphasize on selecting the optimal arm, when in fact selecting a non-optimal arm with a distribution close to the optimal arm may not significantly affect the overall cumulative feedback.

6.3 K-armed bandit problem with Bernoulli distributed feedback

The experiments performed in the k-armed Bernoulli bandit problem are based on the experiments performed by Auer et al. in [16]. The experiment configurations for the k-armed bandit problem with Bernoulli distributed feedback are shown in Table 6.1, where we define three configurations for the 2-armed bandit problem and three configurations for the 10-armed bandit problem. Each entry in the table represents the p parameter of the Bernoulli distribution associated with each arm, and we have included configurations with different degree of difficulty.

Table 6.1: Experiment configurations for the k-armed bandit problem with Bernoulli distributed feedback

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

The overall results of the configurations presented in Table 6.1 are shown in Table 6.2, where the probability of selecting the optimal action in the entire interval 0 to 100 000 is presented. Each row represents a player and each column represents a configuration as given in Table 6.1, and each entry in the table represents the overall probability for selecting the optimal action. Each configuration is replicated 1000 times, and in each replication the optimal arm is selected at random to avoid bias caused by possible patterns during arm selection in the algorithms.

Table 6.2: Results of the 2-armed and 10-armed bandit problem with Bernoulli distributed feedback

Player / Configuration	**1**	**2**	**3**	**4**	**5**	**6**
BLA Bernoulli	1.000	0.999	0.997	0.998	0.988	0.975
BLA Normal unknown σ^2	1.000	0.998	0.992	0.996	0.982	0.968
ε_n-GREEDY c =0.05 [1]	0.981	0.992	0.965	0.996	0.961	0.893
ε_n-GREEDY c =0.15 [1]	1.000	0.999	0.991	0.990	0.988	0.957
ε_n-GREEDY c =0.30 [1]	1.000	0.997	0.997	0.982	0.981	0.977
L_{R-I} 0.05	0.999	0.918	0.985	0.832	0.378	0.526
L_{R-I} 0.01	0.998	0.993	0.993	0.992	0.885	0.958
L_{R-I} 0.005	0.995	0.986	0.986	0.984	0.940	0.951
Pursuit 0.05	1.000	0.970	0.932	0.912	0.699	0.608
Pursuit 0.01	0.999	0.998	0.998	0.998	0.875	0.848
Pursuit 0.005	0.999	0.999	0.998	0.997	0.960	0.924
UCB1	0.999	0.983	0.982	0.979	0.848	0.848
UCB1-TUNED	1.000	0.997	0.997	0.997	0.977	0.978
UCB1-NORMAL	0.995	0.977	0.969	0.959	0.797	0.737
Exp3 0.01	0.990	0.978	0.980	0.913	0.736	0.749
POKER	0.995	0.991	0.876	0.982	0.916	0.812
INTESTIM0.01	0.961	0.949	0.796	0.920	0.905	0.577

[1] Parameter d is set to be the difference between the best arm and the next best arm

We observe from Table 6.2 that the difference in performance between the bandit players becomes more evident as the degree of difficulty increases. The BLA players seem to offer very impressive and stable results across the board, and seem to handle these configurations very well. Note that even BLA Normal unknown σ^2 is able to offer impressive performance in this problem, even though it is not especially designed for this problem and may take feedback in any range. Also note that UCB1-TUNED is the only player that seems to offer similar performance across all the configurations, and seems to be slightly ahead in configuration 3 and 6, which is more evident in the detailed overview of these experiment configurations.

As seen in Table 6.2 we have included several bandit players of the same type in order to investigate the impact of the learning parameters on the performance. We observe that the impact increases as the number of arms increases along with the difficulty of the arm distributions. Note that the ε_n-GREEDY players provide a relative stable performance throughout the configurations, despite different values of the c parameter. However, as noted in the table, these players are given the difference of the best arm and second best arm, which may be regarded as an unfair advantage.

It is worth noting that some of the players presented in Table 6.2 are not tailored to Bernoulli distributed feedback, but as these bandit players are capable of receiving feedback in a wider range than defined by the Bernoulli distribution, these are also applicable in environments with Bernoulli distributed feedback.

Configuration 5 and 6 in Table 6.1 are considered to be the most difficult configurations for the 10-armed bandit problem. We therefore present a more detailed overview of the development of these configurations below. The results from all of the configurations for the k-armed bandit problem with Bernoulli distributed feedback are available in Appendix A.1.

6.3.1 Results of experiment configuration 5

In this section we take a closer look at the results from experiment configuration 5 as defined in Table 6.1. Table 6.3 reports the average probability of selecting the optimal arm in the entire interval from 0 to 10, 100, 1000, 10 000 and 100 000 iterations.

Table 6.3: Detailed overview of the 10-armed bandit problem with Bernoulli distributed feedback with optimal arm $p = 0.9$ and 0.8 on the inferior arms

Player / Iteration	**10**	**100**	**1000**	**10 000**	**100 000**
BLA Bernoulli	0.112	0.197	0.549	0.916	0.988
BLA Normal unknown σ^2	0.107	0.131	0.433	0.881	0.982
ε_n-GREEDY c =0.05 d =0.1	0.101	0.124	0.630	0.898	0.961
ε_n-GREEDY c =0.15 d =0.1	0.105	0.100	0.511	0.911	0.988
ε_n-GREEDY c =0.30 d =0.1	0.099	0.099	0.359	0.872	0.981
L_{R-I} 0.05	0.103	0.119	0.273	0.368	0.378
L_{R-I} 0.01	0.104	0.105	0.156	0.672	0.885
L_{R-I} 0.005	0.102	0.102	0.126	0.518	0.940
Pursuit 0.05	0.100	0.157	0.567	0.682	0.699
Pursuit 0.01	0.098	0.116	0.550	0.840	0.875
Pursuit 0.005	0.101	0.108	0.488	0.910	0.960
UCB1	0.100	0.119	0.166	0.406	0.848
UCB1-TUNED	0.100	0.164	0.425	0.841	0.977
UCB1-NORMAL	0.088	0.099	0.089	0.267	0.797
Exp3 0.01	0.097	0.099	0.104	0.156	0.736
POKER	0.105	0.180	0.444	0.751	0.916
INTESTIM 0.01	0.126	0.194	0.519	0.857	0.905

From the results reported in Table 6.3 we observe that BLA Bernoulli is among the top performers in every interval. Furthermore, it is the top performer when compared to other learning schemes which do not rely on external tuning parameters. We also see that BLA Normal unknown σ^2 offers very good performance, but is initially slow in the lower intervals compared to BLA Bernoulli. However, it seems to offer very good performance when the number of iterations is sufficiently large, such that some confidence about the variance σ^2 has been achieved.

We see that in contrast to L_{R-I} and Pursuit schemes the ε_n-GREEDY schemes seem to be less affected by the tuning parameters in the long run, and ε_n-GREEDY with $c = 15$ achieves similar performance as BLA Bernoulli after 100 000 iterations.

As shown in Figure 6.1 the Bayesian players achieve among the lowest regret when the number of iterations is sufficiently large. Once again only a highly tuned ε_n-GREEDY is able to offer similar performance to BLA Bernoulli.

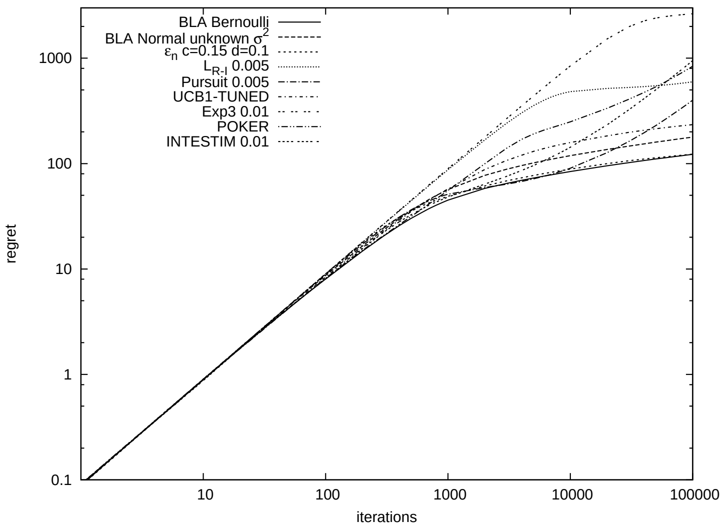


Figure 6.1: Regret in the 10-armed bandit problem with Bernoulli distributed feedback with optimal arm $p = 0.9$ and 0.8 on the inferior arms

6.3.2 Results of experiment configuration 6

The detailed overview of the results from experiment configuration 6 is shown in Table 6.4, which shows the average probability of selecting the optimal in the entire interval from 0 to 10, 100, 1000, 10 000 and 100 000 iterations.

Table 6.4: Detailed overview of the 10-armed bandit problem with Bernoulli distributed feedback with optimal arm $p = 0.55$ and 0.45 on the inferior arms

Player / Iteration	**10**	**100**	**1000**	**10 000**	**100 000**
BLA Bernoulli	0.109	0.150	0.340	0.834	0.975
BLA Normal unknown σ^2	0.101	0.144	0.353	0.828	0.968
ε_n-GREEDY c =0.05 d =0.1	0.102	0.120	0.480	0.787	0.893
ε_n-GREEDY c =0.15 d =0.1	0.103	0.101	0.401	0.835	0.957
ε_n-GREEDY c =0.30 d =0.1	0.097	0.098	0.289	0.847	0.977
L_{R-I} 0.05	0.105	0.123	0.329	0.507	0.526
L_{R-I} 0.01	0.102	0.103	0.158	0.728	0.958
L_{R-I} 0.005	0.101	0.102	0.123	0.526	0.951
Pursuit 0.05	0.102	0.139	0.449	0.589	0.608
Pursuit 0.01	0.098	0.112	0.396	0.793	0.848
Pursuit 0.005	0.104	0.105	0.309	0.838	0.924
UCB1	0.100	0.120	0.165	0.404	0.848
UCB1-TUNED	0.100	0.166	0.401	0.855	0.978
UCB1-NORMAL	0.100	0.099	0.146	0.283	0.737
Exp3 0.01	0.102	0.102	0.105	0.158	0.749
POKER	0.101	0.184	0.378	0.607	0.812
INTESTIM 0.01	0.100	0.174	0.379	0.549	0.577

In this configuration UCB1-TUNED offers slightly better performance than both of the Bayesian players, but when the number of iterations is sufficiently large their performance starts to even out. The good performance by UCB1-TUNED in this configuration seems to be in accordance with the results as reported by Auer et al. in [16]. We also observe that ε_n-GREEDY with $c = 0.30$ offers very good performance.

The regret of the players in experiment configuration 6 is shown in Figure 6.2. We observe that UCB1-TUNED achieves lower regret than both Bayesian players, especially when the number of iterations is quite low. ε_n-GREEDY with $c = 0.30$ is among the best players and offers slightly lower regret than both Bayesian players after 100 000 iterations.

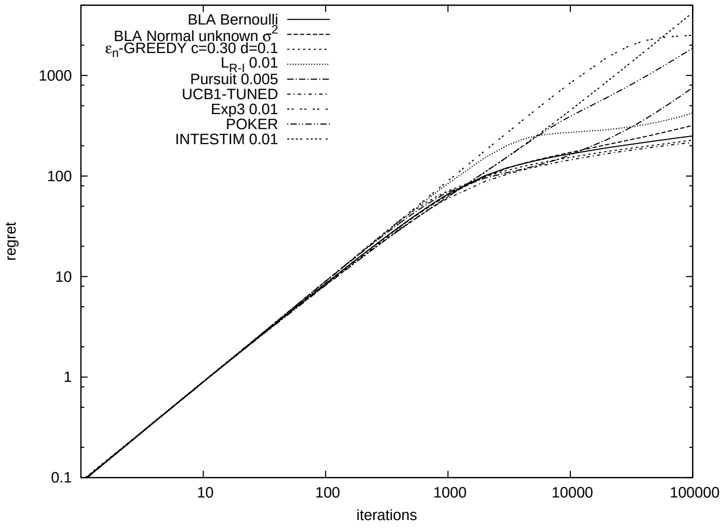


Figure 6.2: Regret in the 10-armed bandit problem with Bernoulli distributed feedback with optimal arm $p = 0.55$ and 0.45 on the inferior arms

6.4 K-armed bandit problem with Poisson distributed feedback

The experiments performed in the k-armed bandit problem with Poisson distributed feedback consist two experiment configurations, the 2-armed bandit problem as well as the 10-armed bandit problem, where the parameter λ is a real number uniformly drawn from the interval [0,10]. The results from both the configurations for the k-armed bandit problem with Poisson distributed feedback are available in Appendix A.2.

Table 6.5: Results of the 10-armed bandit problem with Poisson distributed feedback

Player / Iteration	10	100	1000	10 000	100 000
BLA Poisson	0.113	0.466	0.781	0.927	0.980
BLA Normal unknown σ^2	0.122	0.473	0.769	0.919	0.972
ε_n-GREEDY $c = 0.05$ $d = 0.1$	0.102	0.182	0.656	0.821	0.865
ε_n-GREEDY $c = 0.15$ $d = 0.1$	0.102	0.101	0.517	0.829	0.909
ε_n-GREEDY $c = 0.30$ $d = 0.1$	0.100	0.101	0.355	0.810	0.928
Pursuit 0.050	0.123	0.423	0.633	0.662	0.666
Pursuit 0.010	0.103	0.242	0.716	0.835	0.849
Pursuit 0.005	0.102	0.181	0.656	0.869	0.893
UCB1-NORMAL	0.108	0.099	0.503	0.772	0.910
POKER	0.109	0.518	0.766	0.859	0.874

The results of the 10-armed bandit problem are shown in Table 6.5. The table reports the average probability of selecting the optimal arm over 10, 100, 1000, 10 000 and 100 000 iterations. Note that the d parameter of ε_n-GREEDY is *not* set to be the difference between the best arm and second best arm, as this difference may be out of the range of the excepted value of d, and we therefore set $d = 0.1$.

Overall the BLA players seem to be among the top performers in every interval, and are the top performers when the number of iterations is sufficiently large. POKER is initially ahead, but offers limited improvement in performance as the number of iterations increases and starts to fall behind.

Table 6.5 also shows the impact the tuning parameters have on the performance of the ε_n-GREEDY and Pursuit players. We see that the ε_n-GREEDY with parameters $c = 0.30$ and $d = 0.1$, offers best performance of the ε_n-GREEDY schemes over 100 000 iterations, but we notice that both the other ε_n-GREEDY variants offer better or similar performance over 10, 100, 1000 and 10 000 iterations.

The regret of the bandit players is shown in Figure 6.3. We observe that POKER initially performs very well and achieves the lowest regret, but falls behind in terms of regret towards the end. We observe the same tendencies in Figure 6.4, which shows the regret in the 2-armed bandit problem. Furthermore, we observe from Figures 6.3 and 6.4 that both the BLA players achieve by far the lowest regret in both the 2-armed and 10-armed bandit problem.

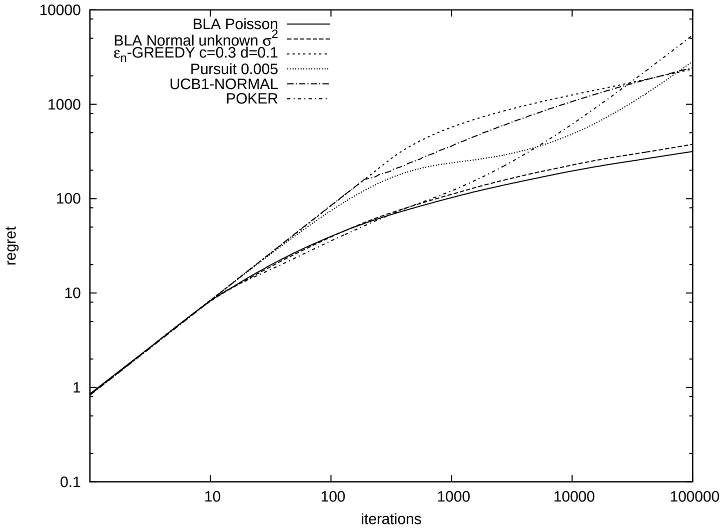


Figure 6.3: Regret in the 10-armed bandit problem with Poisson distributed feedback

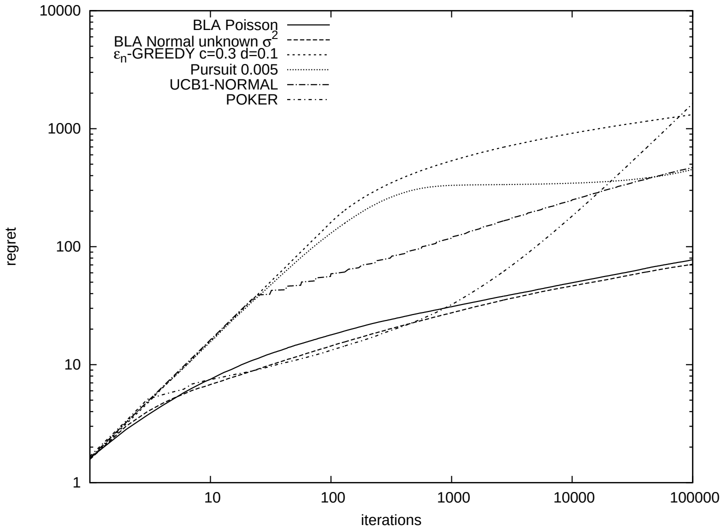


Figure 6.4: Regret in the 2-armed bandit problem with Poisson distributed feedback

These results clearly show that BLA Poisson and BLA Normal unknown σ^2 are superior to the other bandit players in the k-armed bandit problem with Poisson distributed feedback when the number of iterations is sufficiently large.

6.5 K-armed bandit problem with normally distributed feedback

In the experiments performed in the k-armed bandit problem with normally distributed feedback we define two experiment configurations, the 2-armed bandit problem and the 10-armed bandit problem, where both the mean μ and variance σ^2 are drawn uniformly from the interval [0,1]. The results from both the configurations for the k-armed bandit problem with normally distributed feedback are available in Appendix A.3.

Table 6.6: Results of the 10-armed bandit problem with normally distributed feedback

Player / Iteration	**10**	**100**	**1000**	**10 000**	**100 000**
BLA Normal known σ^2	0.100	0.235	0.521	0.804	0.928
BLA Normal known $\sigma^2 = 1$	0.100	0.214	0.490	0.788	0.925
BLA Normal unknown σ^2	0.105	0.300	0.633	0.858	0.955
ε_n-GREEDY c =0.05 [1]	0.103	0.194	0.465	0.692	0.820
ε_n-GREEDY c =0.15 [1]	0.100	0.142	0.378	0.671	0.858
ε_n-GREEDY c =0.30 [1]	0.097	0.118	0.309	0.620	0.833
Pursuit 0.05	0.116	0.347	0.566	0.600	0.605
Pursuit 0.01	0.097	0.191	0.618	0.758	0.775
Pursuit 0.005	0.097	0.152	0.576	0.813	0.842
UCB1-NORMAL	0.113	0.102	0.253	0.553	0.810
POKER	0.106	0.338	0.603	0.800	0.849

[1] Parameter d is set to be the difference between the best arm and the next best arm

The results of the 10-armed bandit problem are shown in Table 6.6. As seen in the table we apply three Bayesian players in this experiment. As mentioned previously, BLA Normal known σ^2 is given the variance of the action with the highest variance. In this configuration we also apply BLA Normal known σ^2 and use the upper limit of the interval as the variance. Knowing the variance might be regarded as an unfair advantage over the other players, however as reported in Table 6.6, the BLA players which are given the variance do not seem to be able to gain any advantage over BLA Normal unknown σ^2, and in fact seem to perform worse than BLA Normal unknown σ^2.

The BLA Normal unknown σ^2 clearly offers best performance over 100 000 iterations, closely followed by the other two BLA players. We observe that POKER achieves very good performance over the first 10 000 iterations, but starts to fall behind when the number of iterations is sufficiently large.

The regret in the 2-armed bandit problem and the 10-armed bandit problem is shown in Figures 6.5 and 6.6. In terms if regret we observe the same performance pattern of POKER as described above. POKER is initially very strong, but falls behind when the number of iterations is sufficiently large. In both configurations the BLA players are superior with regards to regret with a sufficient number of iterations, and especially BLA normal unknown σ^2 offers impressive performance.

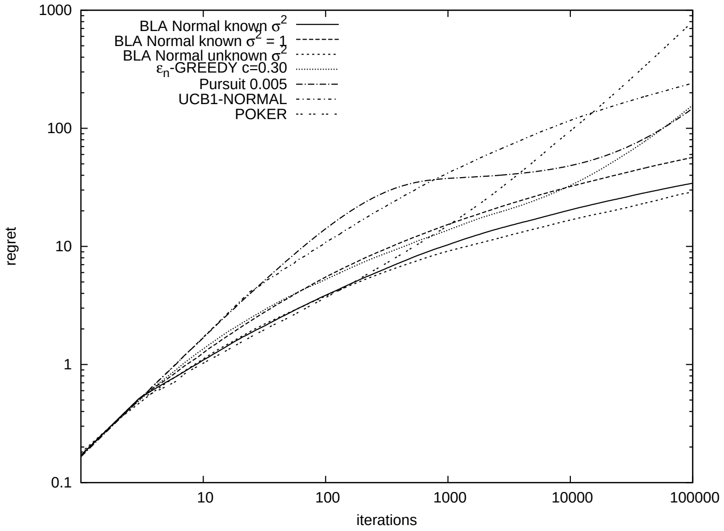


Figure 6.5: Regret in the 2-armed bandit problem with normally distributed feedback

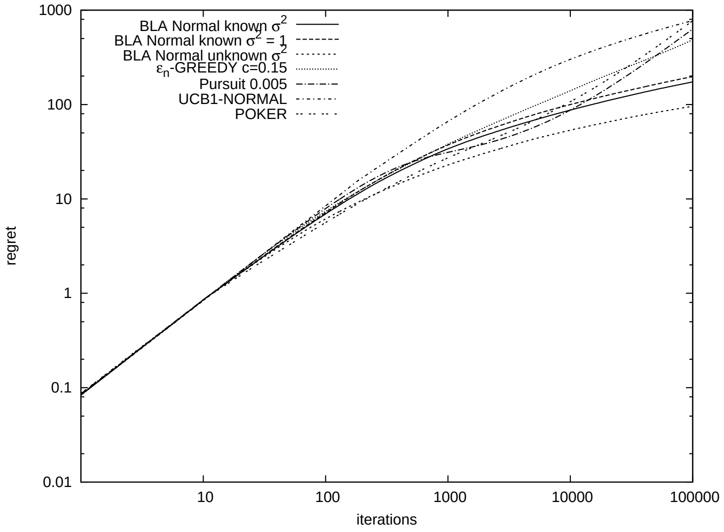


Figure 6.6: Regret in the 10-armed bandit problem with normally distributed feedback

Overall, the experiments show that the BLA players seem to offer excellent performance, and are the top performers when the number of iterations is sufficiently large.

6.6 Goore game

We define three experiment configurations for the Goore game, where the number of players and the optimal number of yes votes vary. The configurations are shown in Table 6.7.

Table 6.7: Experiment configurations for the Goore game

Configuration	Number of players	Optimal number of yes votes
1	10	3
2	50	20
3	100	35

All configurations have been replicated 100 times, where each configuration consists of 100 000 iterations, i.e 100 000 voting iterations.

As the reward function in the Goore game we consider the unimodal function from Section 2.6.2 and generalize the function such that we can adjust the optimal number of yes votes to fit with the experiment configurations in Table 6.7. This yields the following function:

$$f(\theta) = 0.2 + 0.8 * e^{(-0.002(\theta - n)^2)} \tag{6.1}$$

where n denotes the optimal number of yes votes and θ denotes the number of players in the game. In the conducted experiments we define teams of bandit players, where every team only consists of players of the same type.

In the following we present the results of configuration 1 and configuration 3. The results of all three configurations are available in Appendix A.4.

6.6.1 Results of experiment configuration 1

The results of experiment configuration 1 with 10 players, where the optimal number of yes votes is 3, are shown in Table 6.8. The first column denotes the type of bandit players, and the remaining columns denote the average distance from the optimal number of yes votes after 10, 100, 1000, 10 000 and 100 000 iterations.

Table 6.8 reports that only the team of Tsetlin automata and the team of BLA Bernoulli players are able to solve the Goore game in this configuration, since exactly 3 of 10 players in each team voted yes. However, we observe that the team of Tsetlin automata reaches a consensus a lot faster than the team of BLA Bernoulli players. The performance of the Tsetlin automata seems to be in accordance with the results from [11], where a team of Tsetlin automata is proved to solve the game.

In Figure 6.7 we observe how the number of yes votes in each team of bandit players develops as the number of iterations increases, where the straight line denotes the optimal number of yes votes. We again observe how the fast the team of Tsetlin automata is. Furthermore, we see that

Table 6.8: Distance from the optimal number of yes votes, with 10 players and 3 desired yes votes

Player / Iteration	**10**	**100**	**1000**	**10 000**	**100 000**
Tsetlin $L_{2N,2}$ $N = 1$	2.01	0.95	0.00	0.00	0.00
BLA Bernoulli	2.03	1.58	0.82	0.36	0.00
BLA Normal unknown σ^2	2.37	1.67	1.18	0.75	0.04
ε_n-GREEDY c =0.05 d =0.10	2.06	0.80	0.50	0.28	0.24
ε_n-GREEDY c =0.15 d =0.10	2.09	1.09	0.62	0.24	0.21
ε_n-GREEDY c =0.30 d =0.10	2.16	1.24	0.62	0.38	0.35
L_{R-I} 0.01	1.74	2.26	2.09	1.57	1.22
L_{R-I} 0.005	2.13	2.06	2.25	1.77	0.67
Pursuit 0.01	2.05	1.08	0.51	0.24	0.24
Pursuit 0.005	2.09	1.28	0.82	0.24	0.24
UCB1-TUNED	5.03	1.93	1.61	1.00	0.21
UCB1-NORMAL	3.00	2.01	1.91	1.64	0.85
POKER	2.23	2.79	1.09	0.92	0.18

despite having a slow start the team of UCB1-TUNED players gets close to the optimal number of yes votes after 100 000 iterations.

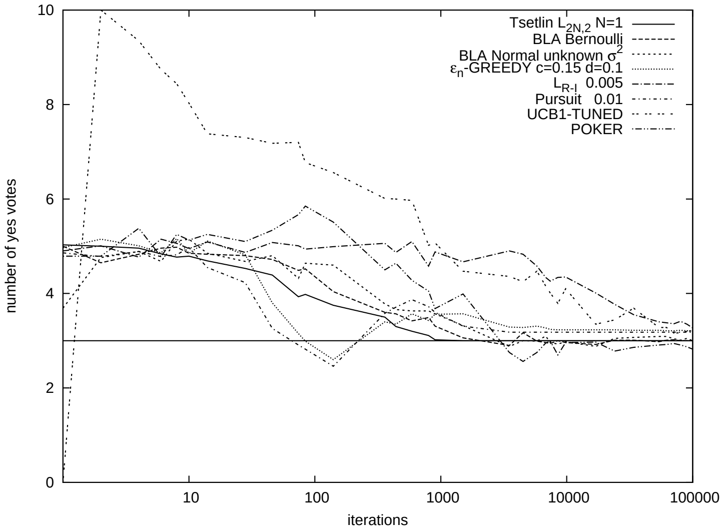


Figure 6.7: Development of the number of yes votes with 10 players and 3 as the optimal number of yes votes

6.6.2 Results of experiment configuration 3

The results of experiment configuration 3 with 100 players, where the optimal number of yes votes is 35, are shown in Table 6.9. From the table we observe that only the Tsetlin $L_{2N,2}$ team is is able to identify the optimal number of yes votes and has identified the optimal number of yes votes already after 1000 iterations.

We observe from Table 6.9 that the team of BLA Normal unknown σ^2 players performs well, but are never able to truly identify the optimal number of yes votes with the number of iterations available.

Table 6.9 reports that both the team of UCB1-NORMAL players and the team of POKER players perform well in this experiment configuration. Furthermore, we observe that despite a slow start the team of UCB1-NORMAL is able to get close to the optimal number of yes votes.

Figure 6.8 shows how the number of yes votes develops through 100 000 iterations, where the straight line denotes the optimal number of yes votes. Again we observe how fast the team of Tsetlin automata is to identify the optimal number of yes votes. Furthermore, we see that the team of Pursuit players with learning rate 0.01 follow the Tsetlin team closely at 1000 iterations, but then falls behind.

Table 6.9: Distance from the optimal number of yes votes in the Goore game with 100 players, with 35 desired yes votes

Player / Iteration	**10**	**100**	**1000**	**10 000**	**100 000**
Tsetlin $L_{2N,2}$ $N=1$	10.43	6.08	0.00	0.00	0.00
BLA Bernoulli	12.88	6.04	1.81	1.20	0.90
BLA Normal unknown σ^2	12.31	6.40	1.83	0.90	0.25
ε_n-GREEDY c =0.05 d =0.10	15.20	6.93	5.22	5.08	4.94
ε_n-GREEDY c =0.15 d =0.10	14.29	7.71	3.95	3.58	3.51
ε_n-GREEDY c =0.30 d =0.10	14.58	10.03	3.57	3.27	3.10
L_{R-I} 0.01	14.66	14.22	8.33	2.56	0.78
L_{R-I} 0.005	15.08	14.52	12.12	3.92	1.39
Pursuit 0.01	14.55	8.91	2.75	2.74	2.74
Pursuit 0.005	14.60	11.75	3.42	3.25	3.25
UCB1-TUNED	4.42	8.57	5.07	1.84	1.51
UCB1-NORMAL	35.00	34.33	15.34	1.35	0.64
POKER	12.74	9.25	2.87	1.97	0.65

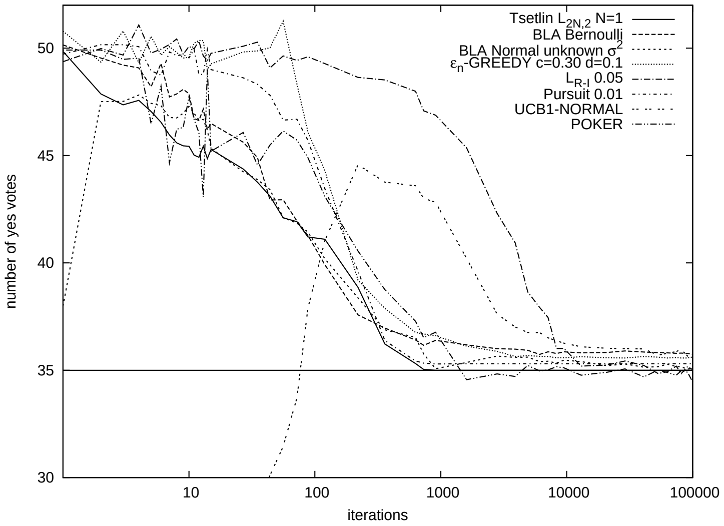


Figure 6.8: Development of the number of yes votes with 100 players and 35 as the optimal number of yes votes

Overall, the teams of BLA players perform well in the Goore game and are among the top performers in the experiment configurations, but are not able to beat the Tsetlin $L_{2N,2}$ team.

6.7 Iterative prisoners' dilemma

The iterative prisoners' dilemma experiment consists of a tournament style game, where each player plays against all other players in addition to a copy of itself. Since many of the learning schemes used in this thesis expect feedback in the interval [0,1], we construct a normalized inverted version of the matrix listed in the initial description of the game in Section 2.6.3. This ensures that the feedback is within the range of the supported feedback of the algorithms, and that a higher feedback value represents a lower sentence. The game matrix is shown in Figure 6.9.

		Player B	
		Not confess	Confess
Player A	Not confess	0.8,0.8	0.0,1.0
	Confess	1.0,0.0	0.4,0.4

Figure 6.9: Inverted normalized game matrix in prisoners' dilemma

The results from this tournament are shown in Table 6.10 as the average reward achieved over 10, 100, 1000, 10 000 and 100 000 iterations. Note that most of the players included in this tournament are not really designed for this game, but are still included as points of reference to the Bayesian players. Tit-For-Tat is also included as a reference in this tournament since it is

Table 6.10: Tournament scores of the iterative prisoners' dilemma

Player / Iteration	**10**	**100**	**1000**	**10 000**	**100 000**
Tit-For-Tat	0.635	0.548	0.491	0.469	0.466
BLA Normal known $\sigma^2 = 1$	0.663	0.542	0.459	0.424	0.420
BLA Normal unknown σ^2	0.585	0.606	0.507	0.463	0.458
ε_n-GREEDY c =0.05 d =0.1	0.624	0.549	0.463	0.421	0.416
ε_n-GREEDY c =0.15 d =0.1	0.626	0.502	0.451	0.418	0.414
ε_n-GREEDY c =0.30 d =0.1	0.624	0.468	0.440	0.417	0.415
L_{R-I} 0.05	0.630	0.505	0.459	0.420	0.414
L_{R-I} 0.01	0.627	0.466	0.419	0.418	0.416
L_{R-I} 0.005	0.629	0.460	0.382	0.411	0.414
Pursuit 0.01	0.630	0.502	0.458	0.421	0.416
Pursuit 0.005	0.629	0.483	0.445	0.420	0.416
UCB1-NORMAL	0.564	0.515	0.459	0.434	0.431
POKER	0.639	0.499	0.376	0.320	0.311
EXP3	0.628	0.459	0.381	0.409	0.413

regarded as a top performer in iterative prisoners' dilemma. However, Tit-for-tat is not really comparable to the bandit players since it is not able to learn by experience, but just replicates the last move of the opponent.

The results from Table 6.10 seem to indicate that most games in the tournament end in the confessing state for both players, as the scores close to 0.4 might indicate. This is not very surprising, as players that are regarded as rational often end in the confessing state in prisoners' dilemma [7].

Note that BLA Normal unknown σ^2 beats Tit-For-Tat in the intervals 0 to 100 and 0 to 1000, but is beaten by Tit-For-Tat when the interval is extended further. All non-Bayesian players seem to be beaten by Tit-For-Tat in every interval listed in the table.

6.8 Two-player zero-sum game

6.8.1 Experiment description

In the two-player zero-sum game we have defined three experiment configurations, where the first two configurations are pure strategy games and the third is a mixed strategy game.

The pure strategy games use the matrices $D1$ and $D2$, shown in Figure 6.10 and 6.11, where $D2$ is considered more difficult than $D1$. In each of these configurations the objective is to observe how fast the players are able to identify the optimal pure strategy.

$$D1 = \begin{bmatrix} 0.6 & 0.8 \\ 0.35 & 0.65 \end{bmatrix}$$

Figure 6.10: Game matrix D1

$$D2 = \begin{bmatrix} 0.55 & 0.525 \\ 0.45 & 0.475 \end{bmatrix}$$

Figure 6.11: Game matrix D2

The matrix $D3$, shown in Figure 6.12, is used in the mixed strategy game, where the optimal solution for player A is to play the arms with probability (0.25, 0.75) and for player B to play the arms with probability (0.5, 0.5). In this experiment configuration we want to observe how fast the players are able to identify and exploit the mixed strategy, if at all.

$$D3 = \begin{bmatrix} 0.8 & 0.2 \\ 0.4 & 0.6 \end{bmatrix}$$

Figure 6.12: Game matrix D3

The mixed strategy configuration is further divided into two experiments. We play each player against itself, i.e same type of player with the same tuning parameters, and look at action selection probability and distance from the optimal reward. In addition we create a tournament style game, where each player plays against all other players.

This time we also have included the player $L_{R-\epsilon P}$ which is known to be able to find the optimal solution in mixed strategy games [11]. The tuning parameters $\alpha = 0.002$ and $\beta = 0.00001$ of $L_{R-\epsilon P}$ and the game matrix $D3$ are the same as used in [11].

Below we present the results of the pure strategy game with the $D2$ matrix together with the results of the mixed strategy game. The results of all the experiment configurations are listed in Appendix A.6.

6.8.2 Pure strategy game results

In Table 6.11 we observe how the probability of selecting the optimal action for both players develops as the number of iterations increases when playing game matrix $D2$. As with the previous experiments, the experiment is replicated 1000 times, where each replication consists of 100 000 iterations.

Table 6.11: Optimal actions selection probability for player A and B in D2 (A,B)

Player / Iteration	**10**	**100**	**1000**	**10 000**	**100 000**
BLA Bernoulli	0.56, 0.49	0.70, 0.55	0.91, 0.63	0.99, 0.92	1.00, 0.99
BLA Normal unknown σ^2	0.61, 0.55	0.81, 0.66	0.96, 0.79	0.99, 0.93	1.00, 0.99
ε_n-GREEDYc =0.05 d =0.10	0.51, 0.54	0.72, 0.54	0.85, 0.60	0.89, 0.63	0.92, 0.67
ε_n-GREEDY c =0.15 d =0.10	0.50, 0.52	0.70, 0.50	0.91, 0.57	0.97, 0.69	0.98, 0.76
ε_n-GREEDY c =0.30 d =0.10	0.50, 0.49	0.61, 0.46	0.93, 0.60	0.99, 0.77	1.00, 0.85
$L_{R-\epsilon P}$ $\alpha = 0.002, \beta = 0.00001$	0.51, 0.50	0.51, 0.53	0.53, 0.50	0.83, 0.55	0.96, 0.90
L_{R-I} 0.05	0.50, 0.50	0.59, 0.53	0.89, 0.58	0.94, 0.66	0.94, 0.66
L_{R-I} 0.01	0.54, 0.52	0.52, 0.49	0.67, 0.49	0.99, 0.83	1.00, 0.98
L_{R-I} 0.005	0.50, 0.49	0.51, 0.50	0.57, 0.51	0.97, 0.68	1.00, 1.00
Pursuit 0.01	0.52, 0.51	0.59, 0.53	0.95, 0.64	0.99, 0.81	0.99, 0.83
Pursuit 0.005	0.50, 0.51	0.54, 0.50	0.91, 0.58	1.00, 0.88	1.00, 0.92
UCB1-TUNED	0.57, 0.48	0.72, 0.54	0.93, 0.65	0.98, 0.92	1.00, 0.99
POKER	0.60, 0.47	0.65, 0.53	0.79, 0.55	0.85, 0.66	0.85, 0.69

The first column denotes the bandit players, while the remaining columns denote the average probability of selecting the optimal action for player A and player B after 10, 100, 1000, 10 000 and 100 000 iterations. Each entry consists of two probabilities, where the leftmost value is the probability of selecting the optimal action for player A, and the rightmost value the probability of selecting the optimal action for player B.

As we observe from Table 6.11 BLA Bernoulli, BLA Normal unknown σ^2 and UCB1-TUNED are able to achieve very good performance throughout the experiment. They all seem to offer almost identical performance, but UCB1-TUNED might be slightly ahead in this configuration.

Note that the ε_n-GREEDY schemes and L_{R-I} schemes are heavily dependent on the tuning parameter, and their performance in a specific environment may require extensive tuning.

6.8.3 Mixed strategy game results

In the mixed strategy game experiment configuration we perform two experiments in order to evaluate the performance of the bandit players. The first experiment consists of a tournament style game, and the second experiment consists of evaluating the players' performance when playing against themselves.

Experiment 1

This is a tournament style experiment where all players play against each other, and also against a copy of itself, and each players play both as player A and player B against each opponent. The experiment has been replicated 100 times, and each replication consists of 100 000 iterations.

The results of the tournament experiment are shown in Table 6.12, where as usual the first column denotes the players, while the remaining columns denote the average scores over 10, 100, 1000, 10 000 and 100 000 iterations.

As reported in Table 6.12, UCB1-TUNED is ahead of the other bandit players until the number of iterations is sufficiently large where the BLA players seem to catch up and pass UCB1-TUNED. However, overall UCB1-TUNED seems to be the top performer in this tournament experiment, with the BLA players following closely.

Ranking of the remaining players is not as easy, as their performance is highly dependent on their tuning parameters. We also observe how the $L_{R-\varepsilon P}$ is able to improve its performance as the number of iterations increases, which seems to fit well with the results in [11], where $L_{R-\varepsilon P}$ is shown to oscillate around the optimal distribution, and eventually converges to the true optimal distribution.

Table 6.12: Two-player zero-sum tournament score with matrix D3

Player / Iteration	**10**	**100**	**1000**	**10 000**	**100 000**
BLA Bernoulli	0.42	4.71	54.1	864	11782
BLA Normal unknown σ^2	0.00	0.65	38.6	841	11523
ε_n-GREEDY c =0.05 d =0.10	-0.07	3.82	6.3	321	7436
ε_n-GREEDY c =0.15 d =0.10	-0.12	2.04	14.5	478	9204
ε_n-GREEDY c =0.30 d =0.10	-0.02	-0.47	24.4	543	9746
$L_{R-\epsilon P}$ $\alpha = 0.002, \beta = 0.00001$	-0.12	-2.49	-26.9	568	10449
L_{R-I} 0.05	0.05	2.04	16.4	-1449	-18068
L_{R-I} 0.01	-0.08	-1.08	32.6	246	-11482
L_{R-I} 0.005	-0.10	-2.17	8.3	579	-3582
Pursuit 0.01	-0.03	-0.24	13.8	-345	-1693
Pursuit 0.005	-0.03	-1.11	16.8	-166	-1355
UCB1-TUNED	0.34	6.05	73.2	978	11751
POKER	-0.21	-11.74	-272.2	-3457	-35712

Experiment 2

In this experiment we let each player play against itself, i.e the same type of player with the same tuning parameters. The experiment has been replicated 1000 times, and each replication consists of 100 000 iterations.

In this experiment we take a closer look at the average distance from the optimal cumulative reward (0), and also at the action selection probability for the players as we want the players to find the optimal mixed strategy. We investigate each of these in the following.

Table 6.13 shows the average cumulative distance from the optimal cumulative reward when each player plays against itself. In this case the $L_{R-\varepsilon P}$ is unbeatable, with BLA Normal unknown σ^2 on second, followed by BLA Bernoulli, one of the ε_n-GREEDY schemes and then UCB1-TUNED.

Table 6.13: Distance of cumulative reward from optimal game reward (0)

Player / Iteration	**10**	**100**	**1000**	**10 000**	**100 000**
BLA Bernoulli	2.79	9.6	37	151	714
BLA Normal unknown σ^2	3.65	19.7	52	145	420
ε_n-GREEDY c =0.05 d =0.10	2.69	11.2	99	964	9881
ε_n-GREEDY c =0.15 d =0.10	2.71	8.9	68	594	5887
ε_n-GREEDY c =0.30 d =0.10	2.73	10.0	56	482	4693
$L_{R-\epsilon P}$ $\alpha = 0.002, \beta = 0.00001$	2.71	7.9	30	90	267
L_{R-I} 0.05	2.80	9.4	82	2127	22849
L_{R-I} 0.01	2.71	8.3	34	476	19870
L_{R-I} 0.005	2.69	8.2	40	129	12632
Pursuit 0.01	2.72	8.7	62	1475	17658
Pursuit 0.005	2.69	8.0	41	1276	17080
UCB1-TUNED	3.13	15.4	67	524	4806
POKER	2.99	8.7	143	1535	15670

In Table 6.14 the development of the action probability distribution for player A is listed for each player. As mentioned earlier, the optimal action distribution in this game is (0.25, 0.75) if the opponent player plays the optimal strategy of (0.5, 0.5). The development of action probability distribution for each player as player B is not included here, but is available in the appendix.

Table 6.14: Development of action selection probability for player A

Player / Iteration	**10**	**100**	**1000**	**10 000**	**100 000**
BLA Bernoulli	0.42, 0.58	0.20, 0.80	0.26, 0.74	0.26, 0.74	0.27, 0.73
BLA Normal unknown σ^2	0.55, 0.45	0.48, 0.52	0.26, 0.74	0.25, 0.75	0.24, 0.76
ε_n-GREEDYc =0.05 d =0.10	0.48, 0.52	0.25, 0.85	0.17, 0.83	0.19, 0.81	0.17, 0.83
ε_n-GREEDY c =0.15 d =0.10	0.51, 0.49	0.27, 0.73	0.21, 0.79	0.23, 0.77	0.20, 0.80
ε_n-GREEDY c =0.30 d =0.10	0.49, 0.51	0.44, 0.56	0.15, 0.85	0.24, 0.76	0.22, 0.78
$L_{R-\epsilon P}$ $\alpha = 0.002, \beta = 0.00001$	0.50, 0.50	0.48, 0.52	0.48, 0.52	0.11, 0.89	0.27, 0.73
L_{R-I} 0.05	0.49, 0.51	0.38, 0.62	0.09, 0.91	0.07, 0.93	0.07, 0.93
L_{R-I} 0.01	0.48, 0.52	0.48, 0.52	0.21, 0.79	0.15, 0.85	0.06, 0.94
L_{R-I} 0.005	0.50, 0.50	0.49, 0.51	0.40, 0.60	0.31, 0.69	0.04, 0.96
Pursuit 0.01	0.49, 0.51	0.44, 0.56	0.04, 0.96	0.03, 0.97	0.01, 0.99
Pursuit 0.005	0.48, 0.52	0.49, 0.51	0.09, 0.91	0.06, 0.94	0.02, 0.98
UCB1-TUNED	0.42, 0.58	0.23, 0.77	0.22, 0.78	0.21, 0.79	0.25, 0.75
POKER	0.48, 0.52	0.18, 0.82	0.11, 0.89	0.12, 0.88	0.08, 0.92

We observe that the BLA players seem to be very close to the optimal probability distribution already after about 1000 iterations, and have at this point already achieved the distribution of (0.26, 0.74). $L_{R-\epsilon P}$, which achieves best score with regards to the average reward distance, is not performing very well in this case, but the results seem to fit very well with the results in [11].

Chapter 7

Discussion and summary of results

As seen in the previous chapter the members of the Bayesian Learning Automaton family seem to offer impressive performance, and its members are among the top performers in every experiment. In the following sections we will go into more details about each BLA player and indicate issues with some members of the family with regards to the conjugate prior distribution. We also point out other application areas of the BLA family and propose areas of interest for further work. In addition we will outline some similarities between the traditional LA field and the Bayesian Learning Automata.

7.1 Performance of BLA Bernoulli

In the experiments with the general k-armed bandit problem with Bernoulli distributed feedback, BLA Bernoulli is the overall top performer, only beaten by UCB1-TUNED and Pursuit in experiment configuration 3, and by UCB1-TUNED and ε_n-GREEDY in experiment configuration 6.

In the various configurations of the two-player zero-sum game BLA Bernoulli and UCB1-TUNED offer very similar performance, and overall it is difficult to distinguish them. However, in the mixed strategy experiment where each player plays against itself, BLA Bernoulli seems to be the top performer of the two with regards to average distance from the optimal cumulative reward. In terms of average action selection probability distribution on the other hand, it could seem like UCB1-TUNED was the top performer of the two. However, average action selection probability distribution might be misleading, since if a distribution is always off by the value $\pm x$ from the optimal distribution, the average distribution might still be spot on.

BLA Bernoulli also performs well in the Goore game and was among the top performers in every experiment configuration. The results of the experiments indicate that a team of BLA Bernoulli offers best performance when the number of players is low, but offers good performance with a relative high number of players as well. Still, it was not able to keep up with the team of Tsetlin $L_{2N,2}$ automata, which offers superior performance in every configuration.

The beta prior distribution as used in BLA Bernoulli offers a great advantage compared to other conjugate priors introduced in this thesis, as a uniform non-informative prior may easily be constructed by setting $\alpha = 1$ and $\beta = 1$. This highly desirable property along with its superior performance makes BLA Bernoulli a very highly applicable and easy to use automaton.

7.2 Performance of BLA Poisson

BLA Poisson is among the top performers in both the 2-armed and 10-armed bandit problem with Poisson distributed feedback. In the 2-armed bandit problem BLA Poisson is only beaten by BLA Normal unknown σ^2 when considering the overall performance. The 10-armed bandit problem may be considered difficult as the expected rate parameter of the inferior arms may be very close to the expected rate parameter of the optimal arm, but BLA Poisson still achieves very good performance.

Bayesian statistics and prior distributions introduce potential problems concerning the prior beliefs that may be incorporated into the conjugate prior. In contrast to BLA Bernoulli, it is not possible to construct a uniform conjugate prior which we are able to sample from in relation to action selection. We discuss this further in Section 7.4.

7.3 Performance of BLA Normal

In Sections 5.4 and 5.5 we introduce BLA players designed for normally distributed feedback from the environment, both with known and unknown variance. Both of the BLA Normal players are able to handle any value of both mean μ and variance σ^2, which potentially make them very applicable. BLA Normal unknown σ^2 is introduced in all the experiments in this thesis and offers surprisingly good performance in every experiment.

A potential problem with BLA Normal is the choice of initial hyperparameters of the conjugate prior distribution, which we discuss in detail in Section 7.4. However, for the BLA Normal unknown σ^2 we were able to construct a single weak and wide prior distribution that offered very good performance throughout all of our experiments. The prior distribution was in fact very wide, and sampled values from the initial distribution could easily be in the interval $(-1 \times 10^{21}, 1 \times 10^{21})$, but the automaton was still able to offer impressive performance even though the value of the true mean μ always was within the interval [0,1]. We think this clearly demonstrates the strength of this automaton.

The normal distribution is a well-known and widely used probability distribution that is applicable to a wide variety of applications. Thus, it is clear that the improved performance offered by BLA Normal can have a significant impact in many such important areas.

7.4 Prior beliefs

The conjugate prior distribution as used in the BLA players is ideally initially chosen to be non-informative, which is achieved by carefully selecting the initial values of the hyperparameters.

The conjugate prior for the Bernoulli distribution, the beta distribution, is convenient since a uniform non-informative prior distribution is easily constructed by setting $\alpha = \beta = 1$. The resulting prior distribution makes it possible to weight every possible value of the parameter p of the Bernoulli distribution equally, where $p \in [0, 1]$.

The conjugate priors for the Poisson and normal distribution are more complicated with regards to the initial beliefs about the unknown parameters. The reason is that the rate parameter λ of the Poisson distribution can take any value in the interval $[0, \infty)$, and the normal mean μ of the normal distribution may take any value in the interval $(-\infty, \infty)$. These intervals make it impossible to make a completely non-informative prior as in the case with the beta prior, as the integrals tend to infinity. This means, that as used in this thesis, we are not able to weight every possible value of λ or μ equally, hence we create a skewed prior where all values are not equally weighted.

It is crucial to the performance of the BLA that we avoid a prior that has a too large impact on the posterior distribution. If a prior is too skewed from the true value of an unknown parameter, and also has a large impact on the posterior, it may take a considerable number of iterations before the unknown parameter is realistically covered by the distribution.

On the other hand, if there exists a prior belief about the true value of an unknown parameter, we might use this information to create an initial prior that weight some values higher. Thus, if we do have a certain belief about an unknown parameter, we may use this information to construct a more specific prior that may greatly improve the performance of the BLA.

In practical usage it is often possible to get a vague idea about the characteristics of an environment and use this information to create a well-functioning initial prior. Indeed, as we have demonstrated with BLA Normal unknown σ^2, we were able to create a well-performing player by creating a weak and wide initial prior distribution.

Alternatively, instead of creating an initial prior before starting an experiment, it is possible to introduce an initialization phase similar as used in UCB1-NORMAL, where each action initially is selected a predetermined number of times. With BLA Normal known σ^2 we introduced such a phase, by initially select each action once and use the received feedback as the initial mean. This could be extended further with several attempts and more complex calculations in order to obtain initial values of the hyperparameters.

7.5 Bayesian Learning Automata and the LA field

In Section 2.4 we gave a brief introduction to the field of Learning Automata. LA started as simple state machines with either deterministic or stochastic state transitions, and later extended to the variable structure automata where transition or action probabilities were able to change.

We argue that the Bayesian Learning Automata may be modeled as an extension of the variable structure automata as presented in Section 2.4.4, where each state represents an action. The BLA do not directly maintain an action probability vector as traditional variable structure automata do, and given the hyperparameters of a single action, it is still not possible to determine the probability of selecting that particular action. As described in Section 5.1.2, the probability of selecting a particular action is determined by each action's associated probability distribution on the parameter θ, and it is the distributions' relative differences that determine the action selection probabilities. By abstracting this a level and considering a series of known probability distributions and parameters associated with each action, there still exists a certain probability of selecting every single action. With this abstraction a BLA may be modeled as a kind of variable structure automaton, in the sense it does not keep track of a probability vector directly, but given an internal state of the automaton it is still a certain probability of selecting each action.

This is in contrast to for instance learning schemes such as UCB1-TUNED, where all randomness is introduced by the environment, and given an internal state of the learning scheme it is predetermined which action that is to be selected.

7.6 Non-stationary environments

All BLA players evaluated and introduced in this thesis are designed for stationary environments, but we have introduced them to non-stationary environments in form of some classical games from game theory. The reason for this is to observe the behavior of the BLA players compared to other bandit players in these games, and to observe if they are able to identify the optimal solutions in these games.

The updating rules of the hyperparameters presented in Chapter 5 indicate that previous received feedback has a large impact on the performance of the automata. Therefore, the BLA are not able to simply "forget" previous received feedback. This causes the BLA to be slower to adapt to possible changes as the number of iterations and corresponding feedback from the environment increase.

An interesting and potential useful extension of the BLA players is to modify the players to tackle non-stationary environments. In a non-stationary environment, i.e where an optimal action either suddenly or gradually changes feedback probability such that it no longer is the optimal action, the BLA players should detect possible changes and focus the attention on identifying the new optimal action.

A potential solution to this could be to introduce some mechanism to make the automata “forget” earlier received feedback, or in some other way detect and adapt in a non-stationary environment.

7.7 Applicability of BLA in cooperative and decentralized systems

In this thesis we have evaluated BLA Bernoulli and BLA Normal unknown σ^2 in the Goore game, due to its decentralized and cooperative nature. These characteristics make the Goore game to a simplified model of more complex cooperative and decentralized learning systems, where automata may be organized in more complex structures.

Such systems typically include a large number of participants, and it may therefore not be possible nor practical to run a simulation until every participant has identified the action that is in the best interest for the system as a whole. Instead, it might be sufficient to stop the simulation at a certain point, and then use the current belief of each automaton to make a global decisions for the entire system.

We think the BLA might be very useful in such systems and consider this to be an area of interest for further work. For instance, we could easily extract mean feedback of every explored action of each automaton in the system, and then use this information in the global decision making. The ability to extract mean feedback for each action is not a unique feature with the BLA, but the highly effective balance between exploration and exploitation could in some situations possibly lead to greatly improved performance of such systems.

Chapter 8

Conclusion and further work

8.1 Conclusion

In this thesis we have have performed an extensive empirical evaluation of the Bayesian Learning Automaton family, and we have extended the BLA family by introducing the three new members BLA Poisson, BLA Normal known σ^2 and BLA Normal unknown σ^2.

In the empirical evaluation of the general k-armed bandit problem the BLA players are among the top performers within their intended feedback distribution. In addition, BLA Normal unknown σ^2 offers good performance in most experiments it is applied in, and seems to be applicable in a wide range of problems.

By using conjugate prior distributions and applying simple updating rules on a set of hyperparameters associated with each action in the BLA, it is possible to utilize Bayesian statistics in a learning automaton in a computationally efficient manner. Bayesian statistics introduces some highly desirable properties, such as the lack of external tuning and accuracy control, which schemes such as ε_n-GREEDY, Pursuit and L_{R-I} are heavily dependent on.

However, the Bayesian Learning Automata are dependent on initial conjugate prior distributions that realistically contain the true values of the unknown parameters θ. If the unknown parameters are unlikely given the prior distribution, the performance might be heavily impacted.

Choice of initial hyperparameters of the conjugate prior may be an issue in some cases, but as we have demonstrated it is possible to construct a reasonable prior which offers excellent performance.

Issues with the choice of initial prior distribution may be avoided by using an initialization phase similar to the one used by UCB1-NORMAL. We may then use the information obtained in the initialization phase to construct a prior distribution that is used throughout the rest of the experiment.

We have also demonstrated that the BLA players are able to offer very good performance in the Goore game, prisoners' dilemma and two-player zero-sum game. However, dynamic changes in feedback distributions as a result of the other players changing action selection preferences seem to hold the automata somewhat back from reaching the optimal outcome in some of these games, especially in the Goore Game. This is however not very surprising in non-stationary environments, as the BLA are designed for use in stationary environments, and are slow to adapt to changes in the feedback distributions.

The experiments with the Goore game indicate that the BLA players are able to handle decentralized and cooperative systems. The teams of BLA players were among the top performers in every experiment configuration. However, the BLA players seem to struggle more when the number of participants increases.

In the iterative prisoners' dilemma the BLA players seem to behave as most rational players do in this game, which is to confess. The experiment indicates that BLA Normal unknown σ^2 is the overall top performer with the exception of Tit-For-Tat in the iterative prisoners' dilemma.

Quite surprisingly, the BLA also seem to be able to solve both pure and mixed strategy two-player zero-sum games in a very efficient manner, and are among the fastest players to identify the optimal mixed strategy when playing against themselves.

Thus, we believe that the family of Bayesian Learning Automata is an important addition to the field of bandit playing algorithms, and provides an interesting avenue for further research. Our experiments indicate that the automata seem to be robust and highly adaptable to different environments, and we believe the BLA could possible lead to improved performance in a wide range of important fields.

8.2 Further work

As mentioned, the choice of initial values of the hyperparameters of the conjugate prior distribution is a potential problem in BLA Poisson and BLA Normal with both known and unknown σ^2. This is an area which has to be solved for easy and widespread use of the new players introduced in this thesis. A possible solution could be to introduce an initialization phase which consists of a predetermined number of trials of each action, and then use the results from these trials to compute initial values of the hyperparameters.

In this thesis we evaluate the BLA players in non-stationary environments, although they are not explicitly designed for such environments. Therefore, an interesting and possibly important extension of the players introduced in this thesis, would be support for non-stationary environments.

It is also of great interest to observe how structures of several BLA players could be used to solve more complex cooperation problems. We believe the Bayesian approach could be of great usage in such systems, where it might be beneficial to be able to stop a simulation at any time and extract the current belief. For instance, the mean feedback received by each automaton in the structure could be extracted and used to make global decisions.

It could also be of great interest to utilize the various members of the BLA family in some real life problems in order to investigate their applicability to such problems.

Bibliography

[1] B. J. Oommen, S. Misra, and O.-C. Granmo, "Routing bandwidth-guaranteed paths in mpls traffic engineering: A multiple race track learning approach," *IEEE Transactions on Computers*, vol. 56, pp. 959–976, 2007.

[2] O.-C. Granmo, B. J. Oommen, S. Myrer, and M. Olsen, "Determining optimal polling frequency using a learning automata-based solution to the fractional knapsack problem," *IEEE Cybernetics and Intelligent Systems*, vol. 7-9, pp. 1–7, 2006.

[3] O.-C. Granmo, B. J. Oommen, S. Myrer, and M. Olsen, "Learning automata-based solutions to the nonlinear fractional knapsack problem with applications to optimal resource allocation," *IEEE Transactions on systems, man, and cybernetics*, vol. 37, pp. 166–175, 2007.

[4] B. J. Ommen, O.-C. Granmo, and A. Pedersen, "Using stochastic al techniques to achieve unbounded resolution in finite player goore games and its applications," *IEEE Symposium on Computational Intelligence and Games*, pp. 161–167, 2007.

[5] O.-C. Granmo, "The bayesian learning automaton - empirical evaluation with two-armed bernoulli bandit problems," *ICMLA Machine Learning and Applications*, pp. 23–30, 2008.

[6] R. Iyer and L. Kleinrock, "Qos control for sensor networks," *Communications, 2003. ICC '03. IEEE International Conference on*, pp. 517– 521 vol.1, 2003.

[7] S. P. H. Heap and Y. Varoufakis, *Game Theory: A Critical Introduction.* Routledge, 1995.

[8] T. M. Mitchell, *Machine Learning.* McGraw Hill, 1997.

[9] M. A. L. Thathachar and P. Sastry, *Networks of Learning automata.* Kluwer Academic Publishers, 2003.

[10] S. J. Russel and P. Norvig, *Artificial Intelligence A Modern Approach, Second Edition.* Prentice Hall, 2003.

[11] K. S. Narendra and M. A. L. Thathachar, *Learning Automata: An Introduction.* Prentice Hall, 1989.

[12] K. S. Narendra and M. Thathachar, "Learning automata: A survey," *IEEE transactions on systems, man, and cybernetics*, vol. SMC-14, pp. 323–334, 1974.

[13] M. Thathachar and P. Sastry, "Varieties of learning automata: An overview," *IEEE transactions on systems, man, and cybernetics*, vol. 32, pp. 711–722, 2002.

[14] N. Cesa-Bianchi and G. Lugosi, *Prediction, Learning, and Games.* Cambridge Univerity Press, 2006.

[15] H. Robbins, "Some aspects of the sequential design of experiments," *Bulletin of the American Mathematical Society*, vol. 58, no. 5, pp. 527–535, 1952.

[16] P. Auer, N. Cesa-Bianchi, and P. Fischer, "Finite-time analysis of the multiarmed bandit problem," *Machine learning*, vol. 47, pp. 235–256, 2002.

[17] M. Tsetlin, *Automaton Theory and Modeling of Biological Systems.* Academic Press, 1973.

[18] B. Tung and L. Kleinrock, "Distributed control methods," *High Performance Distributed Computing, 1993., Proceedings the 2nd International Symposium on*, pp. 206–215, 1993.

[19] W. M. Bolstad, *Introduction to Bayesian Statistics.* John Wiley & Sons Inc, 2007.

[20] K.-R. Koch, *Introduction to Bayesian Statistics, Second Edition.* Springer Verlag, 2007.

[21] M. Fisz, *Probability Theory and Mathematical Statistics.* John Wiley & Sons Inc, 1967.

[22] G. Young and R. L. Smith, *Essentials of Statistical Inference.* Cambridge University Press, 2005.

[23] A. Gelman, J. B. Carlin, H. S. Stern, and D. B. Rubin, *Bayesian Data Analysis.* Chapmen & Hall/CRC, 2003.

[24] J. O. Berger, *Statiscal Decision Theory and Bayesian Data Analysis, Second Edition.* Springer Verlag, 1985.

[25] A. F. M. S. JosÃ¨ M. Bernardo, *Bayesian Theory.* John Wiley & Sons Inc, 2000.

[26] M. Tsetlin, "On the behaviour of finite automata in random media," *Automation and Remote Control*, vol. 22, pp. 1210–19, 1962.

[27] J. Vermorel and M. Mohri, "Multi-armed bandit algorithms and empirical evaluation," *ECML*, pp. 437–448, 2005.

[28] P. Auer, N. Cesa-Bianchi, Y. Freund, and R. E. Schapire, "The non-stochastic multi-armed bandit problem," 2001.

[29] C. J. C. H. Watkins, "Learning from delayed rewards," 1989.

Appendix A

Unabridged experiments and results

In this chapter we present unabridged experiments and results from the empirical evaluation of the Bayesian Learning Automaton family.

In Sections A.1.2, A.2 and A.3 we present the experiments and results of the 2-armed and 10-armed bandit problem with Bernoulli, Poisson and normally distributed feedback, respectively. We then present the experiments and results of the Goore game, iterative prisoner's dilemma and two-player zero-sum game through Sections A.4 - A.6.

A.1 K-armed bandit problem with Bernoulli distributed feedback

Table A.1: Experiment configurations for the k-armed bandit problem with Bernoulli distributed feedback

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

Table A.2: Results of the 2-armed and 10-armed bandit problem with Bernoulli distributed feedback

Player / Configuration	**1**	**2**	**3**	**4**	**5**	**6**
BLA Bernoulli	1.000	0.999	0.997	0.998	0.988	0.975
BLA Normal unknown σ^2	1.000	0.998	0.992	0.996	0.982	0.968
ε_n-GREEDY c =0.05 [1]	0.981	0.992	0.965	0.996	0.961	0.893
ε_n-GREEDY c =0.15 [1]	1.000	0.999	0.991	0.990	0.988	0.957
ε_n-GREEDY c =0.30 [1]	1.000	0.997	0.997	0.982	0.981	0.977
L_{R-I} 0.05	0.999	0.918	0.985	0.832	0.378	0.526
L_{R-I} 0.01	0.998	0.993	0.993	0.992	0.885	0.958
L_{R-I} 0.005	0.995	0.986	0.986	0.984	0.940	0.951
Pursuit 0.05	1.000	0.970	0.932	0.912	0.699	0.608
Pursuit 0.01	0.999	0.998	0.998	0.998	0.875	0.848
Pursuit 0.005	0.999	0.999	0.998	0.997	0.960	0.924
UCB1	0.999	0.983	0.982	0.979	0.848	0.848
UCB1-TUNED	1.000	0.997	0.997	0.997	0.977	0.978
UCB1-NORMAL	0.995	0.977	0.969	0.959	0.797	0.737
Exp3 0.01	0.990	0.978	0.980	0.913	0.736	0.749
POKER	0.995	0.991	0.876	0.982	0.916	0.812
INTESTIM0.01	0.961	0.949	0.796	0.920	0.905	0.577

[1] Parameter d is set to be the difference between the best arm and the next best arm

A.1.1 Bernoulli distributed feedback and the 2-armed bandit problem

Results of experiment configuration 1

Table A.3: Detailed overview of the 2-armed bandit problem with Bernoulli distributed feedback with optimal arm $p = 0.9$ and 0.6 on the inferior arm

Player / Iteration	**10**	**100**	**1000**	**10 000**	**100 000**
BLA Bernoulli	0.666	0.910	0.987	0.998	1.000
BLA Normal unknown σ^2	0.639	0.880	0.977	0.996	1.000
ε_n-GREEDY c =0.05 d =0.3	0.625	0.859	0.944	0.970	0.981
ε_n-GREEDY c =0.15 d =0.3	0.586	0.896	0.984	0.998	1.000
ε_n-GREEDY c =0.30 d =0.3	0.518	0.858	0.978	0.997	1.000
L_{R-I} 0.05	0.516	0.673	0.950	0.995	0.999
L_{R-I} 0.01	0.500	0.538	0.786	0.977	0.998
L_{R-I} 0.005	0.496	0.519	0.670	0.953	0.995
Pursuit 0.05	0.528	0.837	0.983	0.998	1.000
Pursuit 0.01	0.505	0.630	0.935	0.993	0.999
Pursuit 0.005	0.502	0.571	0.876	0.987	0.999
UCB1	0.617	0.774	0.921	0.984	0.999
UCB1-TUNED	0.707	0.912	0.984	0.998	1.000
UCB1-NORMAL	0.519	0.489	0.841	0.969	0.995
Exp3 0.01	0.501	0.519	0.671	0.947	0.990
POKER	0.571	0.856	0.980	0.993	0.995
INTESTIM 0.01	0.624	0.879	0.950	0.960	0.961

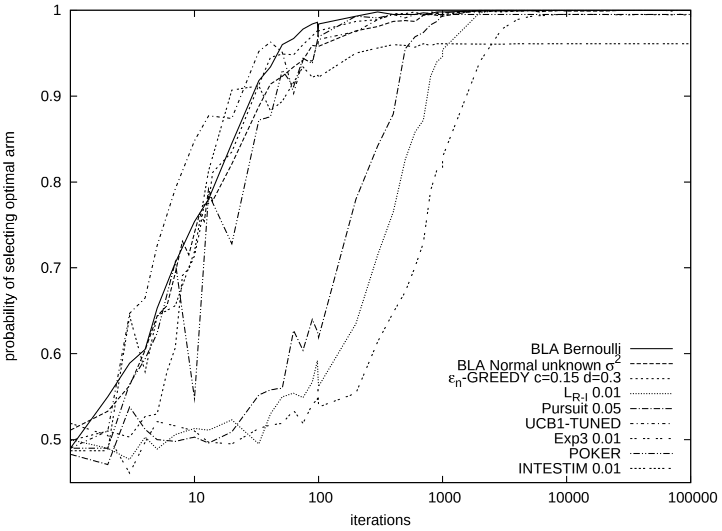


Figure A.1: Optimal action selection probability in the 2-armed bandit problem with Bernoulli distributed feedback, with optimal arm $p = 0.9$ and 0.6 on the inferior arm

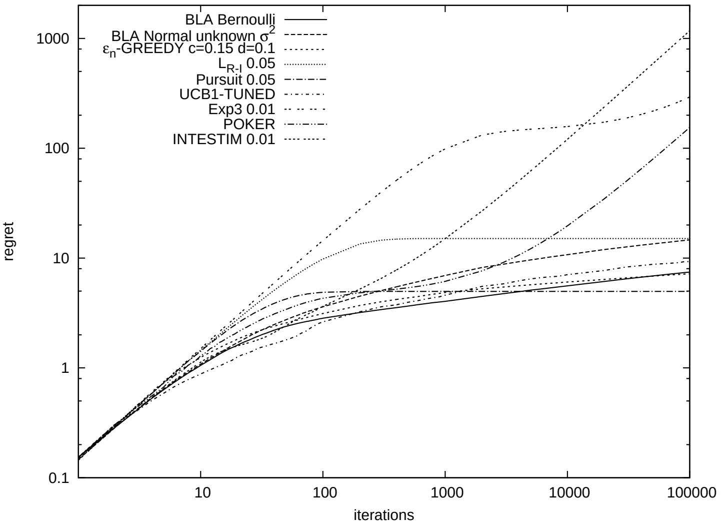


Figure A.2: Regret in the 2-armed bandit problem with Bernoulli distributed feedback with optimal arm $p = 0.9$ and 0.6 on the inferior arm

Results of experiment configuration 2

Table A.4: Detailed overview of the 2-armed bandit problem with Bernoulli distributed feedback with optimal arm $p = 0.9$ and 0.8 on the inferior arm

Player / Iteration	**10**	**100**	**1000**	**10 000**	**100 000**
BLA Bernoulli	0.550	0.727	0.932	0.990	0.999
BLA Normal unknown σ^2	0.551	0.716	0.920	0.986	0.998
ε_n-GREEDY c =0.05 d =0.1	0.500	0.727	0.925	0.974	0.992
ε_n-GREEDY c =0.15 d =0.1	0.496	0.628	0.921	0.989	0.999
ε_n-GREEDY c =0.30 d =0.1	0.495	0.538	0.880	0.981	0.997
L_{R-I} 0.05	0.504	0.556	0.794	0.906	0.918
L_{R-I} 0.01	0.498	0.513	0.616	0.926	0.993
L_{R-I} 0.005	0.502	0.505	0.560	0.858	0.986
Pursuit 0.05	0.505	0.725	0.931	0.965	0.970
Pursuit 0.01	0.508	0.585	0.914	0.990	0.998
Pursuit 0.005	0.502	0.549	0.862	0.986	0.999
UCB1	0.540	0.611	0.750	0.913	0.983
UCB1-TUNED	0.568	0.730	0.910	0.981	0.997
UCB1-NORMAL	0.492	0.423	0.643	0.878	0.977
Exp3 0.01	0.507	0.506	0.560	0.843	0.978
Poker	0.521	0.673	0.905	0.983	0.991
IntEstim 0.01	0.547	0.724	0.896	0.943	0.949

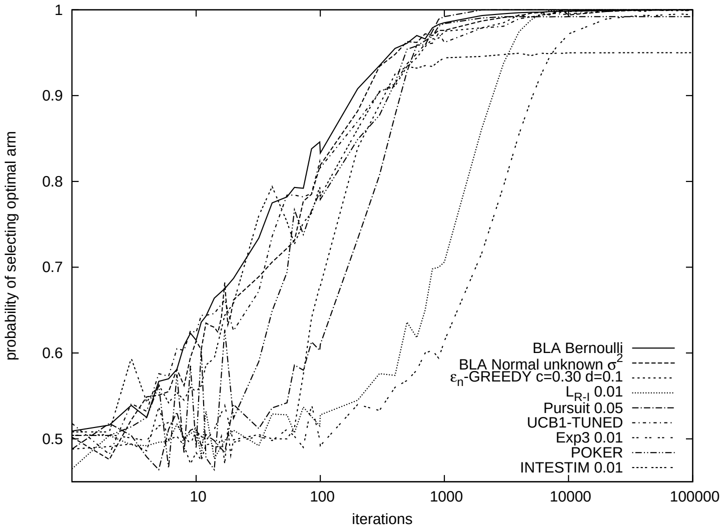


Figure A.3: Optimal action selection probability in the 2-armed bandit problem with Bernoulli distributed feedback, with optimal arm $p = 0.9$ and 0.8 on inferior arm

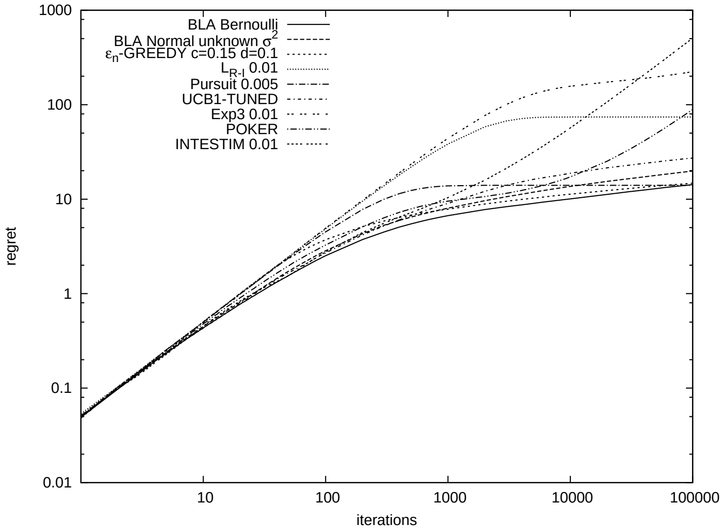


Figure A.4: Regret in the 2-armed bandit problem with Bernoulli distributed feedback with optimal arm $p = 0.9$ and 0.8 on the inferior arm

Results of experiment configuration 3

Table A.5: Detailed overview of the 2-armed bandit problem with Bernoulli distributed feedback with optimal arm $p = 0.55$ and 0.45 on the inferior arm

Player / Iteration	**10**	**100**	**1000**	**10 000**	**100 000**
BLA Bernoulli	0.558	0.670	0.884	0.981	0.997
BLA Normal unknown σ^2	0.544	0.658	0.864	0.969	0.992
ε_n-GREEDY c =0.05 d =0.1	0.505	0.671	0.873	0.943	0.965
ε_n-GREEDY c =0.15 d =0.1	0.497	0.597	0.881	0.972	0.991
ε_n-GREEDY c =0.30 d =0.1	0.505	0.529	0.857	0.978	0.997
L_{R-I} 0.05	0.499	0.562	0.834	0.971	0.985
L_{R-I} 0.01	0.508	0.513	0.618	0.927	0.993
L_{R-I} 0.005	0.503	0.507	0.560	0.859	0.986
Pursuit 0.05	0.510	0.638	0.876	0.926	0.932
Pursuit 0.01	0.504	0.547	0.854	0.984	0.998
Pursuit 0.005	0.500	0.523	0.774	0.975	0.998
UCB1	0.548	0.606	0.752	0.913	0.982
UCB1-TUNED	0.580	0.706	0.896	0.982	0.997
UCB1-NORMAL	0.500	0.580	0.693	0.867	0.969
Exp3 0.01	0.495	0.503	0.560	0.851	0.980
Poker	0.546	0.662	0.797	0.867	0.876
IntEstim 0.01	0.565	0.646	0.752	0.790	0.796

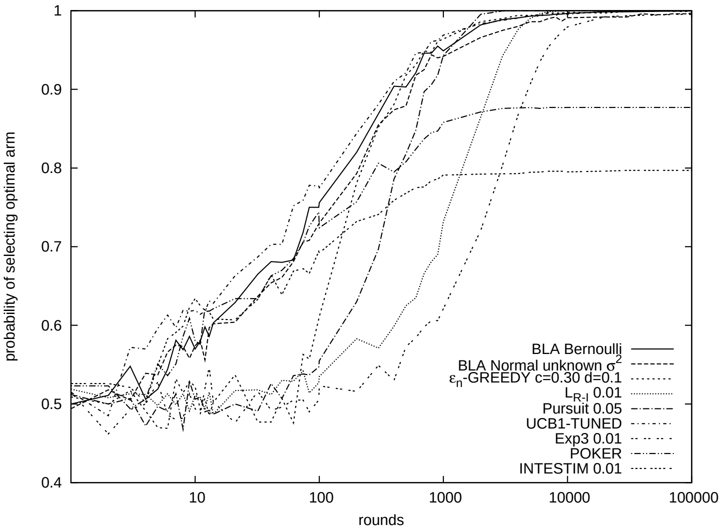


Figure A.5: Optimal action selection probability in the 2-armed bandit problem with Bernoulli distributed feedback, with optimal arm $p = 0.55$ and 0.45 on inferior arm

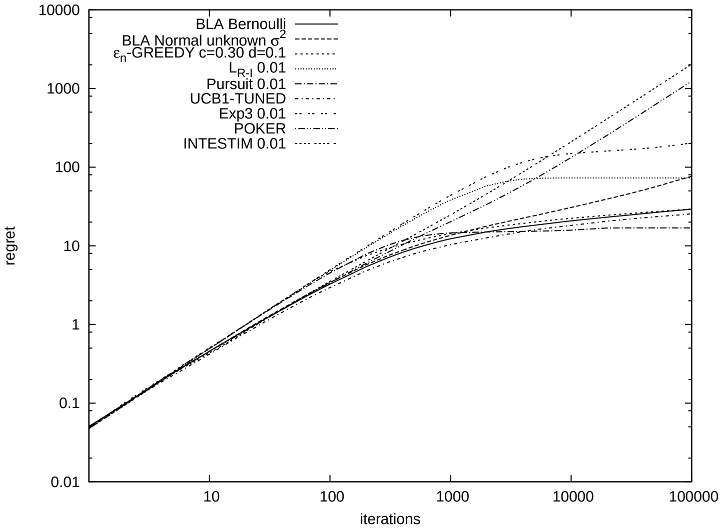


Figure A.6: Regret in the 2-armed bandit problem with Bernoulli distributed feedback with optimal arm $p = 0.55$ and 0.45 on the inferior arm

A.1.2 Bernoulli distributed feedback and the 10-armed bandit problem

Results of experiment configuration 4

Table A.6: Detailed overview of the 10-armed bandit problem with Bernoulli distributed feedback with optimal arm $p = 0.9$ and 0.6 on the inferior arms

Player / Iteration	**10**	**100**	**1000**	**10 000**	**100 000**
BLA Bernoulli	0.131	0.434	0.888	0.984	0.998
BLA Normal unknown σ^2	0.105	0.281	0.808	0.969	0.996
ε_n-GREEDY c =0.05 d =0.3	0.098	0.194	0.810	0.971	0.996
ε_n-GREEDY c =0.15 d =0.3	0.100	0.102	0.607	0.930	0.990
ε_n-GREEDY c =0.30 d =0.3	0.101	0.099	0.405	0.878	0.982
L_{R-I} 0.05	0.105	0.186	0.687	0.818	0.832
L_{R-I} 0.01	0.099	0.111	0.335	0.917	0.992
L_{R-I} 0.005	0.106	0.109	0.197	0.842	0.984
Pursuit 0.05	0.104	0.312	0.834	0.904	0.912
Pursuit 0.01	0.104	0.166	0.810	0.981	0.998
Pursuit 0.005	0.107	0.133	0.697	0.969	0.997
UCB-1	0.100	0.175	0.446	0.862	0.979
UCB1-TUNED	0.100	0.442	0.869	0.979	0.997
UCB1-NORMAL	0.112	0.102	0.161	0.728	0.959
Exp3 0.01	0.105	0.101	0.114	0.351	0.913
POKER	0.121	0.315	0.705	0.946	0.982
INTESTIM 0.01	0.133	0.459	0.839	0.912	0.920

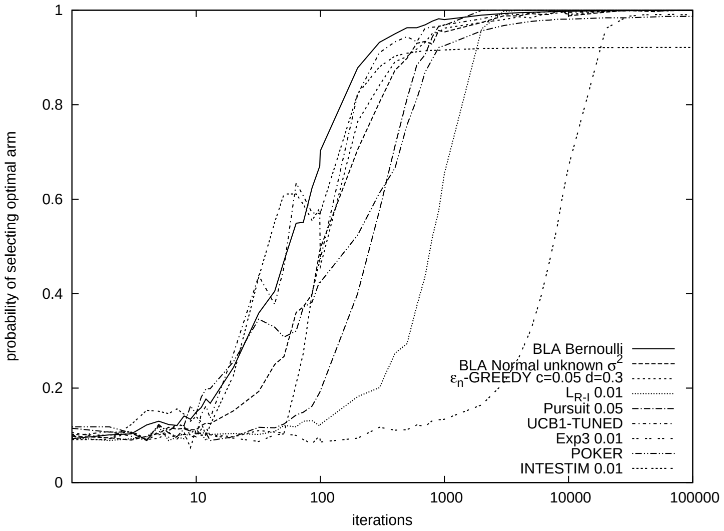


Figure A.7: Optimal action selection probability in the 10-armed bandit problem with Bernoulli distributed feedback, with optimal arm $p = 0.9$ and 0.6 on inferior arms

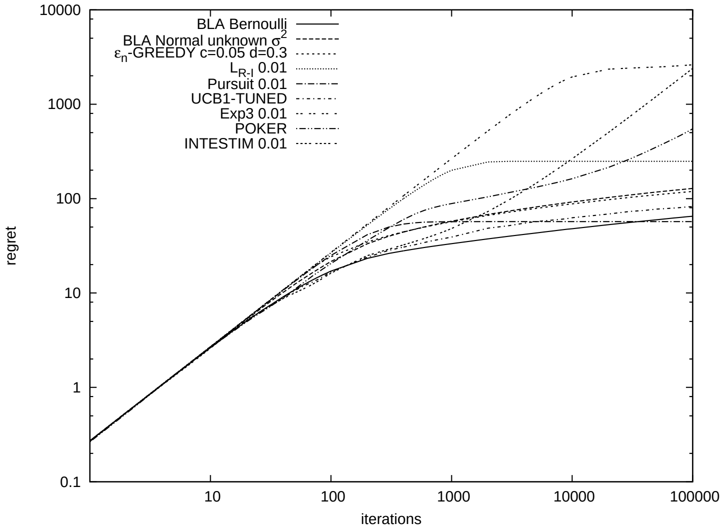


Figure A.8: Regret in the 10-armed bandit problem with Bernoulli distributed feedback with optimal arm $p =$ 0.9 and 0.6 on the inferior arms

Results of experiment configuration 5

Table A.7: Detailed overview of the 10-armed bandit problem with Bernoulli distributed feedback with optimal arm $p = 0.9$ and 0.8 on the inferior arms

Player / Iteration	**10**	**100**	**1000**	**10 000**	**100 000**
BLA Bernoulli	0.112	0.197	0.549	0.916	0.988
BLA Normal unknown σ^2	0.107	0.131	0.433	0.881	0.982
ε_n-GREEDY c =0.05 d =0.1	0.101	0.124	0.630	0.898	0.961
ε_n-GREEDY c =0.15 d =0.1	0.105	0.100	0.511	0.911	0.988
ε_n-GREEDY c =0.30 d =0.1	0.099	0.099	0.359	0.872	0.981
L_{R-I} 0.05	0.103	0.119	0.273	0.368	0.378
L_{R-I} 0.01	0.104	0.105	0.156	0.672	0.885
L_{R-I} 0.005	0.102	0.102	0.126	0.518	0.940
Pursuit 0.05	0.100	0.157	0.567	0.682	0.699
Pursuit 0.01	0.098	0.116	0.550	0.840	0.875
Pursuit 0.005	0.101	0.108	0.488	0.910	0.960
UCB1	0.100	0.119	0.166	0.406	0.848
UCB1-TUNED	0.100	0.164	0.425	0.841	0.977
UCB1-NORMAL	0.088	0.099	0.089	0.267	0.797
Exp3 0.01	0.097	0.099	0.104	0.156	0.736
POKER	0.105	0.180	0.444	0.751	0.916
INTESTIM 0.01	0.126	0.194	0.519	0.857	0.905

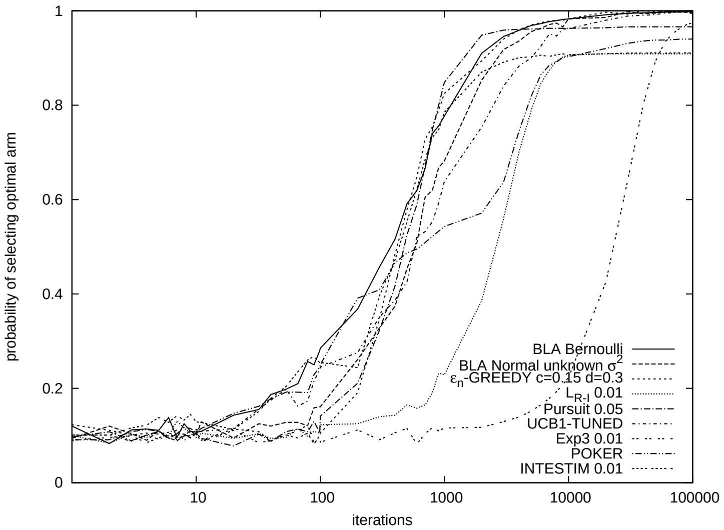


Figure A.9: Optimal action selection probability in the 10-armed bandit problem with Bernoulli distributed feedback, with optimal arm $p = 0.9$ and 0.8 on the inferior arms

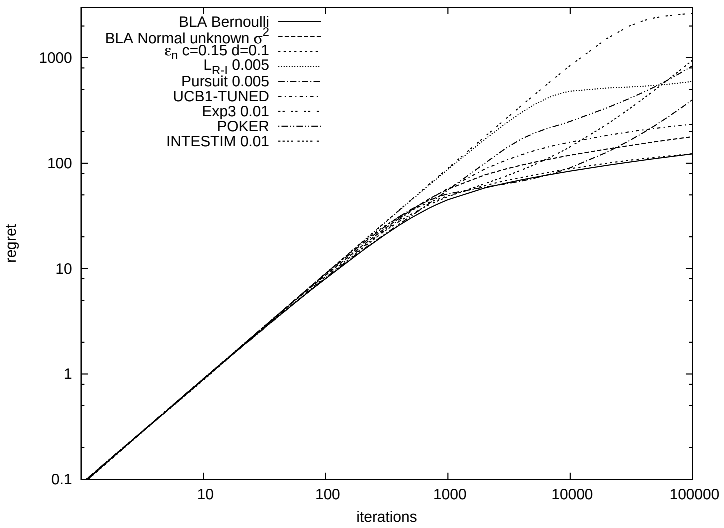


Figure A.10: Regret in the 10-armed bandit problem with Bernoulli distributed feedback with optimal arm $p = 0.9$ and 0.8 on the inferior arms

Results of experiment configuration 6

Table A.8: Detailed overview of the 10-armed bandit problem with Bernoulli distributed feedback with optimal arm $p = 0.55$ and 0.45 on the inferior arms

Player / Iteration	**10**	**100**	**1000**	**10 000**	**100 000**
BLA Bernoulli	0.109	0.150	0.340	0.834	0.975
BLA Normal unknown σ^2	0.101	0.144	0.353	0.828	0.968
ε_n-GREEDY c =0.05 d =0.1	0.102	0.120	0.480	0.787	0.893
ε_n-GREEDY c =0.15 d =0.1	0.103	0.101	0.401	0.835	0.957
ε_n-GREEDY c =0.30 d =0.1	0.097	0.098	0.289	0.847	0.977
L_{R-I} 0.05	0.105	0.123	0.329	0.507	0.526
L_{R-I} 0.01	0.102	0.103	0.158	0.728	0.958
L_{R-I} 0.005	0.101	0.102	0.123	0.526	0.951
Pursuit 0.05	0.102	0.139	0.449	0.589	0.608
Pursuit 0.01	0.098	0.112	0.396	0.793	0.848
Pursuit 0.005	0.104	0.105	0.309	0.838	0.924
UCB1	0.100	0.120	0.165	0.404	0.848
UCB1-TUNED	0.100	0.166	0.401	0.855	0.978
UCB1-NORMAL	0.100	0.099	0.146	0.283	0.737
Exp3 0.01	0.102	0.102	0.105	0.158	0.749
POKER	0.101	0.184	0.378	0.607	0.812
INTESTIM 0.01	0.100	0.174	0.379	0.549	0.577

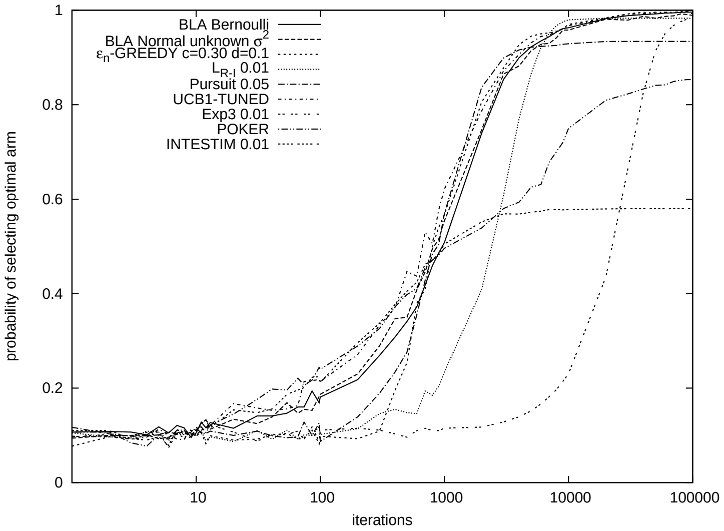


Figure A.11: Optimal action selection probability in the 10-armed bandit problem with Bernoulli distributed feedback, with optimal arm $p = 0.55$ and 0.45 on the inferior arms

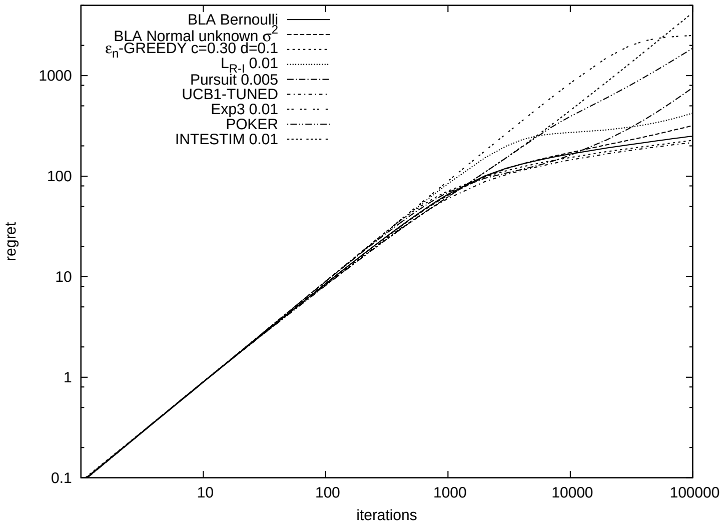


Figure A.12: Regret in the 10-armed bandit problem with Bernoulli distributed feedback with optimal arm $p = 0.55$ and 0.45 on the inferior arms

A.2 K-armed bandit problem with Poisson distributed feedback

Configuration	Number of arms	Range of λ
1	2	$\mathbb{R} \in [0, 10]$
2	10	$\mathbb{R} \in [0, 10]$

Table A.9: Experiment configuration for the k-armed bandit problem with Poisson distributed feedback

A.2.1 Poisson distributed feedback and the 2-armed bandit problem

Table A.10: Results for the 2-armed bandit problem with Poisson distributed feedback

Player / Iteration	10	100	1000	10 000	100 000
BLA Poisson	0.687	0.871	0.956	0.984	0.995
BLA Normal unknown σ^2	0.719	0.885	0.955	0.984	0.995
ε_n-GREEDY $c = 0.05$ $d = 0.1$	0.505	0.798	0.941	0.968	0.972
ε_n-GREEDY $c = 0.15$ $d = 0.1$	0.506	0.652	0.914	0.979	0.990
ε_n-GREEDY $c = 0.30$ $d = 0.1$	0.498	0.540	0.872	0.974	0.993
Pursuit 0.050	0.539	0.842	0.954	0.966	0.968
Pursuit 0.010	0.514	0.656	0.927	0.980	0.986
Pursuit 0.005	0.514	0.588	0.880	0.976	0.987
UCB1-NORMAL	0.486	0.774	0.912	0.962	0.985
POKER	0.711	0.898	0.953	0.968	0.974

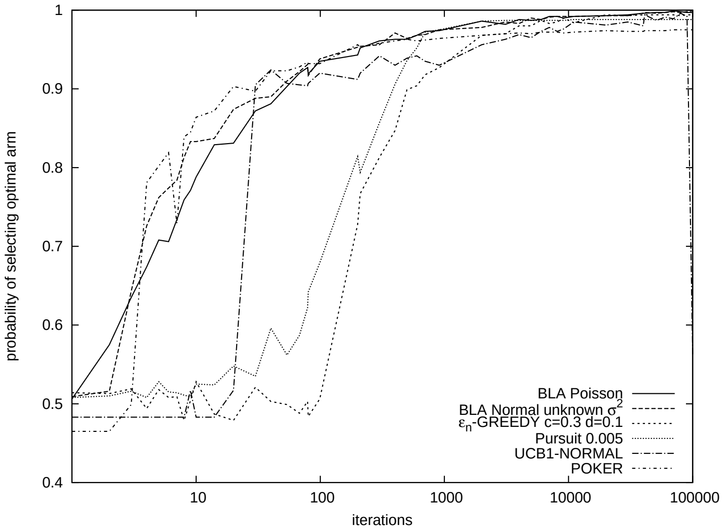


Figure A.13: Optimal action selection probability in the 2-armed bandit problem with Poisson distributed feedback

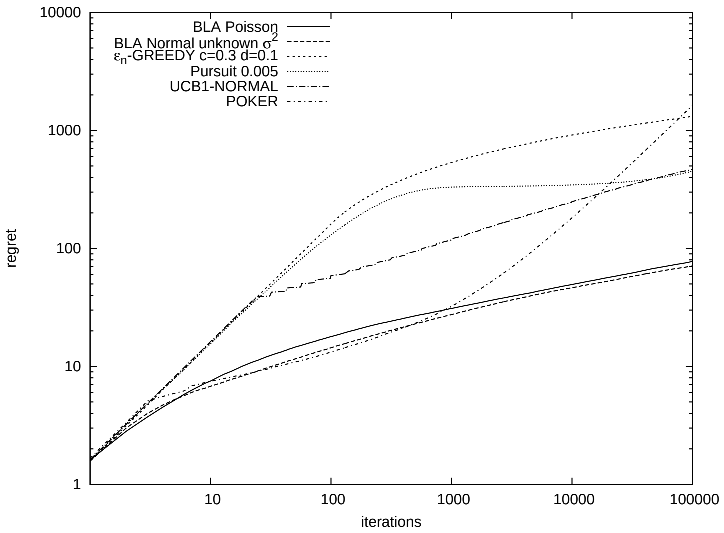


Figure A.14: Regret in the 2-armed bandit problem with Poisson distributed feedback

A.2.2 Poisson distributed feedback and the 10-armed bandit problem

Table A.11: Results of the 10-armed bandit problem with Poisson distributed feedback

Player / Iteration	**10**	**100**	**1000**	**10 000**	**100 000**
BLA Poisson	0.113	0.466	0.781	0.927	0.980
BLA Normal unknown σ^2	0.122	0.473	0.769	0.919	0.972
ε_n-GREEDY $c = 0.05$ $d = 0.1$	0.102	0.182	0.656	0.821	0.865
ε_n-GREEDY $c = 0.15$ $d = 0.1$	0.102	0.101	0.517	0.829	0.909
ε_n-GREEDY $c = 0.30$ $d = 0.1$	0.100	0.101	0.355	0.810	0.928
Pursuit 0.050	0.123	0.423	0.633	0.662	0.666
Pursuit 0.010	0.103	0.242	0.716	0.835	0.849
Pursuit 0.005	0.102	0.181	0.656	0.869	0.893
UCB1-NORMAL	0.108	0.099	0.503	0.772	0.910
POKER	0.109	0.518	0.766	0.859	0.874

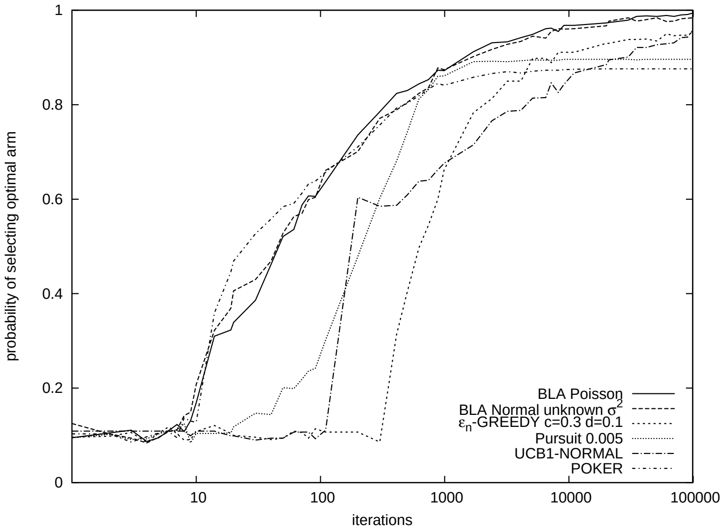


Figure A.15: Optimal action selection probability in the 10-armed bandit problem with Poisson distributed feedback

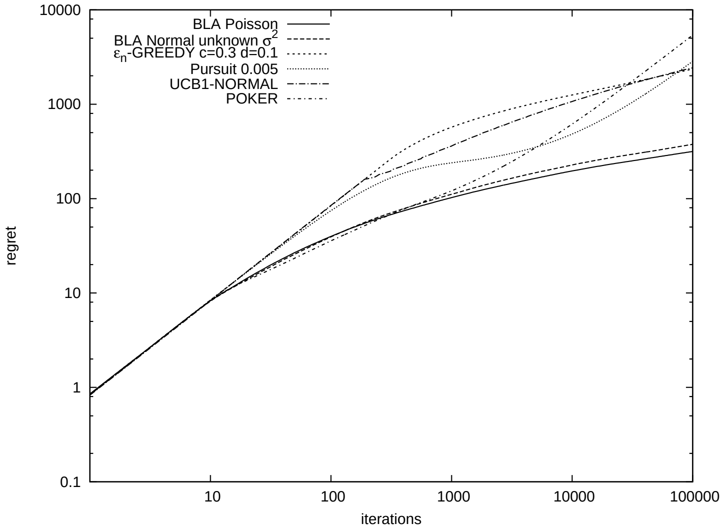


Figure A.16: Regret in the 10-armed bandit problem with Poisson distributed feedback

A.3 K-armed bandit problem with normally distributed feedback

Table A.12: Experiment configuration for the k-armed bandit problem with normally distributed feedback

Configuration	**Number of arms**	**Range of μ and σ^2**
1	2	$\mathbb{R} \in [0, 1]$
2	10	$\mathbb{R} \in [0, 1]$

A.3.1 Normally distributed feedback and the 2-armed bandit problem

Table A.13: Results of the 2-armed bandit problem with normally distributed feedback

Player / Iteration	**10**	**100**	**1000**	**10 000**	**100 000**
BLA Normal known σ^2	0.625	0.802	0.912	0.969	0.990
BLA Normal known $\sigma^2 = 1$	0.594	0.755	0.891	0.960	0.987
BLA Normal unknown σ^2	0.628	0.812	0.923	0.973	0.990
ε_n-GREEDY c =0.05	0.562	0.716	0.811	0.872	0.903
ε_n-GREEDY c =0.15	0.580	0.760	0.894	0.952	0.976
ε_n-GREEDY c =0.30	0.523	0.784	0.903	0.966	0.987
Pursuit 0.05	0.537	0.784	0.907	0.923	0.925
Pursuit 0.01	0.512	0.632	0.898	0.953	0.956
Pursuit 0.005	0.508	0.575	0.859	0.961	0.973
UCB1-NORMAL	0.510	0.631	0.791	0.904	0.964
POKER	0.647	0.806	0.895	0.932	0.945

Parameter d is set to be the difference between the best arm and the next best arm

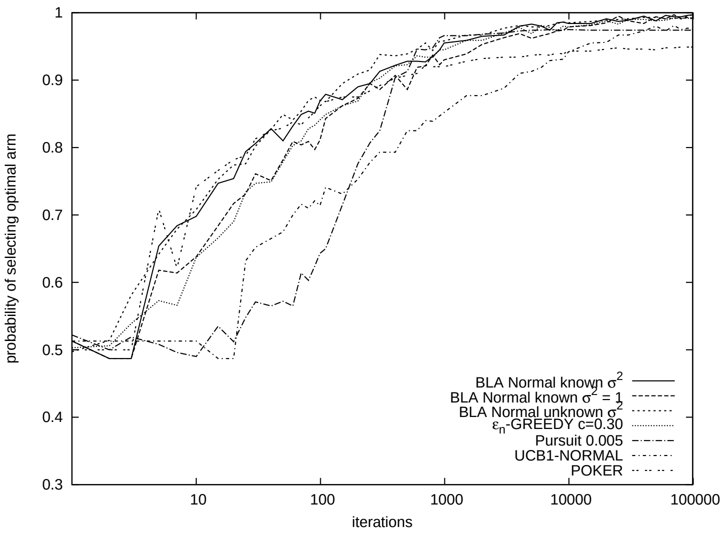


Figure A.17: Optimal action selection probability in the 2-armed bandit problem with normally distributed feedback

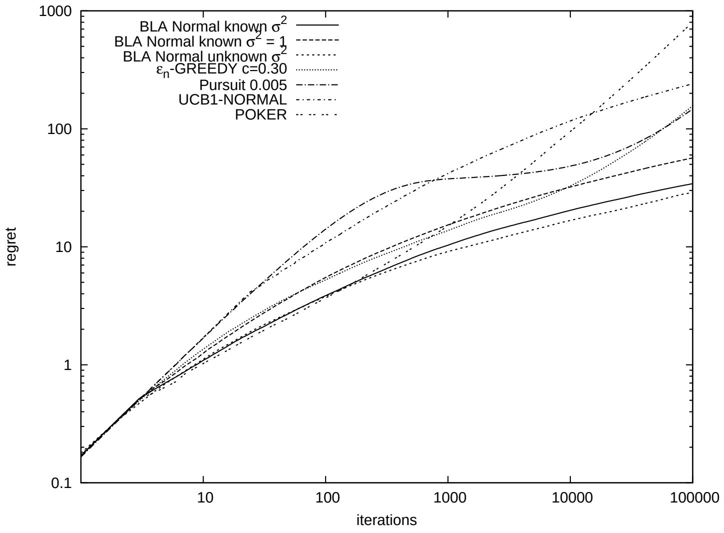


Figure A.18: Regret in the 2-armed bandit problem with normally distributed feedback

A.3.2 Normally distributed feedback and the 10-armed bandit problem

Table A.14: Results of the 10-armed bandit problem with normally distributed feedback

Player / Iteration	**10**	**100**	**1000**	**10 000**	**100 000**
BLA Normal known σ^2	0.100	0.235	0.521	0.804	0.928
BLA Normal known $\sigma^2 = 1$	0.100	0.214	0.490	0.788	0.925
BLA Normal unknown σ^2	0.105	0.300	0.633	0.858	0.955
ε_n-GREEDY c =0.05	0.103	0.194	0.465	0.692	0.820
ε_n-GREEDY c =0.15	0.100	0.142	0.378	0.671	0.858
ε_n-GREEDY c =0.30	0.097	0.118	0.309	0.620	0.833
Pursuit 0.05	0.116	0.347	0.566	0.600	0.605
Pursuit 0.01	0.097	0.191	0.618	0.758	0.775
Pursuit 0.005	0.097	0.152	0.576	0.813	0.842
UCB1-NORMAL	0.113	0.102	0.253	0.553	0.810
POKER	0.106	0.338	0.603	0.800	0.849

Parameter d is set to be the difference between the best arm and the next best arm

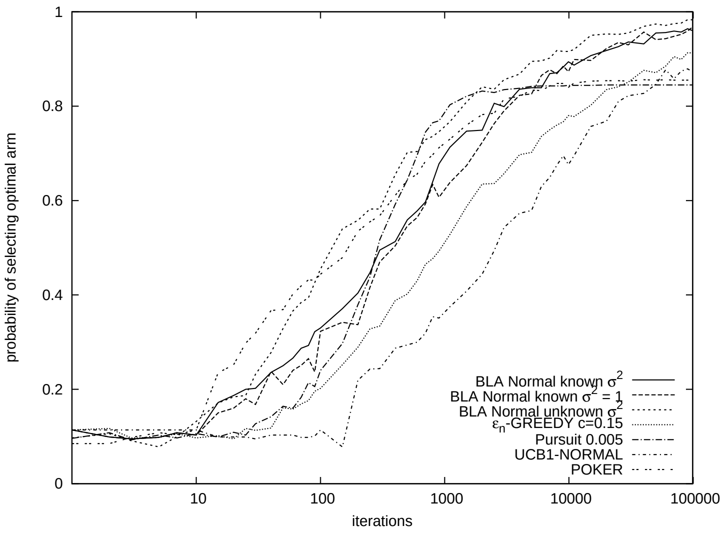


Figure A.19: Optimal action selection probability in the 10-armed bandit problem with normally distributed feedback

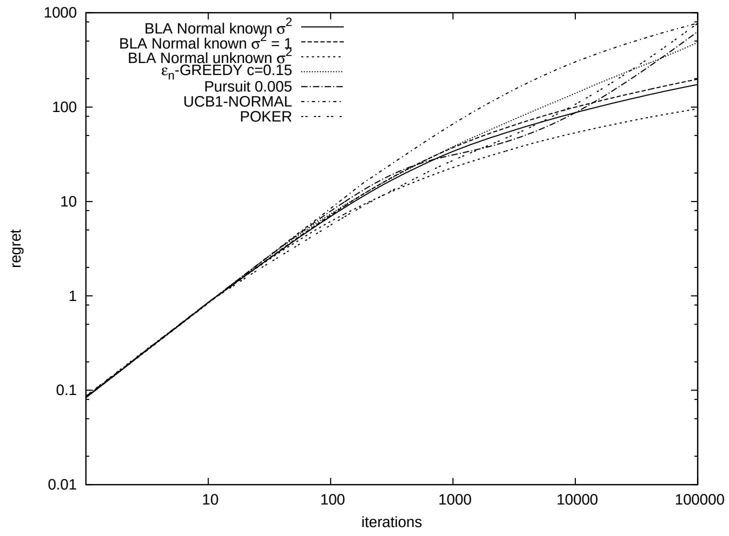


Figure A.20: Regret in the 10-armed bandit problem with normally distributed feedback

A.4 Goore game

Table A.15: Experiment configurations for the Goore game

Configuration	**Number of players**	**Optimal number of yes votes**
1	10	3
2	50	20
3	100	35

A.4.1 Results of experiment configuration 1

Table A.16: Distance from the optimal number of yes votes, with 10 players and 3 desired yes votes

Player / Iteration	**10**	**100**	**1000**	**10 000**	**100 000**
Tsetlin $L_{2N,2}$ $N = 1$	2.01	0.95	0.00	0.00	0.00
BLA Bernoulli	2.03	1.58	0.82	0.36	0.00
BLA Normal unknown σ^2	2.37	1.67	1.18	0.75	0.04
ε_n-GREEDY c =0.05 d =0.10	2.06	0.80	0.50	0.28	0.24
ε_n-GREEDY c =0.15 d =0.10	2.09	1.09	0.62	0.24	0.21
ε_n-GREEDY c =0.30 d =0.10	2.16	1.24	0.62	0.38	0.35
L_{R-I} 0.01	1.74	2.26	2.09	1.57	1.22
L_{R-I} 0.005	2.13	2.06	2.25	1.77	0.67
Pursuit 0.01	2.05	1.08	0.51	0.24	0.24
Pursuit 0.005	2.09	1.28	0.82	0.24	0.24
UCB1-TUNED	5.03	1.93	1.61	1.00	0.21
UCB1-NORMAL	3.00	2.01	1.91	1.64	0.85
POKER	2.23	2.79	1.09	0.92	0.18

Table A.17: Detailed overview of the Goore game with 10 players, where exactly 3 players should vote yes

Player / Iteration	**10**	**100**	**1000**	**10 000**	**100 000**
Tsetlin $L_{2N,2}$ $N = 1$	4.79	3.93	3.00	3.00	3.00
BLA Bernoulli	4.85	4.46	3.20	3.00	3.00
BLA Normal unknown σ^2	5.15	4.25	3.32	2.99	3.00
ε_n-GREEDY c =0.05 d =0.10	4.98	2.64	3.50	3.28	3.24
ε_n-GREEDY c =0.15 d =0.10	4.85	2.79	3.54	3.24	3.21
ε_n-GREEDY c =0.30 d =0.10	5.06	3.76	3.32	3.30	3.27
L_{R-I} 0.01	4.58	5.14	4.85	4.39	4.04
L_{R-I} 0.005	4.95	4.98	5.07	4.49	3.27
Pursuit 0.01	4.97	2.58	3.47	3.18	3.18
Pursuit 0.005	4.99	3.82	3.64	3.10	3.1
UCB1-TUNED	8.03	4.85	4.35	3.58	3.21
UCB1-NORMAL	0.0	4.89	4.75	4.50	3.65
POKER	5.13	5.77	3.63	2.96	2.82

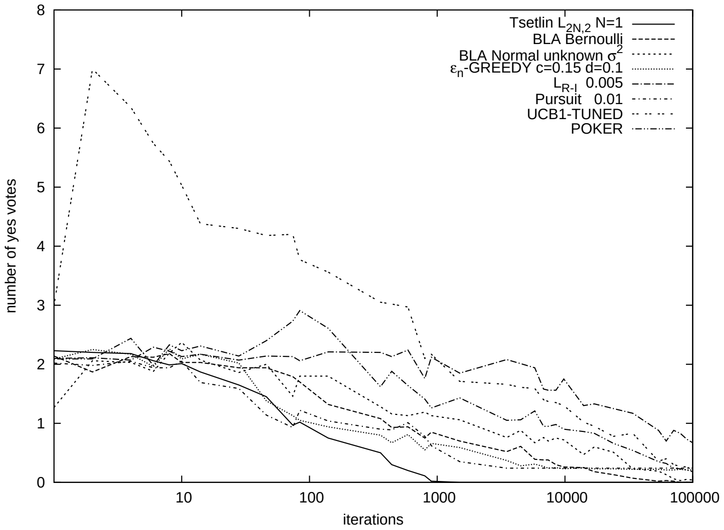


Figure A.21: Development of the distance from the optimal number of yes votes 3, with 10 players

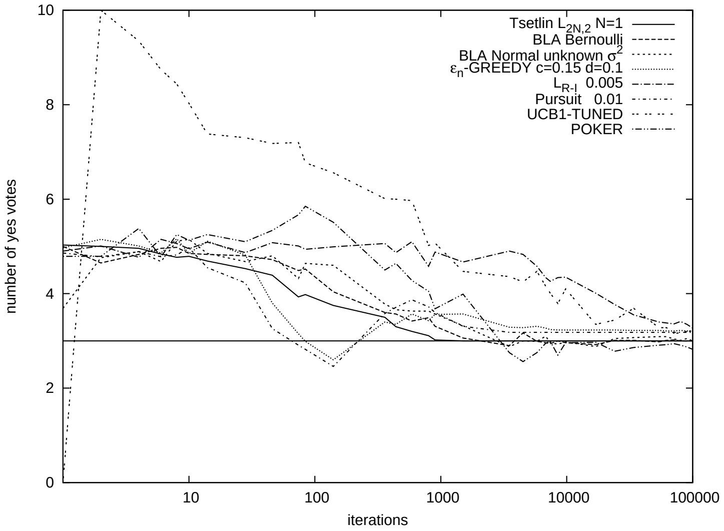


Figure A.22: Development of the number of yes votes with 10 players and 3 as the optimal number of yes votes

A.4.2 Results of experiment configuration 2

Table A.18: Distance from the optimal number of yes votes, with 50 players and 20 desired yes votes

Player / Iteration	**10**	**100**	**1000**	**10 000**	**100 000**
Tsetlin $L_{2N,2}$ $N=1$	3.25	0.54	0.00	0.00	0.00
BLA Bernoulli	4.61	2.61	1.12	1.02	0.15
BLA Normal unknown σ^2	4.87	2.72	2.01	1.43	0.29
ε_n-GREEDY c =0.05 d =0.10	4.50	2.51	1.37	1.07	1.06
ε_n-GREEDY c =0.15 d =0.10	5.20	2.56	1.29	1.0	0.90
ε_n-GREEDY c =0.30 d =0.10	4.64	3.27	1.46	1.32	1.19
L_{R-I} 0.01	5.77	4.83	5.11	2.06	0.96
L_{R-I} 0.005	5.14	5.36	4.92	2.64	0.96
Pursuit 0.01	4.21	2.77	0.89	0.75	0.75
Pursuit 0.005	4.90	3.12	1.11	0.86	0.85
UCB1-TUNED	2.40	6.19	3.69	1.11	0.17
UCB1-NORMAL	20.0	10.52	3.41	2.37	0.19
POKER	5.67	4.55	2.06	1.93	0.47

Table A.19: Number of yes votes with 50 players and 20 desired yes votes

Player / Iteration	**10**	**100**	**1000**	**10 000**	**100 000**
Tsetlin $L_{2N,2}$ $N = 1$	23.07	20.54	20.00	20.00	20.00
BLA Bernoulli	24.33	21.27	20.20	20.20	20.05
BLA Normal unknown σ^2	24.67	21.46	20.25	19.93	20.17
ε_n-GREEDY c =0.05 d =0.10	24.18	21.89	21.35	21.07	21.06
ε_n-GREEDY c =0.15 d =0.10	25.02	20.96	19.77	19.88	19.92
ε_n-GREEDY c =0.30 d =0.10	24.34	21.87	19.44	19.06	19.09
L_{R-I} 0.01	25.59	24.65	24.65	20.92	19.96
L_{R-I} 0.005	24.90	25.24	24.38	20.82	20.10
Pursuit 0.01	23.57	20.09	20.01	20.03	20.03
Pursuit 0.005	24.50	21.76	19.71	19.68	19.69
UCB1-TUNED	19.22	26.13	23.45	20.97	20.15
UCB1-NORMAL	0.00	30.52	22.65	21.55	20.15
POKER	25.67	24.35	19.90	19.75	19.73

Table A.20: Detailed overview of the Goore game with 50 players, where exactly 20 players should vote yes

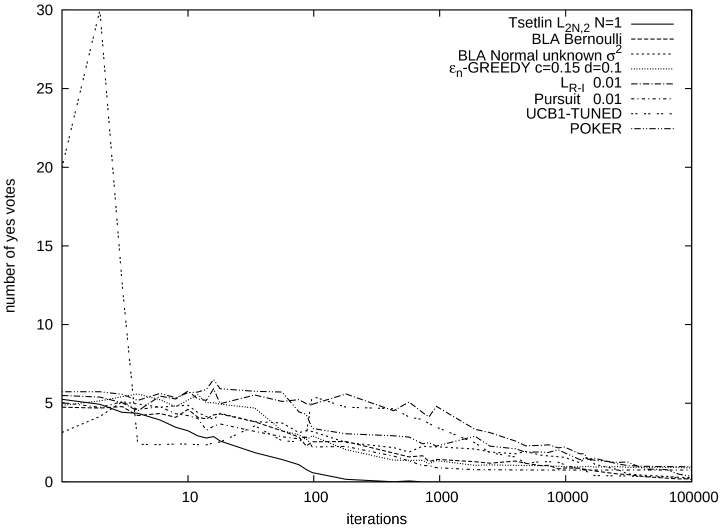


Figure A.23: Development of the distance from the optimal number of yes votes 20, with 50 players

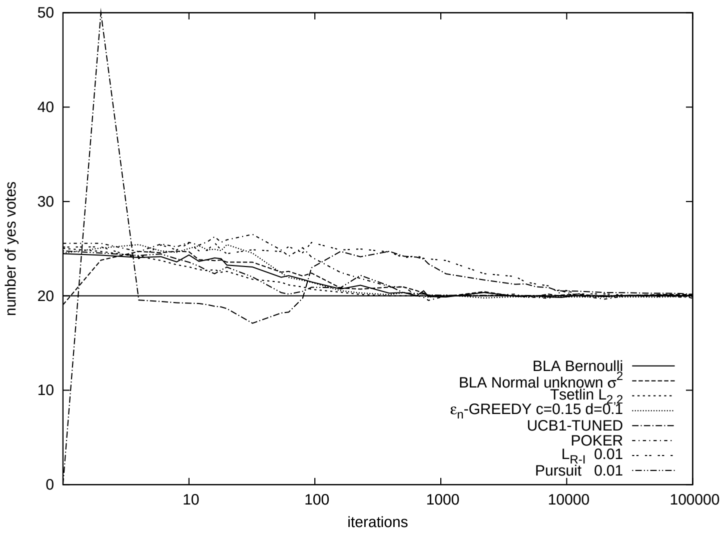


Figure A.24: Development of the number of yes votes with 50 players and 20 as the optimal number of yes votes

A.4.3 Results of experiment configuration 3

Table A.21: Distance from the optimal number of yes votes in the Goore game with 100 players, with 35 desired yes votes

Player / Iteration	**10**	**100**	**1000**	**10 000**	**100 000**
Tsetlin $L_{2N,2}$ $N = 1$	10.43	6.08	0.00	0.00	0.00
BLA Bernoulli	12.88	6.04	1.81	1.20	0.90
BLA Normal unknown σ^2	12.31	6.40	1.83	0.90	0.25
ε_n-GREEDY c =0.05 d =0.10	15.20	6.93	5.22	5.08	4.94
ε_n-GREEDY c =0.15 d =0.10	14.29	7.71	3.95	3.58	3.51
ε_n-GREEDY c =0.30 d =0.10	14.58	10.03	3.57	3.27	3.10
L_{R-I} 0.01	14.66	14.22	8.33	2.56	0.78
L_{R-I} 0.005	15.08	14.52	12.12	3.92	1.39
Pursuit 0.01	14.55	8.91	2.75	2.74	2.74
Pursuit 0.005	14.60	11.75	3.42	3.25	3.25
UCB1-TUNED	4.42	8.57	5.07	1.84	1.51
UCB1-NORMAL	35.00	34.33	15.34	1.35	0.64
POKER	12.74	9.25	2.87	1.97	0.65

Table A.22: Detailed overview of the Goore game with 100 players, where exactly 35 players should vote yes

Players	**10**	**100**	**1000**	**10 000**	**100 000**
Tsetlin $L_{2N,2}$ $N = 1$	45.43	41.08	35.0	35.00	35.00
BLA Bernoulli	47.88	40.92	35.99	35.72	35.74
BLA Normal unknown σ^2	47.29	40.82	35.31	35.48	35.09
ε_n-GREEDY c =0.05 d =0.10	50.2	41.57	39.9	39.8	39.68
ε_n-GREEDY c =0.15 d =0.10	49.29	42.43	38.39	37.88	37.87
ε_n-GREEDY c =0.30 d =0.10	49.58	45.01	36.63	35.73	35.62
L_{R-I} 0.01	49.66	49.22	43.25	35.34	34.78
L_{R-I} 0.005	50.08	49.48	47.06	36.02	35.03
Pursuit 0.01	49.55	43.71	35.31	35.3	35.3
Pursuit 0.005	49.58	46.65	32.18	32.15	32.15
UCB1-TUNED	39.12	43.47	40.05	36.76	36.51
UCB1-NORMAL	0.00	69.33	50.34	36.33	35.62
POKER	47.74	44.23	36.67	35.07	34.45

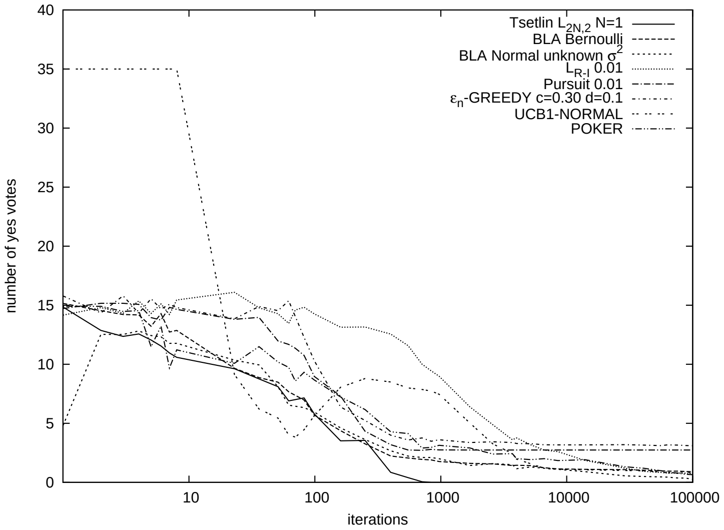


Figure A.25: Development of the distance from the optimal number of yes votes 35, with 100 players

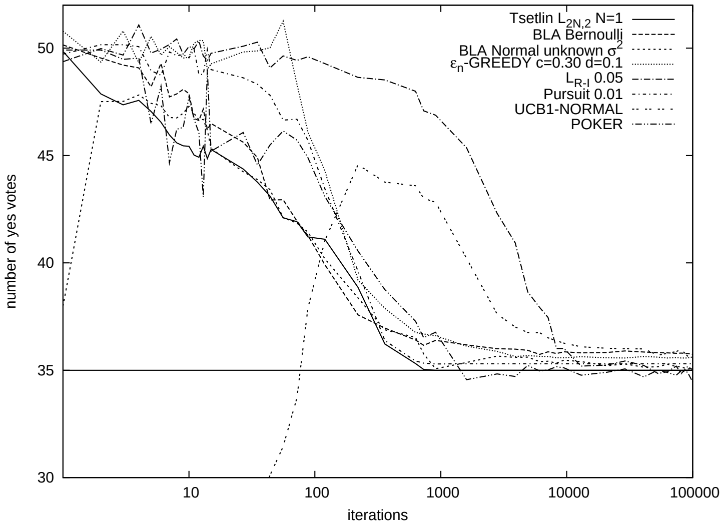


Figure A.26: Development of the number of yes votes with 100 players and 35 as the optimal number of yes votes

A.5 Iterative prisoners' dilemma

		Player B	
		Not confess	Confess
Player A	Not confess	0.8,0.8	0.0,1.0
	Confess	1.0,0.0	0.4,0.4

Figure A.27: Inverted normalized game matrix in prisoners' dilemma

Table A.23: Tournament scores of the iterative prisoners' dilemma

Player / Iteration	**10**	**100**	**1000**	**10 000**	**100 000**
Tit-For-Tat	0.635	0.548	0.491	0.469	0.466
BLA Normal known $\sigma^2 = 1$	0.663	0.542	0.459	0.424	0.420
BLA Normal unknown σ^2	0.585	0.606	0.507	0.463	0.458
ε_n-greedy c =0.05 d =0.1	0.624	0.549	0.463	0.421	0.416
ε_n-greedy c =0.15 d =0.1	0.626	0.502	0.451	0.418	0.414
ε_n-greedy c =0.30 d =0.1	0.624	0.468	0.440	0.417	0.415
L_{R-I} 0.05	0.630	0.505	0.459	0.420	0.414
L_{R-I} 0.01	0.627	0.466	0.419	0.418	0.416
L_{R-I} 0.005	0.629	0.460	0.382	0.411	0.414
Pursuit 0.01	0.630	0.502	0.458	0.421	0.416
Pursuit 0.005	0.629	0.483	0.445	0.420	0.416
ucb1-normal	0.564	0.515	0.459	0.434	0.431
Poker	0.639	0.499	0.376	0.320	0.311
Exp3	0.628	0.459	0.381	0.409	0.413

A.6 Two-player zero-sum game

A.6.1 Pure strategy

$$D1 = \begin{bmatrix} 0.6 & 0.8 \\ 0.35 & 0.65 \end{bmatrix}$$

Figure A.28: Game matrix D1

Table A.24: Optimal actions selection probability for player A and B in D1 (A,B)

Player / Iteration	**10**	**100**	**1000**	**10 000**	**100 000**
BLA Bernoulli	0.69, 0.68	0.93, 0.91	0.99 ,0.99	1.00, 1.00	1.00, 1.00
BLA normal unkown σ^2	0.52, 0.63	0.91, 0.96	0.99, 0.99	1.00, 1.00	1.00, 1.00
ε_n-GREEDY c =0.05 d =0.10	0.50, 0.49	0.93, 0.93	0.99, 0.99	1.00, 1.00	1.00, 1.00
ε_n-GREEDY c =0.15 d =0.10	0.51, 0.49	0.84, 0.86	0.98, 0.99	1.00, 1.00	1.00, 1.00
ε_n-GREEDY c =0.30 d =0.10	0.51, 0.49	0.68, 0.71	0.97, 0.97	1.00, 0.99	1.00, 1.00
$L_{R-\epsilon P}$ $\alpha = 0.002, \beta = 0.00001$	0.48, 0.48	0.51, 0.55	0.59, 0.63	0.98, 0.96	0.99, 0.98
L_{R-I} 0.05	0.51, 0.53	0.71, 0.75	1.00, 1.00	1.00, 1.00	1.00, 1.00
L_{R-I} 0.01	0.51, 0.52	0.57, 0.58	0.91, 0.90	1.00, 1.00	1.00, 1.00
L_{R-I} 0.005	0.51, 0.51	0.53, 0.51	0.74, 0.78	1.00, 1.00	1.00, 1.00
Pursuit 0.01	0.51, 0.52	0.71, 0.63	1.00, 1.00	1.00, 1.00	1.00, 1.00
Pursuit 0.005	0.50, 0.49	0.60, 0.60	0.98, 0.92	1.00, 1.00	1.00, 1.00
UCB1-TUNED	0.73, 0.77	0.94, 0.93	0.96, 0.98	1.00, 1.00	1.00, 1.00
POKER	0.63, 0.74	0.82, 0.90	0.95, 0.94	0.97, 0.98	0.97, 0.98

$$D2 = \begin{bmatrix} 0.55 & 0.525 \\ 0.45 & 0.475 \end{bmatrix}$$

Figure A.29: Game matrix D2

Table A.25: Optimal actions selection probability for player A and B in D2 (A,B)

Player / Iteration	**10**	**100**	**1000**	**10 000**	**100 000**
BLA Bernoulli	0.56, 0.49	0.70, 0.55	0.91, 0.63	0.99, 0.92	1.00, 0.99
BLA Normal unknown σ^2	0.61, 0.55	0.81, 0.66	0.96, 0.79	0.99, 0.93	1.00, 0.99
ε_n-GREEDYc =0.05 d =0.10	0.51, 0.54	0.72, 0.54	0.85, 0.60	0.89, 0.63	0.92, 0.67
ε_n-GREEDY c =0.15 d =0.10	0.50, 0.52	0.70, 0.50	0.91, 0.57	0.97, 0.69	0.98, 0.76
ε_n-GREEDY c =0.30 d =0.10	0.50, 0.49	0.61, 0.46	0.93, 0.60	0.99, 0.77	1.00, 0.85
$L_{R-\epsilon P}$ $\alpha = 0.002, \beta = 0.00001$	0.51, 0.50	0.51, 0.53	0.53, 0.50	0.83, 0.55	0.96, 0.90
L_{R-I} 0.05	0.50, 0.50	0.59, 0.53	0.89, 0.58	0.94, 0.66	0.94, 0.66
L_{R-I} 0.01	0.54, 0.52	0.52, 0.49	0.67, 0.49	0.99, 0.83	1.00, 0.98
L_{R-I} 0.005	0.50, 0.49	0.51, 0.50	0.57, 0.51	0.97, 0.68	1.00, 1.00
Pursuit 0.01	0.52, 0.51	0.59, 0.53	0.95, 0.64	0.99, 0.81	0.99, 0.83
Pursuit 0.005	0.50, 0.51	0.54, 0.50	0.91, 0.58	1.00, 0.88	1.00, 0.92
UCB1-TUNED	0.57, 0.48	0.72, 0.54	0.93, 0.65	0.98, 0.92	1.00, 0.99
POKER	0.60, 0.47	0.65, 0.53	0.79, 0.55	0.85, 0.66	0.85, 0.69

A.6.2 Mixed strategy

$$D3 = \begin{bmatrix} 0.8 & 0.2 \\ 0.4 & 0.6 \end{bmatrix}$$

Figure A.30: Game matrix D3

Table A.26: Two-player zero-sum mixed strategy tournament score with matrix D3

Player / Iteration	**10**	**100**	**1000**	**10 000**	**100 000**
BLA Bernoulli	0.42	4.71	54.1	864	11782
BLA Normal unknown σ^2	0.00	0.65	38.6	841	11523
ε_n-GREEDY c =0.05 d =0.10	-0.07	3.82	6.3	321	7436
ε_n-GREEDY c =0.15 d =0.10	-0.12	2.04	14.5	478	9204
ε_n-GREEDY c =0.30 d =0.10	-0.02	-0.47	24.4	543	9746
$L_{R-\epsilon P}$ $\alpha = 0.002, \beta = 0.00001$	-0.12	-2.49	-26.9	568	10449
L_{R-I} 0.05	0.05	2.04	16.4	-1449	-18068
L_{R-I} 0.01	-0.08	-1.08	32.6	246	-11482
L_{R-I} 0.005	-0.10	-2.17	8.3	579	-3582
Pursuit 0.01	-0.03	-0.24	13.8	-345	-1693
Pursuit 0.005	-0.03	-1.11	16.8	-166	-1355
UCB1-TUNED	0.34	6.05	73.2	978	11751
POKER	-0.21	-11.74	-272.2	-3457	-35712

Table A.27: Two-player zero-sum mixed strategy average distance from optimal reward

Player / Iteration	**10**	**100**	**1000**	**10 000**	**100 000**
BLA Bernoulli	2.79	9.6	37	151	714
BLA Normal unknown σ^2	3.65	19.7	52	145	420
ε_n-GREEDY c =0.05 d =0.10	2.69	11.2	99	964	9881
ε_n-GREEDY c =0.15 d =0.10	2.71	8.9	68	594	5887
ε_n-GREEDY c =0.30 d =0.10	2.73	10.0	56	482	4693
$L_{R-\epsilon P}$ $\alpha = 0.002, \beta = 0.00001$	2.71	7.9	30	90	267
L_{R-I} 0.05	2.80	9.4	82	2127	22849
L_{R-I} 0.01	2.71	8.3	34	476	19870
L_{R-I} 0.005	2.69	8.2	40	129	12632
Pursuit 0.01	2.72	8.7	62	1475	17658
Pursuit 0.005	2.69	8.0	41	1276	17080
UCB1-TUNED	3.13	15.4	67	524	4806
POKER	2.99	8.7	143	1535	15670

Table A.28: Two-player zero-sum mixed strategy action probability distribution A

Player / Iteration	**10**	**100**	**1000**	**10 000**	**100 000**
BLA Bernoulli	0.42, 0.58	0.20, 0.80	0.26, 0.74	0.26, 0.74	0.27, 0.73
BLA Normal unknown σ^2	0.55, 0.45	0.48, 0.52	0.26, 0.74	0.25, 0.75	0.24, 0.76
ε_n-GREEDYc =0.05 d =0.10	0.48, 0.52	0.25, 0.85	0.17, 0.83	0.19, 0.81	0.17, 0.83
ε_n-GREEDY c =0.15 d =0.10	0.51, 0.49	0.27, 0.73	0.21, 0.79	0.23, 0.77	0.20, 0.80
ε_n-GREEDY c =0.30 d =0.10	0.49, 0.51	0.44, 0.56	0.15, 0.85	0.24, 0.76	0.22, 0.78
$L_{R-\epsilon P}$ $\alpha = 0.002, \beta = 0.00001$	0.50, 0.50	0.48, 0.52	0.48, 0.52	0.11, 0.89	0.27, 0.73
L_{R-I} 0.05	0.49, 0.51	0.38, 0.62	0.09, 0.91	0.07, 0.93	0.07, 0.93
L_{R-I} 0.01	0.48, 0.52	0.48, 0.52	0.21, 0.79	0.15, 0.85	0.06, 0.94
L_{R-I} 0.005	0.50, 0.50	0.49, 0.51	0.40, 0.60	0.31, 0.69	0.04, 0.96
Pursuit 0.01	0.49, 0.51	0.44, 0.56	0.04, 0.96	0.03, 0.97	0.01, 0.99
Pursuit 0.005	0.48, 0.52	0.49, 0.51	0.09, 0.91	0.06, 0.94	0.02, 0.98
UCB1-TUNED	0.42, 0.58	0.23, 0.77	0.22, 0.78	0.21, 0.79	0.25, 0.75
POKER	0.48, 0.52	0.18, 0.82	0.11, 0.89	0.12, 0.88	0.08, 0.92

Table A.29: Two-player zero-sum mixed strategy action probability distribution B

Player / Iteration	**10**	**100**	**1000**	**10 000**	**100 000**
BLA Bernoulli	0.34, 0.66	0.38, 0.62	0.44, 0.56	0.50, 0.50	0.46, 0.54
BLA Normal unknown σ^2	0.48, 0.52	0.25, 0.75	0.45, 0.55	0.49, 0.51	0.51, 0.49
ε_n-GREEDYc =0.05 d =0.10	0.50, 0.50	0.25, 0.75	0.51, 0.49	0.49, 0.51	0.48, 0.52
ε_n-GREEDY c =0.15 d =0.10	0.49, 0.51	0.20, 0.80	0.54, 0.46	0.52, 0.48	0.50, 0.50
ε_n-GREEDY c =0.30 d =0.10	0.51, 0.49	0.28, 0.72	0.40, 0.60	0.49, 0.51	0.49, 0.51
$L_{R-\epsilon P}$ $\alpha = 0.002, \beta = 0.00001$	0.51, 0.49	0.51, 0.49	0.40, 0.60	0.49, 0.51	0.46, 0.54
L_{R-I} 0.05	0.47, 0.53	0.31, 0.69	0.60, 0.40	0.66, 0.34	0.66, 0.34
L_{R-I} 0.01	0.52, 0.48	0.47, 0.53	0.24, 0.76	0.64, 0.36	0.84, 0.16
L_{R-I} 0.005	0.48, 0.52	0.47, 0.53	0.32, 0.68	0.32, 0.68	0.87, 0.13
Pursuit 0.01	0.51, 0.49	0.31, 0.69	0.42, 0.58	0.67, 0.33	0.75, 0.25
Pursuit 0.005	0.48, 0.52	0.28, 0.62	0.20, 0.80	0.78, 0.22	0.83, 0.17
UCB1-TUNED	0.28, 0.72	0.40, 0.60	0.41, 0.59	0.44, 0.56	0.44, 0.56
POKER	0.37, 0.63	0.32, 0.68	0.52, 0.48	0.40, 0.60	0.45, 0.55

Appendix B

Paper submitted to European Conference on Machine Learning PKDD 09

Results of some of the experiments conducted in this thesis have been included in the paper *A Generic Solution to Multi-Armed Bernoulli Bandit Problems Based on Random Sampling from Sibling Conjugate Priors* which has been submitted to the European Conference on Machine Learning PKDD 2009.

A Generic Solution to Multi-Armed Bernoulli Bandit Problems Based on Random Sampling from Sibling Conjugate Priors[⋆]

Thomas Norheim[1], Terje Brådland[1],
Ole-Christoffer Granmo[1], and B. John Oommen[2,1]

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

Abstract. The multi-armed Bernoulli bandit (MABB) problem is a classical optimization problem where an agent sequentially pulls one of multiple arms attached to a gambling machine, with each pull resulting in a random reward. The reward distributions are unknown, and thus, one must balance between exploiting existing knowledge about the arms, and obtaining new information.

Although computationally intractable in many cases, Bayesian methods provide a standard for optimal decision making. This paper proposes a novel MABB solution scheme that is inherently Bayesian in nature, yet avoids computational intractability by relying simply on updating the hyper parameters of sibling conjugate distributions, and on random sampling from these posteriors. Although generic in principle, we here study an algorithm for Bernoulli distributed rewards.

Extensive experiments demonstrate that our scheme outperforms recently proposed bandit playing algorithms. We thus believe that our methodology opens avenues for obtaining improved novel solutions.

Keywords: Bandit Problems, Conjugate Priors, Sampling, Bayesian Learning

1 Introduction

The conflict between exploration and exploitation is a well-known problem in Reinforcement Learning (RL), and other areas of artificial intelligence. The multi-armed bandit problem captures the essence of this conflict, and has thus occupied researchers for over fifty years [1]. This paper introduces a new family of *Bayesian* techniques for solving the classical Multi-Armed Bernoulli Bandit (MABB) problem, and reports empirical results that demonstrate its advantages

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

[⋆] The fourth author was partially supported by NSERC, the Natural Sciences and Engineering Research Council of Canada.

over established top performers from two distinct research areas, namely, Learning Automata (LA) and Bandit Playing Algorithms (see [2, 3] and [4] for an overview of current algorithms).

1.1 The Multi-Armed Bernoulli Bandit (MABB) Problem

The MABB problem is a classical optimization problem that explores the trade off between exploitation and exploration in reinforcement learning. The problem consists of an agent that sequentially pulls one of multiple arms attached to a gambling machine, with each pull resulting either in a *reward* or a *penalty*[3]. The sequence of rewards/penalties obtained from each arm i forms a Bernoulli process with an *unknown* reward probability r_i, and a penalty probability $1-r_i$. This leaves the agent with the following dilemma: Should the arm that so far seems to provide the highest chance of reward be pulled once more, or should the inferior arm be pulled in order to learn more about *its* reward probability? Sticking prematurely with the arm that is presently considered to be the best one, may lead to not discovering which arm is truly optimal. On the other hand, lingering with the inferior arm unnecessarily, postpones the harvest that can be obtained from the optimal arm.

With the above in mind, we intend to evaluate an agent's arm selection strategy in terms of the so-called *Regret*, and in terms of the *probability of selecting the optimal arm*[4]. The *Regret* measure is non-trivial, and in all brevity, can be perceived to be *the difference between the sum of rewards expected after N successive arm pulls, and what would have been obtained by only pulling the optimal arm*. To clarify issues, assume that a *reward* amounts to the value (utility) of unity (i.e., 1), and that a *penalty* possesses the value 0. We then observe that the expected returns for pulling Arm i is r_i. Thus, if the optimal arm is Arm 1, the *Regret* after N plays would become:

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

with $\hat{r}_n$ being the expected reward at Arm pull i, given the agent's arm-selection strategy. In other words, as will be clear in the following, we consider the case where rewards are *undiscounted*, as discussed in [5].

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

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

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

1.2 Applications

Solution schemes for bandit problems have formed the basis for dealing with a number of applications. For instance, a Ucb-Tuned scheme [5] is used for move exploration in *MoGo*, a top-level Computer-Go program on 9 × 9 *Go* boards [6]. Furthermore, the so-called Ubc1 scheme has formed the basis for guiding Monte-Carlo planning, and improving planning efficiency significantly in several domains [7].

The applications of LA are many – in the interest of brevity, we list a few more-recent ones. LA have been used to allocate polling resources optimally in web monitoring, and for allocating limited sampling resources in binomial estimation problems [8]. LA have also been applied for solving NP-complete SAT problems [9]. Furthermore, in [10], LA have been used to optimize throughput in MPLS traffic engineering [10]. Note that *Regret* minimizing algorithms also have found applications in network routing [11].

1.3 Contributions and Paper Organization

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

2 Related Work

The MABB problem has been studied in a disparate range of research fields. From a machine learning point of view, Sutton et. al placed an emphasis on computationally efficient solution techniques that are suitable for RL. While there are algorithms for computing the optimal Bayes strategy to balance exploration and exploitation, these are computationally intractable for the general case [12], mainly because of the magnitude of the state space associated with typical bandit problems.

[5] A comparison of Bandit Playing Algorithms can be found in [4], with the Ucb-Tuned distinguishing itself in [5].

From a broader point of view, one can distinguish two distinct fields that address bandit like problems, namely, the field of Learning Automata and the field of Bandit Playing Algorithms. A myriad of approaches have been proposed within these two fields, and we refer the reader to [2,3] and [4] for an overview and comparison of schemes. Although these fields are quite related, research spanning them both is surprisingly sparse. In this paper, however, we will include the established top performers from both of the two fields. These are reviewed here in some detail in order to cast light on the distinguishing properties of BLA, both from an LA perspective and from the perspective of Bandit Playing Algorithms.

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

LA have been used to model biological systems [2,3,13] and have attracted considerable interest in the last decade because they can learn the optimal action when operating in (or interacting with) unknown stochastic environments. Furthermore, they combine rapid and accurate convergence with low computational complexity. For the sake of conceptual simplicity, note that we in this subsection, we assume that we are dealing with a bandit associated with two arms.

More notable approaches include the family of linear updating schemes, with the Linear Reward-Inaction (L_{R-I}) automaton being designed for stationary environments [3]. In short, the L_{R-I} maintains an Arm probability selection vector $\bar{p} = [p_1, p_2]$, with $p_2 = 1 - p_1$. The question of which Arm is to be pulled is decided randomly by sampling from $\bar{p}$. Initially, $\bar{p}$ is uniform. The following linear updating rules summarize how rewards and penalties affect $\bar{p}$ with p_1' and $1 - p_1'$ being the resulting updated Arm selection probabilities:

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

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

A distinguishing feature of the L_{R-I} scheme, and indeed the best LA within the field of LA, is its ϵ-optimality [3]: *By a suitable choice of some parameter of the LA, the expected reward probability obtained from each arm pull can be made arbitrarily close to the optimal reward probability, as the number of arm pulls tends to infinity.*

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

Discretized and Continuous variants of the Pursuit scheme has been proposed [14–16], with slightly superior performances. But, in general, any Pursuit scheme can be seen to be representative of this entire family.

2.2 The ϵ-Greedy and ϵ_n-Greedy Policies

The ϵ-greedy rule is a well-known strategy for the bandit problem [12]. In short, the Arm with the presently highest average reward is pulled with probability $1-\epsilon$, while a randomly chosen Arm is pulled with probability ϵ. In other words, the balancing of exploration and exploitation is controlled by the ϵ-parameter. Note that the ϵ-greedy strategy persistently explores the available Arms with constant effort, which clearly is sub-optimal for the MABB problem (unless the reward probabilities are changing with time).

As a remedy for the above problem, ϵ can be slowly decreased, leading to the ϵ_n-greedy strategy described in [5]. The purpose is to gradually shift focus from exploration to exploitation. The latter work focuses on algorithms that minimizes the so-called *Regret* formally described above. It turns out that the ϵ_n-greedy strategy asymptotically provides a *logarithmically* increasing *Regret*. Indeed, it has been proved that logarithmically increasing *Regret* is the best possible [5] strategy.

2.3 Confidence Interval Based Algorithms

A promising line of thought is the interval estimation methods, where a confidence interval for the reward probability of each Arm is estimated, and an "optimistic reward probability estimate" is identified for each Arm. The Arm with the most optimistic reward probability estimate is then greedily selected [4, 17].

In [5], several confidence interval based algorithms are analysed. These algorithms also provide logarithmically increasing *Regret*, with *U*CB-*T*UNED – a variant of the well-known UBC1 algorithm — outperforming both UBC1,UCB2, as well as the ϵ_n-greedy strategy. In brief, in UCB-TUNED, the following optimistic estimates are used for each Arm i:

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

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

2.4 Bayesian Approaches

The MABB has also been extensively analysed from a Bayesian perspective. For instance, in [18] the MABB is modelled as a partially observable Markov decision process. The authors of the paper demonstrated that the difference in rewards between the actions of stopping learning and acquiring full information goes to zero as the number of arm pulls increases indefinitely.

The use of Bayesian methods in inference problems of this nature has also been reported. The authors of [1] have proposed the use of such a philosophy in their probability matching algorithms. By using conjugate priors, they have resorted to a Bayesian analysis to obtain a closed form expression for the probability that each arm is optimal given the prior observed rewards/penalties. Informally, the method proposes a policy which consists of calculating the probability of each arm being optimal before an arm pull, and then randomly selecting the arm to be pulled next using these probabilities. Unfortunately, for the case of two arms in which the rewards Bernoulli-distributed, the computation time becomes unbounded, and it increases with the number of arm pulls. Furthermore, it turns out that for the multi-armed case, the resulting integrations have no analytical solution. Similar problems surface when the probability of each arm being optimal is computed for the case when the rewards are normally distributed[6]. The authors of [19] take advantage of a Bayesian strategy in a related domain, i.e., in Q-learning. They show that for normally distributed rewards, in which the parameters have a prior normal-gamma distribution, the posteriors also have a normal-gamma distribution, rendering the computation efficient. They then integrate this into a framework for Bayesian Q-learning by maintaining and propagating probability distributions over the Q-values, and suggest that a non-approximate solution can be obtained by means of random sampling for the normal distribution case. It would be interesting to investigate the applicability of these results for the MABB.

[6] It turns out that in the latter case, the approximate Bayesian solution reported by [1] is computationally efficient

2.5 Boltzmann Exploration and POKER

One class of algorithms for solving MABB problems is based on so-called Boltzmann exploration. In brief, an arm i is pulled with probability $p_k = \frac{e^{\hat{\mu}_i/\tau}}{\sum_{j=1}^{n}}$ where $\hat{\mu}_i$ is the sample mean and τ is defined as the temperature of the exploration. A high temperature τ leads to increased exploration since each arm will have approximately the same probability of being pulled. A low temperature, on the other hand, leads to arms being pulled proportionally to the size of the rewards that can be expected. Typically, the temperature is set to be high initially, and then is gradually reduced in order to shift from exploration to exploitation. Note that the Exp3 scheme, proposed and detailed in [20], is a more complicated variant of Boltzmann exploration. In brief, this scheme calculates the arm selection probabilities p_k based on dividing the rewards obtained with the probability of pulling the arm that produced the rewards [4].

The "Price of Knowledge and Estimated Reward" (Poker) algorithm proposed in [4] attempts to combine the following three principles: (1) Reducing uncertainty about the arm reward probabilities should grant a bonus to stimulate exploration; (2) Information obtained from pulling arms should be used to estimate the properties of arms that have not yet been pulled; and (3) Knowledge about the number of rounds that remains (the horizon) should be used to plan the exploitation and exploration of arms. We refer the reader to [4] for the specific algorithm that incorporates these three principles. Note, however, that principle (2) and (3) assumes knowledge about the MABB that is not universally available. E.g., the reward probabilities of the arms may in many cases be independent and accordingly, information about the reward probability of one arm, does not necessarily provide information about the reward probabilities of other arms. Furthermore, in many cases, one must assume that the MABB are to be played for an unlimited number of rounds, with undiscounted rewards, which leaves the concept of a horizon inappropriate.

3 The Bayesian Learning Automaton (BLA)

Bayesian reasoning is a probabilistic approach to inference which is of significant importance in machine learning because it allows quantitative weighting of evidence supporting alternative hypotheses, with the purpose of allowing optimal decisions to be made. Furthermore, it provides a framework for analyzing learning algorithms [21].

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

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

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

and the corresponding cumulative distribution function:

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

Essentially, the BLA uses the *Beta* distribution for two purposes. First of all, the *Beta* distribution is used to provide a *Bayesian estimate* of the reward probabilities associated with each of the available bandit Arms - the latter being valid by virtue of the Conjugate Prior [22] nature of the Binomial parameter. Secondly, a novel feature of the BLA is that it uses the *Beta* distribution as the basis for an *Order-of-Statistics*-based *randomized* Arm selection mechanism.

The following algorithm contains the essence of the BLA approach.

```
Algorithm: BLA-MABB Input: Number of bandit Arms $r$.
Initialization: $\alpha_1^1 = \beta_1^1 = \alpha_2^1 = \beta_2^1 = \ldots = \alpha_r^1 = \beta_r^1 = 1$.
Method:
For $N = 1, 2, \ldots$ Do
  1. For each Arm $j \in \{1, \ldots, r\}$, draw a value $x_j$ randomly from the associated
     Beta distribution $f(x_j; \alpha_j^N, \beta_j^N)$ with the parameters $\alpha_j^N, \beta_j^N$.
  2. Pull the Arm $i$ whose drawn value $x_i$ is the largest one of the randomly
     drawn values:
         $i = \underset{j \in \{1,\ldots,r\}}{\text{argmax}} \ x_j.$
  3. Receive either Reward or Penalty as a result of pulling Arm $i$, and update
     parameters as follows:
     – Upon Reward: $\alpha_i^{N+1} = \alpha_i^N + 1$; $\beta_i^{N+1} = \beta_i^N$; and $\alpha_j^{N+1} = \alpha_j^N$, $\beta_j^{N+1} =$
       $\beta_j^N$ for $j \neq i$.
     – Upon Penalty: $\alpha_i^{N+1} = \alpha_i^N$; $b_i^{N+1} = b_i^N + 1$; and $\alpha_j^{N+1} = \alpha_j^N$, $\beta_j^{N+1} =$
       $\beta_j^N$ for $j \neq i$.
End Algorithm: BLA-MABB
```

As seen from the above BLA algorithm, N is a discrete time index and the parameters $\phi^N = \langle \alpha_1^N, \beta_1^N, \alpha_2^N, \beta_2^N, \ldots, \alpha_r^N, \beta_r^N \rangle$ form an infinite discrete $2 \times r$-dimensional state space, which we will denote with Φ. Within Φ the BLA navigates by iteratively adding 1 to either α_1^N, β_1^N, α_2^N, $\beta_2^N, \ldots, \alpha_r^N$ or β_r^N.

Since the state space of BLA is both discrete and infinite, BLA is quite different from both the *Variable Structure*- and the *Fixed Structure* LA families [2], traditionally referred to as *Learning Automata*. In all brevity, the novel aspects of the BLA are listed below:

1. In traditional LA, the action chosen (i.e, Arm pulled) is based on the so-called action probability vector. The BLA does not maintain such a vector, but chooses the arm based on the *distribution* of the components of the *Estimate* vector.
2. The second difference is that we have not chosen the arm based on the *a posteriori* distribution of the estimate. Rather, it has been chosen based on the magnitude of a *random sample* drawn from the *a posteriori* distribution, and thus it is more appropriate to state that the arm is chosen based on the *order of statistics* of instances of these variables[7].
3. The third significant aspect is that we can now consider the design of Pursuit LA in which the estimate used is not of the ML family, but on a Bayesian updating scheme. As far as we know, such a mechanism is also unreported in the literature.
4. The final significant aspect is that we can now devise solutions to the Multi-Armed Bandit problem even for cases when the Reward/Penalty distribution is not Bernoulli distributed. Indeed, we advocate the use of a Bayesian methodology with the appropriate Conjugate Prior [22].

In the interest of notational simplicity, let *Arm* 1 be the Arm under investigation. Then, for any parameter configuration $\phi^N \in \Phi$ we can state, using a generic notation[8], that the probability of selecting *Arm* 1 is equal to the probability $P(X_1^N > X_2^N \wedge X_1^N > X_3^N \wedge \dots \wedge X_1^N > X_r^N | \phi^N)$ — the probability that a randomly drawn value $x_1 \in X_1^N$ is greater than all of the other randomly drawn values $x_j \in X_j^N, j \neq i$, at time step N, when the associated stochastic variables $X_1^N, X_2^N, \dots, X_r^N$ are *Beta* distributed, with parameters $\alpha_1^N, \beta_1^N, \alpha_2^N, \beta_2^N, \dots, \alpha_r^N, \beta_r^N$ respectively. In the following, we will let $p_1^{\phi^N}$ denote this latter probability.

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

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

4 Empirical Results

In this section we evaluate the BLA by comparing it with the best performing algorithms from [4, 5], as well as the L_{R-I} and Pursuit schemes, which can be

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

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

seen as established top performers in the LA field. Based on our comparison with these "reference" algorithms, it should be quite straightforward to also relate the BLA performance results to the performance of other similar algorithms.

For the sake of fairness, we base our comparison on the experimental setup for the MABB found in [5]. Although we have conducted numerous experiments using various reward distributions, we here report, for the sake of brevity, results for the experiment configurations enumerated in Table 1.

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

For these experiment configurations, an ensemble of 1000 independent replications with different random number streams was performed to minimize the variance of the reported results[9]. In each replication, 100 000 arm pulls were conducted in order to examine both the short term and the limiting performance of the evaluated algorithms.

Note that real-world instantiations of the bandit problem, such as Resource Allocation in Web Polling [8], may exhibit any reward probability in the interval $[0, 1]$. Hence, a solution scheme designed to tackle bandit problems in general, should perform well across the complete space of reward probabilities.

For all of the experiment configurations in the table, we compared the performance of both the BLA, ϵ_n-GREEDY, L_{R-I}, Pursuit, UCB-1, UCB-TUNED, EXP3, POKER, and INTESTIM. In Table 2 we report the average probability of pulling the best arm over 100 000 arm pulls. By taking the average probability over all the arm selections, a low learning pace is penalized, however, long term performance is emphasized. As seen in the table, BLA provides either equal or better performance than any of the compared algorithms, expect for experiment

[9] Some of the tested algorithms were unstable for certain reward distributions, producing a high variance compared to the mean regret. This confirms the observations from [23] where the high variance of e.g. UCB-TUNED was first reported. Thus, in our experience 100 replications were too few to unveil the "true" performance of these algorithms.

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

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

configuration 6 where Ucb-Tuned provides slightly better performance than BLA. Also note that the ϵ_n-greedy algorithm is given the difference between the best arm and the second best arm, thus giving it an unfair advantage.

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

Algorithm/Config.	**1**	**2**	**3**	**4**	**5**	**6**
BLA Bernoulli	1.000	0.999	0.997	0.998	0.988	0.975
ϵ_n − greedy c =0.05 [1]	0.981	0.992	0.965	0.996	0.961	0.893
ϵ_n − greedy c =0.15 [1]	1.000	0.999	0.991	0.990	0.988	0.957
ϵ_n − greedy c =0.30 [1]	1.000	0.997	0.997	0.982	0.981	0.977
L_{R-I} 0.05	0.999	0.918	0.985	0.832	0.378	0.526
L_{R-I} 0.01	0.998	0.993	0.993	0.992	0.885	0.958
L_{R-I} 0.005	0.995	0.986	0.986	0.984	0.940	0.951
Pursuit 0.05	1.000	0.970	0.932	0.912	0.699	0.608
Pursuit 0.01	0.999	0.998	0.998	0.998	0.875	0.848
Pursuit 0.005	0.999	0.999	0.998	0.997	0.960	0.924
Ucb1	0.998	0.982	0.983	0.979	0.848	0.848
Ucb-tuned	1.000	0.997	0.997	0.997	0.977	0.978
Exp3 $\gamma = 0.01$	0.990	0.978	0.980	0.913	0.736	0.749
Poker	0.995	0.991	0.876	0.982	0.916	0.812
IntEstim 0.01	0.961	0.949	0.796	0.920	0.905	0.577

[1] Parameter d is set to be the difference in reward probability between the best arm and the second best arm

Both learning accuracy and learning speed governs the performance of bandit playing algorithms in practice. Table 3 reports the average probability of selecting the best arms after 10, 100, 1000, 10 000, and 100 000 arm pulls for experiment configuration 5. As seen from the table, IntEstim provides the best performance after 10 arm pulls, being slightly better than BLA. After 100 arm pulls, however, BLA provides the best performance. Then, after 1000 arm pulls, one of the parameter configurations of ϵ_n-greedy as well as the Pursuit scheme provide slightly better performance than BLA, with BLA being clearly superior after 10 000 and 100 000 arm pulls.

We now consider the *Regret* of the algorithms. *Regret* offers the advantage that it does not overly emphasize the importance of pulling the best arm. Indeed, pulling one of the non-optimal arms will not necessarily affect the overall amount of rewards obtained in a significant manner if for instance the reward probability of the non-optimal arm is relatively close to the optimal reward probability. For *Regret* it turns out that the performance characteristics of the algorithms are mainly decided by the reward distributions, and not by the number of arms. Thus, we will now consider the configurations in pairs. Fig. 1 contains the comparison on experiment configuration 1 and 4, with the best arm having reward probability 0.9 and the inferior arms having reward probability 0.6. The plots show the accumulation of regret with the number of arm pulls. Because of the logarithmically scaled x- and y-axes, it is clear from the plots that both BLA and Ucb-Tuned attain a logarithmically growing regret. Moreover, the performance of BLA is significantly better than that of the other algorithms, with the Pursuit scheme catching up from the final 10 000 to 100 000 rounds. Note that if the learning speed of the Pursuit scheme is increased to match that of BLA, the accuracy of the Pursuit schemes becomes significantly lower than that of BLA. Surprisingly, both of the LA schemes converge to constant regret. This can be explained by their ϵ-optimality and the relatively low learning speed parameter used ($a = 0.01$). In brief, the LA converged to only selecting the optimal arm in all of the 1000 replications.

For experiment configuration 5, however, it turns out that the applied learning accuracy of the LA is too low to always converge to only selecting the optimal arm ($a = 0.005$). In some of the replications, the LA also converges to selecting the inferior arm only, and as seen in Fig. 2, this leads to linearly growing regret. Note that the LA can achieve constant regret in this latter experiment too, by increasing learning accuracy. However, this reduces learning speed, which for the present setting already is worse than that of BLA and Ucb-Tuned. As also seen in the plots, the BLA continues to provide the best performance.

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

From the above results, we conclude that BLA is the superior choice for MABB problems in general, providing significantly better performance in most of the experiment configurations. Only in two of the experiment configurations does it provide *slightly* lower performance than the second best algorithm for

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

Algorithm/#Arm Pulls	**10**	**100**	**1000**	**10000**	**100000**
BLA Bernoulli	0.112	0.197	0.549	0.916	0.988
ϵ_n − greedy c =0.05 d =0.10	0.101	0.124	0.630	0.898	0.961
ϵ_n − greedy c =0.15 d =0.10	0.105	0.100	0.511	0.911	0.988
ϵ_n − greedy c =0.30 d =0.10	0.099	0.099	0.359	0.872	0.981
L_{R-I} 0.05	0.103	0.119	0.273	0.368	0.378
L_{R-I} 0.01	0.104	0.105	0.156	0.672	0.885
L_{R-I} 0.005	0.102	0.102	0.126	0.518	0.940
Pursuit 0.05	0.100	0.157	0.567	0.682	0.699
Pursuit 0.01	0.098	0.116	0.550	0.840	0.875
Pursuit 0.005	0.101	0.108	0.488	0.910	0.960
ucb-1	0.100	0.119	0.166	0.406	0.848
ucb1-tuned	0.100	0.164	0.425	0.841	0.977
Exp3 $\gamma = 0.01$	0.097	0.099	0.104	0.156	0.736
Poker	0.105	0.180	0.444	0.751	0.916
IntEstim 0.01	0.126	0.194	0.519	0.857	0.905

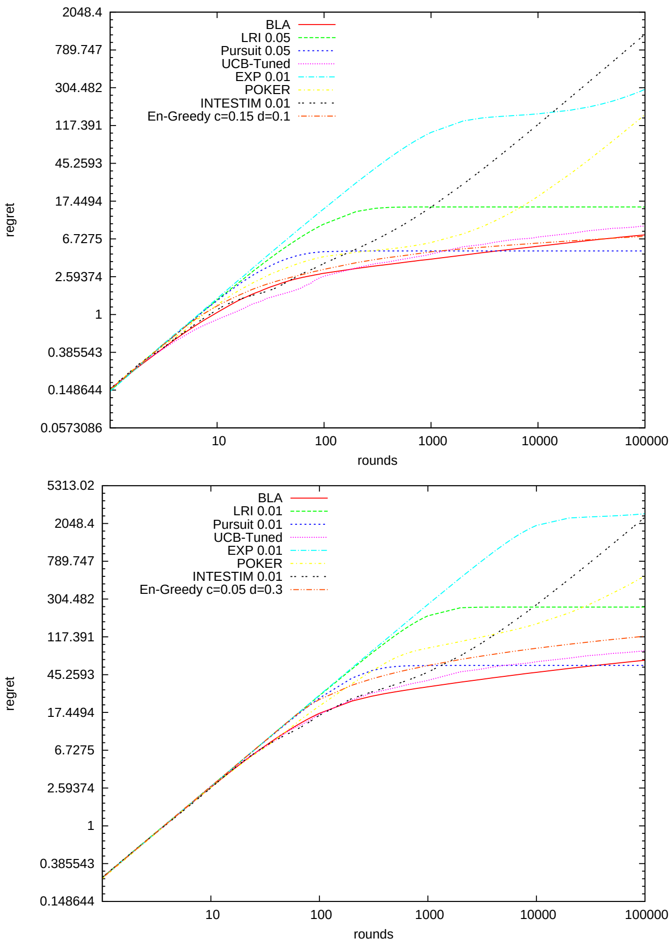


Fig. 1. Regret for experiment config. 1 (top) and config. 4 (bottom)

those configurations. Finally, BLA does not rely on fine-tuning some learning parameter to achieve this performance.

5 Conclusion and Further Work

In this paper we presented the Bayesian Learning Automaton (BLA) for tackling the classical MABB problem. In contrast to previous LA and regret minimizing approaches, BLA is inherently Bayesian in nature. Still, it relies simply on counting of rewards/penalties and random sampling from a set of sibling beta distributions. Thus, to the best of our knowledge, BLA is the first MABB algo-

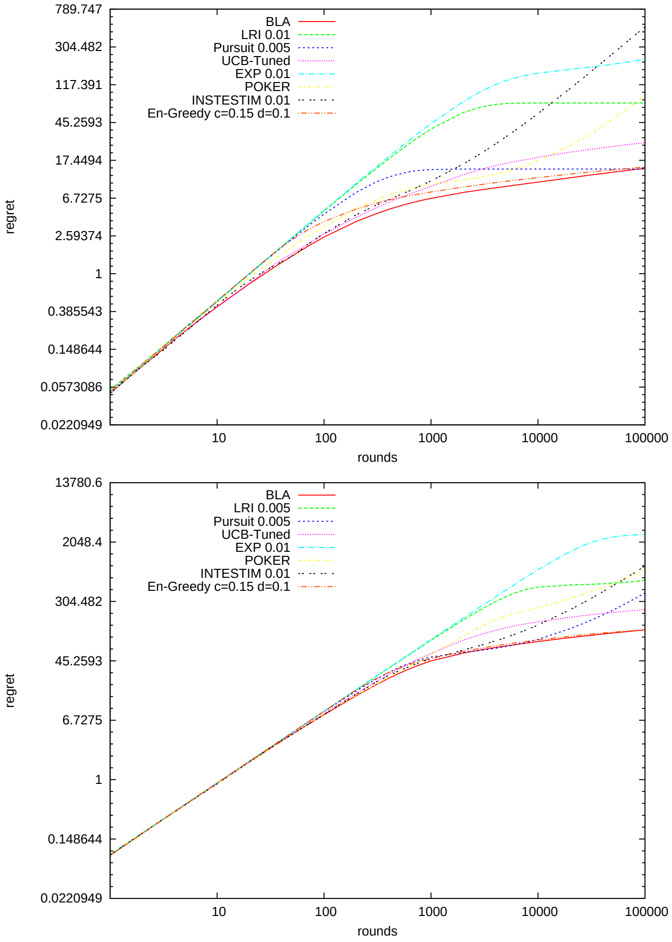


Fig. 2. Regret for experiment config. 2 (top) and config. 5 (bottom)

rithm that takes advantage of Bayesian estimation in a computationally efficient manner.

Extensive experimental results demonstrated that, unlike the ϵ_n-GREEDY, L_{R-I}, and IINTESTIM schemes, BLA does not rely on external learning speed/accuracy control. The BLA also outperformed UCB-TUNED, achieving logarithmically growing regret.

Accordingly, in the above perspective, it is our belief that the BLA represents a new promising avenue of research. E.g., incorporating other reward distributions, such as Gaussian and multinomial distributions, into our scheme is of interest. Secondly, we believe that our scheme can be modified to tackle bandit problems that are non-stationary, i.e., where the reward probabilities are

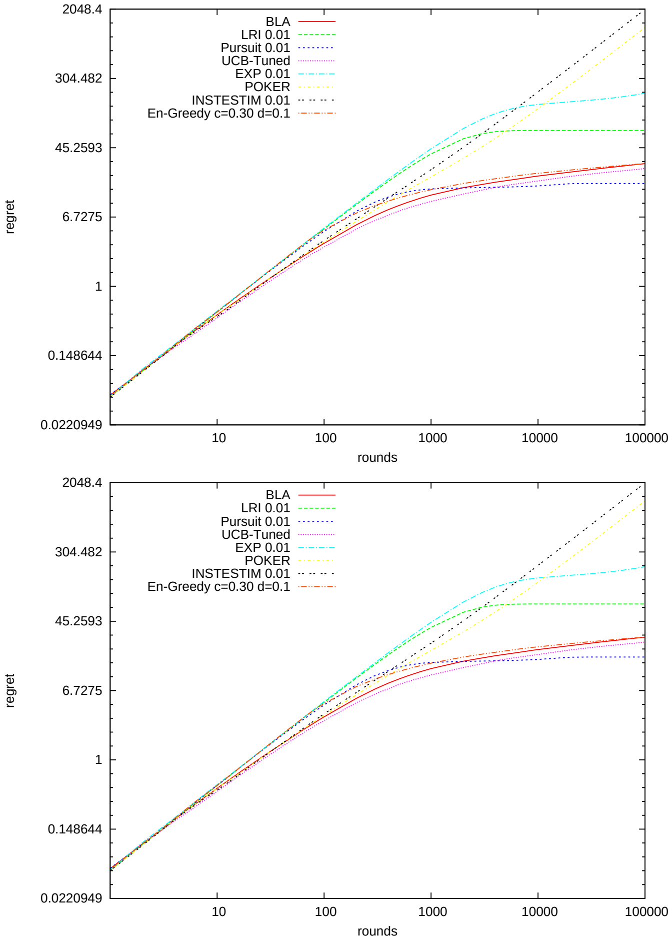


Fig. 3. Regret for experiment config. 3 (top) and config. 6 (bottom)

changing with time. Finally, systems of BLA can be studied from a game theory point of view, where multiple BLAs interact forming the basis for multi-agent systems.

References

1. Wyatt, J.: Exploration and Inference in Learning from Reinforcement. PhD thesis, University of Edinburgh (1997)
2. Thathachar, M.A.L., Sastry, P.S.: Networks of Learning Automata: Techniques for Online Stochastic Optimization. Kluwer Academic Publishers (2004)
3. Narendra, K.S., Thathachar, M.A.L.: Learning Automata: An Introduction. Prentice Hall (1989)

4. Vermorel, J., Mohri, M.: Multi-armed bandit algorithms and empirical evaluation. In: Proceedings of ECML 2005, Springer (2005) 437–448
5. Auer, P., Cesa-Bianchi, N., Fischer, P.: Finite-time Analysis of the Multiarmed Bandit Problem. Machine Learning **47** (2002) 235–256
6. Gelly, S., Wang, Y.: Exploration exploitation in Go: UCT for Monte-Carlo Go. In: Proceedings of NIPS-2006, NIPS (2006)
7. Kocsis, L., Szepesvari, C.: Bandit Based Monte-Carlo Planning. In: Proceedings of ECML 2006, Springer (2006) 282–293
8. Granmo, O.C., 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) (2007) 166–175
9. Granmo, O.C., Bouhmala, N.: Solving the Satisfiability Problem Using Finite Learning Automata. International Journal of Computer Science and Applications, **4**(3) (2007) 15–29
10. Oommen, B.J., Misra, S., Granmo, O.C.: Routing Bandwidth Guaranteed Paths in MPLS Traffic Engineering: A Multiple Race Track Learning Approach. IEEE Transactions on Computers **56**(7) (2007) 959–976
11. Blum, A., Even-Dar, E., Ligett, K.: Routing Without Regret: On Convergence to Nash Equilibria of Regret-Minimizing Algorithms in Routing Games. In: Proceedings of the Twenty-Fifth Annual ACM SIGACT-SIGOPS Symposium on Principles of Distributed Computing (PODC 2006), ACM (2006) 45–52
12. Sutton, R.S., Barto, A.G.: Reinforcement Learning: An Introduction. MIT Press (1998)
13. Tsetlin, M.L.: Automaton Theory and Modeling of Biological Systems. Academic Press (1973)
14. Agache, M., Oommen, B.J.: Generalized pursuit learning schemes: New families of continuous and discretized learning automata. IEEE Transactions on Systems, Man, and Cybernetics-Part B: Cybernetics **32**(6) (2002) 738–749
15. Lanctôt, J.K., Oommen, B.J.: Discretized estimator learning automata. IEEE Transactions on Systems, Man, and Cybernetics **SMC-22**(6) (1992) 1473–1483
16. 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** (2001) 277–287
17. Kaelbling, L.P.: Learning in Embedded Systems. PhD thesis, Stanford University (1993)
18. Bhulai, S., Koole, G.: On the Value of Learning for Bernoulli Bandits with Unknown Parameters. IEEE Transactions on Automatic Control **45**(11) (2000) 2135–2140
19. Dearden, R., Friedman, N., Russell, S.: Bayesian q-learning. In: In AAAI/IAAI, AAAI Press (1998) 761–768
20. Auer, P., Cesa-Bianchi, N., Freund, Y., Schapire, R.E.: Gambling in a Rigged Casino: the Adversial Multi-Armed Bandit Problem. In: Proceedings of the 36th Annual Symposium on Foundations of Computer Science (FOCS'95), IEEE (1995) 322–331
21. Mitchell, T.M.: Machine Learning. McGraw-Hill (1997)
22. Duda, R., Hart, P., Stork, D.: Pattern Classification. 2nd edn. John Wiley and Sons, Inc., New York, NY (2000)
23. Audibert, J.Y., Munos, R., Szepesvarri, C.: Tuning bandit algorithms in stochastic environments. In: Proceedings of the 18th International Conference of Algorithmic Learning Theory, Springer Verlag (2007) 150–165