Beetle Bandit:

Evaluation of a Bayesian-adaptive reinforcement learning algorithm for bandit problems with Bernoulli rewards.

A Thesis presented

by

Bart Jan Buter

in partial fulfillment of the requirements

for the degree of

Bachelor of Science in Artificial Intelligence

FACULTEIT DER NATUURWETENSCHAPPEN, WISKUNDE EN INFORMATICA

Bachelor's programme in Artificial Intelligence

Faculty of Science
Universiteit van Amsterdam
The Netherlands
June 2006

Dedicated to
Mom and Dad
for their ongoing display
of unconditional love.

Beetle Bandit

Bart J. Buter

Submitted for the degree of

Bachelor of Science in Artificial Intelligence
June 2006

Abstract

A novel approach to Bayesian Reinforcement learning (RL) named Beetle has recently been presented; this approach nicely balances exploration vs. exploitation while learning is performed online. This has produced an interest into experimental results obtained from the Beetle algorithm. This thesis gives an overview of bandit problems and modifies the Beetle algorithm. The new Beetle Bandit algorithm is applied to the multi-armed bandit class of problems, thereby comparing the resulting Beetle Bandit algorithm with traditional and current Bayesian inspired approaches.

Declaration

The work in this thesis is based on research carried out at the Faculty of Science of the Universiteit van Amsterdam The Netherlands. No part of this thesis has been submitted elsewhere for any other degree or qualification and it all my own work performed under the guidance of my project supervisor unless referenced to the contrary in the text.

Acknowledgments

Nikos Vlassis for his inspiring courses and his guidance during this final bachelor's project. My mentor Leo Dorst for his guidance and advice during my bachelors programme, though we still have to really drink coffee some day. All teachers of the courses I have taken during my bachelors programme. My fellow students for our pleasant co-operation and indulging classes. Radboud Winkels and Frits de Vries for giving me the opportunity to be student assistant for their courses. Friends and family for their many proof-readings and general support during my education. Especially, sis for taking care of your little brother as you so often do. Dad for his ongoing support, insight and guidance, and Mom though your mind may be leaving your spirit never has.

Thank you all.

Contents

List of Tables

Chapter 1

Introduction

1.1 Introduction

Reinforcement learning (RL) is presented as a field especially suited for problems where planning and learning has to take place simultaneously. A common theme in these problems is the exploration-exploitation trade-off; is it best to exploit the knowledge obtained or do you explore to increase the knowledge base? A prevalent solution to this trade-off is amortization of the cost of exploring, leading to asymptotically optimal behavior, see [17] for a variety of these algorithms.

Bayesian approaches to RL offer the prospect of optimal behavior because an informed trade-off can be made between the cost of obtaining new information and the future reward exploitation this new information will likely provide. This is achieved by learning a transition-reward model, a prior distribution is defined over transition and reward models, the prior distribution and new observation are used to determine the posterior distribution, that is the updated transition-reward model. A drawback of the Bayesian approach to RL has been the intractability of these approaches.

However, recent developments have made approximate solutions to partially observable Markov decision processes (POMDPs) tractable. This, coupled with the knowledge that Bayesian adaptive Markov decision processes (BAMDPs) can be modeled as POMDPs [8], has created a renewed interest in Bayesian approaches. One of the results of this renewed interest is Beetle (Bayesian Exploration Exploitation Trade-off in LEarning) [14].

Multi-armed bandit problems are prototypical problems which display the exploration- exploitation trade-off. These bandit problems are first described by Robbins [15]. In these problems an agent gets to pull one of multiple levers connected to a slot-machine and every lever pays off according to an underlying unknown probability distribution. The objective is to gain maximum reward by pulling the levers, or, more common in this setting, minimizing regret, which is the amount of possible reward lost because of pulling a sub-optimal lever. The exploration-exploitation trade-off now becomes a question of pulling the lever with the highest payoff so far versus pulling a sub-optimal lever but gaining more certainty on the underlying distributions.

This thesis is the result of a four week project, which completes the qualifications needed in obtaining the Bachelor of Science degree in Artificial Intelligence. The project is aimed at evaluating the Beetle algorithm in the multi-armed bandit setting. This thesis is organized as follows: Chapter 1 will give a layman's overview aimed towards friends and family, then it will give a description of the project and its setup. Chapter 2 is aimed to give a literature overview on bandit problems in the field of reinforcement learning and it will give insight into the Beetle Bandit algorithm. Chapter 3 will describe algorithms used in the comparison of the new Beetle Bandit algorithm. Chapter 4 will describe experimental results. Chapter 5 will give a conclusion to this project and give ideas for possible future research.

1.2 Layman's overview

Suppose you enter a casino and the manager welcomes you as the one millionth visitor. Of course, this means you win a special prize. You are let into a room with a few different slot machines. You are allowed to play these one-armed bandits for free, for a fixed period, wanting to extract as much money as quickly as possible from the machines. How will you be able to achieve this goal? By playing the machines you have to find out which machine on average pays out the most and then play this best paying machine, while minimizing the plays on the machine that pays out the least. The problem you are faced with is a classical dilemma between exploration and exploitation. In this case the exploration means playing the machine which has, so far, on average given less payout, but by playing it you are gaining more knowledge and confidence that this machine is worse. Exploiting means playing the machine that has shown the best average payout until now, but how can you be sure this is indeed the best machine to play?

Robbins [15] came up with this multi-armed bandit problem to investigate the exploration-exploitation trade-off. These problems were first studied as statistical problems; however, with the advent of computing it is being studied in computer science. Reinforcement learning (RL) is a machine learning setting where an agent learns a policy (how to behave) by receiving positive or negative rewards while interacting with the environment. This setting is well suited for the bandit problem because both deal with actions and subsequent rewards. This thesis investigates the bandit problem in a RL setting and presents results for different algorithms, one of which is our new Beetle bandit algorithm, playing the multi-armed bandit. More specifically, it investigates the problem where a choice has to be made between machines which pay out 0 or 1 each time, but on average pay a fixed amount per play.

An intuitive way of dealing with this problem is keeping an estimate of the payout probability, thus an average for every machine, and updating this estimate every time a machine is played and more information about the real probability is received. This estimate will be called a belief; with all the information we have received until now we believe the underlying probability to be our estimate. Updating beliefs in the light of new evidence can mathematically be done with Bayes' theorem.

However, this first intuitive solution loses the notion of certainty about your belief. For example observing an average of 0.8 for arm 2 over 100 plays gives more certainty of what is to be expected next time arm 2 is pulled than observing the same average over 10 plays. This is why we take our belief to be a probability distribution over our estimate, meaning we do not say we believe the real probability for arm 2 is 0.8, but we now say we think 0.8 is the most probable real probability. The real probability might also be 0.5, but this is less likely, though 0.5 is more likely with 10 observations than with 100. We can now make an informed decision between gaining short term rewards and/or improving the certainty of our estimates. This new approach, however, causes computational problems.

Since our beliefs are no longer a fixed set of averages but all possible distributions, we cannot compute the best actions to take but we have to make due with estimates. Previous Bayesian approaches to RL slowly computed the estimates online ("while playing") a game. They are therefore impractical in real life applications, For example when a router has to choose the best peer to send a packet it has to be able to decide in milliseconds. However, if this router could pre-compute intermediate results offline (before it is brought into service) it could provide fast online service.

Due to certain mathematical properties of the bandit problem which will be derived in this thesis, we are capable of such a fast online algorithm. We do this by first imagining that we are playing multi-armed bandit problems and then pre-computing the belief updates; playing the game has now become considerably less intensive computationally for the online decision making.

Poupart et al. [14] have recently described a new algorithm from which the new Beetle Bandit algorithm is derived. Beetle is capable of making a good trade-off between exploration and

exploitation while still making fast online decisions. There is a demand for experimental results from this algorithm. This is why it has been adapted for the multi-armed bandit problem. We have compared it to several other algorithms and found that the new Beetle Bandit algorithm performs equal to, if not better than, any of the other reviewed classical algorithms, as long as the problem is kept small. For larger problems other algorithms such as Wang's sampling methods [20] have been found to be better.

1.3 Problem description

The project has primarily been setup to gain experimental data on the Beetle algorithm; more specifically, we have chosen to gather experimental data in order to compare the Beetle algorithm with existing algorithms. The project aims are three-fold:

- Adapting the Beetle algorithm to the domain of bandit problems.
- Review literature on bandit problems to see how this new algorithm compares to existing approaches.
- Gather experimental results to compare Beetle Bandit with algorithms found in literature.

Chapter 2

Bandits Problems

2.1 Historical Background

Bandit problems are a prototypical example of the exploration-exploitation trade-off seen in machine learning algorithms in general and reinforcement learning in particular. The problem was first proposed by Robbins [15]. Put simply, a slot machine with multiple arms, all with different payoff probabilities, can be played with the aim of maximizing reward. There have been many variations on this basic bandit problem. For instance, Duff [7] attaches a reward process to each arm where the reward is dependent on the arm chosen and the internal state of the process attached to this arm. Cicirello et al. [5] try to maximize the largest single reward, while Hardwick et al. [12] study a bandit problem with delayed reward signal. This variety in bandit problems is due to the simplicity, complexity and universality of the problem. That is, the problem statement is easy to grasp and understand, yet it manifests all complexity of the exploration-exploitation trade-off. This is why it has found it's way from sequential design in statistics into machine learning [1], and in particular, reinforcement learning [17] and practical applications such as clinical trial design [11].

Three works are of particular importance to the field of bandit problems, namely those by Bellman [3], Gittins and Jones [10] and Lai and Robbins [13]. Bellman proved that the optimal Bayesian solution was intractable. Gittins and Jones' paper is often described as making the problem tractable. They prove that the optimal decision can be found by computing an index, independent of other arms, for every separate arm after which the optimal choice is the arm with the highest index. However, the computation of the indices is not trivial and needs information about the reward processes [1]. Lai and Robbins proved that optimal exploration policies can be achieved where the regret grows logarithmically with respect to the horizon of the problem.

2.2 Problem Statement

The basic bandit problem is well stated by Auer et al. [1], which we will follow closely in the problem description of the multi-armed bandit with Bernoulli distributed rewards.

A K-armed bandit problem is defined by random variables $X_{i,m}$ for $1 \leq a \leq K$ and $m \geq 1$, where m denotes time and a is the index of an arm or lever of the bandit. By playing the K-armed bandits ath arm rewards $X_{a,1}, X_{a,2}, ...$ are received, these are independent and identically distributed according to the Bernoulli distribution with unknown expectation μ_a. This means $Pr(X_{a,m} = 1) = \mu_a$ where $0 \leq \mu_a \leq 1$. The rewards across arms are also independent; that is $X_{a,s}$ and $X_{b,t}$ are independent for each $1 \leq a \leq b \leq K$ and each $s, t \geq 1$.

A policy, or allocation strategy, A is an algorithm that chooses the action to take, thus which arm to pull, based on the previous plays and subsequent obtained rewards. Let $X_{a,m}$be the reward received when arm a was pulled at time m. Then the *regret* of A after m plays is defined by

$$\mu^{\star} m - \sum_{t=1}^{m} E[X_{\pi_t,t}] \tag{2.1}$$

where

$$\mu^{\star} = \max_{1 \leq a \leq K} \mu_a \tag{2.2}$$

and $E[\cdot]$ denotes expectation, π_t, denotes the action taken at time t when following policy π. Thus the regret is the expected loss due to the fact that the policy does not always play the best arm.

A division can be made between problems with finite vs. infinite horizons. Bandit problems with finite horizons stop after a certain number of rounds, whereas the infinite horizon problems never stop. A consequence of the infinite problem setting is that future cumulative reward also becomes infinite. This is why future cumulative rewards for infinite problems need to be discounted by a term $0 < \gamma < 1$, cumulative future reward then becomes

$$R = \sum_{m=1}^{\infty} \gamma^{m-1} r_m \tag{2.3}$$

where r_m is the reward received at time m. For finite horizon problems the discount factor $\gamma = 1$ can be used to cancel discounting. Since regret is related to reward, future cumulative regret also has to be discounted.

2.3 Important Concepts

Two concepts are of particular importance to Bandit problems, the Gittins index and the proof by Lai and Robins about “optimal" regret, both will be discussed in this section. It is important to note that both concepts are about the infinite bandit case, the Gittins index needs infinite cumulative rewards, while Lai and Robbins claims hold when $m \to \infty$

2.3.1 Gittins Index

This section will describe and give an intuition into the so called Gittins index [9, 10].

Suppose we have a 1-armed bandit with rewards drawn from an unknown Bernoulli distribution. Let n_a denote the number of times a positive reward of 1 is observed after pulling arm a and $\bar{n}_a$, denotes the number of times a reward of 0 has been observed after pulling a, and let γ be a discount factor where $0 < \gamma < 0$. We can now express a value function that is dependent on observations of rewards in the following way

$$V(n, \bar{n}) = \frac{n}{n + \bar{n}} \left[1 + \gamma V(n+1, \bar{n})\right] + \frac{\bar{n}}{n + \bar{n}} \gamma V(n, \bar{n}+1) \tag{2.4}$$

This means we multiply the the chance of receiving a reward $r_m \in \{0, 1\}$, by the obtained reward and all future rewards for both the reward 0 and 1. The future rewards are expressed as a value function with updated observations. This is a Bayesian approach, because the prior probability is used to update the posterior probability with the newly observed rewards.

Suppose we now add an arm to this bandit which gives reward 1 with probability p, this probability is known to the player of the bandit. The discounted reward at round m will then be $\gamma^{m-1} r_m$ and the infinite cumulative reward is given by equation 2.3, for a fixed reward r this will be $\frac{r}{1-\gamma}$. The new value function for this problem becomes

$$V(n, \bar{n}) = \max \left\{ \frac{p}{1-\gamma}, \frac{n}{n + \bar{n}} \left[1 + \gamma V(n+1, \bar{n})\right] + \frac{\bar{n}}{n + \bar{n}} \gamma V(n, \bar{n}+1) \right\} \tag{2.5}$$

The new formula expresses that a choice has to be made between pulling the new arm with fixed probability p or the arm with unknown probability. The Gittins index is defined as the p where the value of pulling the arm with unknown probability is equal to the one with known probability.

For a multi-armed bandit the best action is to pull the arm with the highest Gittins index. However, the Gittins index is not easy to compute, it can be solved iteratively, but this is not a trivial computation.

2.3.2 Lai and Robbins

Lai and Robbins [13] showed that for families of reward distributions, including the Bernoulli distribution, there exist optimal exploration policies where the regret grows logarithmically with the size of the horizon m. this section will stick closely to the formulation of Auer et al. [1]. There exist allocation policies satisfying

$$E[T_a(m)] \leq \left(\frac{1}{D(p_a||p^\star)} + o(1)\right) \ln m \tag{2.6}$$

$$\leq (c_a + o(1)) \ln m \tag{2.7}$$

where $E[]$, denotes expectation, $T_a(m)$ denotes the number of times arm a has been played at time m $o(1) \to 0$ as $m \to \infty$ and $D(p_a||p^\star)$, $a \neq i*$ is the Kullback-Leiber distance or relative entropy [6]. This means that the greater the difference of probabilities between the optimal arm and arm a, the less likely; arm a is to be pulled. It also means that the regret grows logarithmically in the size m.

2.4 The Bandit problem as MDPs

Here the K-armed bandit will be stated as a Markov Decision Process (MDP). We will build on the MDP to create a Bayesian adaptive Markov decision process (BAMDP) stated as partially observable Markov decision process (POMDP). The Math will be derived in a similar fashion to Poupart et al. [14].

2.4.1 Bandits as a Markov decision process

The K-armed bandit modeled as a Markov decision process (MDP) can be formally described as a tuple $<\mathcal{S}, \mathcal{A}, T, R>$. Here the state space $\mathcal{S}$ is the set of states s, there is only one (1) state and this can thus be omitted for convenience. The action space $\mathcal{A}$ is the set of actions $a = \{a_1, ..., a_k\}$, where k denotes the number of arms. The reward model $T(s, a, s') = 1$; since there is only one (1) state after all actions the state will stay the same. The reward model $R(s, a, s') = Pr(r|a)$ encodes the reward received after action a is chosen, where $r \in \{0, 1\}$. Thus in its simplest form the K-armed bandit MDP can be seen as the tuple $<\mathcal{A}, \mathcal{R}>$. The policy describing the actions of an agent, normally $\pi : \mathcal{S} \longmapsto \mathcal{A}$, a mapping from states to actions, now only specifies an action a.

Stationary policies do not change over time, they do not take into account new information obtained over time, for instance randomly selecting an arm is a stationary policy. More interestingly are policies that learn from information that is obtained over time, these non-stationary policies can decide their actions on all available information. policies found by Bayesian adaptive approaches specifically make use of new information obtained, besides taking into account all current available information, they look ahead and also decide their actions on the possibility of gaining new information.

2.4.2 Bandits as POMDP or BAMDP

When we cast the MDP problem as a Bayesian model-based reinforcement learning problem, we are able to learn the reward model while making informed decisions between exploitation and exploration. We do this by introducing a belief about the reward model; this belief is updated after an action is taken and subsequent reward is received. We can model this as a POMDP defined by the tuple $<\mathcal{S}_P, \mathcal{A}_P, \mathcal{O}_P, T_P, Z_P, R_P>$, using the unknown parameter θ_a, which describes the parameter of the Bernoulli distributed reward function μ_a. The state space $\mathcal{S}_P = \{\theta_a\}$, the action space $\mathcal{A}_P = A_{MDP}$. The observation space $\mathcal{O}_P = \{0, 1\}$ is the set of possible rewards. The transition function $T_P(\theta, a, \theta') = Pr(\theta'|\theta, a)$, because θ is assumed not to change, $Pr(\theta'|\theta, a) = \delta_\theta(\theta')$ where $\delta_\theta(\theta)$ is a Kronecker delta with value 1 when $\theta' = \theta$, otherwise zero (0). The observation function $Z_P(\theta', a, o) = Pr(o|\theta', a) = R_{MDP}(a)$ means we observe the reward received after action a. Last, the reward function $\mathcal{R}_P(\theta, a, \theta') = R_{MDP}(a)$ is the same as the underlying MDP.

Rewards are drawn from a Bernoulli distribution with chance of p to obtain reward r, after taking action a

$$p(r|a) = \theta_a^r(1-\theta_a)^{1-r} \tag{2.8}$$

where θ_a denotes the chance of success thus getting reward $r = 1$, and $1 - \theta_a$ is the chance of failure where $r = 0$. Our beliefs have to express a belief about the unknown parameter θ, which parametrizes the Bernoulli distributions attached to each arm. Thus the belief $b(\theta)$ about θ is related to all separate θ_a attached to arm a in the following way.

$$b(\theta) = \prod_{a \in \mathcal{A}} b(\theta_a) \tag{2.9}$$

$$b(\theta_a) = Beta(\theta_a; \overrightarrow{n_a}) \tag{2.10}$$

Where $n_a - 1$ is the count of the number of successes obtained after action a, $\bar{n}_a - 1$ is the count failures obtained after action a, $\overrightarrow{n_a}$ is a vector consisting of both $n_a, \bar{n_a}$. A belief $b(\theta_a)$ about θ_a can be expressed by using a Beta distribution, because Beta distributions describe the chance that a certain Bernoulli distribution is responsible for our observations. The Beta distributions also provide a family of conjugate prior distributions for Bernoulli distributions. This means that a Beta distribution has the property that if it is used as a prior probability in Bayes' rule, the posterior probability will also be a Beta distribution. The formula for the Beta distribution is

$$Beta(\theta_a; n_a, \bar{n}_a) = \frac{1}{B(n_a, \bar{n}_a)}(\theta_a)^{n_a}(1-\theta_a)^{\bar{n}_a} \tag{2.11}$$

$$B(n_a, \bar{n}_a) = \int_0^1 \theta^{n_a}(1-\theta_a)^{\bar{n}_a} d\theta \tag{2.12}$$

Where $B(n_a, \bar{n}_a)$ is the Beta function which is needed to normalize the integral of the distribution to 1. There is a closed form for the $B(n_a, \bar{n}_a)$ function, that involves gamma functions.

We can now learn the reward model θ with belief updates after we perform an action and receive a reward. Using Bayes' theorem the belief update can be written as :

$$
\begin{aligned}
b_a^r(\theta) &= kb(\theta)Pr(r|\theta,a) && (2.13)\\
&= kb(\theta)\theta_a^r && (2.14)\\
&= k\prod_{a\in\mathcal{A}} b(\theta_a)\theta_a^r && (2.15)\\
&= \prod_{a\in\mathcal{A}} Beta(\theta_a;\overrightarrow{n_a})\theta_a^r && (2.16)\\
&= \begin{cases} \prod_{\hat{a}\in\mathcal{A}} Beta(\theta_{\hat{a}};n_{\hat{a}}+1,n\bar{a}) & if\ r=1,\hat{a}=a \\ \prod_{\hat{a}\in\mathcal{A}} Beta(\theta_{\hat{a}};n_{\hat{a}},n\bar{a}+1) & if\ r=0,\hat{a}=a \\ \prod_{\hat{a}\in\mathcal{A}} Beta(\theta_{\hat{a}};n_{\hat{a}},n\bar{a}) & otherwise \end{cases} && (2.17)
\end{aligned}
$$

This means that belief updating for the K-armed bandit can be done by increasing the count relating to the action made and the reward that is received.

2.5 Beetle Bandit

Beetle Bandit is derived from Poupart et al. [14], this paper can give further insight into math of the Beetle Bandit algorithm. In section 2.4 it was described how a K-armed bandit problem can be described as a POMDP, together with the way in which a belief state can be modeled and updated, with new information. What is still left to do is the math to show how the proposed POMDP description can be used to solve the Bellman equation.

2.5.1 The Bellman Equation

The Bellman equation is a way to express the value function for a state, in the case of POMDPs a belief state, when following a policy. Once the optimal value for a state is known, the optimal policy can be derived by selecting that action which has the best trade-off between immediate reward and future reward. The Bellman equation is the manner in which this optimal trade-off is expressed.

First some prerequisites, the chance of receiving a reward r after a history obtained, expressed in belief b, and performing action a is equal to the expectation of the updated Beta reward, which is equal to the updated average reward.

$$
\begin{aligned}
Pr(r|b,a) &= \int_\theta b(\theta)Pr(r|\theta,a) && (2.18)\\
&= \int_\theta \theta_a b(\theta) && (2.19)\\
&= \begin{cases} \frac{n_a}{n_a+\bar{n}_a} & if\ r=1 \\ \frac{\bar{n}_a}{n_a+\bar{n}_a} & if\ r=1 \end{cases} && (2.20)
\end{aligned}
$$

We can now take the Bellman equation for POMDPs and plug in the reward expectation

$$
\begin{aligned}
V^\star(b) &= \max_a \sum_o Pr(o|b,a)\left[R(b,a,b_a^o)+\gamma V^*(b_a^o)\right] && (2.21)\\
&= \max_a \sum_{r\in\{0,1\}} Pr(r|b,a)\left[r+\gamma V^*(b_a^r)\right] && (2.22)\\
&= \max_a \frac{n_a}{n_a+\bar{n}_a}\left[1+\gamma V^*(b_a^{r=1})\right]+\frac{\bar{n}_a}{n_a+\bar{n}_a}\gamma V^*(b_a^{r=0}) && (2.23)
\end{aligned}
$$

where $V^\star(b)$ is the optimal value function and b_a^r means the new belief reached after taking action a and receiving reward r .

Equation 2.23, is still not solvable in the normal sense, because it expresses a recursion, that is, the next state value $V^*(b_a^r)$ has to be computed with the same equation.

2.5.2 Value functions

Luckily value functions exhibit certain characteristics which make it possible to accomplish approximate value iteration [16]. Because value functions are piecewise linear and convex, they can be expressed as the inner product of a belief state and an alpha function

$$V_n(b) = \max_{\{\alpha_n^i\}_i} \int b \cdot \alpha_n^i \tag{2.24}$$

where $\left\{\alpha_n^i\right\}$ is the set of vectors which parametrize the value function V_n at stage n. Since our Bayesian approach deals with a continuous belief space, we can write the optimal value function as the following two equations, in the second equation 2.26, the max has been replaced by the optimal $\alpha^\star_{b_a^r(\theta)} = \arg\max_\alpha \alpha(b_a^r)$, which will be written as $\alpha^\star(\theta)$, it binds the $b_a^r(\theta)$, though this is not written in the subsequent formulas.

$$\begin{aligned} V^\star(b_a^r) &= \max_{\alpha(\theta)} \int_\theta \alpha \cdot b_a^r(\theta) d\theta & (2.25)\\ &= \int_\theta \alpha^\star_{b_a^r(\theta)} \cdot b_a^r(\theta) d\theta & (2.26) \end{aligned}$$

Suppose we have the optimal value function V^k at stage k than we can compute the optimal value function V^{k+1} at the next stage. First we state the formula, the α-functions are plugged in and the max is absorbed into $\alpha^\star$, then $Pr(r|b,a)$ can be expressed in terms of θ as in equation 2.18. Last, i can swap the integral and sum.

$$\begin{aligned} V^{k+1}(b) &= \max_a \sum_{r\in\{0,1\}} Pr(r|b,a)\left[r + \gamma V^*(b_a^r)\right] & (2.27)\\ &= \sum_{r\in\{0,1\}} Pr(r|b,a)\left[r + \gamma \int_\theta \alpha^\star \cdot b_a^r(\theta) d\theta\right] & (2.28)\\ &= \sum_{r\in\{0,1\}} \left[\int_\theta b(\theta) Pr(r|\theta,a)[r + \gamma\alpha^\star(\theta)] d\theta\right] & (2.29)\\ &= \int_\theta b(\theta) \left[\sum_{r\in\{0,1\}} \theta_a [r + \gamma\alpha^\star(\theta)] d\theta\right] & (2.30) \end{aligned}$$

The resulting value function is dependent on r and θ it can thus be a new α-function, Theorem 1 by Poupart [14] et al. proves that because the the α-functions in Bayesian RL have the derived properties, they are multivariate polynomials.

$$\alpha_{b,r}(\theta) = \sum_r [\theta_a [r + \gamma\alpha^\star(\theta)]] \tag{2.31}$$

2.6 Beetle Bandit Algorithm

The Beetle algorithm works in the following way, first it samples the belief space, it acquires game states by simulating bandit games and playing them with a random policy. For every belief state that is sampled the approximate optimal value function can be found using the Perseus point based value iteration algorithm [16]. This is done by considering all actions and

subsequent reward by using formula 2.32, the second formula 2.33 can be used to calculate the optimal action. Finally the last formula 2.34 is used to construct a new α-function from the found optimal α and a, r.

$$\alpha^{\star}_{b^r_a(\theta)} = \arg\max_{\alpha} \alpha(b^r_a) \tag{2.32}$$

$$a^r_b = \arg\max_{\alpha} \sum_{r\in\{0,1\}} Pr(r|b,a)\left[r + \gamma\alpha^{\star}_{b^r_a(\theta)}(b^r_a)\right] \tag{2.33}$$

$$\alpha_{b,r}(\theta) = \textstyle\sum_r \left[Pr(r|\theta,a)[r + \gamma\alpha^{\star}_{b^r_a(\theta)}(\theta)]\right] \tag{2.34}$$

At each Perseus backup step a new α-function is made from other α-functions, this means that with every backup the number of monomials, which together form the α -functions, grows. Therefore a projection step has to take place, to project the new α-function, back onto a fixed basis set. This basis set is build up out of the sampled belief states at the beginning of the algorithm. These computations are all performed offline. Thus making the actual online playing of the K-armed bandit a matter selecting the maximal α-function which is selected from the dot products of the current belief and basis functions. For details on this projection step see the original Beetle paper [14].

2.7 Beetle Bandit-2 and other improvements

Since the growing number of monomials and subsequent basis-projections keep Beetle and Beetle Bandit from scaling to problems with more than a few unknown variables, Beetle Bandit-2 is proposed and implemented. The main idea behind Beetle Bandit-2 is that the belief space is built up out of a Beta-distribution for each arm. All these Beta distributions have a maximum, which lies at the mean of the observed rewards. If the distributions that make up the belief space get sorted according to the mean, the belief space shrinks with a faculty! term. For example a belief space for two actions with Beta-distribution “maximum" of (0.2 and 0.8) will become the same as one with (0.8 and 0.2) When the maxima have the same probability, the number of “observations" ($\overrightarrow{n_a}$) can be used to sort, the belief space. This reduces the belief space and makes it possible for the algorithm to execute faster, or the belief space can be sampled deeper.

A second not implemented improvement comes from the following observation, according to Lai and Robbins, see section 2.3.2, arms get selected with a term which is $\propto log(m)$ where m the current round of the game. This means that an (nearly-) optimal exploration algorithm will visit the action with worst average payoff less than a better one. This means that a large part of the belief space will never get visited when the bandit game is played. Thus the fictitious play of a random policy at the start of the game can be improved, by a more realistic sampling.

Chapter 3

Previous approaches

This chapter describes previous approaches to the multi-armed bandit problem. All algorithms described are finite algorithms, because they do not use any discount factor γ. Usually finite algorithms perform better in an experiment with a short horizon, we therefore feel that these algorithms form a good benchmark for our new Beetle Bandit algorithm.

All algorithms are taken from Vermorel et al. [18], with accompanying implementations from sourceforge.bandit.net, unless stated otherwise. For a more in depth discussion I would like to direct the reader to the Vermorel et al. paper. Caveat Lector, some claims are not substantiated; for instance, in the description of ϵ-decreasing strategy it is claimed that Auer et al. [1] found this strategy to be as good as other strategies described in the cited article. We concur with this statement, however the algorithm described by Vermorel and Auer are not the same. Because Vermorel uses $\epsilon_m = \min\left\{1, \frac{\epsilon_0}{m}\right\}$, where Auer uses $\epsilon_m = \min\left\{1, \frac{\epsilon_0 K}{d^2 m}\right\}$ where $0 < d < 1$. Nonetheless, these algorithms are included in the research because we are confident in the experimental results obtained.

In our multi-armed bandit setting, every action has a chance of returning a success when chosen. This success chance can be estimated, this is called the value estimate, this value estimate of an action a is usually equal to the mean reward obtained after performing action a. If the action with the highest value estimate is chosen, this is called a greedy action.

3.1 ϵ-Greedy algorithms

For all ϵ-greedy algorithms the book by Sutton and Barto [17] is a good source for a more detailed description of these methods.

3.1.1 basic ϵ-greedy

The ϵ-greedy algorithm is one of the simplest RL-algorithms. In its basic form a parameter ϵ is set where $0 \leq \epsilon \leq 1$. The algorithm with a probability ϵ will choose a random action, and the algorithm with probability $1 - \epsilon$ will take a greedy action, it will exploit its current knowledge and choose the action with the highest estimated action value. The extreme parameter setting of $\epsilon = 0$ leads to a completely greedy policy, while the other extreme $\epsilon = 1$ leads to a completely random policy.

3.1.2 ϵ-first

ϵ-First is a variation of the basic ϵ-greedy; here the algorithm starts with an exploration phase, where all actions are chosen at random, after which a completely greedy policy is followed. This means two parameters have to be set, ϵ and the horizon of the problem M, where $0 \leq \epsilon \leq 1$. The exploration phase lasts ϵM rounds while the greedy exploitation phase lasts $(1 - \epsilon)M$ rounds.

3.1.3 ϵ-decreasing

Because the ϵ-greedy algorithm keeps exploring, it does not converge to the optimal policy. Convergence is possible when more exploration is done in the beginning, while an increasingly more greedy policy is followed near the end. ϵ-decreasing is one of the methods to achieve this. Each round the ϵ value is decreased until it reaches 0. Parameter for this algorithm is $\epsilon_0 > 0$, the ϵ_m at each round $1 \leq m \leq M$ is calculated by $\epsilon_m = min\left\{1, \frac{\epsilon_0}{m}\right\}$.

3.1.4 LeastTaken

LeastTaken is another ϵ-greedy variety although the policy does not take an arbitrary random action. The least taken arm is pulled with probability $\epsilon_m^a = 4/(4 + l^2)$, where l is the number of times the least taken action has been taken. With probability $1 - \epsilon_m^a$, the greedy policy is taken.

3.2 Softmax algorithms

For the first two algorithms, the book by Sutton and Barto [17] is again a good starting point for additional information.

3.2.1 Gibbs-Softmax

This algorithm will simply be called Softmax. Like the ϵ-greedy algorithms, Softmax also gives the highest chance of selection to the greedy action. All other actions are given a probability weighted by their value estimate according to a Gibbs distribution. The algorithm has parameter τ; this is the temperature of the Gibbs distribution. This is expressed in the following formula

$$Pr(a) = \frac{e^{Q_m(a)/\tau}}{\sum_{b=1}^{K} e^{Q_m(b)/\tau}} \tag{3.1}$$

Thus, the chance of picking action a at time m depends on the current action-value estimate $Q_m(a)$, which is the mean, the number of actions k and temperature parameter τ. When τ is set high, the part $Q_m(a)$ plays in the equation becomes less, thus the policy will perform more randomly and explore more. On the other hand, when $\tau \to 0$ the greedy action will be performed.

3.2.2 Softmix

What Softmax still misses is the behavior of ϵ-decreasing where more exploration is done in the beginning and more exploitation in the end. This can be done by adjusting temperature τ every round, with a higher τ in the beginning for more exploration and a lower temperature for exploitation. Softmix does exactly this; it is parametrized by τ_0 the starting temperature and is adjusted every round by $\tau_m = \tau_0 \log(t)\ /t$.

3.2.3 Exp3

The Exponential weight algorithm for exploration and exploitation is described by Auer et al. [2]. This bandit problem algorithm was designed without statistical assumptions about the process generating the payoffs of the slot machines. Exp3 generates new weights for the played arm which is dependent on the reward received, as well as the chance this action was actually selected. The formulas for calculating the probabilities that action a gets chosen at time m are,

$$p_a(m) = (1 - \gamma)\frac{w_a(m)}{\sum_{b=1}^{K} w_b(m)} + \frac{\gamma}{K} \tag{3.2}$$

$$w_a(m+1) = w_a(m) \exp\left(\gamma \frac{r_a(m)}{p_a(m)K}\right) \tag{3.3}$$

The only parameter γ can be set $0 \leq \gamma \leq 1$, it controls the importance which is given to this weight w_a. Here a higher γ means a more random policy.

3.3 Interval estimation

With interval estimation, a reward estimate within a confidence interval is kept for every action. This confidence interval gets tighter around the estimate each time the action is taken because more information means more confidence. A decision bound is set to a percentage of the upper bound of the confidence interval. An action is taken greedily with respect to this decision bound. This means that actions which have not been chosen often, and therefore have a high interval upper bound, are likely to be explored. After a few selections of every action, the confidence interval upper bound of actions with low expected reward will be considerably lower because of a low reward estimate and a tighter interval. Thus, after an exploring start phase, only actions with highest expected reward will remain to be exploited.

IntEstim is parametrized by α where $0 < \alpha < 1$. This parameter sets the decision bound u_a used for action selection as a portion of the upper bound in the following way, $\mu_a = \mu(1-\alpha)$, where μ is the upper bound. This means a low α value causes more exploration because the decision bound is kept high. When $\alpha \to 0$ there is no interval and the reward estimate itself is used for action selection. μ_a is calculated with the following formulas,

$$\mu_a = \hat{\mu} + \frac{\hat{\sigma}}{\sqrt{2\pi}} cdf(1-\alpha)^{-1} \tag{3.4}$$

$$cdf(x; \hat{\mu}, \hat{\sigma}) = \frac{1}{\hat{\sigma}\sqrt{2\pi}} \int_{-\inf}^{x} \exp(-\frac{(u-\mu)^2}{2\sigma^2}) du \tag{3.5}$$

Where $\hat{\mu}$ and $\hat{\sigma}$ are the empirical mean and standard deviation, $cdf(x; \hat{\mu}, \hat{\sigma})$ is the cumulative normal distribution function. This algorithm makes the assumption that the arm mean is normally distributed. Therefore algorithm needs two observations per action to make the math work; therefore, in the beginning, random actions are taken until an action is chosen twice. Then, obtained $\hat{\sigma}$ for actions with more than 1 observation are averaged and used as an estimate for actions with only one observation. This averaged $\hat{\sigma}$ is also used for actions with no observations together with the average of means $\hat{\mu}$.

3.4 Poker

The Price of Knowledge and Estimated Reward algorithm POKER is built with three ideas in mind. The POKER algorithm makes a trade-off between exploration and exploitation by pricing knowledge gained by exploration. The horizon of the problem is specifically taken into account, for example, the last possible action before the game stops should always be greedy, because information gained will be useless. Last, the algorithm assumes that if there are more possible actions than rounds, and thus no chance of observing all actions, the observed actions-rewards can be used to estimate unobserved action-rewards. The main pricing formula which unites these concepts is given by

$$p_a = \hat{\mu}_a + Pr\left[\mu_a \geq \hat{\mu}^\star + \delta_\mu\right] \delta_\mu M \tag{3.6}$$

with M is the horizon, μ_a is the reward mean, $\hat{\mu_a}$ is the reward mean estimate, $\hat{\mu}^*$ is the highest reward mean estimate, and $\delta_\mu = E[\mu^* - \hat{\mu}^*]$, is the expected reward mean improvement.

For an algorithm, Vermorel et al. [18] describes it extensively.

3.5 UCB2

The UCB2 algorithm [1], is not taken from [18], it is chosen because it is presented as a Bernoulli bandit algorithm, which performs almost as good as a tuned ϵ-greedy algorithm. The main practical advantage of this algorithm is that it is more robust, thus the parameter setting does not need to be tuned, a sufficiently low setting is enough. A theoretical advantage is that it achieves logarithmic regret uniformly spread over time. Lai and Robbins, see section 2.3.2, demonstrated and proved that logarithmic regret is possible asymptotically, UCB2 does this uniformly, no asymptotical behavior is needed.

UCB2 selects the arm which maximizes $\mu_a + o_{n,r_a}$ and plays this for an episode of $\tau(r_a+1) - \tau(r_a)$ times , μ_a is the current average reward for action a o_{n,r_a} is calculated by the formulas

$$o_{n,r} = \sqrt{\frac{(1+\alpha)\ln(em/\tau(r))}{2\tau(r)}} \tag{3.7}$$

$$\tau(r) = \begin{cases} 0 & for\, r < 0. \\ \lceil (1+\alpha)^r \rceil & r \geq 0. \end{cases} \tag{3.8}$$

where m is the total number of rounds played until now, r is a counter which is initialized with 0, and gets increased at the end of an episode(which can last multiple rounds).

3.6 Optimistically initialized Greedy

This algorithm will be called Greedy [4, 17]. This algorithm gets initialized as if it has already done an observation for every action, which all gave a positive reward. The optimistic action-reward averages are updated ever round. An action is greedily chosen with respect to these optimistic action-reward averages.

Chapter 4

Experiments

4.1 Overview of experimental setup

This chapter will describe the experiments which where designed to be able to compare the newly proposed Beetle Bandit algorithm with other algorithms. There will be two setups: a 2-armed bandit setup and a 5-armed setup. These two setups are chosen because the 2-armed bandit problem is the simplest problem which still exhibits the whole complexity of the exploration exploitation trade-off. The 5-armed bandit is chosen so the results obtained can be compared to the article of Wang et al. [20]. The obtained results will be discussed at the end of the chapter. One has to take into account that Beetle Bandit is an infinite horizon algorithm where the other algorithms it is being compared to are not. Usually finite horizon algorithms perform better than infinite horizon algorithms on problems with a small horizon.

4.2 2-Armed bandit experiments

Since we are confident the newly reported Beetle Bandit algorithm will make a good trade-off between exploration and exploitation, we have designed an experiment with a short horizon. This means algorithms with good asymptotic behavior but poor "start" behavior are expected to not perform well on this setup. This is because they generally over explore in the beginning, then over exploit in the end, making the poor start behavior. Algorithms are run with parameters set in multiple ways; the best results for every algorithm are reported in this section. UCB2 is the only algorithm that was run with only 1 parameter because the authors have put this algorithm forward to be robust as long as the parameter α is set low. The authors optimal setting for UCB2 is used.

4.2.1 Experimental setup

The setting uses 2-armed bandit problems. The arms generate Bernoulli distributed rewards, with means drawn uniformly from the interval (0,1) (though all algorithms play the same game) this is done for 20 different games. The game horizon is 20. Every game is played 1000 times. Reward and regret per round will be reported.

Beetle Bandit has a number of parameters, the following parameters are used. To test the influence of infinite vs finite games played by Beetle Bandit, discount factors $\gamma = 0.99$ and 1 were used. The number of that are sampled is 2000 and the maximal number of basis functions that were created was 100. 100 Perseus backups were performed. Every game new basis vectors are initialized. State sampling of the belief space is performed with the same horizon as the problem.

Table 4.1: Best average rewards and regrets per round for 2-armed Bandit

Algorithm	avg-Reward	avg-Regret
Beetle Bandit, $\gamma = 1$	0.46298	0.21850
Beetle Bandit, $\gamma = 0.99$	**0.62192**	**0.05956**
Beetle Bandit-2, $\gamma = 0.99$	**0.64077**	**0.04071**
Greedy	**0.62459**	**0.05688**
UCB2, $\alpha = 0.001$	0.60819	0.07328
POKER	0.58526	0.09622
ϵ-greedy, $\epsilon = 0.15$	0.56346	0.07328
ϵ-first, $\epsilon = 0.15$	0.58661	0.07328
ϵ-decreasing, $\epsilon = 1$	0.58769	0.09379
LeastTaken $\epsilon = 0.15$	0.58661	0.11220
SoftMax $\tau = 0.15$	0.61387	0.06761
SoftMix $\tau_0 = 0.1$	0.49146	0.19002
Exp3 $\gamma = 0.4$	0.53718	0.14572
IntEstim $\alpha = 0.1$	0.59307	0.08841

4.2.2 Discussion of results 2-armed Bandit experiments

The Beetle Bandit algorithm performs badly without discounting when $\gamma = 1$. This is because the Perseus algorithm used is made for infinite problems, without discounting future rewards, every next value function will be higher than the last. Convergence onto the optimal value function is no longer guaranteed. This is why the Beetle Bandit, $\gamma = 1$ sometimes performs best, but on average performs poorly see Table 4.1. However, normal Beetle Bandit performs with $\gamma = 0.99$ almost best, only optimistically initialized Greedy is better. Beetle Bandit-2 is an improvement on the Beetle Bandit algorithm, it reduces the size of the belief space by ordering the Beta-distributions which make up the belief space. This has a positive effect, Beetle Bandit-2 performs best of all algorithms.

4.3 5-Armed Beetle experiment

The 5-arm experiment is designed to make the results comparable to Wang et al. [20]. Which gives results pertaining to other algorithms than the ones used in the 2-armed Bandit experiment. The Beetle Bandit $\gamma = 0.99$ and Beetle Bandit-2 $\gamma = 0.99$ algorithms are both run in a setup similar to Wang. Though the original paper by Wang only gives graphs, through personal correspondence with the author the precise results have been obtained [19]. Results displayed will be by Wang with the Beetle Bandit Algorithms added.

4.3.1 Experimental setup

The described setup is for the Beetle Bandit and other algorithms described in this thesis, Wang et al. [20] report averages taken from 1000 to 10000 episodes. The setting uses 5-armed bandit problems. The arms generate Bernoulli distributed rewards, with means drawn uniformly from the interval (0,1), this is done for 20 * 1000 different games. The game horizon is 5, 10, 15 and 20. Every game is played once. Optimal parameters of the 2-armed bandit experiment are used. Reward and regret per round will be reported.

Beetle Bandit has a number of parameters, the following parameters are used. A discount factor of $\gamma = 0.99$. The number of belief states that are sampled is 2000 and the maximal number of basis functions that were created was 100. 100 Perseus backups were performed. Initialization of new basis vectors will occur every 1000 games, thus 20 times.. State sampling of the belief

space is performed with the same horizon as the problem. State sampling of the belief space is done with the same horizon as the problem.

Table 4.2: Results by Wang et al. compared with Beetle Bandit, 5-armed bandit reward per round is reported

Horizon	5	10	15	20
ϵ-greedy	0.5905	0.6311	0.6511	0.6703
Bolzman	0.574	0.6244	0.6494	0.6758
Interval Est.	0.498	0.5858	0.6061	0.6313
Thompson	0.5389	0.5965	0.6141	0.6412
MVPI	0.5511	0.6029	0.613	0.6445
Peret Garcia	0.555	0.5887	0.6097	0.6479
Sparse Samp.	0.6075	**0.6598**	**0.6806**	**0.7148**
Bayes Samp	**0.6161**	**0.6686**	**0.6868**	**0.7212**
Beetle Bandit, $\gamma = 0.99$	**0.6109**	0.6496	0.6673	0.7027
Beetle Bandit-2, $\gamma = 0.99$	0.5986	0.6580	0.6697	0,6987

4.3.2 Discussion of results 5-armed Bandit experiments

Table 4.2 shows that Beetle Bandit and Beetle Bandit-2 are comparable in performance on the 5-armed bandit. The results for Beetle Bandit are better than those for the classical approaches. However, the Sparse and Bayes sampling techniques are generally a bit better than Beetle Bandit. There are two reason why Beetle Bandits performance does not grow equally with Bayesian sampling. Beetle Bandit is an infinite horizon algorithm, where Bayesian sampling is a finite horizon algorithm. Infinite horizon algorithms usually perform slightly worse than finite horizon algorithms which do not discount rewards.

Beetle Bandit also suffers from the size of the belief space when the horizon of the problem grows. The reachable belief space can be seen as a tree with the root at the start of a game, the tree branches out with every possible action and reward, this means the size of the tree will be $(2K)^H$, where K is the number of arms and H the horizon. The sampling of the belief space which beetle performs at the beginning of the algorithm is not capable of getting a representative sample of the whole belief space, this the performance of the algorithm drops as the size of the belief space grows larger.

Chapter 5

Conclusion

In this thesis I have given an overview of Bandit Problems, in particular those with rewards obtained from Bernoulli distributions. I have given an overview of algorithms with which our newly developed Beetle Bandit algorithm was compared, as well as the math needed to adjust Beetle into Beetle Bandit. The experiments show that for Bandit problems with limited arms, horizon, and rewards distributed according to a Bernoulli distribution, the Beetle Bandit is performing better than classical approaches and slightly worse than other current Bayesian inspired approaches. These results occur because Beetle Bandit is an infinite horizon algorithm whereas the compared algorithms are finite horizon algorithms which are usually better at dealing with short horizon setups such as those presented in this paper. The other cause of these results is the problem Beetle Bandit has with adequate sampling of large belief spaces. The Bayesian approach taken with Beetle Bandit offers the possibility of an optimal exploration vs. exploitation trade-off. Beetle Bandit has shown through offline pre-computation that the Bayesian approach can give best results while still being able to make fast online decisions.

The main advantage of Beetle Bandit is its performance on problems with a small number of variables. For these problems, it offers one of the best exploration exploitation decision making available. However, a problem still facing Beetle and Beetle Bandit are larger problems with more variables. This is partly due to the size of the belief space, as sampling this efficiently becomes increasingly harder and the projection onto a fixed set of basis functions becomes more prone to errors as the size of the belief space increases. I have made a start at improving this vulnerability of Beetle Bandit by sorting the distributions making up the belief space. The results of this approach are promising, as for small problems the 'adapted' Beetle Bandit-2 does not perform worse, but sometimes even better than Beetle Bandit. Making Beetle Bandit scale to larger problems is the main area of future research. One may think of more intelligent belief space sampling, better belief space representations, or better projection onto basis functions. An other way can be to keep the look ahead decision tree small by intelligent sampling techniques. A first step into this future research has been given with Beetle Bandit-2. For in terms of practical applications, one can think of intelligent adaptive routers which can use Beetle Bandit in discovering what peers should be selected for optimal service. Beetle Bandit can also be used in adaptive medical trials where the goal is to decide which is the better treatment while, balancing this with the the least possible negative impact.

Bibliography

[1] P. Auer, N. Cesa-Bianchi, and P. Fischer. Finite-time analysis of the multiarmed bandit problem. *Machine Learning*, 47(2/3):235–256, 2002.

[2] P. Auer, N. Cesa-Bianchi, Y. Freund, and R. E. Schapire. Gambling in a rigged casino: the adversarial multi-armed bandit problem. In *Proceedings of the 36th Annual Symposium on Foundations of Computer Science*, pages 322–331. IEEE Computer Society Press, Los Alamitos, CA, 1995.

[3] R. Bellman. A problem in the sequential design of experiments. *Sankhya A*, 16:221–229, 1956.

[4] R. I. Brafman and M. Tennenholtz. R-MAX — a general polynomial time algorithm for near-optimal reinforcement learning. In *IJCAI*, pages 953–958, 2001.

[5] V. Cicirello and S. Smith. The max k-armed bandit: A new model for exploration applied to search heuristic selection. In *20th National Conference on Artificial Intelligence (AAAI-05)*, July 2005. Best Paper Award.

[6] T. M. Cover and J. A. Thomas. *Elements of Information Theory*. John Wiley, New York, 1991.

[7] M. O. Duff. Q-learning for bandit problems. In *International Conference on Machine Learning*, pages 209–217, 1995.

[8] M. O. Duff. *Optimal learning: computational procedures for Bayes-adaptive Markov decision processes*. PhD thesis, University of Massachusetts Amherst, 2002. Director-Andrew Barto.

[9] J. Gittins. *Bandit Processes and Dynamic Allocation Indices*. John Wiley, 1989.

[10] J. C. Gittins and D. M. Jones. A dynamic allocation index for the sequential design of experiments. *Progress in Statistics*, 1:241–66, 1974.

[11] J. Hardwick. *IMS Lecture Notes – Monograph Series*, chapter A modified bandit as an approach to ethical allocation in clinical trials, pages 65–87. Institute of Mathematical Statistics, 1995.

[12] J. Hardwick, R. Oehmke, and Q. F. Stout. New adaptive designs for delayed response models. *Journal of Sequential Planning and Inference*, 136:1940–1955, 2006.

[13] T. Lai and H. Robbins. Asymptotically efficient adaptive allocation rules. *Advances in Mathematics*, 6:4–22, 1985.

[14] P. Poupart, N. Vlassis, J. Hoey, and K. Regan. An analytic solution to discrete Bayesian reinforcement learning. In *Proceedings of the International Joint Conference on Machine Learning*, pages 249–256, Hakodate, Japan, May 2006.

[15] H. Robbins. Some aspects of the sequential design of experiments. *Bulletin American Mathematical Society*, 55:427–535, 1952.

[16] M. T. J. Spaan and N. Vlassis. Perseus: Randomized point-based value iteration for POMDPs. *Journal of Artificial Intelligence Research*, 24:195–220, 2005.

[17] R. S. Sutton and A. G. Barto. *Reinforcement Learning*. MIT Press, Cambridge, MA, 1998.

[18] J. Vermorel and M. Mohri. Multi-armed bandit algorithms and empirical evaluation. In *Machine Learning: ECML 2005: 16th European Conference on Machine Learning, Porto, Portugal, October 3-7, 2005. Proceedings*, volume Volume 3720, pages 437 – 448, Nov 2005.

[19] T. Wang. Personal correspondence, June 2006.

[20] T. Wang, D. Lizotte, M. Bowling, and D. Schuurmans. Bayesian sparse sampling for on-line reward optimization. In *ICML '05: Proceedings of the 22nd international conference on Machine learning*, pages 956–963, New York, NY, USA, 2005. ACM Press.