ABSTRACT

Title of dissertation: LEARNING TECHNIQUES IN MULTI-ARMED BANDITS

Neha Gupta, Doctor of Philosophy, 2012

Dissertation directed by: Professor Ashok K. Agrawala
Department of Computer Science

Multi-armed bandit problem is a classic example of the exploration vs. exploitation dilemma in which a collection of one-armed bandits, each with unknown but fixed reward probability, is given. The key idea is to develop a strategy, which results in the arm with the highest reward probability to be played such that the total reward obtained is maximized. Although seemingly a simplistic problem, solution strategies are important because of their wide applicability in a myriad of areas such as adaptive routing, resource allocation, clinical trials, and more recently in the area of online recommendation of news articles, advertisements, coupons, etc. to name a few.

In this dissertation, we present different types of Bayesian Inference based bandit algorithms for Two and Multiple Armed Bandits which use Order Statistics to select the next arm to play. The Bayesian strategies, also known in literature as "Thompson Method" are shown to function well for a whole range of values, including very small values, outperforming UCB and other commonly used strategies. Empirical analysis results show a significant improvement in both synthetic and real

datasets.

In the second part of the dissertation, two types of Successive Reduction (SR) strategies - 1) Successive Reduction Hoeffding (SRH) and 2) Successive Reduction Order Statistics (SRO) are introduced. Both use an Order Statistics based Sampling method for arm selection, and then successively eliminate bandit arms from consideration depending on a confidence threshold. While SRH uses Hoeffding Bounds for elimination, SRO uses the probability of an arm being superior to the currently selected arm to measure confidence. The empirical results show that the performance advantage of proposed SRO scheme *increasing* persistently with the number of bandit arms while the SRH scheme shows similar performance as pure Thompson Sampling Method.

In the third part of the dissertation, the assumption of the reward probability being fixed is removed. We model problems where reward probabilities θ are *drifting*, and introduce a new method called *Dynamic Thompson Sampling (DTS)* which adapts the rewarad probability estimate, $\hat{\theta}$, faster than traditional schemes and thus leads to improved performance in terms of lower regret. Our empirical results demonstrate that DTS method outperforms the state-of-the-art techniques, namely pure Thompson Sampling, UCB-Normal and UCB_f, for the case of dynamic reward probabilities. Furthermore, the performance advantage of the proposed DTS scheme increases persistently with the number of bandit arms.

In the last part of the dissertation, we delve into arm space decomposition and use of multiple agents in the Bandit process. The three most important characteristics of a multi-agent systems are 1) Autonomy – agents are completely or partially

autonomous, 2) Local views – agents are restricted to a local view of information, and 3) Decentralization of control – each agent influences a limited part of the overall decision space. We study and compare Centralized vs. Decentralized Sampling Algorithm in Multi-Armed Bandit problems in the context of *common payoff games*. In the Centralized Decision Making, a central agent maintains a global view of the currently available information and makes a decision to choose the next arm just as the regular Bayesian Algorithm. In Decentralized Decision Making, each agent maintains a local view of the arms and makes decisions just based on the local information available at its end without communicating with other agents. The Decentralized Decision Making can be modeled as a *Game Theory* problem. Our results show that the Decentralized systems perform well for both the cases of Pure as well Mixed Nash equilibria and their performance scales well with the increase in the number of arms due to reduced dimensionality of the space.

We thus believe that this dissertation establishes Bayesian Multi-Armed bandit strategies as one of the prominent strategies in the field of bandits and opens up venues for new interesting research in the future.

LEARNING TECHNIQUES IN MULTI-ARMED BANDITS

by

Neha Gupta

Dissertation submitted to the Faculty of the Graduate School of the
University of Maryland, College Park in partial fulfillment
of the requirements for the degree of
Doctor of Philosophy
2012

Advisory Committee:
Professor Ashok K. Agrawala, Chair/Advisor
Professor Hal Daumé III
Professor Amol Deshpande
Professor Atif Memon
Professor Charles B. Silio, Jr.

Acknowledgments

First and foremost I am grateful to my advisor, Dr. Ashok Agrawala for his guidance and support. As an international student with no family in the USA, coming to this country was like rebirth for me. Prof. Agrawala help me make this transition smoothly and guided me throughout while instilling confidence and curiosity in me. He taught me how to unfold problems and understand them conceptually before taking them further, two very essential tools for academic research. He also made learning fun for me once again. He let me explore different areas and gave me freedom to choose my topic of research, and I am very thankful to him for this.

I would like to thank Dr. Ole-Christoffer Granmo for collaborating with me and sharing his understanding about the Multi-Armed problem. He helped me improve my presentation and writing skills which made my work more effective. I m looking forward to working with him in future. I would also like to thank Dr. Hal Daumé III for providing feedback on the problem and also for encouraging my work. I m also grateful to my committee members Dr. Amol Deshpande, Dr. Atif Memon and Dr. Charles Silio for their feedback and review of my work. I would also like to thank all my teachers who have taught me wonderful things all my life.

Also, working in the MINDLAB has given me to opportunity to work in a very stimulated environment on many different research topics. I would like to thank by collaborators and friends : Christian Almazan, Matthew Mah, Shiv Krishnamoorthy, Preeti Bhargava, David Thaw, Raghu Narasimhan, Gleneesha Johnson, Rincy

Mathew, for working with me. I have learnt a lot from each one of them and I am very grateful for this opportunity. Also, I would like to thank Sandeep Nawathe and Pradeep Javangula for introducing me to the problem of online learning and arising my curiosity in the field. I would also like to thank Tumri, Inc. for their support in this work. Also, thanks to Jennifer Story and Fatima Bangura for helping me get through the graduate school rules and formalities.

All of this would not have been possible without the endless love of my husband Nirmal Sharma. His ability to look the brighter side of things and make me laugh in the worst of times has kept me going. My parents, Hari Mohan Gupta and Vinod Gupta, thank you for your unconditional love and support. I also appreciate my brother Mohit Gupta and my in-laws for understanding and believing in me.

I am also very grateful to my friends - Monica Syal, Neha Mukherjhee-Nathani, Priyanka Roshyan, Arvind Agrawal, Harish Nathani, Manmeet Singh and Satinder Pal Singh, for being there for me at all times. Thanks to everyone else who has helped me.

Dedicated to

My parents and my husband

Table of Contents

List of Tables

List of Figures

List of Abbreviations and Symbols

k	Bandit arm
K	Total number of bandit arms
θ^k	Random variable denoting reward probability of arm k
n	Total no of bandit trials
n^k	Total number of trials of arm k
$\hat{\theta}_n^k$	Estimated reward probability of arm k
$E(\theta)$	Expected value of random variable θ
$Var(\theta)$	Variance of random variable θ
Δ_{ij}	Difference between reward probability θ_i and θ_j
Φ_n	Infinite state space of $2K$ dimensions $\{(\alpha_n^1, \beta_n^1), (\alpha_n^2, \beta_n^2), ..., (\alpha_n^K, \beta_n^K)\}$.
ν_n	Gaussian Noise

MAB	Multi-Armed Bandit
TAB	Two-Armed Bandit
BGP	Beta Geometric Probabilistic
BS	Beta Sampling
BGD	Beta Geometric Deterministic
TS	Thompson Sampling
UCB	Upper Confidence Bound
SRH	Successive Reduction Hoeffding
SRO	Successive Reduction Order Statistics
DTS	Dynamic Thompson Sampling
DeTS	Decentralized Thompson Sampling
EP-d	Epsilon Decreasing Strategy
EP-n	ϵ_n Strategy
RMSE	Root Mean Square Error

Chapter 1

Introduction

To introduce the problem, let us consider a Bernoulli distribution having outcomes $\{0, 1\}$ with a probability θ. If θ is unknown, then we can obtain an estimate of θ by observing the samples of Bernoulli trials. After N samples with n_s successes, $\hat{\theta}$ is given by

$$\hat{\theta} = \frac{n_s}{N} \tag{1.1}$$

$$Std.Dev(\hat{\theta}) = \frac{\sqrt{\theta(1-\theta)}}{\sqrt{N}} \tag{1.2}$$

As we increase the number of trials, the standard deviation of $\hat{\theta}$ approaches 0, and hence the accuracy of $\hat{\theta}$ increases. We can consider this problem of estimating unknown θ from the Bernoulli trials as a "one-armed bandit" problem.

Now, suppose we are given two bandits with probabilities θ^1 and θ^2. At each trial, only one arm can be pulled, and we need to maximize the rewards obtained from the observations of the output. The decision of pulling the arms can be made in several ways. We can observe a few samples of θ^1 and θ^2 and then only select the arm whose $\hat{\theta}$ is higher for the future pulls. In this case, we explore the arms first and then exploit the "optimal arm". Although the accuracy of this estimate will depend on the number of past observations made since as the number the past observations reduces, the standard deviation increases. But if there is a cost associated with each

pull of an arm and the total reward of the arms needs to be maximized, then the exploration of the "optimal arm" needs to be done together with the "exploitation". This dilemma of exploration vs. exploitation can be addressed by the Multi-Armed Bandit problem.

The bandit problem has been considered hard to solve in general. According to Whittle [7] — "the problem is a classic one; it was formulated during the war, and efforts to solve it so sapped the energies and minds of Allied analysts that the suggestion was made that the problem be dropped over Germany, as the ultimate instrument of intellectual sabotage."

1.1 Importance of Topic

The dilemma of "exploration vs. exploitation" is applicable in many different areas of research not only in the field of computer science, but in electrical engineering and business as well.

In the Multi-Armed setting, there could be different types of rewards obtained from playing the arms - rewards could be stochastic or deterministic, they could be based on a parametric model such as Gaussian, Poisson, etc; rewards could also be dynamic in nature. Although the application areas are many, below we only discuss two application areas in the field of computer science.

1.1.1 Optimization in the Internet Domain

Earlier, the trend was that the Internet publishers hand-picked information items for a website based on their knowledge of their readers. But with increasing number of information items as well as a large variety of users, hand-picking information which serves a huge user base, is no longer feasible. A recent trend has been the use of algorithmic optimization for showing information items dynamically based on the likelihood of users to pay attention or respond to an information item by clicking on it, engaging (reading, doing a mouse over) with it, visiting a related webpage, etc. to name a few. These information items could be advertisements, news articles, coupons, etc. to name a few. This problem of Internet optimization can be directly mapped to the problem of MAB with each information item being an arm and each impression being a trial.

1.1.2 Networks - Routing

The role of a router in a wireless network is to find the fastest route for the packets as sending a packet on a congested route can lead to an unnecessary delay, hence low performance of the network. The router has to explore a number of traffic paths and at the same time exploit the information it has for sending the packets across through the minimum delay route. Thus, the problem of choosing an optimal route by a router can be mapped to the bandit problem.

1.2 Background

1.2.1 Reinforcement Learning

Reinforcement learning addresses the question of how an autonomous agent having the ability to sense the feedback (reward) obtained from the environment can learn to choose the options available to him and achieve its goals [19]. The agent must discover the optimal action using the scalar reward from each selected action, also referred to as reinforcement. Since the goal of the agent is to maximize the long term rewards, the agent tries to learn and identify the optimal action by trying several different options.

In a stochastic environment the feedback will typically be distributed in accordance to a probability distribution with the parameters of the probability distribution being unknown to the agent. To obtain knowledge about these parameters, the agent needs to perform a sequence of trials, and obtain a view of the probability distribution of the rewards. Reinforcement learning is different from traditional supervised learning in many ways, most important being 1) In Reinforcement Learning, the agent receives the immediate reward on the current action but does not receive the total reward. Thus, the agent needs to learn about different possible states and actions of the system before deciding which actions to take in order to maximize his total reward. 2)In Reinforcement Learning, learning is done concurrently with applying the current learning which means exploring along with exploitation.

In Reinforcement learning, the environment consisting of a system and agent, is usually modeled as a *Markov Decision Process* (MDP).

1.2.2 Markov Decision Process

Markov Decision Process is a discrete time, stochastic, control process in which the state and reward of the agent depends probabilistically on the actions the agent has taken at each step. The process can be defined in terms of:

- discrete set of states S.
- discrete set of actions A.
- set of transition probabilities $\mathrm{P}(s, s')$ of going from any state s to s' and corresponding rewards $\mathrm{Ra}(s, s')$.

At each time step, the system is in some state s, and the decision maker chooses one of the actions, say a, available in state s which changes the system to another state s' and gives the agent a corresponding reward $\mathrm{Ra}(s, s')$. Thus, the next state s' depends on the current state s and the agent's action a. But given s and a, s' and $\mathrm{Ra}(s, s')$ are conditionally independent of all previous states and actions; in other words, the state transitions of an MDP possess the "Markov property".

The problem of Multi Armed Bandits is a single state Markov Decision process in which the state of the system remains the same, but the agent has to learn the system parameters using multiple trials based on the stochastic rewards obtained from the environment.

1.2.3 Classic Multi Armed Bandits - Performance Evaluators

In the MAB setting, each pull of an arm can be considered as a Bernoulli trial with output in the set {0,1} and defined by a single parameter θ^k which is the probability of success denoted by {1}. The goal is to maximize the rewards and minimize the regret. The key performance evaluators for the Multi-Armed Bandit settings are as follows:

Let $\hat{\theta^k}$ denote the estimated reward probability of arm k and θ^* denote the reward probability of the best arm, n being to number of plays so far then,

- % Best Arm Played: The % of times the best arm is played.
- Expected Reward is defined as:

$$\Sigma_{k=1}^{K} \theta^k E[T^k(n)] \tag{1.3}$$

- Expected Regret defined as:

$$\theta^* n - \Sigma_{k=1}^{K} \theta^k E[T^k(n)] \tag{1.4}$$

where $E[T^k(n)]$ is the expected value of the number of times k^{th} arm is played.

The strategies which ensure no pulls of the suboptimal arms asymptotically are called *zero-regret strategies*.

1.3 Contributions

Below we describe the important contributions made in this thesis:

I The most commonly used algorithms for solving MAB problems have been the different variants of the UCB algorithm [2]. However, the UCB algorithms do not scale well for small values of reward probabilities (< 0.01). In this work, we present two main types of Bayesian Bandit algorithms – "Beta Geometric Probabilistic", and "Beta Sampling" ("Thompson Sampling") for the case of Two and Multiple Armed Bandits which use Order Statistics for selecting the next arm to play. The Bayesian Thompson Method based strategies are shown to function for non-rare as well as rare reward probabilities, outperforming UCB and other commonly used strategies. Empirical analysis done in this dissertation shows that a significant improvement in the results is obtained by using Bayesian algorithms for both synthetic and real datasets.

II In the second part of this dissertation, we present two types of Successive Reduction (SR) strategies - 1) Successive Reduction Hoeffding (SRH) and 2) Successive Reduction Order Statistics (SRO). Both use an Order Statistics based Thompson Sampling method for arm selection, and then successively eliminate bandit arms from consideration based on a confidence threshold. While SRH uses Hoeffding Bounds for elimination, SRO uses the probability of an arm being superior to the currently selected arm to measure confidence. A computationally efficient scheme for pairwise calculation of the latter probability is also presented in this dissertation. Using SR strategies, sampling resources and arm pulls are not wasted on arms that are unlikely to be the optimal one. To demonstrate the scalability of our proposed schemes, we compare them with

two state-of-the-art approaches, namely pure Thompson Sampling and UCB-Tuned. The empirical results show that the performance advantage of proposed SRO scheme *increasing* persistently with the number of bandit arms while the SRH scheme shows similar performance as base Thompson Method. We thus believe that SR algorithms will open up for improved performance in Internet based on-line optimization where it is common to have a few hundred trials, and tackling of larger problems.

III The most common assumption made when solving such MAB problems is that the unknown reward probability θ of each bandit arm is fixed. However, this assumption rarely holds in practice simply because real-life problems often involve underlying processes that are dynamically evolving. In the next part of this dissertation, we model Multi-Armed Bandit in the framework where reward probabilities θ are *drifting*, and introduce a new method called *Dynamic Thompson Sampling (DTS)* that facilitates Order Statistics based Thompson Sampling for dynamically evolving MAB problems. The DTS algorithm adapts the success probability estimate, $\hat{\theta}$, faster than traditional Thompson Sampling schemes and thus leads to improved performance in terms of lower regret. Our experiments demonstrate that DTS method outperforms current state-of-the-art approaches, namely pure Thompson Sampling, UCB-Normal and UCB_f, for the case of dynamic reward probabilities. Furthermore, the performance advantage of the proposed DTS scheme increases persistently with the number of bandit arms.

IV The presence of multiple agents has risen in the present day systems, with the rise in the system complexity and tasks, for which a single agent has become increasingly incapable. These multiple agents are typically autonomous, maintain local view of information, and have a finite cost of communication. In this dissertation, we study and compare Centralized vs. Decentralized Thompson Sampling Algorithm in Multi-Armed Bandit problems in the context of *common payoff games*. In the Centralized Decision Making for Thompson Sampling, a central agent maintains a global view and makes a decision to choose the next arm just as the regular Thompson Sampling Algorithm. While in Decentralized Decision Making, each agent maintains a local view of the arms and makes decisions just based on the local information available without communicating with other agents. The Decentralized Decision Making is modeled as a *Game Theory* problem. Our results show that the Decentralized systems perform well for both the cases of Pure as well Mixed Nash equilibria and their performance scale well with the increase in the number of arms due to reduced dimensionality of the space in the same.

Chapter 2

Related Work: Non-Bayesian Techniques in Multi-Armed Bandits

2.1 Introduction

Seminal work on Multi-Armed Bandit policies was done by Lai and Robbins [20]. They proved that for certain reward distributions, such as Bernoulli, Poisson, and uniform, there exists an asymptotic bound on regret (the loss experienced due to playing the suboptimal arms) that only depends on the logarithm of the number of trials and the Kullback-Leibler value of each reward distribution. The main idea behind the strategy is to calculate an upper confidence index for each arm, which only depends on the previous rewards of that arm. At each trial the arm which has the maximum upper confidence value is played, thus enabling a deterministic play. Agrawal [1] improved the results obtained by Lai and Robbins by proposing strategies that are independent of the reward distributions. Auer et al. [2] further proved that instead of an asymptotic logarithmic upper bound, an upper bound on the regret could be obtained in finite time for algorithms such as UCB-1, UCB-2 and some variants. The pioneering Gittins Index based strategy [7], for instance, performs a Bayesian look ahead at each step in order to decide which arm to play. This look ahead makes the Gittins technique intractable in practice, however it enables an optimal performance. We will discuss these algorithms one by one in detail. We first start with Confidence bound based algorithms.

2.1.1 UCB Algorithms

Upper Confidence Bound based algorithm is a deterministic algorithm in which at each step an index number is calculated which is used to select the next arm to play. The UCB-1 [2] algorithm computes an Upper Confidence Bound (UCB) for each arm: $(\hat{\theta}_n^k + \sqrt{\frac{2\ln\ n}{n^k}})$ as shown in Alg. 1. Here, $\hat{\theta}_n^k$ is the average reward obtained from the arm k when the number of times arm k has been played is n^k and n is the overall number of plays so far. In this algorithm, the arm which has the maximum UCB value is played and the confidence bounds are updated at each trial.

UCB-2 is a similar strategy in which the picked arm is played for a certain periods which are calle "epochs". The upper bounds of UCB-2 is calculated by the formula $\sqrt{\frac{(1+\gamma)\ln(en/\tau(r))}{\tau(r)}}$, where γ is a constant, r denotes the current epoch of the machine and $\tau(r) = \lceil (1+\gamma)^r \rceil$ as shown in Alg. 2.

UCB-Normal is a modification of UCB algorithm for the case of Gaussian reward probabilities and is shown in Alg. 3. The bounds for the UCB-Normal becomes $(\theta_n^k + \sqrt{16.\frac{q_n^k - n^k(\hat{\theta}_n^k)^2}{n_k - 1}\frac{\ln\ (n-1)}{n^k}})$ where q_n^k is the squared sum of rewards of arm k.

UCB-tuned [2] is also a slight modification of UCB-1 that uses an upper bound of $\sqrt{\frac{\ln\ n\ min(1/4, \mathrm{V^k}(n))}{n}}$, where $\mathrm{V^k}(n)$ denotes the estimated variance of arm k. There are no theoretical proofs supporting UCB-tuned, but in empirical results, its performance turns out to be better than that of UCB-1, UCB-2.

Algorithm 1 Algorithm: Upper Confidence Bound (UCB-1)

Play each arm once.

loop

Compute the Upper Confidence Bound (UCB) for each arm: $(\hat{\theta}_n^k + \sqrt{\frac{2\ln\ n}{n^k}})$.

Play the arm with the highest value of the Upper Confidence bound.

end loop

Algorithm 2 Algorithm: Upper Confidence Bound (UCB-2)

Set $r_k = 0$ for $k = 1, 2, 3, ..., K$. Play each arm once.

loop

Select machine k maximizing $\hat{\theta}_n^k + \sqrt{\frac{(1+\gamma)\ln(en/\tau(r))}{\tau(r)}}$

Play the arm k exactly $\tau(r^k + 1) - \tau(r^k)$ times.

Set $r_k = r_k + 1$.

end loop

Algorithm 3 Algorithm: Upper Confidence Bound - Normal (UCB-Normal)

loop

If there is an arm which has been played less than $\lceil 8\log n \rceil$ times, play the arm.

Select machine k having the maximum value $(E(\hat{\theta}_n^k) + \sqrt{16.\frac{q_n^k - n^k(\hat{\theta}_n^k)^2}{n^k - 1}\frac{\ln\ (n-1)}{n^k}})$.

end loop

2.1.2 Greedy Algorithms

ϵ-greedy algorithms are one of the simplest strategies in solving bandit problems and is shown in Alg. 4. The key idea behind the algorithm is that with probability ϵ within the range [0,1], any random arm is pulled from the set of arms and with probability $1-\epsilon$ the best arm is pulled. There are several different variants of the epsilon greedy strategy. One of the ϵ-strategies is described in [2] as ϵ_n-greedy strategy (EP-n). EP-n uses a factor $\epsilon_n = min(1, \frac{cK}{d^2 n})$ where c, d are constants and $0 \leq d \leq min(\theta^k - \theta^*)$. Second, form of ϵ strategy is the ϵ-decreasing (EP-d) strategy [31], where the probability ϵ is a function of n and decreases asymptotically with the number of trials such that $\epsilon_n = \frac{\epsilon_0}{n}$ where $\epsilon_0 > 0$.

Algorithm 4 Algorithm: ϵ-greedy

loop

With probability ϵ, play any arm at random and with probability $1 - \epsilon$, play arm $k = argmax(\hat{\theta}^k), k \in K$

end loop

2.2 Dynamic (Restless) Bandits

The most common assumption made when solving such MAB problems is that the unknown reward probability θ of each bandit arm is fixed. However, this assumption may not hold true in all cases and the bandit arms may change with time, or in other words be *dynamic*. Dynamic Bandits are also called *Restless Bandits* [32] . Restless Bandits were introduced by Whittle and are considered to be

PSPACE-hard. Guha et al. [12] introduced approximation algorithms for a special setting of the Restless Bandit problem. Below we discuss some of the techniques used to solve the problem of "Dynamic Bandits".

2.2.1 Adaptive Restless Bandits with Brownian Motion

UCB_f algorithm [26] is a more general form of $UCB-1$ algorithm which includes the case of reward probabilities varying as Brownian motion with reflecting boundaries. In UCB_f, an extra confidence interval term is introduced, $\sigma^k\sqrt{8N\log\ N}$ where σ^k is the known standard deviation of the Brownian motion of arm k. The algorithm is shown in Alg. 5.

Algorithm 5 Algorithm: Upper Confidence Brownian Motion (UCB_f)

Play each arm once.

loop

Compute the Upper Confidence Bound (UCB) for each arm: $(\theta_n^k + \sqrt{\frac{2\ln\ n}{n_k}}) + \sigma^k\sqrt{8N\log\ N}$.

Play the arm with the highest value of the Upper Confidence bound.

end loop

2.2.2 Adversarial Bandits - Exp3.

Auer et al. [3] also introduced another version of restless bandits called the Adversarial Bandits but the technique introduced was designed to perform against an all powerful adversary and hence let to very loose bounds for the rewards. The algorithm is shown in Alg. 7.

Algorithm 6 Algorithm: Hedge

Parameter: A real number $\eta > 0$.

Initialization: Set $G_i(0) := 0$ for $i = 1, 2, 3, ..., K$.

loop

Choose action i_t according to the distribution $p(t)$, where

$$p_i(t) = \frac{\exp(\eta G_i(t-1))}{\Sigma_{j=1}^{K} \exp(\eta G_j(t-1))}$$

Receive the reward vector $\mathrm{x}(t)$ and score gain $x_i(t)$.

Set $G_i(t) = G_i(t-1) + x_i(t)$ for $i = 1, 2, ..., K$.

end loop

Algorithm 7 Algorithm: Exp3

Parameter: A real number $\eta > 0$ and $\gamma \in (0, 1]$.

Initialization: Initialize **Hedge(η)** as in Alg. 6.

loop

Get the distribution $p(t)$ from **Hedge**.

Select action i_t to be j with probability $\hat{p_j}(t) = (1-\gamma)p_j(t) + \frac{\gamma}{K}$.

Receive reward $x_i(t) \in [0, 1]$.

Feed the simulated reward vector $\hat{\mathrm{x}}(t)$ back to Hedge, where $\hat{x_j}(t) = \frac{x_i(t)}{\hat{p_i}(t)}$

end loop

2.3 Bandits with separate Exploration and Exploitation phases

Some versions of the bandits have been proposed in literature in which the exploration phase is considered separate from the exploitation phase. We discus few of them below, which handle similar problems as discussed in this dissertation.

2.3.1 Best Arm Identification Bandits

In [18], an algorithm for best arm identification by successively rejecting the sub-optimal arms has been proposed for the problem of pure exploitation. The problem of finding the best arm by doing pure exploitation is different from the problem discussed in this dissertation. In the later exploration and exploitation have to be done together, to maximize the objective function which is the total reward obtained, while in the former the objective function is to find the arm closest to the "optimal" arm.

2.3.2 Multi-Agent Systems for Bandits

Multi agent systems have been studied for a long time [27, 4, 30, 29]. The problem discussed in [30] is of particular interest in context of this work since it deals with common payoff games. Verbeeck et al. introduce a Learning Automata based algorithm for Common Interest Games which consist of two phases – exploration phase and synchronization phase. In the exploration phase, the agents independently optimize using the *learning automata* based algorithm and the objective is to converge to the best arm. In the synchronization phase, the agents compare the

best arm obtained in the current exploration phase to the best arm obtained in the previous exploration phase. If the current best arm is better than the previous best arm, the former is removed temporarily from the next exploration phase so that the system can converge to another better arm, if one exists. If the current best arm is worse than the previous best arm, than the former is removed permanently from the system as it is sub-optimal. This technique works for scenarios in which exploration and exploitation are done in separate phases but not for the techniques in which exploration and exploitation have to be performed together as in the current setting of this dissertation.

Chapter 3

Bayesian Inference and Order Statistics

3.1 Statistical Inference

The process of drawing conclusions based on data is called Statistical Inference. There could be two major assumptions made about the data depending on which there are two types of Statistical models - 1) Parametric Models — in which data is assumed to be generated from a certain distribution family, the members of which are distinguished by parameter values. The Normal distribution family is the most commonly used in parametric models. 2) Non-parametric Models — in which no prior assumptions are made about the process generating the data.

Also, there are two major school of thoughts regarding the type of inferencing 1) Classical /Frequentist Inference – is based on the principle that the parameter of a distribution are constants. 2) Bayesian Inference - is an approach in which all forms of uncertainty are expressed in terms of a subjective probability distribution called prior distribution. A prior distribution over the unknown parameters of the model is formulated, which is meant to capture the beliefs about the situation before seeing the data. After, each data is observed the Bayes' Rule is applied to obtain a posterior distribution for these unknowns using both the prior and the data.

$$p(\theta|y) = \frac{p(\theta, y)}{p(y)} \tag{3.1}$$

3.2 Bayes Theorem and parameter estimation

As shown in the previous section, the Bayes Theorem [5] forms the basis of Bayesian Inference. Using the Bayes theorem, the prior distribution of the unknown parameter θ is updated on observing data y leading to a posterior distribution.

The posterior distribution of θ given y is :

$$p(\theta|y) = \frac{p(\theta, y)}{p(y)} \tag{3.2}$$

p(y) acts like a normalization constant since its value is independent of θ. Thus,

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

The probability $p(y|\theta)$ is called the *likelihood*, $p(\theta)$ is the *prior* and $p(\theta|y)$ is the *posterior*.

$$\text{posterior} \propto \text{prior} \times \text{likelihood} \tag{3.4}$$

3.2.1 Types of Priors

3.2.1.1 Informative Priors

If some prior knowledge is available about the distribution of the parameter θ, then it could be incorporated in the prior y to get a better estimate of the posterior. Such priors are called *informative priors*.

3.2.1.2 Non-informative Priors

"Let the data speak for themselves" seems to imply that it would be a violation of scientific objectivity to be influenced by other considerations such as prior knowledge about a hypothesis. Hence, non-informative priors which typically assume equal probabilities for the occurrence of all events are very commonly used.

3.2.1.3 Conjugate Priors

Conjugate Priors are the priors having the same functional form as the *likelihood*, which leads to the *posterior* distribution being of the same form as the *prior*. For example, Gaussian distribution is a conjugate prior to itself, Beta distribution is a conjugate prior to Bernoulli, Binomial and Geometric distributions.

3.3 Possible Distributions for Bernoulli MAB

In the MAB setting, each pull of an arm can be considered as a Bernoulli trial with output in the set {0,1} and defined by a single parameter θ which is the probability of success denoted by {1}. We use conjugate priors to model the MAB reward probabilities θ.

3.3.1 Binomial Distribution

In the MAB setting, each pull of an arm can be considered as a Bernoulli trial with output in the set {0,1} and defined by a single parameter θ which is the probability of success denoted by {1}. The probability distribution of the number of

successes, denoted by S, obtained in n Bernoulli trials is known to have a Binomial distribution, $S \sim Binomial(n, \theta)$.

$$p(S = s|\theta) = \binom{n}{s}(1-\theta)^{n-s}\theta^s \tag{3.5}$$

$$\tag{3.6}$$

3.3.2 Geometric Distribution

The probability distribution of the number of Bernoulli trials needed to get one success is known to have a geometric distribution. If θ is the probability of success of the bernoulli trial, then the probability of getting $(k-1)$ failures before getting a success at the k-th trial is given by,

$$p(X = k|\theta) = (1-\theta)^{k-1}\theta \tag{3.7}$$

$$E(X) = \frac{1}{\theta} \tag{3.8}$$

$$Var(X) = \frac{1-\theta}{\theta^2} \tag{3.9}$$

It is known that Beta distribution is a conjugate prior for the Binomial and Geometric distribution [13], hence we assume θ to be a Beta distribution with parameters (α_0, β_0):

$$p(\theta; \alpha_0, \beta_0) = \frac{\theta^{\alpha_0-1}(1-\theta)^{\beta_0-1}}{B(\alpha_0, \beta_0)} \tag{3.10}$$

If a success is received at the n^{th} trial, α and β are updated as,

$$\alpha_n = \alpha_{n-1} + 1, \beta_n = \beta_{n-1} \tag{3.11}$$

or if a failure is received at the n^{th} trial, α and β are updated as,

$$\alpha_n = \alpha_n, \beta_n = \beta_{n-1} + 1 \tag{3.12}$$

After s successes and r failures, the parameters of Beta distribution become $(\alpha_0 + s,\ \beta_0 + r)$.

$$\theta \sim Beta(\alpha_0 + s, \beta_0 + r) \tag{3.13}$$

The estimated mean and variance after n trials become,

$$\hat{\theta}_n = \frac{\alpha_n}{\alpha_n + \beta_n} \tag{3.14}$$

$$\hat{\sigma}_n^2 = \frac{(\alpha_n \beta_n)}{(\alpha_n + \beta_n + 1)(\alpha_n + \beta_n)^2} \tag{3.15}$$

3.3.3 Properties of Bayesian Inferences in Beta-Distribution

Property 1 (Estimate of Mean) *Irrespective of the estimate of initial value of θ, the expected value of θ will become unbiased and will converge to the true value.*

Proof 3.1 *Let θ be beta-distributed random variable with prior parameters $(\alpha_0,\ \beta_0)$. After n samples of the random variable suppose we get r successes, hence new parameters $(\alpha_0 + r, \beta_0 + n - r)$ are obtained.*

$$E[\theta_{prior}] = \frac{\alpha_0}{\alpha_0 + \beta_0}$$

$$E_R[\theta_{post}] = \frac{\alpha_0 + r}{\alpha_0 + \beta_0 + n}$$

$$E_R[\theta_{post}] = \xi_n \frac{r}{n} + (1 - \xi_n)\frac{\alpha_0}{\alpha_0 + \beta_0}$$

where

$$\xi_n = \frac{n}{\alpha_0 + \beta + n} \tag{3.16}$$

Hence,

$$E_R[\theta_{post}] = \xi_n \frac{r}{n} + (1 - \xi_n) E[\theta_{prior}] \tag{3.17}$$

Thus, the posterior estimate of θ is the weighted average of the prior mean and the likelihood estimate. The average depends on ξ_n and the value of $(1 - \xi_n)$ converges to 0 as n tends to infinity. Hence, irrespective of the initial values of α_0 and β_0, θ will converge to the true value.

Property 2 (Estimate of Variance) *In a single series, estimate of the variance converges to 0 as the number of trials go to infinity.*

Proof 3.2 *The following equation describes the relationship between the prior and the expected value of the posterior variance in beta distribution [13].*

$$E_R({Var_{post}}^{(n,r)}) = \frac{\alpha_0 + \beta_0}{\alpha_0 + \beta_0 + n} Var_{prior} \tag{3.18}$$

The above equation clearly verifies that as n tends to infinity, the variance will converge to 0.

3.4 Order Statistics

Order statistics refers to statistical methods that depend only on the ordering of the data and not on its numerical values. Given any random variables $X_1, X_2, \ldots, X_K$, the order statistics $X_{(1)}, X_{(2)}, \ldots, X_{(K)}$ are also random variables, defined by sorting the values of $X_1, X_2, \ldots, X_K$ in increasing order. Although the average of the data is an important estimate of its central value, it is not an order statistic. However, the median is an order statistic parameter.

In the current setting of exploration vs. exploitation, order statistics plays a very important part. At each point in the trials, a decision has to be made regarding

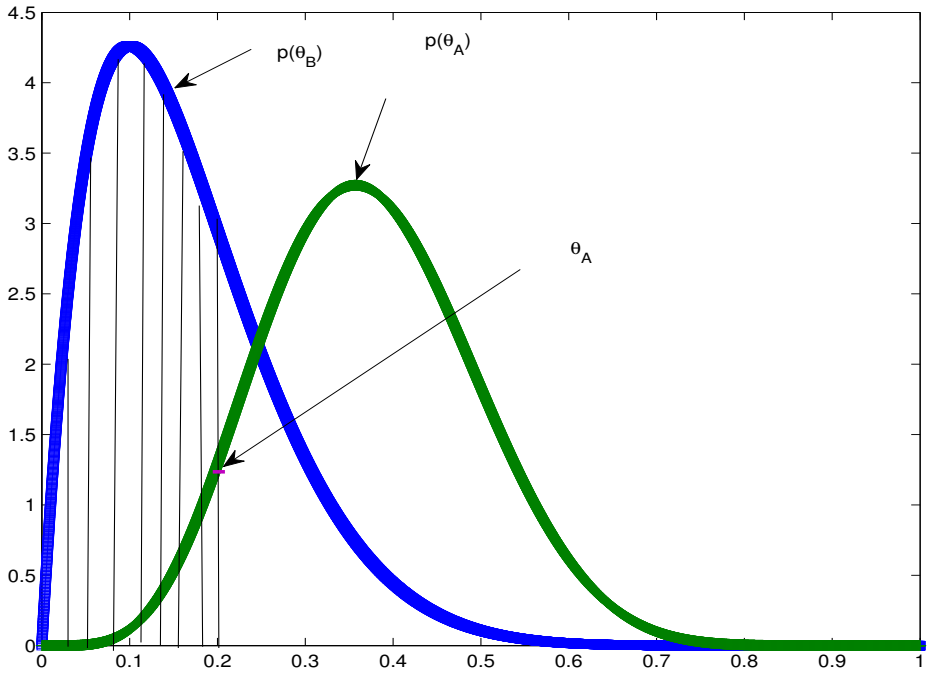

Figure 3.1: Computation of $Probability(\theta^A > \theta^B)$ for one value of θ^A.

the next arm to pull in order to optimize the objective function. But the current estimates of θ may not be accurate, hence a deterministic decision solely based on the current values may lead to sub-optimal values in the long term.

To solve this dilemma of *exploration vs. exploitation* in a two-armed bandit setting, let us assume that we are given a two arms reward probabilities – θ^A and θ^B. $P(\theta^A > \theta^B)$ is the probability of the random variable θ^A being greater than θ^B based on the current samples.

The formula for $P(\theta^A > \theta^B)$ when θ^A and θ^B are beta-distributed is shown below.

$$P(\theta^A > \theta^B)$$

$$= \int_0^1 p(\theta^A) \int_0^{\theta^A} p(\theta^B) \mathrm{d}\theta^B \mathrm{d}\theta^A$$
$$= \int_0^1 \frac{\theta^{A\alpha^A - 1}(1-\theta^A)^{\beta^A - 1}}{B(\alpha^A, \beta^A)} \int_0^{\theta^A} \frac{\theta^{B\alpha^B - 1}(1-\theta^B)^{\beta^B - 1}}{B(\alpha^B, \beta^B)} \mathrm{d}\theta^B \mathrm{d}\theta^A$$

Fig. 3.1 shows two graphs corresponding to $p(\theta^A)$, $p(\theta^B)$ with $\theta^A > \theta^B$. The shaded area represents the area under curve for $\int_0^{\theta^A} p(\theta^B)\mathrm{d}\theta^B$, the internal integral, for single sample of θ^A.

We prove the following for the case of symmetric distributions of random variable θ^A and θ^B with expected values of $\hat{\theta}^A$ and $\hat{\theta}^B$.

Property 3 (Values of $P(\theta^A > \theta^B)$) *If $\hat{\theta}^A = \hat{\theta}^B$, then $P(\theta^A > \theta^B) = 0.5$, If $\hat{\theta}^A > \hat{\theta}^B$, then $P(\theta^A > \theta^B) > 0.5$, else $\hat{\theta}^A < \hat{\theta}^B$, then $P(\theta^A > \theta^B) < 0.5$.*

Proof 3.3 *We prove the above results for symmetric distributions since the beta distribution converges to normal distribution for large values of n. $\hat{\theta^A}$ denotes the expected value of θ^A.*

Case: $\alpha^A = \alpha^B$ and $\beta^A = \beta^B \Rightarrow \hat{\theta}^A = \hat{\theta}^B$.

$$P(\theta^A > \theta^B)$$
$$= \int_0^1 p(\theta^A)[P_{\theta^B}(\theta < \theta^A)]\mathrm{d}\theta^A$$
$$= \int_0^{\hat{\theta^A}} p(\hat{\theta^A} - \theta^A)[P_{\theta^B}(\theta < \hat{\theta^A} - \theta^A) + P_{\theta^B}(\theta < \hat{\theta^A} + \theta^A)]\mathrm{d}\theta^A$$
$$= \int_0^{\hat{\theta^A}} p(\hat{\theta^A} - \theta^A)[P_{\theta^B}(\theta > \hat{\theta^A} + \theta^A) + P_{\theta^B}(\theta < \hat{\theta^A} + \theta^A)]\mathrm{d}\theta^A$$
$$= \int_0^{\hat{\theta^A}} p(\hat{\theta^A} - \theta^A) \times 1 \mathrm{d}\theta^A$$
$$= \frac{1}{2}$$

Hence, when

$$\hat{\theta}^A = \hat{\theta}^B => P(\theta^A > \theta^B) = 0.5 \tag{3.19}$$

Case: $\hat{\theta}^A > \hat{\theta}^B$

$$\begin{aligned}
&P(\theta^A > \theta^B) \\
&\quad = \int_0^1 p(\theta^A)[P_{\theta^B}(\theta < \theta^A)]\mathrm{d}\theta^A \\
&\quad = \int_0^{\hat{\theta^A}} p(\hat{\theta^A} - \theta^A)[P_{\theta^B}(\theta < \hat{\theta^A} - \theta^A) + P_{\theta^B}(\theta < \hat{\theta^A} + \theta^A)]\mathrm{d}\theta^A \\
&\quad = \int_0^{\hat{\theta^A}} p(\hat{\theta^A} - \theta^A) \times (value > 1)\mathrm{d}\theta^A \\
&\quad > \frac{1}{2}
\end{aligned}$$

Case: $\hat{\theta}^A < \hat{\theta}^B$

$$\begin{aligned}
&P(\theta^A > \theta^B) \\
&\quad = \int_0^1 p(\theta^A)[P_{\theta^B}(\theta < \theta^A)] \\
&\quad = \int_0^{\hat{\theta^A}} p(\hat{\theta^A} - \theta^A)[P_{\theta^B}(\theta < \hat{\theta^A} - \theta^A) + P_{\theta^B}(\theta < \hat{\theta^A} + \theta^A)]\mathrm{d}\theta^A \\
&\quad = \int_0^{\hat{\theta^A}} p(\hat{\theta^A} - \theta^A) \times (value < 1)\mathrm{d}\theta^A \\
&\quad < \frac{1}{2}
\end{aligned}$$

Hence, this probability of $P(\theta^A > \theta^B)$ could be used for deciding the next arm to pull leading to a randomized strategy for the case of two-armed bandit which is discussed in detail in the following chapters.

Chapter 4

Two armed Bandit

4.1 Introduction

Multi-armed bandit (MAB) problem [2, 7, 20, 1] is the problem in which a collection of one armed bandits, each with unknown but fixed reward probability θ, is given. The key idea is to develop a strategy which results in the arm with the highest reward probability to be played such that the total reward obtained is maximized. This problem is a classic example of the exploration vs. exploitation dilemma.

In order to develop effective solutions for the MAB problem, we first develop techniques for the problem of two arms i.e *two armed bandits (TAB)*, and then extend them to address the MAB problem. In this chapter, we focus on the TAB problem for the case of non-rare as well as rare events. We define *rare events* as the events whose success probabilities are < 0.01. Such situations naturally arise in the domain of display advertising where clicks (rewards) are rare, and are also common in areas such as reliability engineering, environmental sciences, genomics, etc. To address the TAB problem, in this work, we present a set of Bayesian algorithms -**Beta-Geometric Probabilistic (BGP), Beta (Geometric) Sampling (BGS) and Beta-Geometric Deterministic (BGD)** which are based on the Order Statistics based Thompson Method [28].

The main idea behind Thompson Method is the use of Bayesian inferences and subjective probabilities to compute the probability of one random variable being greater than the other ($P(\theta^A > \theta^B)$) to select an arm to play, where θ^A, θ^B are the reward probabilities. Wyatt [33] used factorial look up tables for the evaluation of the function $P(\theta^A > \theta^B)$ based on the Beta-Binomial model. In this work, we use the Beta-Geometric model and present an efficient technique for computing $P(\theta^A > \theta^B)$ which reduces the computation complexity and makes the algorithm practically feasible. We also introduce a deterministic form of the Thompson Method called BGD algorithm which shows better performance than the randomized algorithms BGP and BGS. Note that the real time computation of $P(\theta^A > \theta^B)$ is not only useful for TAB but many other areas. A way to utilize this approach is to draw samples from the subjective probability distributions of θ^A, θ^B and select an arm corresponding to the larger outcome [9, 8, 24], which is known as Thompson Sampling.

The key **advantages** of Thompson Method based Beta Bayesian algorithms are: 1) Unlike many other algorithms, Beta Bayesian algorithms yield good results over whole range of reward probabilities including very low probabilities of the order of 0.001 or less. 2) Beta Bayesian algorithms are less sensitive to the prior parameters initialized during the start of the experiment.

4.2 Two armed Bandit Policy

In this section, we present three different types of two armed bandit strategies which use the Bayesian inference based Beta distribution, two of them are randomized strategies while the third one is deterministic.

We assume that we are given two arms with reward probabilities θ^A and θ^B. Suppose after n trials, each arm has been played a certain number of times such that r_A successes are received in arm A and r_B successes are received in arm B and the trials at which these successes are received are $X_A = k_1, k_2, k_3, ..., k_{rA}$ and $X_B = k_1{}', k_2{}', ..., k_{rB}{}'$ respectively. We could express these series of Bernoulli trials in two forms: 1) Geometric Distribution 2) Binomial Distribution.

In terms of Geometric distribution, we express θ^A and θ^B in the following form. Here α_0 and β_0 are the prior parameters.

$$\theta^A \sim Beta(\alpha^A, \beta^A) \tag{4.1}$$

$$\theta^B \sim Beta(\alpha^B, \beta^B) \tag{4.2}$$

where,

$$\alpha^A = \alpha_0 + r_A \tag{4.3}$$

$$\alpha^B = \alpha_0 + r_B \tag{4.4}$$

$$\beta^A = \beta_0 + \Sigma_{i=1}^{r_A}(k_i - 1) \tag{4.5}$$

$$\beta^B = \beta_0 + \Sigma_{i=1}^{r_B}(k_i{}' - 1) \tag{4.6}$$

In terms of Binomial distribution, we say that if a success is received at n^{th}

trial, α becomes,

$$\alpha_n = \alpha_{n-1} + 1 \tag{4.7}$$

or if a failure is received at n^{th} trial, β is updated to,

$$\beta_n = \beta_{n-1} + 1 \tag{4.8}$$

For both the above distributions, after s successes and r failures, the parameters of Beta distribution become $(\alpha_0 + s,\ \beta_0 + r)$.

$$\theta \sim Beta(\alpha_0 + s, \beta_0 + r) \tag{4.9}$$

Below we describe how we use the above model to formulate three strategies - BGP, BGS, BGD.

4.2.1 Beta Geometric Probabilistic (BGP)

BGP algorithm is described in Alg. 11. According to the algorithm, at the start of the experiment, we play both arms A and B with equal probabilities (0.5) until a success is received in any of the arms. We select Arm A with probability $P(\theta^A > \theta^B)$ and arm B with $1 - P(\theta^A > \theta^B)$. If an arm k (either A or B) is the selected arm, then arm k is played, and on receiving a success $\alpha^k = \alpha^k + 1$ is updated whereas on receiving failure $\beta^k = \beta^k + 1$. *BGP* algorithm only computes $P(\theta^A > \theta^B)$ for the arms after a success is received which makes it computationally less intensive.

Algorithm 8 Algorithm: BGP

Initialize α_0, β_0, $P(\theta^A > \theta^B) = 0.5$.

loop

Select arm A with probability $P(\theta^A > \theta^B)$ and arm B with probability $1 - P(\theta^A > \theta^B)$.

Play the selected arm i.

if success is received in arm i **then**

Update the values for $\alpha^i = \alpha^i + 1$.

Recompute the $P(\theta^A > \theta^B)$.

else

Update the values for $\beta^i = \beta^i + 1$.

end if

end loop

4.2.2 Beta Geometric Sampling (BGS)

BGS algorithm is also known in literature as Thompson Sampling and has been discussed in [9, 24]. It is described in Algo. 12. Instead of actually computing the values of $P(\theta^A > \theta^B)$, we samples the values π^A and π^B from the probability distributions of θ^A and θ^B, and play the arm with higher value of the sample.

Algorithm 9 Algorithm: BGS

Initialize α_0, β_0.

loop

Draw a value of π^k randomly from Beta$(\alpha_k, \beta_k) \forall k \in \{A, B\}$.

Arrange the samples in decreasing order.

Select the arm i s.t $\pi^i = max_k(\pi_k)$, $\forall k \in \{A, B\}$.

Pull arm i.

if success is received in arm i **then**

Update the values for $\alpha^i = \alpha^i + 1$.

else

Update the values for $\beta^i = \beta^i + 1$.

end if

end loop

4.2.3 Beta Geometric Deterministic (BGD)

BGD algorithm is a deterministic form of the Beta-Geometric algorithms and is described in Alg. 10 . In the algorithm, n_A and n_B represent the number of

times arm A and B have been pulled respectively. At the start of the algorithm, we initialize n_A and n_B as 1 for both arms. The basic idea used in this algorithm is the same as in the BGP and BGS algorithms with the key difference being the use of the ratio $\frac{n_A}{n_A+n_B}$, which makes it deterministic. If $P(\theta^A > \theta^B) \geq \frac{n_A}{n_A+n_B}$, the arm we pull arm A, otherwise arm B.

In Fig. 4.1, 4.2 and 4.3, we plot the values of $P(\theta^A > \theta^B)$ and $\frac{n_A}{n_A+n_B}$ against the number of trials for a single experiment for different sets of probabilities, (θ^A θ^B) = (0.4,01), (0.009,0.006) and (0.004, 0.001) respectively. In Fig. 4.1, $(\theta^A, \theta^B) = (0.4, 0.1)$, the plot shows convergence to 1 for $P(\theta^A > \theta^B)$, however it is not the case in Fig. 4.2 and 4.3. In Fig. 4.2, it can be seen that although the value of probability $P(\theta^A > \theta^B)$ does not reach 1 but still towards the end, the best arm is tried deterministically which is different from the case of the random algorithm in which the best arm would have been tried with probability less than 1. The graph also depicts the self-correcting nature of the algorithm.

4.2.4 Computation of $P(\theta^A > \theta^B)$

Below we show how to compute the value of $P(\theta^A > \theta^B)$ in real time for Beta-Geometric distributions. We do this computation at each success using the current values of distribution parameters $\{(\alpha^A, \beta^A), (\alpha^B, \beta^B)\}$.

$$P(\theta^A > \theta^B) \tag{4.10}$$

$$= \int_0^1 p(\theta^A) \int_0^{\theta^A} p(\theta^B) \mathrm{d}\theta^B \mathrm{d}\theta^A \tag{4.11}$$

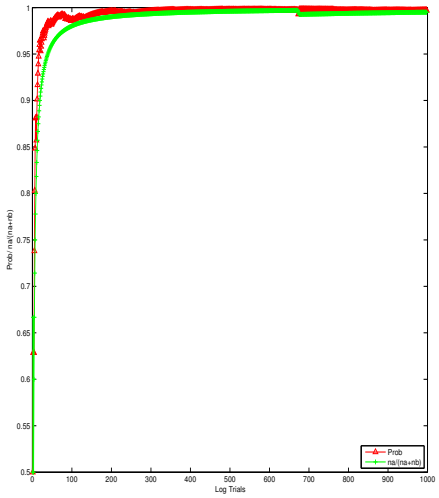

Figure 4.1: Plot for $P(\theta^A > \theta^B)$ vs. trials and $\frac{n_A}{n_A+n_B}$ vs. trials for one experiment consisting of 1000 trials for $\theta^A = 0.4$ and $\theta^B = 0.1$ with % best arm played=99.5%.

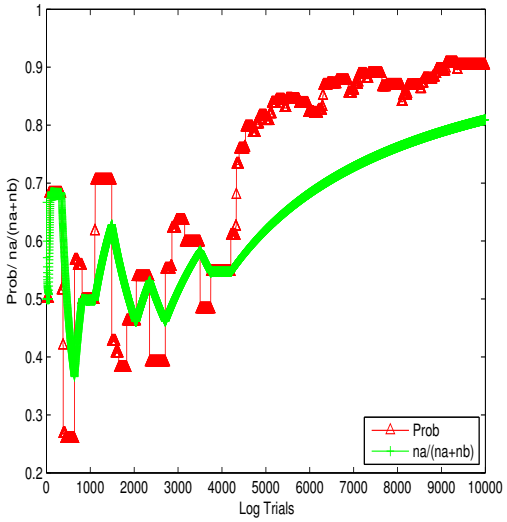

Figure 4.2: Plot for $P(\theta^A > \theta^B)$ vs. trials and $\frac{n_A}{n_A+n_B}$ vs. trials for one experiment consisting of 10000 trials for $\theta^A = 0.009$ and $\theta^B = 0.006$ with % best arm played=80.61% and regret = 14 .

Algorithm 10 Algorithm: BGD

Initialize α_0, β_0, $P(\theta^A > \theta^B) = 0.5$, $n^A = 1$, $n^B = 1$.

loop

if $(P(\theta^A > \theta^B) \geq \frac{n^A}{n^A + n^B})$ **then**

Pull arm A.

if arm A receives success **then**

Update the values for $\alpha^A = \alpha^A + 1$ and compute $P(\theta^A > \theta^B)$.

else

Update the values for $\beta^A = \beta^A + 1$.

end if

else

Pull arm B.

if arm B receives success **then**

Update the values for $\alpha^B = \alpha^B + 1$ and compute $P(\theta^A > \theta^B)$.

else

Update the values for $\beta^B = \beta^B + 1$.

end if

end if

end loop

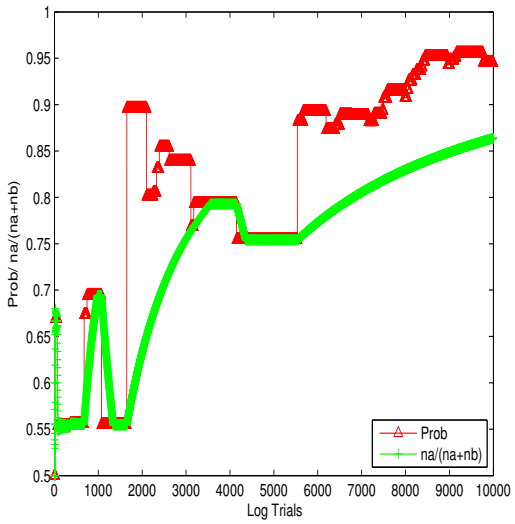

Figure 4.3: Plot for $P(\theta^A > \theta^B)$ vs. trials and $\frac{n_A}{n_A+n_B}$ vs. trials for one experiment consisting of 10000 trials for $\theta^A = 0.004$ and $\theta^B = 0.001$ with % best arm played=86.38% and regret =1.

$$= \frac{\alpha^A + \beta^A - 1}{\alpha^A + \beta^A + \alpha^B + \beta^B - 2} \times \Sigma_{j=\alpha^B}^{\alpha^B+\beta^B-1} \frac{\binom{\alpha^B+\beta^B-1}{j}\binom{\alpha^A+\beta^A-2}{\alpha^A-1}}{\binom{\alpha^B+\beta^B+\alpha^A+\beta^A-3}{\alpha^A+j-1}} \tag{4.12}$$

$$= \frac{\alpha^A + \beta^A - 1}{\alpha^A + \beta^A + \alpha^B + \beta^B - 2} \times \Sigma_j \frac{\binom{m}{j}\binom{n}{k}}{\binom{m+n}{j+k}} \tag{4.13}$$

$$m = \alpha^B + \beta^B - 1, n = \alpha^A + \beta^A - 2, k = \alpha^A - 1 \tag{4.14}$$

Each term in the summation can be simplified as,

$$\frac{\binom{m}{j}\binom{n}{k}}{\binom{m+n}{j+k}} = (\prod_i \frac{j+k-i}{k-i}) \times A \times B \tag{4.15}$$

where

$$A = \frac{n}{(n+m)} \frac{(n-1)}{(n+m-1)} .... \frac{(n-k+1)}{(n+m-k+1)} \tag{4.16}$$

$$B = \frac{m}{(m+n-k)} \frac{(m-1)}{(m+n-k)} ... \frac{(m-j+1)}{(m+n-k-j+1)} \tag{4.17}$$

Further, each term in the summation can be computed from its previous term,

$$\frac{\binom{m}{j+1}\binom{n}{k}}{\binom{m+n}{j+k+1}} = \frac{m-j}{j+1} \times \frac{k+1}{m+n-j-k} \times \frac{\binom{m}{j}\binom{n}{k}}{\binom{m+n}{j+k}} \tag{4.18}$$

Simplifying the formula in the above manner ensures that the number of terms in the numerator and denominator are equal. To ensure that overflow does not occur in the above computation, instead of multiplying the numerators and denominators separately, we first divide on a term by term basis and then multiply each result. The above simplifications ensure a real time computation of the quantity $P(\theta^A > \theta^B)$, thereby eliminating the need of factorial look-up suggested in [33].

4.3 Comparative Analysis

We perform comparative analysis on two types of events - rare and non-rare events. Table 4.1 shows the parameter settings for different experiments on 5 simulated datasets. We compare our *Beta Bayesian algorithms - BGD, BGP* and *BGS* with UCB-tuned, EP-d and EP-n, for both rare and non-rare events for simulated dataset as well as for rare events for datasets obtained from real advertisements.

Note that these models require the prior parameters, α_0 and β_0 The priors α_0 and β_0 can be initialized depending on the problem domain. Fig. 4.4 shows the Beta-distribution curves for different priors. As seen, the priors can be initialized in multiple ways, however in the absence of any domain knowledge we initialize as $\alpha_0 = 2$ and $\beta_0 = 2$ so that the curve is unimodal and spreads from 0 to 1 with mean at 0.5 (non-informative case). But on the other hand, if the domain knowledge is available then the priors can be initialized to be of the same order as θ^A and θ^B.

4.3.1 Non-rare Events

Non-rare events are defined as events when probability of success (> 0.01). For non-rare events, we compare the performance for $(\theta^A, \theta^B) = ((0.7, 0.3), (0.4, 0.1))$.

Each experiment is performed using independently generated random variables and averaged over 100 runs. For each experiment, we show a plot of % Best arm played and regret on a semi-logarithmic scale. Figs. 4.5 and 4.6 for $(\theta^A, \theta^B) = (0.4, 0.1)$, UCB-tuned shows the highest percentage of best arm played with Beta Bayesian algorithms performing slightly worse. EP-d algorithm performs the worst. All algorithms except EP-d perform nearly same on regret.

Figs. 4.7, 4.8 for $(\theta^A, \theta^B) = (0.7, 0.3)$, BG-d performs the best with UCB-tuned performing slightly worse. EP-d algorithm performs the worst and shows unstable behavior. All algorithms except EP-d perform same on the regret. EP-d and EP-n are amongst the bottom performers.

4.4 Rare Events

In display advertising and many other applications clicks are rare events. In this section of my dissertation, we discuss as to why the theoretical bounds in the algorithms UCB-1 and UCB-2 will not work at low probabilities. Theorem 1 in [2] proves that the expected value of the number of times the suboptimal arm is played is bounded by

$$E[T_i(n)] \leq \frac{8lnn}{\Delta_i^2} + \frac{\pi^2}{3} + 1 \tag{4.19}$$

$\hat{\theta^A}$	$\hat{\theta^B}$	BGD	BGP	BGS	UCB-T	EP-d	EP-n
0.7	0.3	α_0=2,β_0=2; α_0=2,β_0 = 8	α_0=2,β_0=2	α_0=2, β_0=2	NA	ϵ_0 = .30	c=0.15, d=0.4
0.4	0.1	α_0=2,β_0=2; α_0=2,β_0 = 8	α_0=2,β_0=2	α_0=2, β_0=2	NA	ϵ_0 = .30	c=0.15, d=0.3
0.001	0.004	α_0=2,β_0=2	α_0=2, β_0=2	α_0=2, β_0=2	NA	ϵ_0 = 20	c=10, d=0.003
0.009	0.006	α_0=2, β_0=2	α_0=2, β_0=2	α_0=2, β_0=2	NA	ϵ_0 = 20	NA
0.0055	0.0045	α_0=2, β_0=2	α_0=2, β_0=2	α_0=2,β_0=2	NA	ϵ_0 = 20	NA

Table 4.1: Table shows the initial parameters for all algorithms used in the simulated dataset. UCB-T (UCB-tuned) does not require any prior.

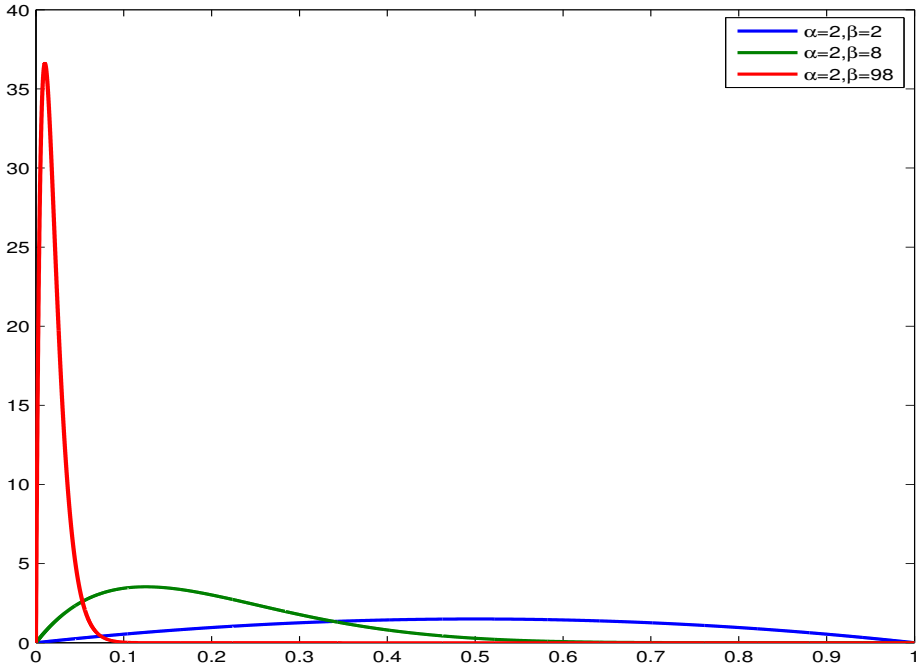

Figure 4.4: Comparing beta distribution for three sets of alpha and beta parameters.

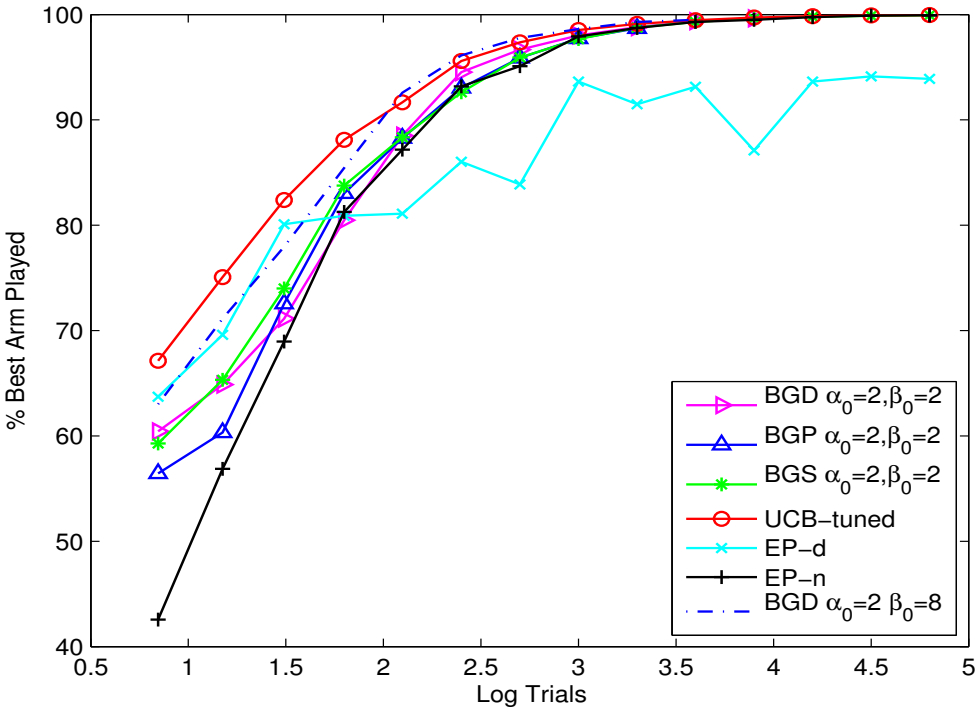

Figure 4.5: % Best Arm Played when θ^A=0.4 and $\theta^B = 0.1$

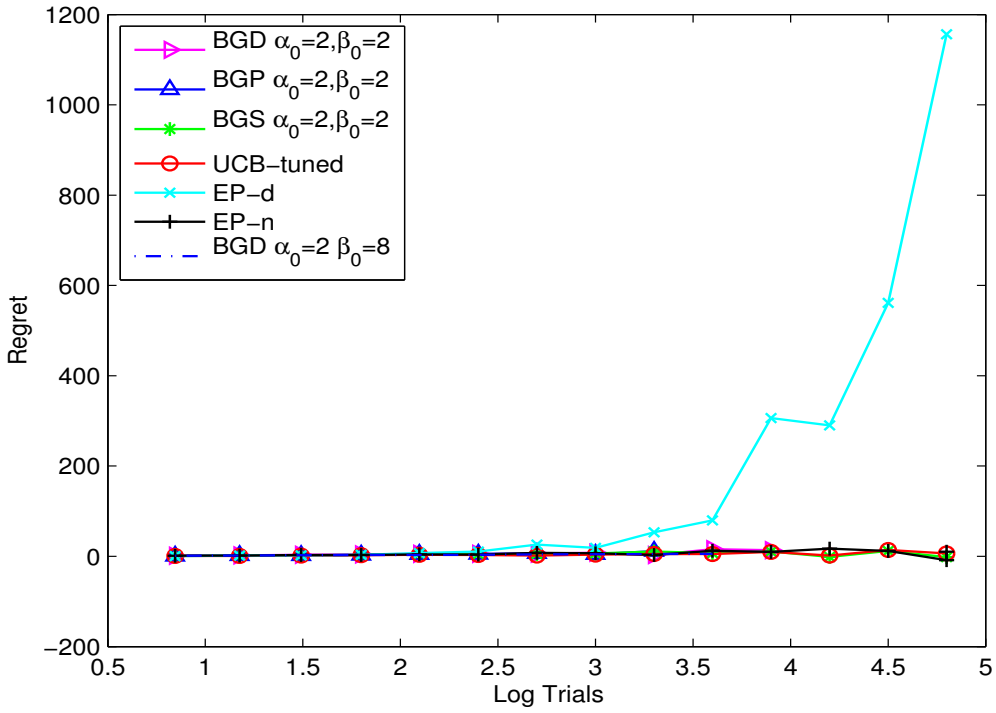

Figure 4.6: Regret when θ^A=0.4 and $\theta^B = 0.1$

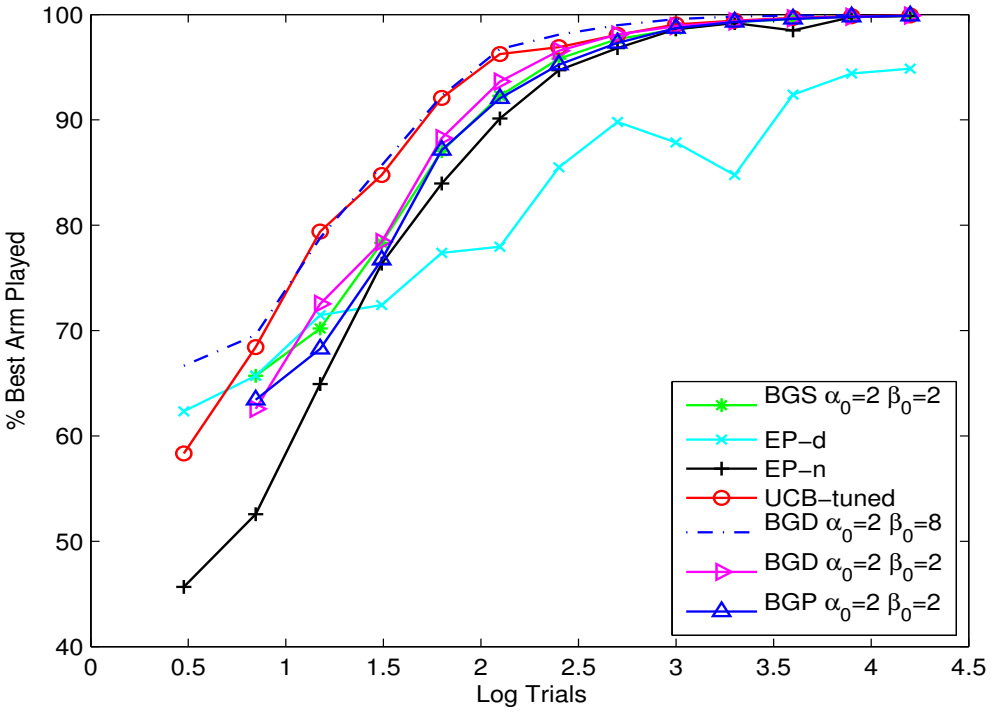

Figure 4.7: % Best Arm Played when θ^A=0.7 and $\theta^B = 0.3$

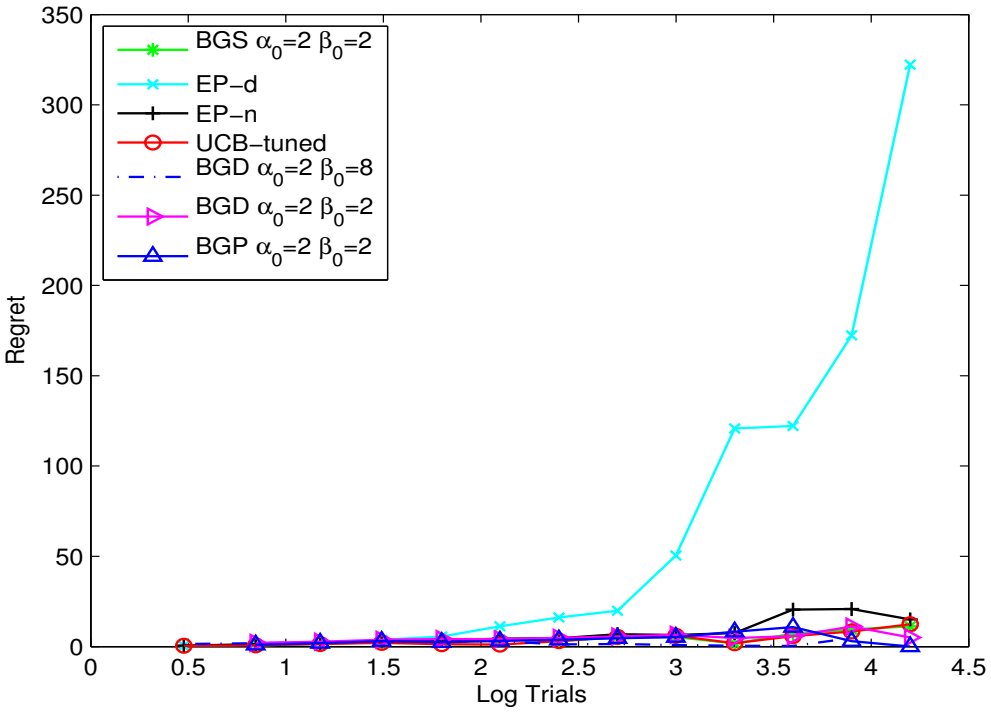

Figure 4.8: Regret when θ^A=0.7 and $\theta^B = 0.3$

Δ_i	n	$E[T_i(n)]$ UCB-1	n	$E[T_i(n)]$ UCB-2
0.009	10K	395K	500K	3.360E+8
0.008	10K	500K	500K	4.253E+8
0.007	10K	653K	500K	5.555E+8
0.006	10K	889K	500K	7.561E+8
0.005	10K	1280K	500K	1.0888E+9
0.004	10K	2000K	500K	1.70126E+9
0.003	10K	3555K	500K	3.02445E+9
0.002	10K	8000K	500K	6.80502E+9
0.001	10K	32000K	500K	2.72201E+10

Table 4.2: Expected Number of times sub optimal arms are played

$$\Delta_i = \hat{\theta^*} - \hat{\theta}_i \tag{4.20}$$

Table 4.2 shows the expected value of the number of times suboptimal arms will be played for probabilities ranging from 0.001 - 0.009. The above numbers clearly show that for rare events the upper bound on the expected number of times suboptimal arms are played are very very weak.

Similarly, the formula for the expected number of trials for the suboptimal arm of UCB-2 algorithm as proved in [2] is,

$$E[T_i(n)] \leq \frac{(1+\alpha)(1+4\alpha)ln(2en\Delta_i^2)}{2\Delta_i^2} + \frac{c_\alpha}{\Delta_i^2} \tag{4.21}$$

where

$$n \geq \max(\frac{1}{2\Delta_i^2}) \tag{4.22}$$

and

$$c_\alpha = 1 + \frac{(1+\alpha)e}{\alpha^2} + \frac{1+\alpha}{\alpha}^{(1+\alpha)}[1 + \frac{11(1+\alpha)}{5\alpha^2 ln(1+\alpha)}] \tag{4.23}$$

Table 4.2 shows the expected value of the number times the suboptimal arms will be played Δ_i for a two-armed case UCB-2. The value of n is to be taken according to Eqn. 4.22, hence we choose the value 500K which is suitable for all Δ_i in the range $0.001 - 0.009$ and we set α to 0.001. The above computations clearly show that for the reward probabilities of the order of 1 in 100 the above bounds are very weak, and they get worse as the values of Δ_i is further reduced since $E[T_i(n)] \propto \frac{1}{\Delta_i^2}$ in both cases.

4.5 Experiments- Rare Events

In this section, we present an extensive set of experiments to show that the Beta Bayesian algorithms show a good performance for rare events. We perform experiments on simulated datasets for probabilities ranging from 0.001-0.009. We set the values of c, ϵ_0, d according to Table 4.1.

We also evaluate the performance of our methods on two real datasets obtained from the logs of a dynamic display advertising company. These datasets were collected from the data of real advertisements, hence specific information cannot be

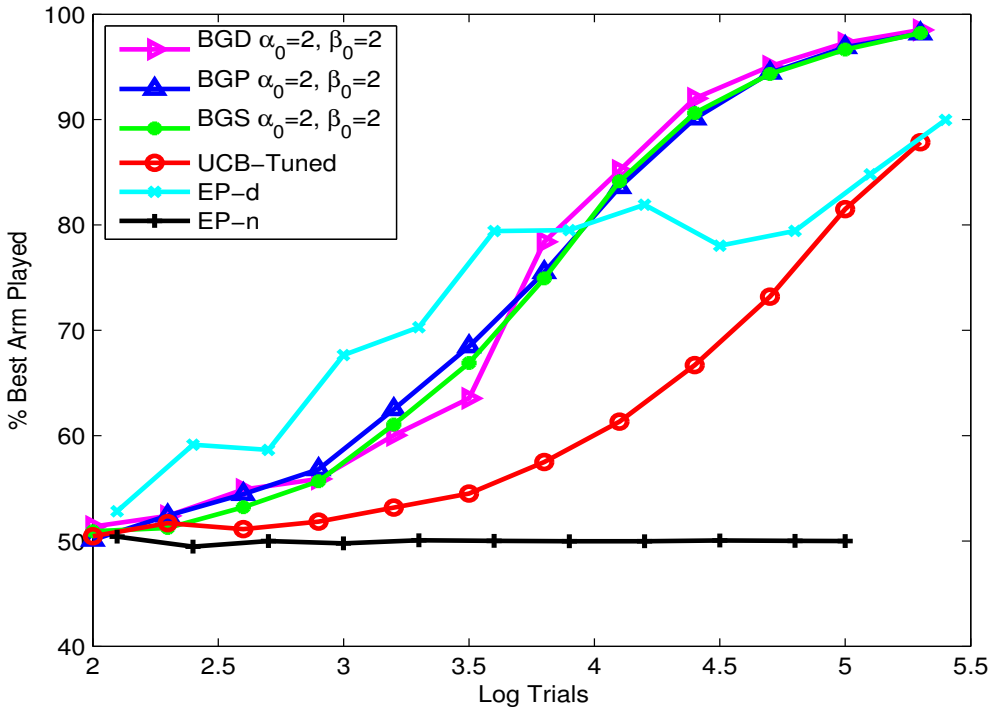

Figure 4.9: %Best Arm Played when θ^A=0.004 and $\hat{\theta^B} = 0.001$

made public, we call these two datasets as *dataset1*, *dataset2*. In our setting, the publishing urls of the advertisements are changed on a daily basis hence to ensure stationary distribution over the reward, we only use one day worth of data consisting of around $300,000 - 400,000$ total impressions.

Each experiment on simulated dataset is run 100 times, and the average & standard deviation obtained are reported in the graphs on a semi-logarithmic scale (base 10).

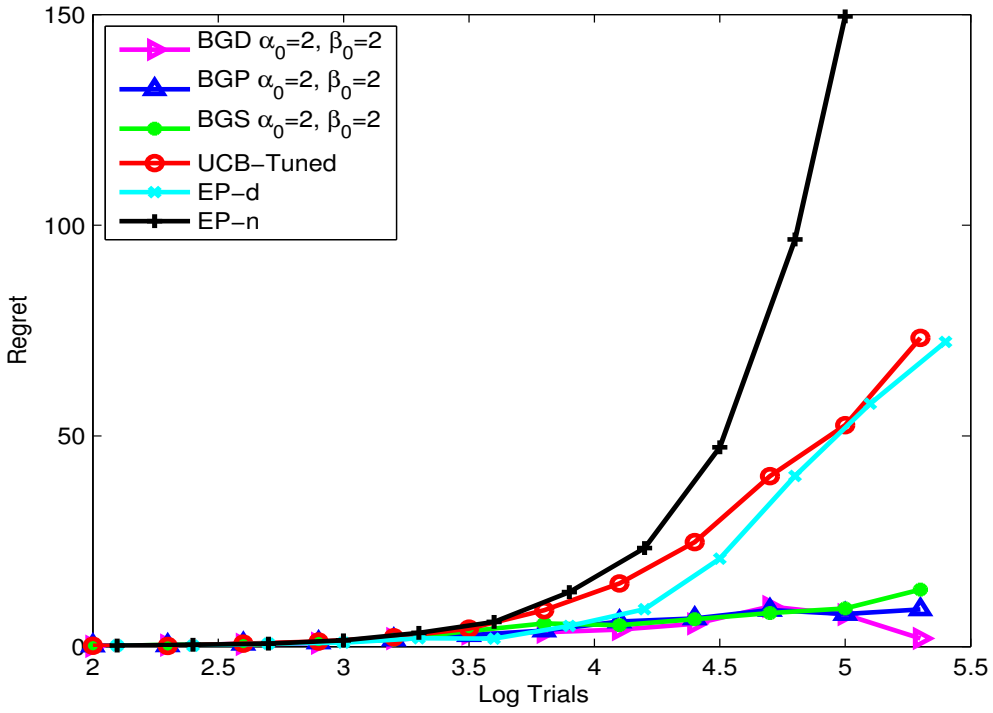

Figure 4.10: %Regret when θ^A=0.004 and $\theta^B = 0.001$

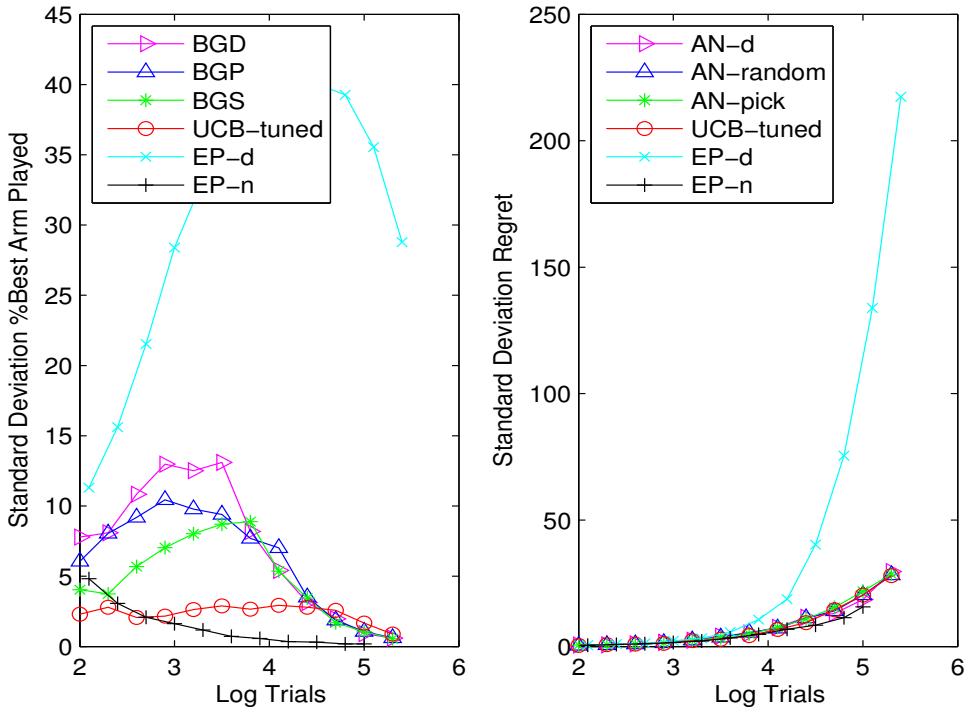

Figure 4.11: Standard Deviation in %Best Arm Played and Regret when θ^A=0.004 and $\theta^B = 0.001$

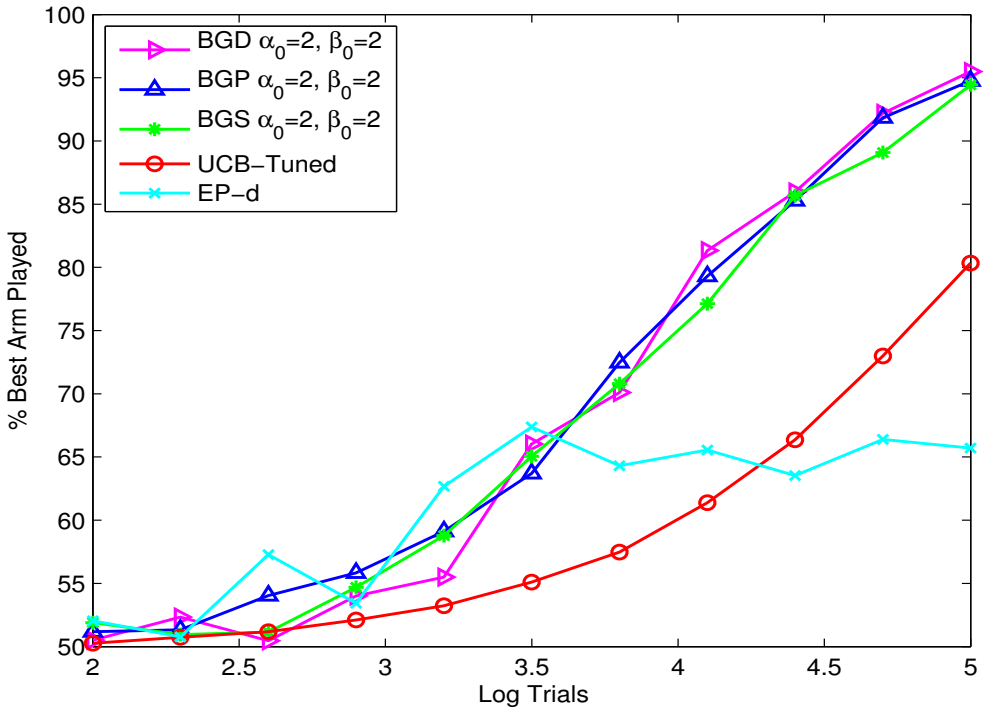

Figure 4.12: %Best Arm Played and Regret when θ^A=0.009 and θ^B = 0.006

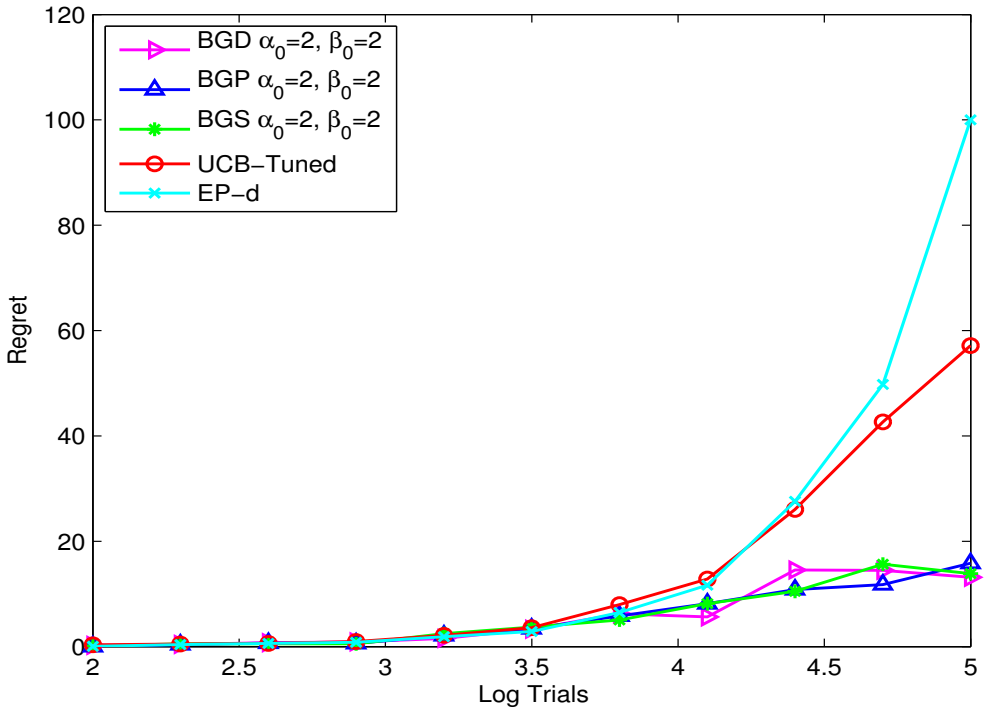

Figure 4.13: Regret when θ^A=0.009 and θ^B = 0.006

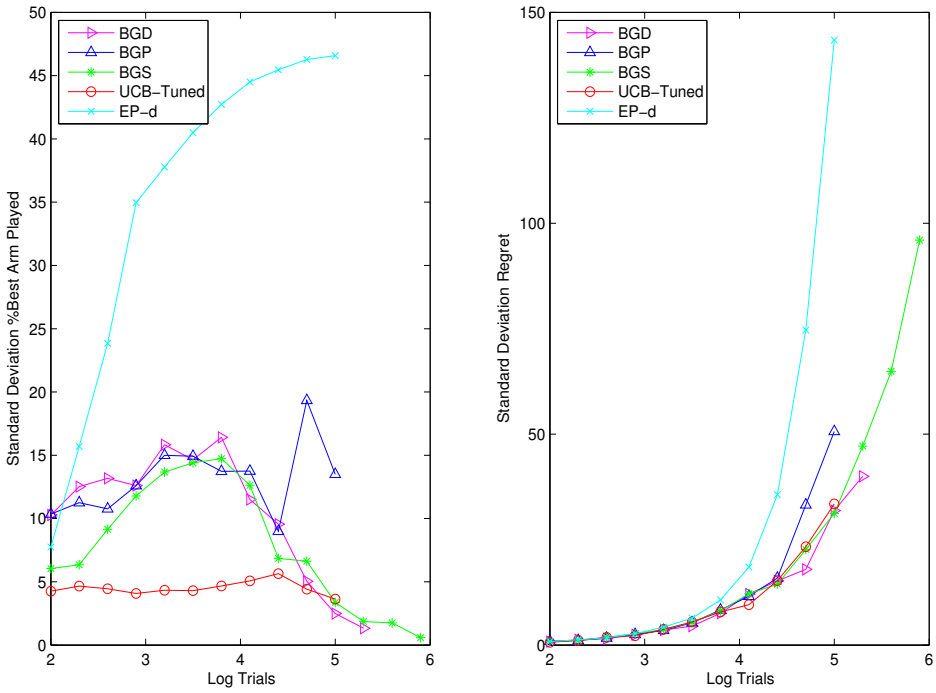

Figure 4.14: Standard Deviation in %Best Arm Played and Regret θ^A=0.009 and $\theta^B = 0.006$

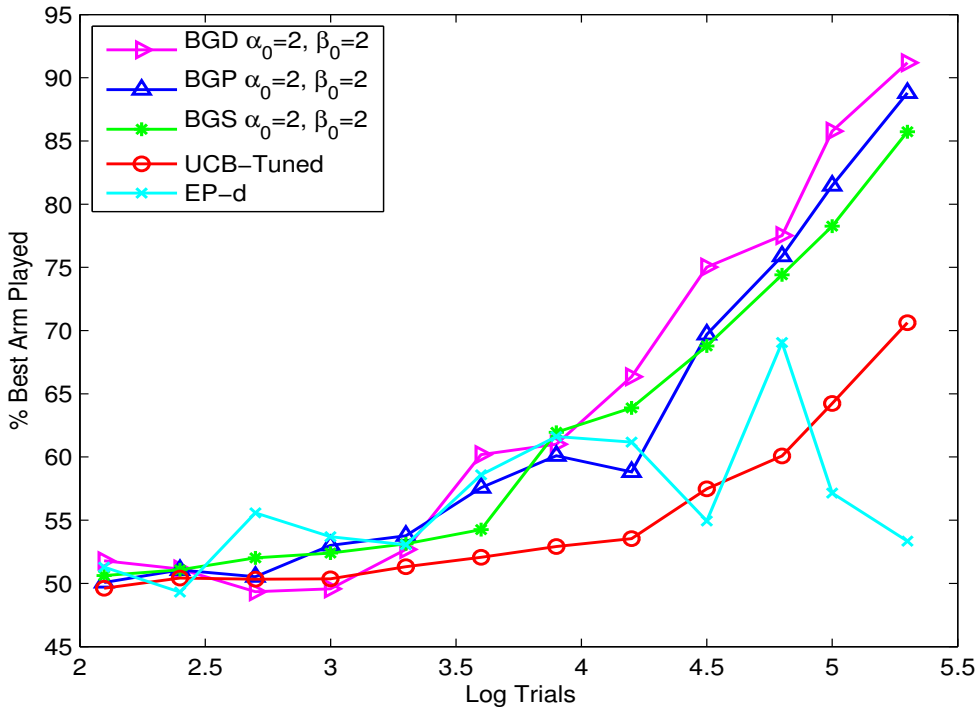

Figure 4.15: %Best Arm Played when θ^A=0.0055 and θ^B = 0.0045

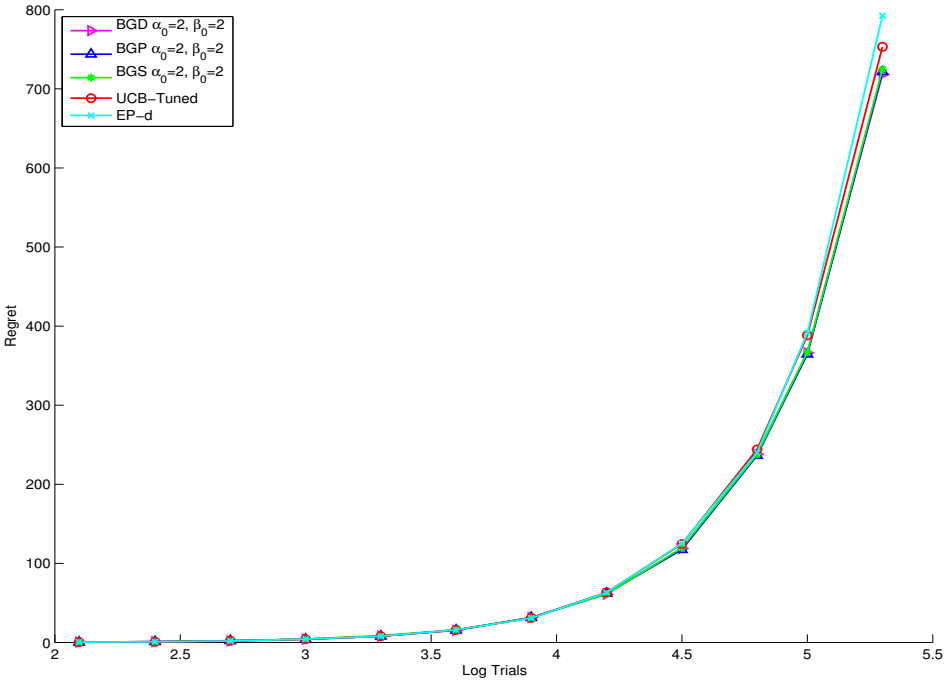

Figure 4.16: Regret when θ^A=0.0055 and θ^B = 0.0045

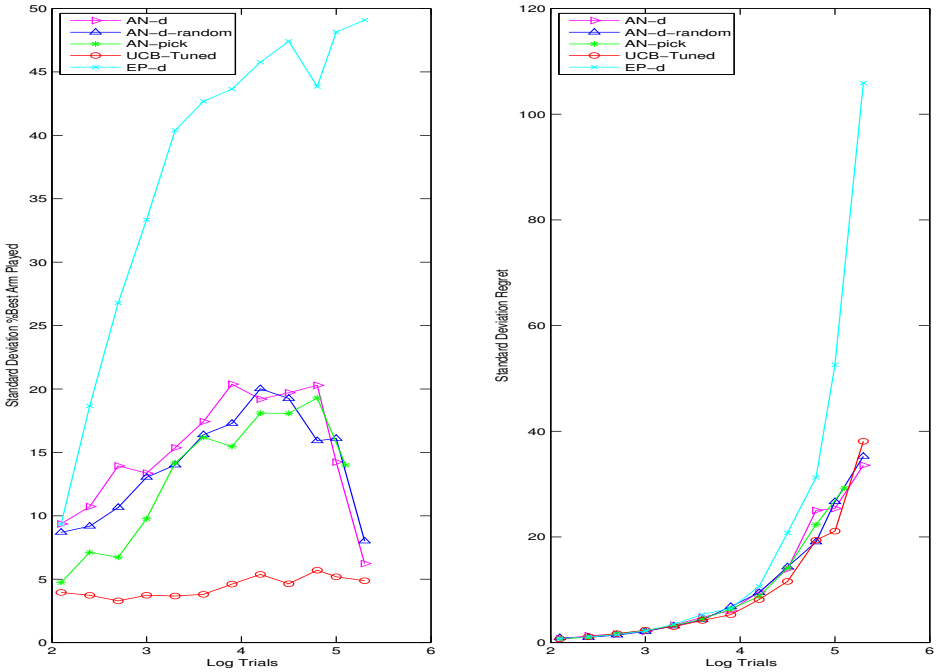

Figure 4.17: Standard Deviation in %Best Arm Played and Regret θ^A=0.0055 and $\theta^B = 0.0045$

4.5.1 Simulated Dataset for Rare Events

We simulate the data for rare events using the bernoulli trials for 3 distributions as shown in table. 4.1.

The first set of experiment was performed for probabilities $(\theta^A, \theta^B) = (0.001, 0.004)$. The graphs for regret and % best arm played are shown in Figs.4.9 and 4.10 respectively. Fig. 4.11 shows the standard deviation in the regret and % best arm played for the above case.

The graphs show that the Beta Bayesian algorithms perform the best both in regret obtained and % of times the best arm is played. UCB-Tuned algorithm does not perform well for rare-events and EP-d shows unstable behavior. EP-n performs poorly due to the d^2 factor in the denominator $\frac{ck}{d^2 n}$ [2] , where $0 \leq d \leq min(\hat{\theta}_i - \hat{\theta^*})$, which makes it try the worst arm more than the best arm. Hence we do not consider EP-n in our future experiments [2]. The standard deviation of the Beta Bayesian algorithms is higher initially but start decreasing rapidly after about $10,000$ trials for the best arm played. The standard deviation in regret is the similar for all but EP-d and EP-n.

The second experiment is performed for $(\theta^A, \theta^B) = (0.009, 0.006)$ is shown in Fig. 4.12, 4.13. Beta Bayesian algorithms show the best performance in this case also. Moreover, the standard deviation for Beta Bayesian algorithms both in regret and best arm played is much less as compared to EP-d algorithm as shown in Fig. 4.14.

In the third experiment, the probabilities $(\theta^A, \theta^B) = (0.0055, 0.0045)$ are used

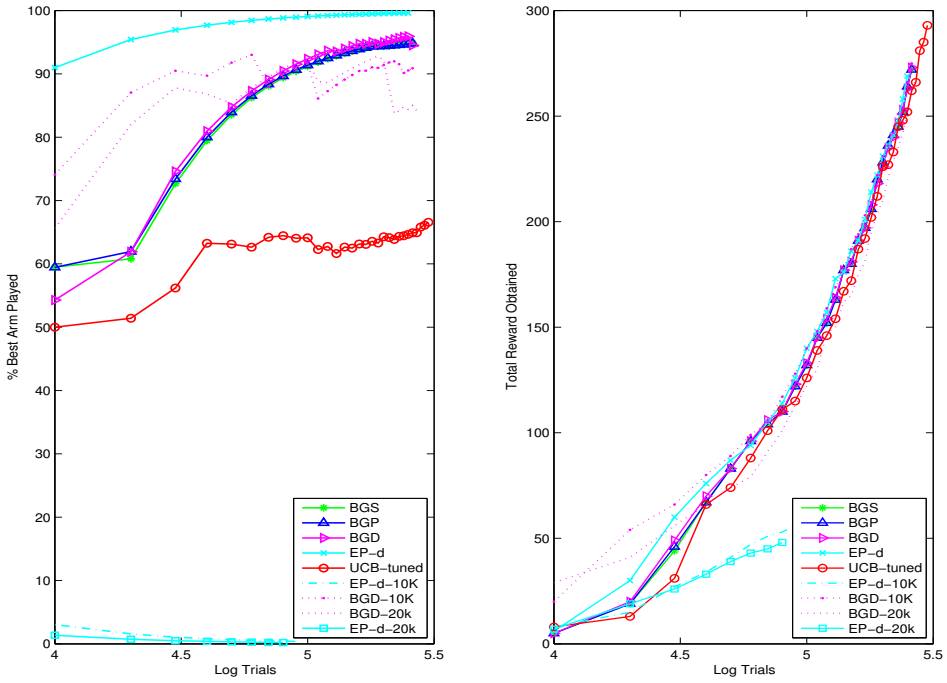

Figure 4.18: % Best Arm Played and Total Reward Obtained for *dataset1*

to evaluate the performance of the algorithms as shown in Figs. 4.15, 4.16 and 4.17. It is considered a weak case since the two probabilities are very close. The results obtained are similar to the previous two cases and BG-algorithms perform better other algorithms. Since the probabilities are very close, regret is almost same in all cases.

4.5.2 Real Dataset : Online Display Advertising

We use the real datasets available from 2 advertisers for our experiments.

The first dataset is called *dataset1* where we use one day worth of log for an advertiser and use click probabilities as the measure of performance. The total number

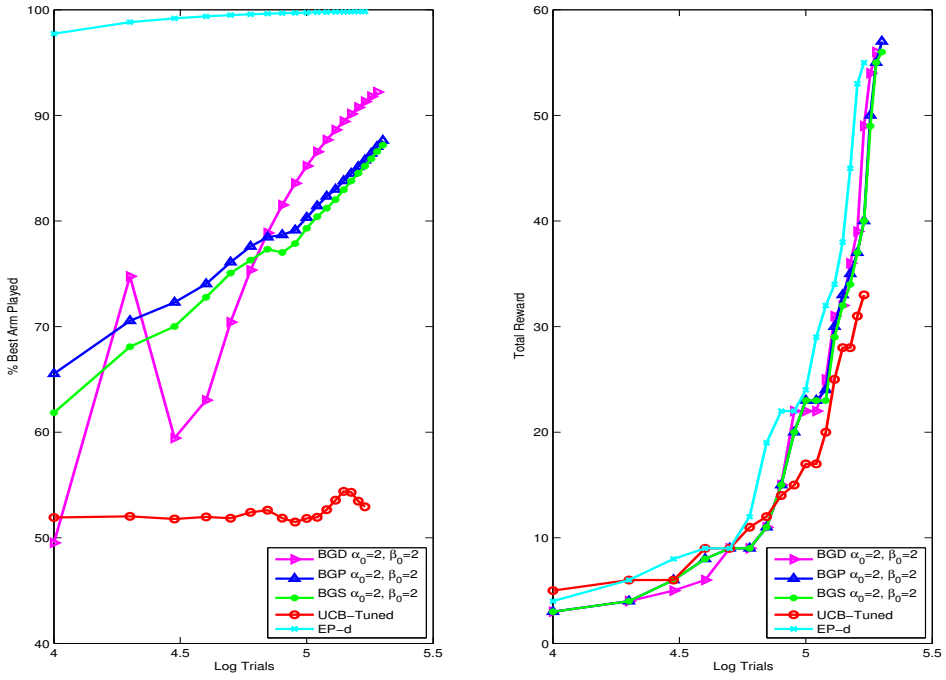

Figure 4.19: % Best Arm Played and Total Reward Obtained for *dataset2*

of impressions for both the advertisements was around 350K, with advertisement 1 having 150K and advertisement 2 having 200K impressions. Fig. 4.18 shows the plots of the graphs for 5 different algorithms.

For EP-d and Beta Bayesian algorithms, we took different starting points for *dataset1* such as first 10k values ignored, 20K ignored, ..., etc. and notice a flip-flop behavior in EP-d algorithm. This shows that EP-d algorithm is sensitive to the initial values of the samples while the BGD algorithm is stable. BGD algorithm leads in the total reward obtained. The second set of experiments was done using *dataset2* is shown in Fig. 4.19. The total number of impressions for both the advertisements was around 280K, with advertisement 1 having 180K and advertisement 2 having 100K impressions. Beta Bayesian algorithms outperforms UCB-tuned algorithm both in the total reward obtained and the % of times the best arm is played. EP-d algorithm performs the best but again EP-d is an unstable algorithm and has a tendency to converge based on the initial samples leaving no scope for self-correction.

4.6 Conclusion

Multi-armed bandit has been a problem of great interest in the Internet domain. Recently, a lot of attention has been given to the application of UCB algorithm in the area of online advertising and Internet news article recommendation [25, 6, 21]. This chapter brings attention to the Bayesian learning based Thompson methods which work better for all range of reward probabilities.

The work presented here is the study of Two Armed Bandit problem using

Thompson Method and applying the solution to the problem of Online Display Advertising for which the probabilities of success are rather small. The empirical results show that the **Beta Bayesian algorithms - BGD, BGP, BGS** perform better than all other commonly used algorithms including UCB-tuned and epsilon-greedy for solving this problem.

Chapter 5

Multi armed Bandit

5.1 Introduction

In the previous chapter, we have presented variants of the Beta Bayesian Models for the case of Two-armed bandits. In this chapter, we advance our work to the problem of generalized case of K-armed Bandits and change the previously introduced algorithms BGP and BGS (TS) for the case of K-arms.

5.1.1 Beta Geometric Probabilistic (BGP)

The generalized form of *BGP algorithm* for the case of K-arms is presented in Algo. 11. As per the algorithm, at the start of the experiment each of the K arms are played with equal probabilities of success until a success is received in any of the arms. The estimate of reward probabilities of each arm k, after n total trials, is modeled as $p(\hat{\theta}_n^k) \sim Beta(\alpha_n^k, \beta_n^k)$ and the state of the system after n trials is denoted by $\{(\alpha_n^1, \beta_n^1), (\alpha_n^2, \beta_n^2), ...(\alpha_n^K, \beta_n^K)\}$.

The probability of any arm k being played at trial n is $P(\theta^k > \theta^1 \wedge \theta^k > \theta^2 \wedge \theta^k > \theta^3 ... \theta^k > \theta^K)$.

$$P(\theta^k > \theta^1 \wedge \theta^k > \theta^2 \wedge \theta^k > \theta^3 ... \theta^k > \theta^f K) \tag{5.1}$$

$$= \int_0^1 p(\theta^k) \int_0^{\theta^k} p(\theta^1) \mathrm{d}\theta^1 \int_0^{\theta^k} p(\theta^2) \mathrm{d}\theta^2 ... \int_0^{\theta^k} p(\theta^K) \mathrm{d}\theta^K \mathrm{d}\theta^K \tag{5.2}$$

$$\tag{5.3}$$

$$\int_0^{\theta^k} p(\theta^1) \mathrm{d}\theta^1 \tag{5.4}$$

$$= \int_0^{\theta^k} \frac{{\theta^1}^{\alpha^1 - 1} (1 - \theta^1)^{\beta^1 - 1} \mathrm{d}\theta^1}{B(\alpha^1, \beta^1)} \tag{5.5}$$

$$= \Sigma_{j=\alpha^1}^{\alpha^1 + \beta^1 - 1} \binom{\alpha^1 + \beta^1 - 1}{j} (\theta^k)^j (1 - \theta^k)^{\alpha^1 + \beta^1 - 1 - j} \tag{5.6}$$

Similarly,

$$\int_0^{\theta^k} p(\theta^2) \mathrm{d}\theta^2 \tag{5.7}$$

$$= \int_0^{\theta^k} \frac{{\theta^2}^{\alpha^2 - 1} (1 - \theta^2)^{\beta^2 - 1} \mathrm{d}\theta^2}{B(\alpha^2, \beta^2)} \tag{5.8}$$

$$= \Sigma_{j=\alpha^2}^{\alpha^2 + \beta^2 - 1} \binom{\alpha^2 + \beta^2 - 1}{j} (\theta^k)^j (1 - \theta^k)^{\alpha^2 + \beta^2 - 1 - j} \tag{5.9}$$

$$P(\theta^k > \theta^1 \wedge \theta^k > \theta^2 \wedge \theta^k > \theta^3 ... \theta^k > \theta^K) = \tag{5.10}$$

$$\int_0^1 \frac{\Gamma(\alpha^k)\Gamma(\beta^k)}{\Gamma(\alpha^k + \beta^k)} (\theta^k)^{\alpha^k - 1} (1 - \theta^k)^{\beta^k - 1} \times \tag{5.11}$$

$$\Sigma_{j=\alpha^1}^{\alpha^1 + \beta^1 - 1} \binom{\alpha^1 + \beta^1 - 1}{j} (\theta^k)^j (1 - \theta^k)^{\alpha^1 + \beta^1 - 1 - j} \times \tag{5.12}$$

$$\Sigma_{j=\alpha^2}^{\alpha^2 + \beta^2 - 1} \binom{\alpha^2 + \beta^2 - 1}{j} (\theta^k)^j (1 - \theta^k)^{\alpha^2 + \beta^2 - 1 - j} \times ... \tag{5.13}$$

$$\Sigma_{j=\alpha_K}^{\alpha_K + \beta_K - 1} \binom{\alpha_K + \beta_K - 1}{j} (\theta^k)^j (1 - \theta^k)^{\alpha_K + \beta_K - 1 - j} \mathrm{d}\theta^k \tag{5.14}$$

The above form leads to an exponential growth in the number of summation terms which makes the above formula intractable.

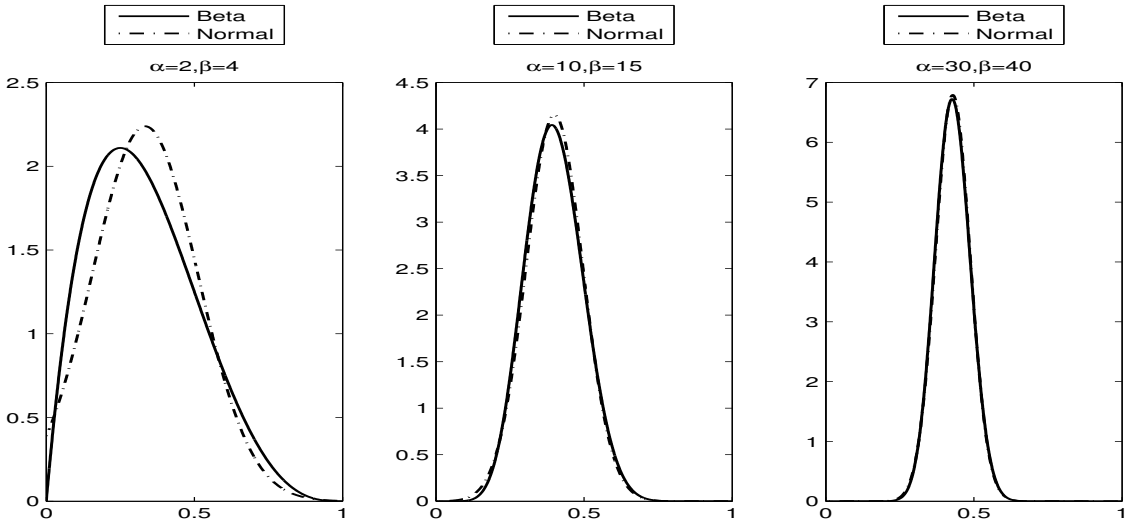

Figure 5.1: Plot comparing Beta and Normal distributions for different parameter values. The curves are overlapping for large (α, β) values.

To make the above computation tractable, Beta distribution is approximated to Normal distribution for calculating the value $P(\theta^k > \theta^1 \wedge \theta^k > \theta^2 \wedge \theta^k > \theta^3 ... \theta^k > \theta^K)$. Fig. 5.1 shows the curves for Beta and Normal distributions for two different values of (α, β). For the first case, when $(\alpha = 2, \beta = 4)$, the two curves look slightly different, but for the case of $(\alpha = 10, \beta = 15)$ and $(\alpha = 30, \beta = 40)$, the curves are almost the same (overlapping). Hence, Normal Distributions could be used as for Beta approximations.

In case of Normal distribution, the incomplete integral can be easily computed using the error function. The error function (erf) is available as a direct look-up, in most programming languages.

$$\int_0^{\theta^1} p(\theta^2) \mathrm{d}\theta^2 = \frac{1}{2}[1 + \mathrm{erf}(\frac{\theta^1 - \hat{\theta}_2}{\hat{\sigma}_2 \sqrt{2}})] \tag{5.15}$$

$$P(\theta^1 > \theta^2) = \int_0^1 \frac{1}{\sqrt{2\pi\hat{\sigma}_1^2}} \mathrm{e}^{(\theta^1-\hat{\theta}_1)/(2\hat{\sigma}_1^2)} \frac{1}{2}[1 + \mathrm{erf}(\frac{\theta^1 - \hat{\theta}_2}{\hat{\sigma}_2\sqrt{2}})]d\theta^1 \tag{5.16}$$

$$\hat{\theta}_1 = \frac{\alpha_1}{\alpha_1 + \beta_1}, \hat{\theta}_2 = \frac{\alpha_2}{\alpha_2 + \beta_2}, ... \tag{5.17}$$

$$\hat{\sigma}_1 = \sqrt{\frac{\alpha_1\beta_1}{(\alpha_1 + \beta_1)^2(\alpha_1 + \beta_1 + 1)}}, \hat{\sigma}_2 = \sqrt{\frac{\alpha_2\beta_2}{(\alpha_2 + \beta_2)^2(\alpha_2 + \beta_2 + 1)}}, ... \tag{5.18}$$

A generalized case for K-arms can be written as,

$$P(\theta^k > \theta^1 \wedge \theta^k > \theta^2 \wedge \theta^k > \theta^3 ... \theta^k > \theta^K) = \tag{5.19}$$

$$= \int_0^1 \frac{1}{\sqrt{2\pi\hat{\sigma}_k^2}} \mathrm{e}^{(\theta^k-\hat{\theta}^k)/(2\hat{\sigma}_k^2)} \frac{1}{2}[1 + \mathrm{erf}(\frac{\theta^k - \hat{\theta}^1}{\hat{\sigma}_1^2\sqrt{2}})] \tag{5.20}$$

$$\frac{1}{2}[1 + \mathrm{erf}(\frac{\theta^k - \hat{\theta}^2}{\hat{\sigma}_2^2\sqrt{2}})] \tag{5.21}$$

$$\frac{1}{2}[1 + \mathrm{erf}(\frac{\theta^k - \hat{\theta}^K}{\hat{\sigma}_K^2\sqrt{2}})]d\theta^k \tag{5.22}$$

5.1.2 Beta Geometric Sampling / Thompson Sampling (TS)

Beta/ Thompson Sampling is a randomized algorithm based on Bayesian modeling of the random distributions of the reward probabilities of the arms in the MAB setting. The algorithm is shown in Algorithm. 12. The estimate of reward probabilities of each arm k, after n total trials, is modeled as $p(\hat{\theta}_n^k) \sim Beta(\alpha_n^k, \beta_n^k)$ and the state of the system after n trials is denoted by $\{(\alpha_1^n, \beta_1^n), (\alpha_2^n, \beta_2^n), ...(\alpha_K^n, \beta_K^n)\}$.

For arm selection at each trial, one sample per arm is drawn from $Beta(\alpha_K^n, \beta_K^n)$ and the arm with the maximum value for the sample is played. The probability of an arm k being played is $P(\theta^k > \theta^1 \wedge \theta^k > \theta^2 \wedge \theta^k > \theta^3 ... \theta^k > \theta^K)$ but there is no need to explicitly compute this value. Theoretical proofs of this method have been discussed in [10, 24].

Algorithm 11 Algorithm: BGP

Initialize α_0^k, $\beta_0^k = 2, P(\theta^k > \theta^1 \wedge \theta^k > \theta^2 \wedge \theta^k > \theta^3 ... \theta^k > \theta^K) = 1/K, \forall k \in K$

loop

Select an arm k with probability $P(\theta^k > \theta^1 \wedge \theta^k > \theta^2 \wedge \theta^k > \theta^3 ... \theta^k > \theta^K), \forall k \in K$

Play the selected arm i.

if success is received in arm i **then**

Update the values for $\alpha^i = \alpha^i + 1$.

Recompute the values of $P(\theta^k > \theta^1 \wedge \theta^k > \theta^2 \wedge \theta^k > \theta^3 ... \theta^k > \theta^K), \forall k \in K$.

else

Update the values for $\beta^i = \beta^i + 1$.

end if

end loop

Algorithm 12 Algorithm: Beta Geometric / Thompson Sampling (TS)

Initialize α_0^k=2, $\beta_0^k = 2$.

loop

Draw a value of π^k randomly from $Beta(\alpha^k, \beta^k) \forall k \in K$.

Arrange the samples in decreasing order.

Select the arm i s.t $\pi^i = max_k(\pi^k)$, $\forall k \in K$.

Pull arm i.

if Arm i is successful **then**

Update the values for $\alpha_n^i = \alpha_{n-1}^i + 1$.

else

Update the values for $\beta_n^i = \beta_{n-1}^i + 1$.

end if

end loop

Experiment/Arms	1	2	3	4	5	6	7	8	9	10
1	0.9	0.8	0.8	0.8	0.7	0.7	0.7	0.6	0.6	0.6
2	0.9	0.6	0.6	0.6	0.6	0.6	0.6	0.6	0.6	0.6
3	0.009	0.008	0.008	0.008	0.007	0.007	0.007	0.006	0.006	0.006
4	0.009	0.006	0.006	0.006	0.006	0.006	0.006	0.006	0.006	0.006

Table 5.1: Table shows the values of reward probabilities for both non-rare and rare events for the experiments done in this dissertation.

The main differences between Thompson Sampling and Beta Geometric Probabilistic Algorithms are 1) In Thompson Sampling, there is no need to explicitly compute the value of $P(\theta^k > \theta^1 \wedge \theta^k > \theta^2 \wedge \theta^k > \theta^3 ... \theta^k > \theta^K)$. 2) Beta Geometric Algorithm is a more passive form since the value of $P(\theta^k > \theta^1 \wedge \theta^k > \theta^2 \wedge \theta^k > \theta^3 ... \theta^k > \theta^K)$ is computed only after each success, although the values of α, β are updated at each step as in the case of Thompson Sampling, hence the selections are made with the same probability until the next success. While in the Sampling algorithm, since a sample is drawn from a new distribution of Beta every time, it a more active technique.

5.2 Comparative Analysis

We perform comparative analysis on two types of events - rare as well as non-rare events. Table. 5.1 shows the different experiments run on 4 simulated datasets. We compare our *Bayesian algorithms - BGP and TS* with UCB-tuned, EP-d and

EP-n for both non-rare and rare events for simulated datasets and show the results in terms of regret obtained and the % Best arm played.

5.2.1 Experiments Non-rare Events

Non-rare events are defined as events when probability of success (> 0.01). For non-rare events, we compare the performance when the reward probabilities are as shown in rows 1 and 2 Table. 5.1.

Each experiment is performed using independently generated random variables and averaged over 100 runs. For each experiment we show a plot of % Best arm played and regret on a semi-logarithmic scale. Fig. 5.4 and Fig. 5.5 show the results for Experiment 1. Thompson Sampling shows the highest percentage of best arm played and lowest regret with BGP algorithm performing slightly worse. EP-d and UCB-Tuned algorithms performs the worst.

In Fig. 5.2 and Fig. 5.3 for the case of Experiment 2, Thompson Sampling performs the best with BGP performing slightly worse. EP-n algorithms performs the worst and shows a lot of unstable behavior.

5.2.2 Experiments Rare Events

We simulate the data for rare events for two sets of reward probabilities as shown in Table. 5.1.

The first set of experiment was performed for probabilities shown in Experiment 3 in Table 5.1 in which the optimal and sub-optimal arms are very close to

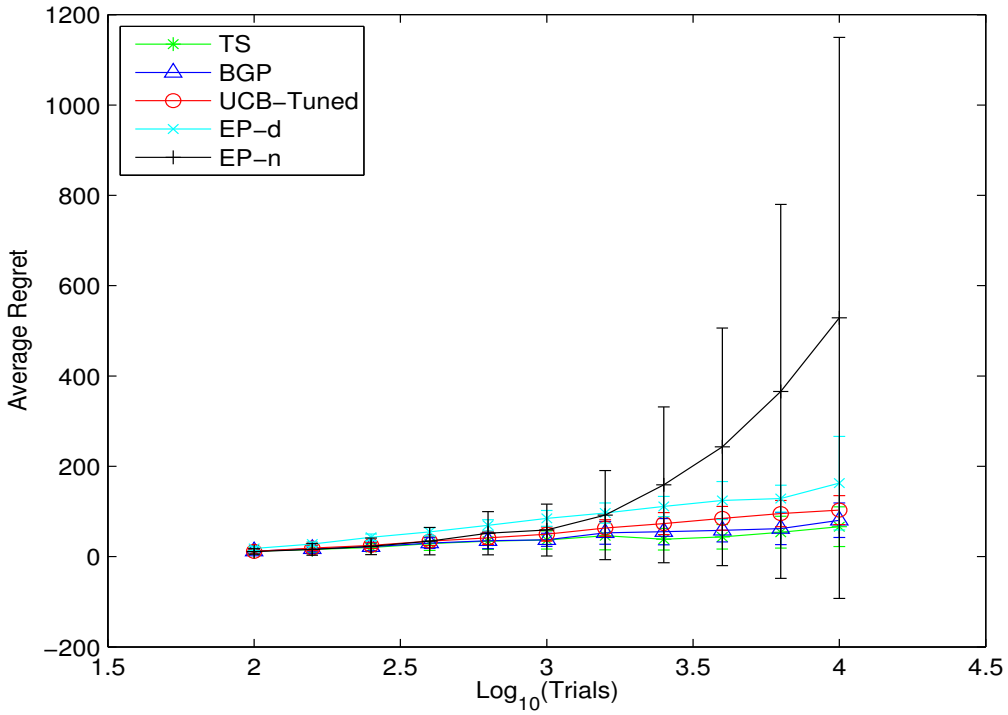

Figure 5.2: Regret for the case of Experiment 1

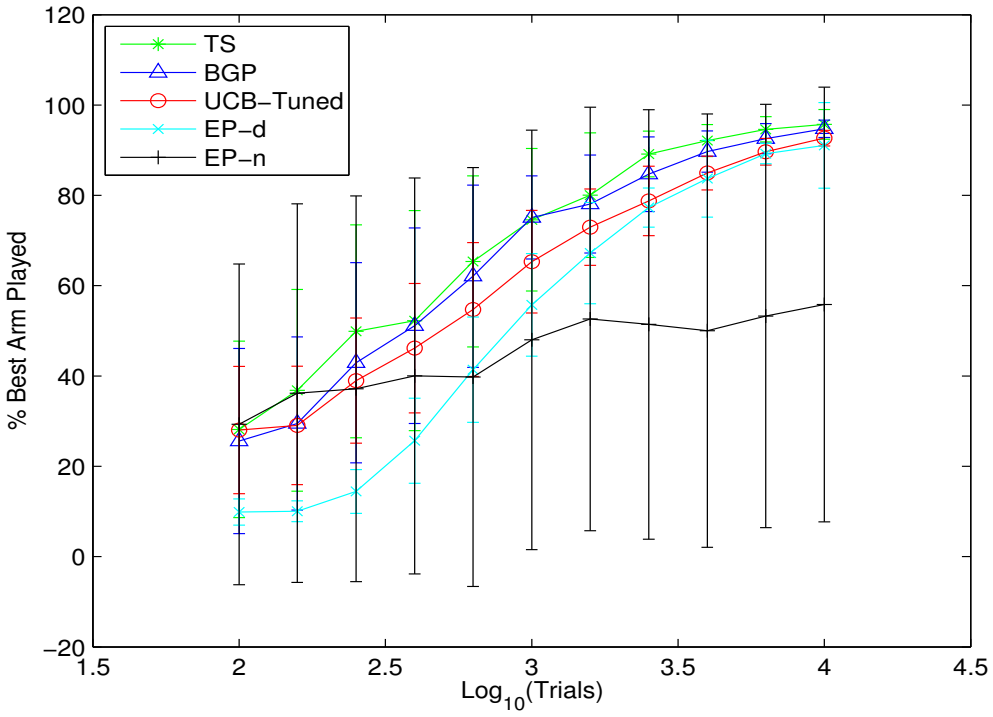

Figure 5.3: % Best Arm Played for the case of Experiment 1

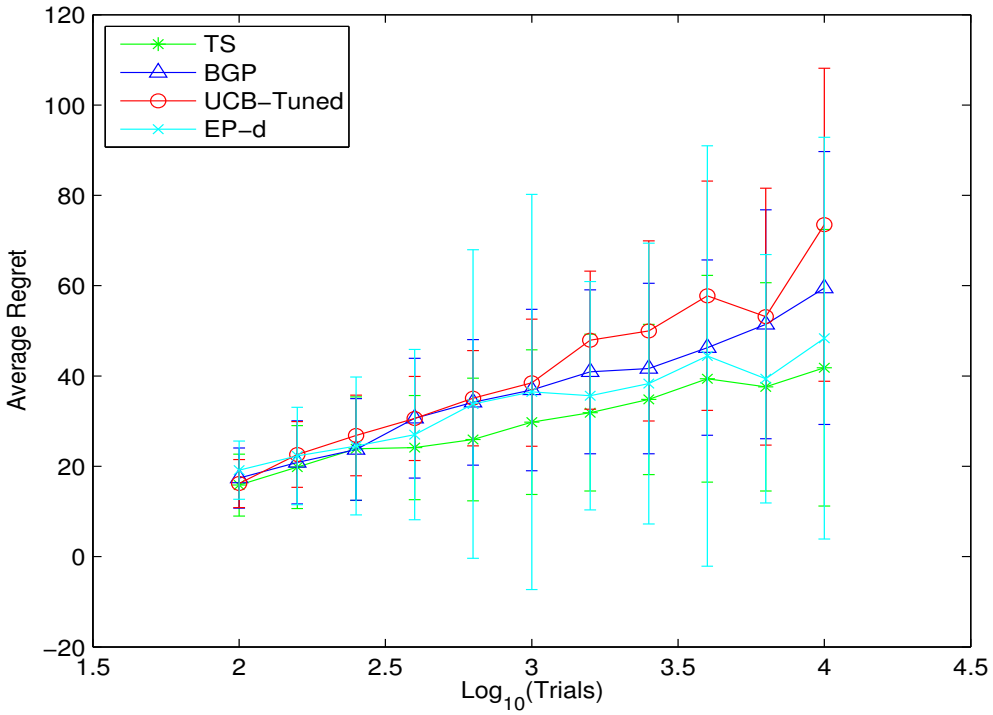

Figure 5.4: Regret for the case of Experiment 2

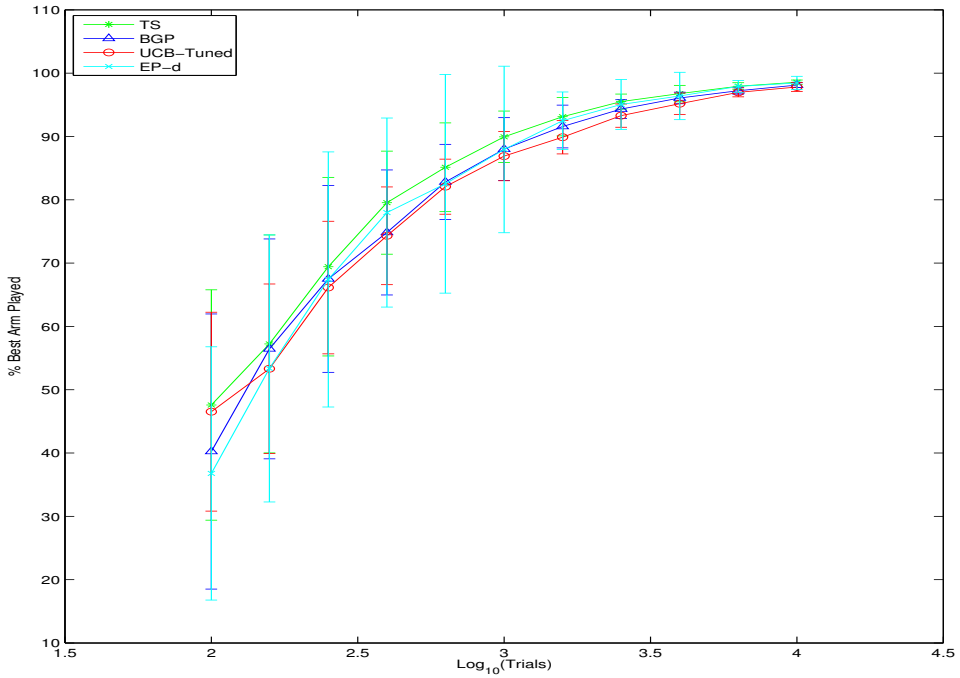

Figure 5.5: % Best Arm Played for the case of Experiment 2

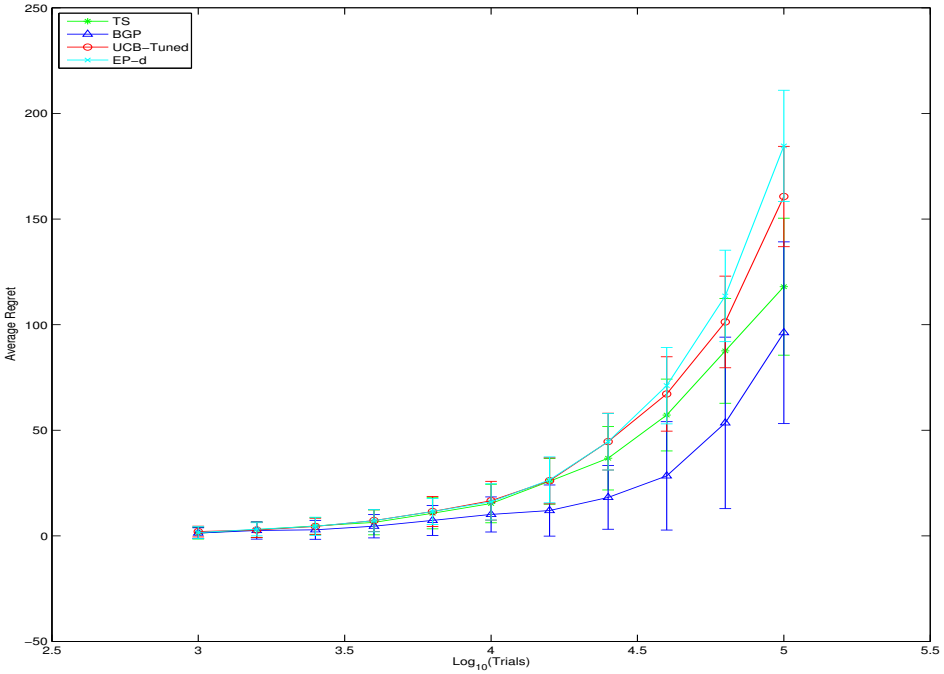

Figure 5.6: Regret for the case of Experiment 3

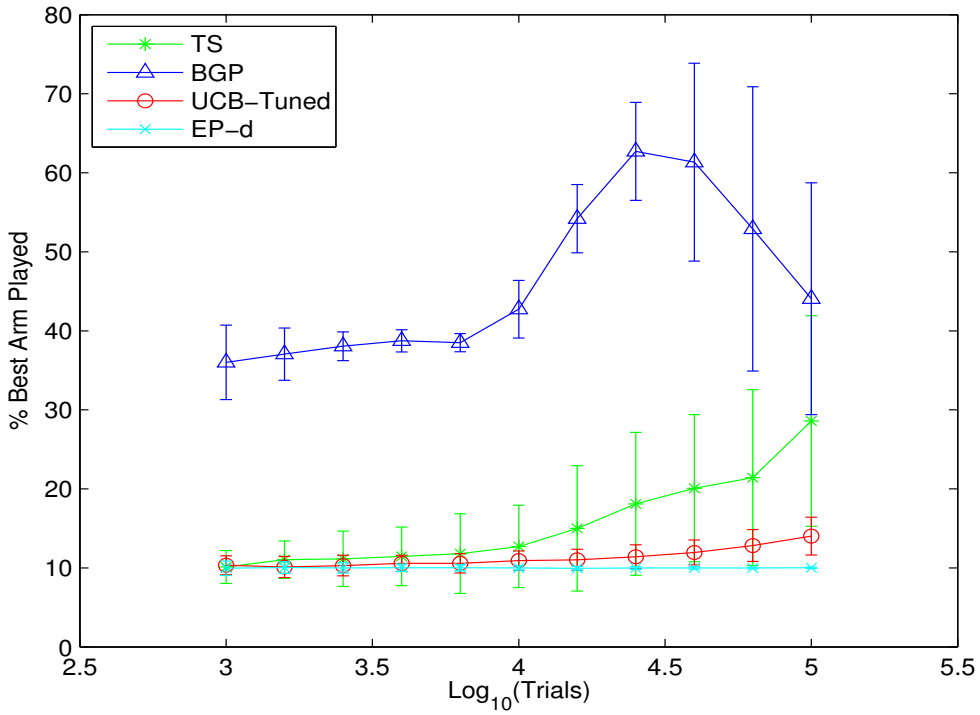

Figure 5.7: % Best Arm Played for the case of Experiment 3

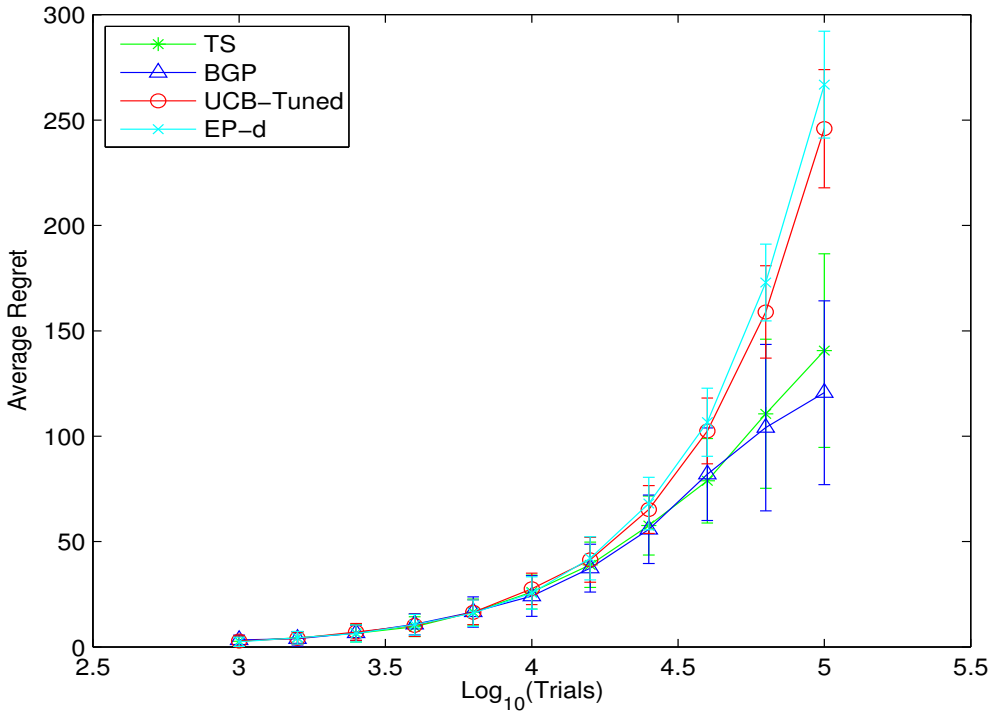

Figure 5.8: Regret for the case of Experiment 4

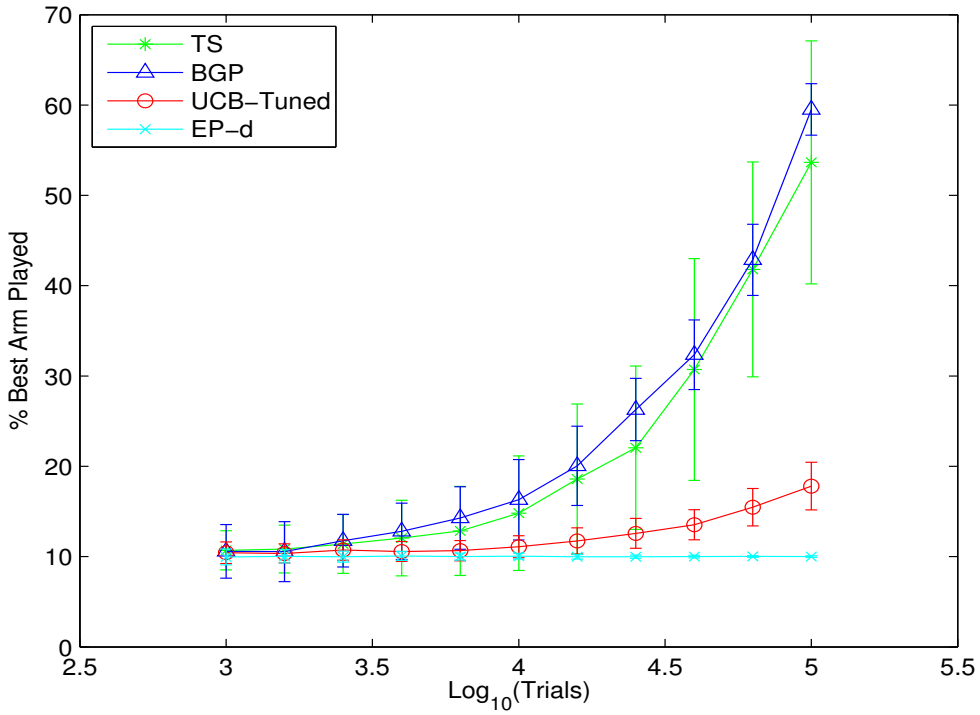

Figure 5.9: % Best Arm Played for the case of Experiment 4

each other ($\Delta_{opt-subopt} = 0.001$). The graphs for the regret and the % best arm played are shown in Fig. 5.6 and Fig. 5.7 respectively. The graphs show that the Beta Bayesian algorithms perform the best both in regret obtained and % of times the best arm is played. UCB-Tuned algorithm does not perform well for rare-events and EP-d shows unstable behavior.

The second experiment is performed for reward probabilities given in Experiment 4, and the results are shown in Fig. 5.8 and 5.9. Beta Bayesian algorithms show the best performance in this case also.

5.3 Conclusion

In this chapter, we analyzed and demonstrated the Bayesian inferencing based BGP and BGS (TS) methods which use *Order Statistics* to decide an arm to play. The empirical results show that the **Beta Bayesian algorithms - BGP and TS** perform better than all other commonly used algorithms including UCB-tuned and epsilon-greedy for solving this problem for a wide range of probabilities. Both BGP and BGS (TS) perform almost equally well. While in BGP we need to explicitly compute the value of $P(\theta^k > \theta^1 \wedge \theta^k > \theta^2 \wedge \theta^k > \theta^3 ... \theta^k > \theta^K)$, we do not need to do so in the Sampling Algorithm. Hence, we use the BGS (TS) algorithm for future comparisons. Since the sampling algorithm came from the Thompson Method, we will call it "Thompson Sampling" in the work further.

Chapter 6

Successive Reduction in Multi-Armed Bandits

6.1 Introduction

In the previous chapter, we showed that Bayesian based solution strategies are top performers when it comes to solving MABs with Bernoulli distributed rewards. For every arm, there is a chance that exactly that arm is the one with the largest reward probability, and thus being the optimal choice. By pulling the available arms with frequencies that are proportional to their probabilities of being optimal, Thompson Methods gradually moves from exploration to exploitation, converging towards only selecting the optimal arm. Unfortunately, an inherent limitation of Thompson Method emerges when the number of arms grows large. In order for the optimal arm to be chosen in Thompson Methods, it has to "beat" *all* of the inferior arms in a pair-wise manner. As the number of inferior arms grows, the probability of the superior arm winning thus deteriorates. The effect of this deterioration is easily seen in large scale real life applications, such as those found in the Internet domain, where it is common with several hundreds arms in a single MAB problem.

In this dissertation, we introduce a rather radical strategy — the Successive Reduction (SR) strategy — that addresses the above weakness directly. We propose two kinds of SR strategies: 1) Successive Reduction Hoeffding (SRH) and 2) Successive Reduction Order Statistics (SRO) [14]. Both use an Order Statistics based

Thompson Sampling Method method for arm selection, and then successively eliminates bandit arms from consideration based on a confidence threshold. While SRH uses Hoeffding Bounds for elimination, SRO uses the probability of the arms being superior to the currently selected arm to measure confidence. In effect, the Thompson Methods is focused strictly on the arms that still are promising candidates for being the optimal choice. The gains are two-fold: 1) the total number of reward increases since the eliminated sub-optimal arms are not considered in future trials, and 2) the number of arms are reduced, hence information stored and managed are reduced too.

6.2 Successive Reduction using Hoeffding Bounds (SRH)

Hoeffding Bounds [17] are important theoretical bounds and have been applied to a large number of areas such as algorithmic and learning theory, networking, machine learning. Maron et al. used Hoeffding Bounds to quickly discard bad models in order to accelerate model selection search for classification and function approximation [23]. In this dissertation, we apply Hoeffding Bounds as one of the measures for arm elimination in the Thompson Sampling based MAB strategy. The bound states that for $\theta^k \in [0,1]$with confidence δ,

$$P(|\theta^k_{true} - \theta^k_n| > \epsilon) < 2\mathrm{e}^{-2n_k\epsilon^2} \tag{6.1}$$

Hence, for the estimated mean to be within ϵ of the true mean with confidence $(1-\delta)$, ϵ becomes

$$\epsilon^k_n = \sqrt{\frac{log(2/\delta)}{2n_k}} \tag{6.2}$$

In the SRH algorithm, we first do Thompson Sampling Method and select an arm to play. But at the end of each trial, we eliminate the arms whose best possible mean (upper bound) is less than the worst (lower bound) of the best arm.

6.3 Successive Reduction using Order Statistics (SRO)

In order statistics based SRO algorithm, we select arm i according to Thompson Sampling Method but to reduce the arms, we compare all the other arms in the set with the selected arm i and compute on a pairwise basis the probability of arm i being greater than an arm k by computing $P(\theta^i > \theta^k)$. If this value is greater than a threshold say 99% then arm i is removed as shown in Algo. 14. Next, we illustrate how given two arms A and B we compute the $P(\theta^1 > \theta^2)$ in real-time.

The value of $P(\theta^1 > \theta^2)$ depends on distance between the means of the two random distributions given by $\Delta^{12} = \theta^1 - \theta^2$ and the variance of the random distributions. Fig. 6.1 shows the plot for the Beta distributions for four different sets of values for the case of two arms. In the top-left figure, $P(\theta^1 > \theta^2) = 0.99 \sim 1$ since $\theta^1 > \theta^2$ and the variance is small while in top right figure, the value of $P(\theta^1 > \theta^2) = 0.791$ due to high variance leading to high overlap. In the bottom figures, $P(\theta^1 > \theta^2) = 0.806$ due to high overlap whereas in the bottom-right corner $P(\theta^1 > \theta^2) = 0.001$ since $\theta^1 < \theta^2$ with very little overlap. We have shown earlier that the value of $P(\theta^A > \theta^B)$ in two ways by using - Beta distribution and Normal approximation. The equations and derivations are given in Chapter 5.

Setting the priors: Beta distribution can take many different forms de-

Algorithm 13 Algorithm: Successive Reduction Method based on Hoeffding Bounds (SRH)

Initialize all α_0^k=2, $\beta_0^k = 2, \forall k \in K$, set the threshold equal to δ .

loop

Do Thompson Sampling Method as given in Algo. 12

Identify the arm i which has the highest lower bound, $i = \{k : \max_{\mathrm{k}}(\theta_n^k - \epsilon_n^k)\}$.

for all arms k excluding arm i **do**

if $(\theta_n^i - \epsilon_n^i) > (\theta_n^k + \epsilon_n^k)$ **then**

Remove arm k.

end if

end for

end loop

Algorithm 14 Algorithm: Successive Reduction Method based on Order Statistics (SRO)

Initialize all α_0^k=2, $\beta_0^k = 2, \forall k \in K$, set the threshold equal to P^* .

loop

Do Thompson Sampling Method as given in Algo. 12 and play arm i.

for all arms k excluding arm i **do**

Compute $p = P(\theta^i > \theta^k)$.

if $p > P^*$ **then**

Remove arm k.

end if

end for

end loop

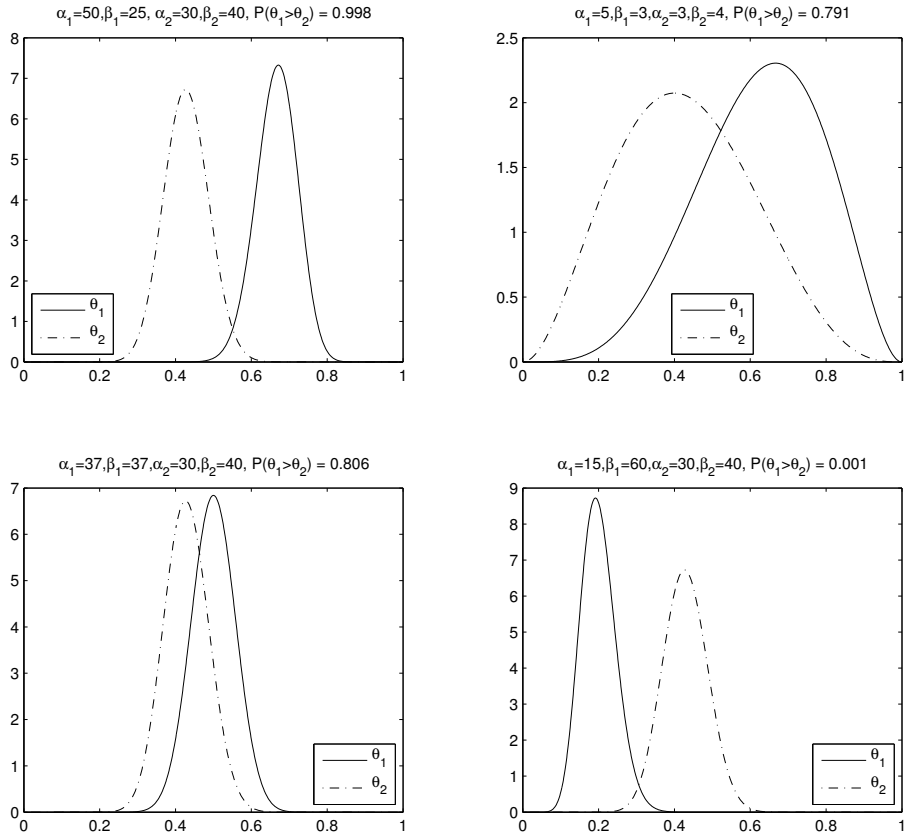

Figure 6.1: Plots for Beta distributions for two examples for the case of two arms with mean values denoted by θ^1, θ^2

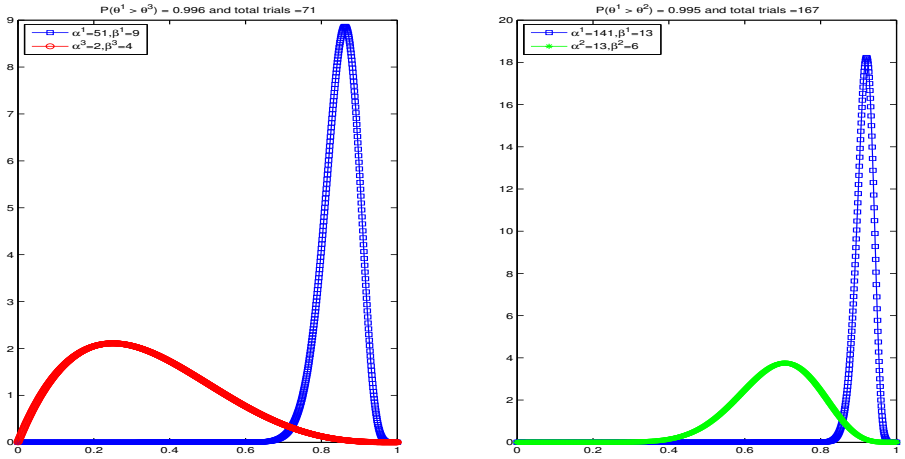

Figure 6.2: Plots for Beta distribution curves comparing the selected and the suboptimal arm at the trial when the suboptimal arm is eliminated.

pending on the values of the parameters α, β. When $\alpha, \beta \geq 2$, the curve becomes unimodal. For a better approximation to normal distribution, it is desirable to have unimodal curves. Hence, we initialize the priors to be $\alpha_0 = 2, \beta_0 = 2$.

Example SRO algorithm

To illustrate the working of the SRO algorithm, we take a case of 3 armed bandits with reward probabilities equal to $(\theta^1 = 0.9, \theta^2 = 0.6, \theta^3 = 0.3)$ as an example. As per the algorithm, arms are selected according to Thompson Sampling Method, but are discarded based on the Order statistics method with a threshold value of 99.5%. Initially, the state of the system is $\Phi_0 = ((2,2),(2,2),(2,2))$. Fig. 6.2 represents probability distributions at the trials when the suboptimal arms are eliminated. The first curve plots the Beta distribution curves for $((\alpha^1 = 51, \beta^1 = 9), (\alpha^3 = 2, \beta^3 = 4))$ when θ^3 is eliminated at 71^{st} trial at $\Phi_{71} = ((51,9),(12,5),(2,4))$.

At this point, $P(\theta^1 > \theta^3) = 0.99652$. Since, the second arm has a higher value of the reward probability, it is removed much later at 167^{th} trial when $\Phi_{167} = ((141, 13), (13, 6), (-, -))$ and $P(\theta^1 > \theta^2) = 0.995$. So, at the end of 167^{th} trial, only the optimal arm is left in the set and only that is tried henceforth, thereby maximizing the rewards.

6.4 Empirical Analysis

In this section, we evaluate the performance of SRO and SRH algorithms by comparing them with Thompson Method and UCB-Tuned algorithms. Each experiment is repeated 100 times from independent random streams and average values and standard deviations are reported in the results. We report the total reward obtained as the measure of performance of the MAB strategies. Reward is defined as:

$$\Sigma_{k=1}^{K} \theta^k \mathrm{E}[n^k] \tag{6.3}$$

where $E[n^k]$ is the expected value of the number of times k^{th} arm is played.

6.4.1 Experiment 1: Varying Threshold

In the first experiment, we vary the threshold values for SRH and SRO algorithms to analyze its effect on the performance of the SR strategies. We consider a total of 50 arms, $\theta_{opt} = 0.6$, and all the other 49 arms are generated from Bayesian distribution U$(0.6, 0)$. Table. 6.1 shows the reward obtained by using different thresholds for the SR method for 10K trials. The different values of threshold are

90%, 95%, 99%, 99.5%. We see that the SRH algorithm does not show any improvement over the Thompson Sampling Method algorithm for any value of the thresholds while the SRO algorithm gives a significant improvement over the Thompson Sampling Method and UCB-T algorithms.

In the variable threshold Var_T experiment for SRO algorithm, we vary the threshold based on the number of times an arm considered for elimination has been played. The intuition behind the variable threshold is that the larger the number of plays of an arm, the lower is its variance and more closer is it to it's actual value, hence more risk could be taken while eliminating the arm. The values for variable threshold Var_T are 99.5% when $n^k \leq 10$, 99% when n_k lies in range $(10, 50)$ and 95% otherwise, where n^k is the number of times arm k has been tried. These values of n^k work well for Uniformly distributed probabilities $\theta^k \in [0, 1]$.

Table. 6.1 also reports the standard deviations in the total reward obtained and we see that the variable threshold has the least standard deviation. The standard deviation obtained in the SRO algorithm with threshold = 90% is vey high while its reward is the one of the highest, hence it is a high risk threshold.

The number of arms left in the set at the end of the 10K trials is also shown in the column *Final Arms*. We see that out of 100 arms, less than 5 arms remain at the end of 10K trials for the case of SRO algorithm in Table 6.1 while for the case of SRO algorithm more than 20 arms remain. The average number of trials after which only a single optimal arm is left are also given in the column *Trial End*.

Table 6.1: The results obtained by varying threshold for a range of probabilities from (0.6,0) where $\theta_{opt} = 0.6$. for Experiment 1

Method	Threshold %	Reward	Std Dev	Final Arms	Trial End
SR -Order Stats.	90	5814.57	218.51	1.03	1242.86
SR -Order Stats.	95	**5861.82**	97.91	1.3	4487.06
SR -Order Stats.	99	5764.56	83.99	3.14	9663.09
SR -Order Stats.	99.5	5744.77	75.39	3.97	9945.72
SR -Order Stats.	Var_T	5781.96	**58.23**	1.65	6867.33
SR -Hoeffding	90	5539.26	63.09	21.92	10000
SR -Hoeffding	95	5534.59	69.08	26.93	10000
SR -Hoeffding	99	5560.1	55.308	49.03	10000
SR -Hoeffding	99.5	5541.78	70.22	49.98	10000
Thompson	-	5540.22	63.76	-	-
UCB-T	-	5640.78	74.90	-	-

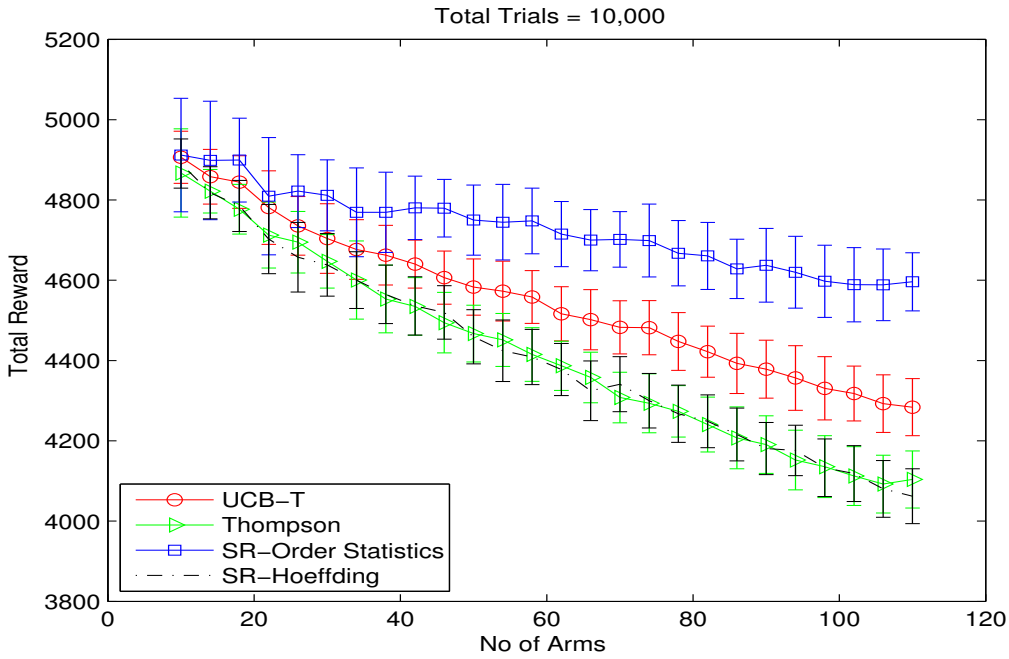

Figure 6.3: Reward & remaining arms obtained when number of arms are varied for a total of 10K trials for Experiment 2. Remaining arms for all other algorithms except SRO overlap.

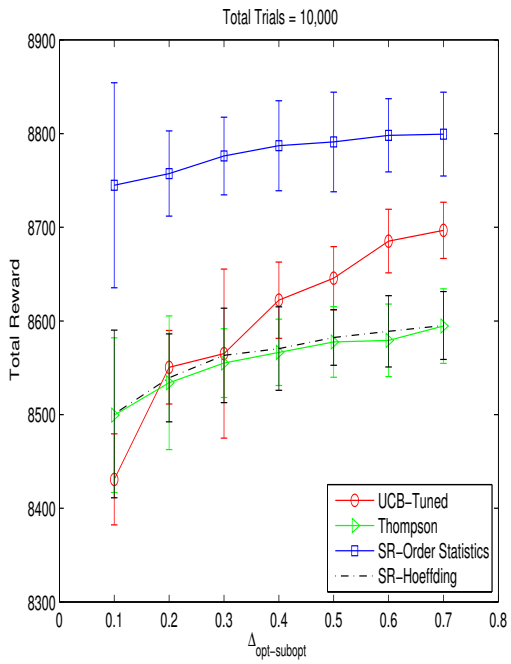

Figure 6.4: Reward when $\Delta_{opt-subopt}$ is varied for Experiment 3.

6.4.2 Experiment 2: Increasing Number of Arms

We perform this experiment to show the effect of increasing the number of arms on the reward obtained. We set $\theta_{opt} = 0.5$ and initially randomly generate a set of 9 arms with reward probabilities in interval $(0.5, 0)$ using Uniform distribution, and add four arms from the same set $\mathrm{U}(0.5, 0)$ for a total of 10K trials. We use Var_T threshold for SRO algorithm and 99% threshold for the SRH algorithm. As shown in Fig. 6.3, the SRO algorithm performs significantly better than the Thompson Sampling Method and UCB-Tuned algorithm. Also for the SRO algorithm, we notice that the total number of arms remaining at the end of 10K trials has an average value of < 5 even when the initial number of arms are > 100. SRH algorithm does not show any improvement in the reward values relative to Thompson Sampling

Method and also the number of arms eliminated at the end of the experiment is almost null.

6.4.3 Experiment 3: Increasing $\Delta_{opt-subopt}$

In this experiment, we systematically vary the difference in the optimal and the suboptimal arm and compare the performance of SR algorithms with UCB-Tuned and Thompson Sampling Method. We take a total of 100 arms and increase the difference in the optimal and sub-optimal arms denoted by $\Delta_{opt-subopt}$. The optimal arm is set to $\theta^{opt} = 0.9$ and different values of θ^{subopt} are chosen from the set $\{0.8, 0.7, 0.6, 0.5, 0.4, 0.3, 0.2\}$ such that $\Delta_{opt-subopt}$ varies from $(0.1 - 0.7)$. Rest other arms are generated from a Uniform random distribution between $\mathrm{U}(\theta^{subopt}, 0)$. We use Var_T as threshold for SRO algorithm and a constant 99% threshold for the SRH algorithm. The results in Fig. 6.4 show that the SRO algorithm performs significantly better than Thompson and UCB-T algorithms. SRH algorithm does not show any improvement over Thompson Sampling Method.

6.5 Conclusion

From the experimental results and analysis done in this dissertation we conclude that current state-of-art methods such as UCB, Thompson Sampling Method do not work well for large number of arms, hence new schemes need to be developed to handle the challenge of scalability in the multi-armed bandit setting. Towards this direction, we presented SR strategies, SRO and SRH, for solving large

scale multi-armed bandit problems in the scenario where no prior information was available about the arms. Our experiments reveal that SRO strategy significantly outperforms Thompson Sampling Method, UCB-Tuned and SRH algorithms and the performance increase is more significant with increasing number of arms. Hoeffding Bounds are loose bounds which Thompson Sampling Method already takes care of, hence no improvement in performance is noticed in the SRH algorithm. Although this dissertation discusses Bernoulli bandits with Beta distributions, the SR algorithms are applicable for all forms of random distributions.

Chapter 7

Dynamic Multi-Armed Bandits

7.1 Introduction

Bayesian Method based MAB strategies have been established as top performers when it comes to solving Bernoulli distributed rewards. For every arm, there is a chance that exactly that arm is the one with the largest reward probability, and thus being the optimal choice. By pulling the available arms with frequencies that are proportional to their probabilities of being optimal, Thompson Sampling gradually moves from exploration to exploitation, converging towards only selecting the optimal arm. This behavior is ideal when the reward probabilities of the bandit arms are fixed. However, in cases where the reward probabilities are dynamically evolving, one would instead prefer schemes that explore and track potential reward probability changes. Apart from the Kalman filter based scheme proposed in [11], the latter problem area is largely unexplored when it comes to Thompson Sampling based schemes. Another obstacle in solving the problem is due to the fact that we cannot sample noisy instances of θ directly, as done in [11]. Instead, we must rely on samples obtained from Bernoulli trials with *reward probability* θ, which makes the problem unique and one not studied before.

In this dissertation, we introduce a novel strategy — Dynamic Thompson Sampling. Order Statistics based Thompson Sampling is used for arm selection, but

the reward probability θ is tracked using an exponential filtering technique, allowing adaptive exploration. In brief, we explicitly model changing θ as an integrated part of an Order Statistics based sampling and arm selection method, considering changes in reward probability to follow a Brownian motion – one of the most popular stationary process which has been extensively used in many fields including economics to model stock markets, commodity pricing, etc.

In this dissertation, we look at the problem of dynamic bandits in which the reward probabilities of the arms follow bounded Brownian motion. In [26], the authors consider a similar scenario of Brownian bandits with reflective boundaries assuming that a sample from the current distribution of θ itself is observed at each trial. Granmo et al. introduced the Order Statistics based Kalman Filter Multi-Armed Bandit Algorithm [11]. In their model, reward obtained any arm is affected by Gaussian noise $\sim N(0, \sigma_{ob}^2)$ and an independent Gaussian perturbations $\sim N(0, \sigma_{tr}^2)$ at each trial. A key assumption in [11] is that at each trial a noisy sample of the reward is observed. In our work, estimation of the reward probability θ is done by only using the Bernoulli outcomes $r \sim Bernoulli(\theta)$. Our work is well suited for modeling of reward probabilities in the Internet domain where, when an item such as a newspaper article, advertisement, etc. is shown, the system receives a $\{0, 1\}$ reward in the form of a click/engagement, using which it has to estimate θ, which is the click/engagement rate in this situation. Also, instead of using reflective boundaries we consider absorbing and simple boundaries, which are more suited to the Internet domain.

7.2 PROBLEM DEFINITION

7.2.1 Constant Rewards

In the MAB setting, each pull of an arm can be considered as a Bernoulli trial with output in the set {0,1} and defined by a single parameter θ which is the probability of success denoted by {1}. The probability distribution of the number of successes, denoted by S, obtained in n Bernoulli trials is known to have a Binomial distribution, $S \sim Binomial(n, \theta)$.

$$p(S = s|\theta) = \binom{n}{s}(1-\theta)^{n-s}\theta^{s} \tag{7.1}$$

Bayesian estimation may be used for estimating θ by considering its estimate $\hat{\theta}$ to have a defined probability distribution. When the distribution of $\hat{\theta}$ is a conjugate distribution to the actual distribution of θ and the observation, the Bayesian estimate simplifies significantly. As it is known that Beta distribution is a conjugate prior for the Binomial distribution [13]. Thus, when providing a Bayesian estimate for θ, it is natural to assume that $\hat{\theta}$ possesses a Beta distributed prior, fully specified by the parameters (α_0, β_0):

$$p(\hat{\theta}; \alpha_0, \beta_0) = \frac{\theta^{\alpha_0 - 1}(1-\theta)^{\beta_0 - 1}}{B(\alpha_0, \beta_0)} \tag{7.2}$$

If a success is received at the n^{th} trial, α and β are updated as,

$$\alpha_n = \alpha_{n-1} + 1, \beta_n = \beta_{n-1} \tag{7.3}$$

or if a failure is received at the n^{th} trial, α and β are updated as,

$$\alpha_n = \alpha_n, \beta_n = \beta_{n-1} + 1 \tag{7.4}$$

After s successes and r failures, the parameters of Beta distribution become $(\alpha_0 + s,\ \beta_0 + r)$.

$$\hat{\theta} \sim Beta(\alpha_0 + s, \beta_0 + r) \tag{7.5}$$

The estimated mean and variance after trial n become,

$$\hat{\theta}_n = \frac{\alpha_n}{\alpha_n + \beta_n} \tag{7.6}$$

$$\hat{\sigma}_n^2 = \frac{(\alpha_n \beta_n)}{(\alpha_n + \beta_n + 1)(\alpha_n + \beta_n)^2} \tag{7.7}$$

7.2.2 Dynamically Changing Rewards

The key assumption made in static MAB algorithms is that the value of reward probability remains constant. In practical situations, it is rare to have constant reward probabilities and the problem we address here explicitly takes into account changing reward probabilities.

Brownian motion is a simple stochastic process in which the value of a random variable at step n is the sum of its value at time $n-1$ and a Gaussian noise term $\sim N(0, \sigma^2)$. In this dissertation, we consider that the reward probability θ to follow a simple Brownian motion in the range $[0, 1]$.

$$System\ Equation : \theta_n = \theta_{n-1} + \nu_n \tag{7.8}$$

$$\nu_n \sim N(0, \sigma^2) \tag{7.9}$$

As θ is a probability, it must be between [0, 1], as a consquence we need to used bounded Brownian motion to model the changes in it. We define two types of boundary conditions:

- Simple Bounded : The reward probability is bounded between $[0, 1]$ and once it reaches boundaries it remains there until the next outcome moves it out of the boundary.

$$\theta_n = \begin{cases} \theta_{n-1} + \nu_n \\ 1 \ if \ \theta_n \geq 1 \\ 0 \ if \ \theta_n \leq 0 \end{cases}$$

- Absorbing Boundary : In absorbing boundaries, when θ_n reaches any boundary at trial n, it remains there forever.

$$\theta_n = \begin{cases} \theta_{n-1} + \nu_n \\ 1 \quad \exists i \leq n : \theta_i \geq 1 \\ 0 \quad \exists i \leq n : \theta_i \leq 0 \end{cases}$$

The performance of the estimation technique is measured in terms of the *Regret* which is defined as,

$$Regret = \Sigma_{n=0}^{N}(r_n^* - r_n^k)$$

where N is the total number of trials, r_n^* is the Bernoulli output received after playing an arm with the highest θ_n^k at trial n and r_n^k is the reward obtained after sampling the k^{th} arm according to the applied algorithm. Note that the arm corresponding to r_n^* may change as the values of θ_n^k evolves. Hence, regret is a measure of the loss occurred when using an algorithm as compared to the optimal algorithm.

7.2.3 Dynamic Thompson Sampling Algorithm (DTS)

Unlike the static MAB problems, the goal of the DTS algorithm is to minimize the regret by tracking the changing values of θ_n^k at trial n as closely as possible [15].

The Dynamic Thompson Sampling algorithm proposed here dynamically estimates the changing values of θ_n^k for each arm k and at each step with the goal of minimizing the regret. Note that in our model θ_n^k changes according to Eqn. 7.8 whether arm k is played or not. In DTS algorithm, unlike the update rules of Eqn. 7.3 and 7.4, we propose to use the update rules for α and β as follows:

If $\alpha_n + \beta_n \leq C$,

$$\alpha_n = \alpha_{n-1} + r_n \tag{7.10}$$

$$\beta_n = \beta_{n-1} + (1 - r_n) \tag{7.11}$$

But if $\alpha_n + \beta_n > C$,

$$\alpha_n = (\alpha_{n-1} + r_n)\frac{C}{C+1} \tag{7.12}$$

$$\beta_n = (\beta_{n-1} + (1 - r_n))\frac{C}{C+1} \tag{7.13}$$

Hence,

$$\begin{aligned} \alpha_n + \beta_n &= (\alpha_{n-1} + \beta_{n-1} + 1)\frac{C}{C+1} && (7.14)\\ &= (C+1)\frac{C}{C+1} && (7.15)\\ &= C && (7.16) \end{aligned}$$

Updating the values of α_n, β_n in the above form, leads to estimates of α_n, β_n which give more weight to the more recent values of the reward as compared to the old values. If we further substitute the value of α_{n-1} in the above expression, we get

$$\begin{aligned} \alpha_n &= ((\alpha_{n-2} + r_{n-1})\frac{C}{C+1} + r_n)\frac{C}{C+1} && (7.17)\\ &= \alpha_{n-2}(\frac{C}{C+1})^2 + r_{n-1}(\frac{C}{C+1})^2 + r_n\frac{C}{C+1} && (7.18) \end{aligned}$$

In the same way, we could express β_n as a discounted sum of previous outputs of the Bernoulli trials and priors. According to the Beta distribution, the estimated mean $\hat{\theta}_n$ at time step n is,

$$\begin{aligned}
\hat{\theta}_n &= \frac{\alpha_n}{\alpha_n + \beta_n} && (7.19)\\
&= \frac{\alpha_{n-1} + r_n}{C} \times \frac{C}{C+1} && (7.20)\\
&= \frac{\alpha_{n-1}}{C}\frac{C}{C+1} + r_n\frac{1}{C+1} && (7.21)\\
&= \frac{C}{C+1}\frac{\alpha_{n-1}}{\alpha_{n-1}+\beta_{n-1}} + \frac{1}{C+1}r_n && (7.22)\\
&= \Delta\hat{\theta}_{n-1} + (1-\Delta)r_n && (7.23)
\end{aligned}$$

where $\Delta = \frac{C}{C+1}$

Clearly, this approach yields *exponential filtering(smoothing)* of $\hat{\theta}_n$ [22]. Note that the estimated variance is,

$$\hat{\sigma}_n^2 = \frac{(\alpha_n\beta_n)}{(\alpha_n + \beta_n + 1)(\alpha_n + \beta_n)^2}$$

The product of α_n and β_n is maximum when $\alpha_n = \beta_n = C/2$ and will be minimum when one of them is equal to the prior$(\alpha_0, \beta_0 = 2)$. In Thompson Sampling, initial priors are set to 2 to ensure unimodal distribution. Hence,

$$\frac{2(C-2)}{C^2(C+1)} \leq \hat{\sigma}_n^2 \leq \frac{1}{4(C+1)} \tag{7.24}$$

The DTS algorithm is described in Algorithm. 15 for the generalized case of K-arms having reward probabilities $(\theta^1, \theta^2, \theta^3, .., \theta^K)$ which vary using Brownian motion. The algorithm starts by initializing same priors for all the arms and then gradually changing the values of α, β parameters of the selected arms as shown in

Algorithm 15 Algorithm: Dynamic Thompson Sampling (DTS)

Initialize α_0^k=2, $\beta_0^k = 2$.

loop

Draw a value of π^k randomly from $Beta(\alpha^k, \beta^k) \forall k \in K$.

Arrange the samples in decreasing order.

Select the arm A s.t $\pi^A = max_k(\pi^k)$, $\forall k \in K$.

Pull arm A.

if Arm A is successful **then**

if $\alpha_{n-1}^A + \beta_{n-1}^A \leq C$ **then**

Update the values for $\alpha_n^A = (\alpha_{n-1}^A + 1)$, $\beta_n^A = \beta_{n-1}^A$.

else

Update the values for $\alpha_n^A = (\alpha_{n-1}^A + 1)\frac{C}{C+1}$, $\beta_n^A = (\beta_{n-1}^A)\frac{C}{C+1}$.

end if

else

if $\alpha_{n-1}^A + \beta_{n-1}^A > C$ **then**

Update the values for $\alpha_n^A = \alpha_{n-1}^A$, $\beta_n^A = \beta_{n-1}^A + 1$.

else

Update the values for $\alpha_n^A = (\alpha_{n-1}^A)\frac{C}{C+1}$, $\beta_n^A = (\beta_{n-1}^A + 1)\frac{C}{C+1}$.

end if

end if

end loop

Algorithm. 15. The algorithm adapts the values of the θ based on the exponential updates and leads to a better approximation to the drifting reward probabilities which in turn lead to a better performance.

7.3 Experiments

In this section, we primarily evaluate the performance of DTS algorithm by comparing it with UCB_f, TS and UCB-Normal algorithms. Though we performed significant experiments over several values of the reward distributions, we only report the most important and relevant experiments in this dissertation due to limited space. We report the regret obtained as the measure of performance of the different strategies. As DTS is a randomized algorithm, the regret becomes a random variable. The expected value of the regret is calculated by repeating each experiment 400 times.

7.3.1 Varying value of standard deviation σ

To get an insight into the Brownian motion of the reward probability θ, we performed experiments in which we simulated the dynamics of θ for different values of standard deviation. In Fig. 7.1, we show a sample plot of the curves for 4 values of $\sigma = \{0.05, 0.01, 0.005, 0.001\}$. The curve with standard deviation $\sigma = 0.05$ is cutting across the boundaries 0 and 1 very often and any kind of learning seems impossible in this situation. The other graphs with standard deviations $\sigma = \{0.01, 0.005, 0.001\}$ are more stable and seem more appropriate for learning.

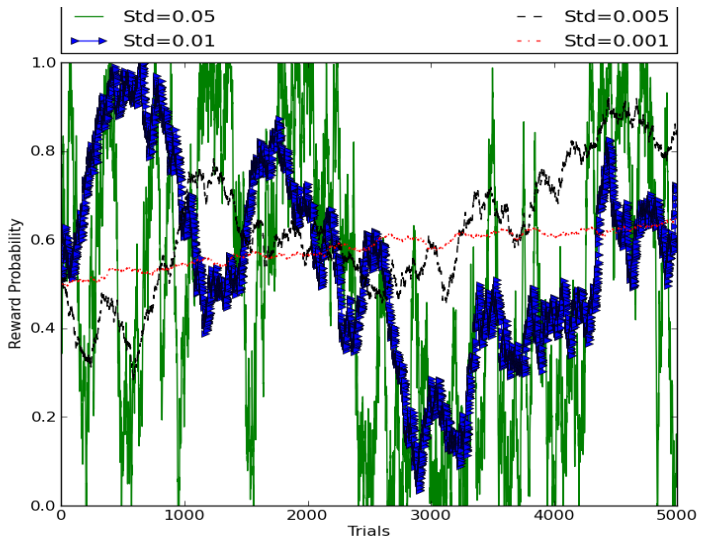

Figure 7.1: Typical variations of the reward probability θ for different values of standard deviations. $\theta_0 = 0.5$ in all cases.

7.3.2 Estimation vs. Actual

We perform these experiments to show how closely the estimated values of $\hat{\theta}$ are to the actual value of θ for the case of TS and DTS algorithms for a single arm. The two graphs, Fig. 7.2 and Fig. 7.3, show the results for the estimated and actual values of θ. We see that the DTS algorithm makes a much closer estimate of θ by using exponential filtering technique in the formulation of $\hat{\theta}$ as compared to the TS algorithm.

7.3.3 Tuning parameter C for DTS algorithm

Fig. 7.4 shows a plot of the root mean square error (RMSE) obtained for different values of C and standard deviation σ for 10, 000 trials in the DTS and TS

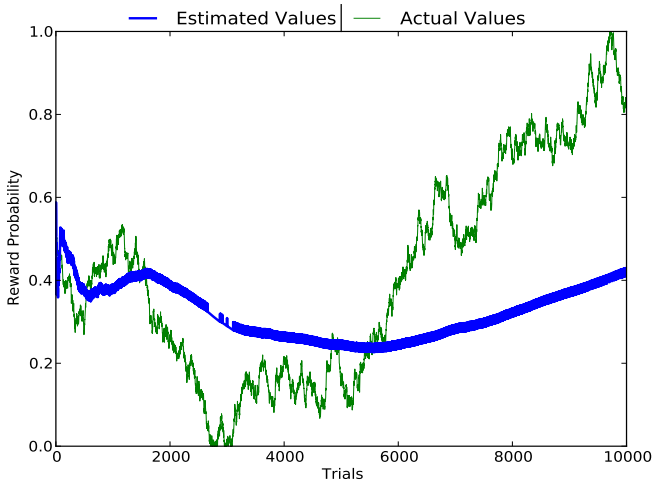

Figure 7.2: Plot shows the estimated and actual values of θ for the case of a single arm. Estimated values are calculated based on TS algorithm.

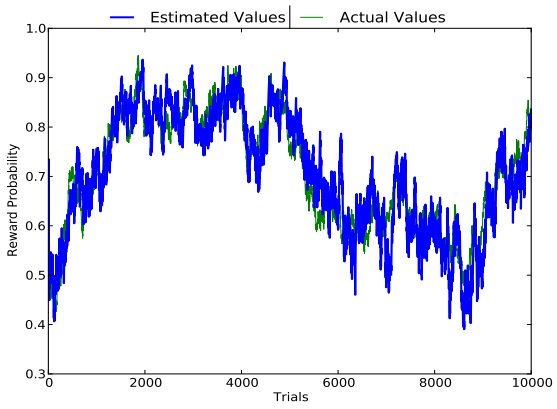

Figure 7.3: Plot shows the estimated and actual values of θ_t for the case of a single arm. Estimated values are calculated based on DTS algorithm.

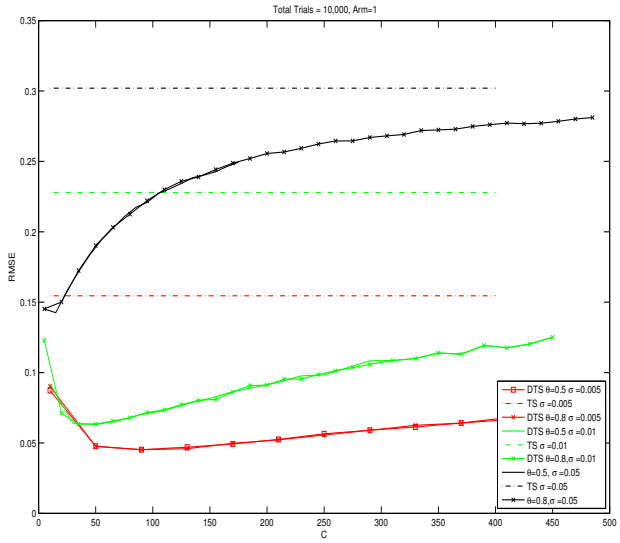

Figure 7.4: Plots for RMSE for two different values of θ, 3 different values of standard deviation σ and with/without the exponential filtering for θ

algorithm for a single arm. RMSE is measured as :

$$RMSE = \sqrt{\frac{\Sigma_{n=1}^{N}(\theta_n - \hat{\theta}_n)^2}{N}} \tag{7.25}$$

Note here that RMSE values averaged over 400 runs are reported in the graph. In this experiment, we take two different values of $\theta = \{0.8, 0.5\}$ and choose the standard deviation in the set $\{0.005, 0.01, 0.05\}$. We notice that the graphs for different values of θ but same standard deviation are overlapping. We also notice that the value at which the RMSE is minimum reduces with the increasing value of σ since higher the value of σ, the more dynamic the arms are, hence lesser past history is required for a better estimation of θ.

We next present an empirical evaluation of the different MAB algorithms using tuned values of the model parameters.

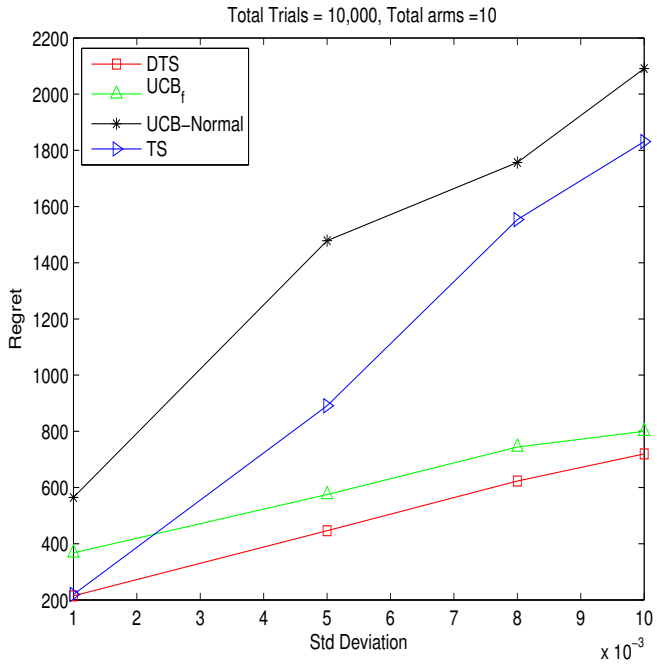

Figure 7.5: Plots of Regret comparing DTS with UCB_f, UCB-Normal and TS algorithms for the case of Simple Boundaries

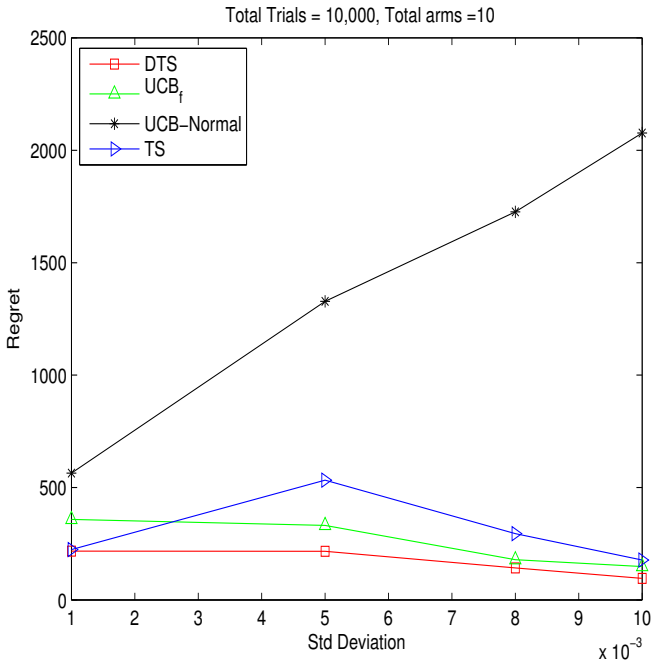

Figure 7.6: Plots of Regret comparing DTS with UCB_f, UCB-Normal and TS algorithms for the case of Absorbing Boundaries

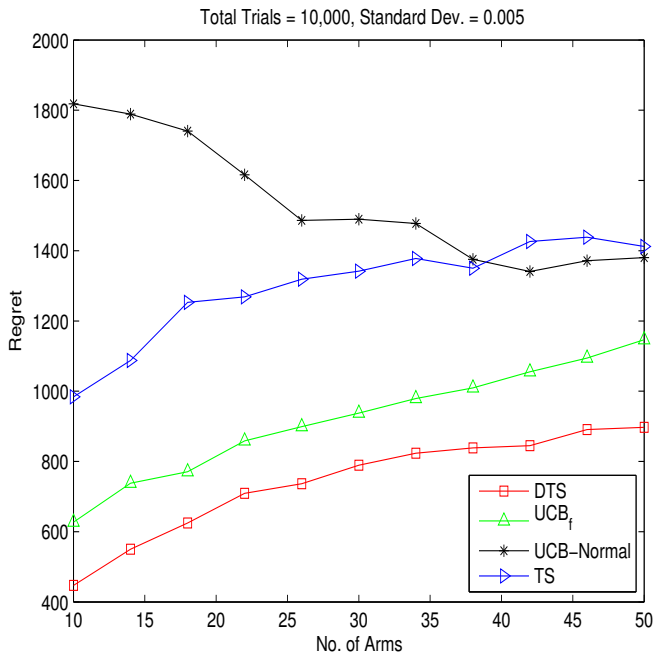

Figure 7.7: Plots of Regret comparing DTS with UCB_f, UCB-Normal and TS algorithms for the case of Simple Boundaries

7.3.4 Varying Standard Deviation

In the first experiment to evaluate the performance of different MAB strategies, we vary the standard deviation σ of θ. We consider a total of 10 arms, $\theta_{opt} = 0.6$, and all the other 9 arms are generated from Uniform distribution U$(0.6, 0)$. Fig. 7.5, 7.6 show the regret obtained by using different standard deviations for the SR method for 10, 000 trials for the case of simple and absorbing boundaries. The different values of threshold are $\{0.001, 0.005, 0.008, 0.01\}$. We see that the DTS algorithm shows the least regret as compared to other MAB strategies for both the cases of absorbing as well as simple boundaries.

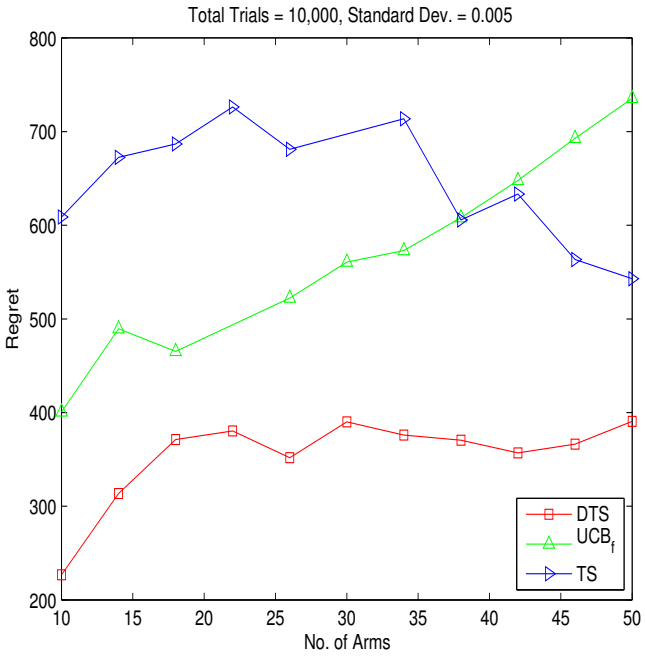

Figure 7.8: Plots of Regret comparing DTS with UCB_f, UCB-Normal and TS algorithms for the case of Absorbing Boundaries

7.3.5 Changing the number of arms

We perform this experiment to show the effect of increasing the number of arms on the regret obtained for the case of Brownian bandits. We set $\theta_{max} = 0.6$ and initially randomly generate a set of 9 arms with reward probabilities in interval $(0.6, 0)$ using Uniform distribution, and add four arms from the same set U$(0.6, 0)$ for a total of 10K trials. We use $\sigma = 0.005$ as standard deviation for the DTS algorithm. As shown in Fig. 7.7, 7.8, the DTS algorithm performs much better than the UCB_f, Thompson Sampling and UCB-Normal algorithm. The difference between UCB_f and DTS algorithm grows as the number of arms increase which shows that the UCB_f algorithm is not scalable with the number of arms for both absorbing and simple boundaries. We do not show the results of UCB-Normal algorithm in Fig. 7.8 as it consistently shows poor results for the case of absorbing boundaries also.

7.4 Conclusion

In this dissertation, we presented *Dynamic Thompson Sampling (DTS)* algorithm which uses Order Statistics based Thompson Sampling framework but extends it using exponential filtering to track the dynamic changes in the reward probabilities and minimizes the total regret. The experimental results and analysis presented in this dissertation show that the DTS algorithm significantly outperforms current state-of- art methods such as UCB_f, Thompson Sampling and UCB- Normal for the case of dynamic reward probabilities following bounded Brownian motion. We also observe an increasing performance improvement as the number of arms increases,

which demonstrates the usefulness of our proposed algorithm to large-scale MAB problems. The DTS strategy can be further extended to include variations such as playing the top-k arms instead of a single arm, and adding immunity from elimination for some arms. We are working on proving the theoretical bounds of the DTS algorithm and exploring the possibilities of extending it in other dimensions such as mortal bandits, and hierarchical bandits.

Chapter 8

Centralized vs. Decentralized Decision Making in Multi-Armed Bandits for Common PayOff Games

8.1 Introduction

In this chapter, we study the case of *Decentralized* vs. *Centralized* decision making for MAB for the case of two Samplers whose rewards depend on the decisions of each other. Instead of having a single Thompson Sampler pulling K arms, suppose we take two Thompson Samplers $\{A, B\}$ each being responsible for $\sqrt{K}$ "virtual" arms. Let sampler A govern the virtual arms $(a_1, a_2, ..., a_{\sqrt{K}})$ while sampler B governs the virtual arms $(b_1, b_2, ..., b_{\sqrt{K}})$. The rewards received by each then depend on their joint pulling of virtual arms since this joint choice is mapped into one of the K original arms. This, in turn, means for instance that the rewards for both samplers will be different for arm pull combinations (a_1, b_1) and (a_1, b_2), even though A pulls the same virtual arm in both cases. The problem that we address in this dissertation is to find the best joint choice (a_i, b_j) in a decentralized manner, with each sampler maximizing its own rewards, without communicating with the other sampler. As we will see in the following sections, this formulation can be mapped to a multi-agent system modeled in terms of a *common pay-off game* from *game theory*.

8.2 Problem Definition

8.2.1 Game Theory

Thompson Sampling decomposed as a pair of agents can be modeled as a *game theory* [16] problem and below we discuss the pertinent aspects of game theory used in this dissertation.

- **Dominated vs. Dominant Strategies:** A strategy is *dominated* if it never is the best response, whatever the choice of the opposition. Conversely, a strategy is *dominant* if it is the best strategy regardless of the opposition strategy. The fact that one of the two players have a dominant strategy enables the players to pinpoint a single outcome which is the best for both.

- **Nash Equilibrium:** An outcome is in *equilibrium* if it is brought about by strategies that agents have good reason to follow. A set of rationalizable strategies (one for each player) are in a *Nash Equilibrium* if their implementation confirms the expectations of each player about the others choice. *Nash strategies* are the only rationalizable strategies, that if implemented, confirm the expectations on which they are based.

Common pay-off games are a special class of common interest games with identical payoffs as exemplified by Matrix-I and Matrix-II in Table 8.1. In common interest games, the rationality of the group is aligned with the rationality of each player and a *Pareto Optimal Nash Equilibrium* exists. An outcome of a game is Pareto optimal if there is no other outcome that makes every player at least as well

	Matrix-I			**Matrix-II**		
A/B	b_1	b_2	b_3	b_1	b_2	b_3
a_1	0.25	0.5	**1.0**	0.25	0.275	**0.4375**
a_2	0.125	0.25	0.5	0.35	0.25	0.275
a_3	0.0625	0.125	0.25	**0.625**	0.35	0.25

Table 8.1: An example of common interest game with Pure and Multiple Nash Equilibria. Game Matrix -I has pure Nash equilibrium at (a_3, b_1) and Matrix-II has mixed Nash equilibrium at (a_3, b_1) and (a_1, b_3).

off and at least one player strictly better off. Matrix-I in Table 8.1 has a single Nash equilibrium at (a_1, b_3) and Matrix-II shows a game with two Nash equilibria (a_3, b_1) and (a_1, b_3) with (a_3, b_1) being the Pareto Optimal Nash equilibrium.

8.2.2 Multi-Armed Bandit Strategies

8.2.2.1 Decentralized Thompson Sampling (DeTS)

In Decentralized Thompson Sampling, each of the samplers A and B maintains its own local view of information. First of all, each agent maintains its own state space: $\{(\alpha^n_{A1}, \beta^n_{A1}), (\alpha^n_{A2}, \beta^n_{A2}), \ldots, (\alpha^n_{A\sqrt{K}}, \beta^n_{A\sqrt{K}})\}$ for A, and $\{(\alpha^n_{B1}, \beta^n_{B1}), (\alpha^n_{B2}, \beta^n_{B2}), \ldots, (\alpha^n_{B\sqrt{K}}, \beta^n_{B\sqrt{K}})\}$ for B. Furthermore, at each trial, each sampler selects an arm from his local set of virtual arms $(a_1, a_2, ..., a_{\sqrt{K}})$ using Thompson Sampling as shown in Alg. 16. After the samplers has selected a pair

A/B	b_1	b_2	...	$b_{\sqrt{K}}$
a_1	$q/2^{\sqrt{K}-1}$	$q/2^{\sqrt{K}-2}$	...	$q/2^0$
a_2	$q/2^{\sqrt{K}}$	$q/2^{\sqrt{K}-1}$	...	$q/2^1$
a_3	$q/2^{\sqrt{K}+1}$	$q/2^{\sqrt{K}}$	...	$q/2^2$
...	...	..	...	
$a_{\sqrt{K}}$	$q/2^{2\sqrt{K}-2}$	$q/2^{2\sqrt{K}-4}$	...	$q/2^{\sqrt{K}-1}$

Table 8.2: Common Interest Game matrix used in the experiments section.

(a_i, b_j) of virtual arms, the pair is mapped to the corresponding arm k in the K-dimensional space, which is pulled to obtain a reward. Accordingly, the rewards that each sampler receives are dependent on the decisions of the other sampler too. In this way, each agent has to decide only amongst $\sqrt{K}$ decisions as compared to the K arms available to the Pure Thompson Sampling algorithm.

Algorithm 16 Decentralized Thompson Sampling (DeTS)

For each agent, A and B, initialize $\alpha^0_{A1}, ..., \alpha^0_{A\sqrt{K}} = 2, \beta^0_{A1}, ..., \beta^0_{A\sqrt{K}} = 2$, $\alpha^0_{B1}, ..., \alpha^0_{B\sqrt{K}} = 2$, $\beta^0_{B1}, ..., \beta^0_{B\sqrt{K}} = 2$.

loop

for $i = 1 \rightarrow 2$ **do**

Select an arm i_k based on Thompson sampling for each agent i.

end for

Play arm (a_k, b_k).

end loop

A/B	b_1	b_2	b_3	b_4	b_1	b_2	b_3	b_4
	G'_0 $(r = 0)$				G'_1 $(r = 0.1)$			
a_1	0.125	0.0625	0.03125	0.015625	0.125	0.081	0.078	0.11
a_2	0.25	0.125	0.0625	0.03125	0.23	0.125	0.081	0.078
a_3	0.5	0.25	0.125	0.0625	0.45	0.23	0.125	0.081
a_4	**1**	0.5	0.25	0.125	**0.90**	0.45	0.23	0.125
	G'_2 $(r = 0.2)$				G'_3 $(r = 0.3)$			
a_1	0.125	0.1	0.125	**0.2125**	0.125	0.12	0.17	**0.31**
a_2	0.2125	0.125	0.1	0.125	0.19	0.125	0.12	0.17
a_3	0.40625	0.2125	0.125	0.1	0.3125	0.175	0.125	0.138
a_4	**0.803125**	0.40625	0.2125	0.125	**0.70**	0.36	0.19	0.125
	G'_4 $(r = 0.4)$				G'_5 $(r = 0.5)$			
a_1	0.125	0.1375	0.21875	**0.409375**	0.125	0.156	0.27	**0.508**
a_2	0.175	0.125	0.1375	0.21875	0.156	0.125	0.15625	0.26
a_3	0.3125	0.175	0.125	0.1375	0.265	0.156	0.125	0.156
a_4	**0.606**	0.313	0.175	0.125	**0.508**	0.266	0.156	0.125

Table 8.3: Matrix G' for common interest game $r = 0, 0.1, 0.2, 0.3, 0.4, 0.5$ starting from top left used in the experiment for the case of $K = 4$. The Nash equilibria are shown in bold.

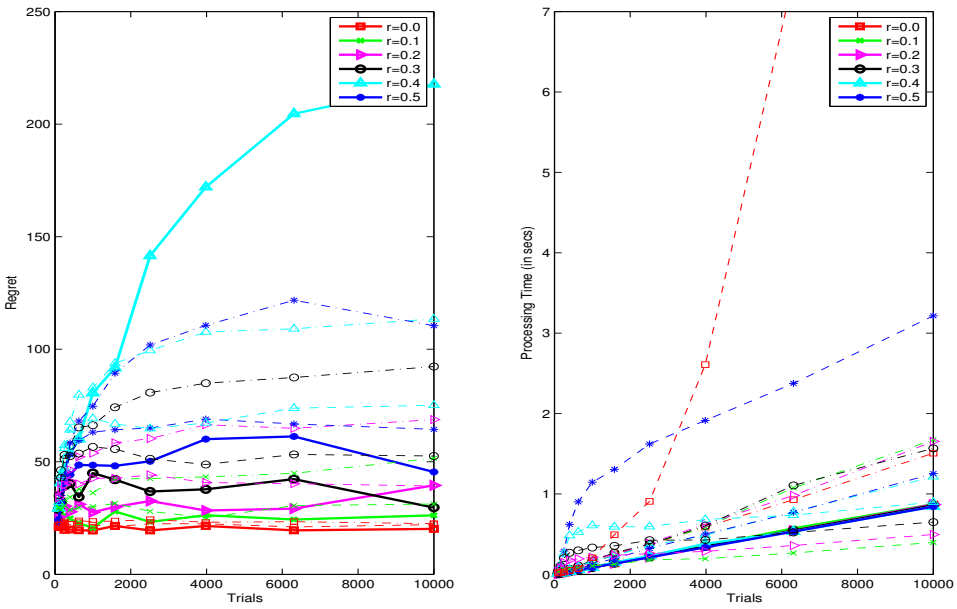

Figure 8.1: Regret and Processing time for the case of 4×4 arms for TS, DeTS, SR algorithms. (-.) line represents TS algorithm, (- -) line represents SR algorithm and (-) solid line represents DeTS algorithm.

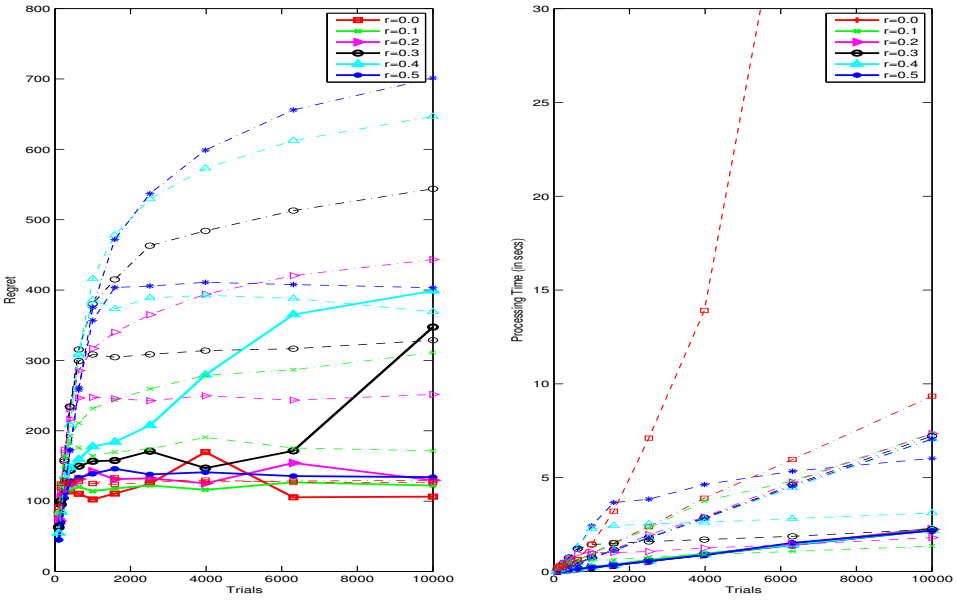

Figure 8.2: Regret and Processing time for the case of 10×10 arms for TS, DeTS, SR algorithms. (-.) line represents TS algorithm, (- -) line represents SR algorithm and (-) solid line represents DeTS algorithm.

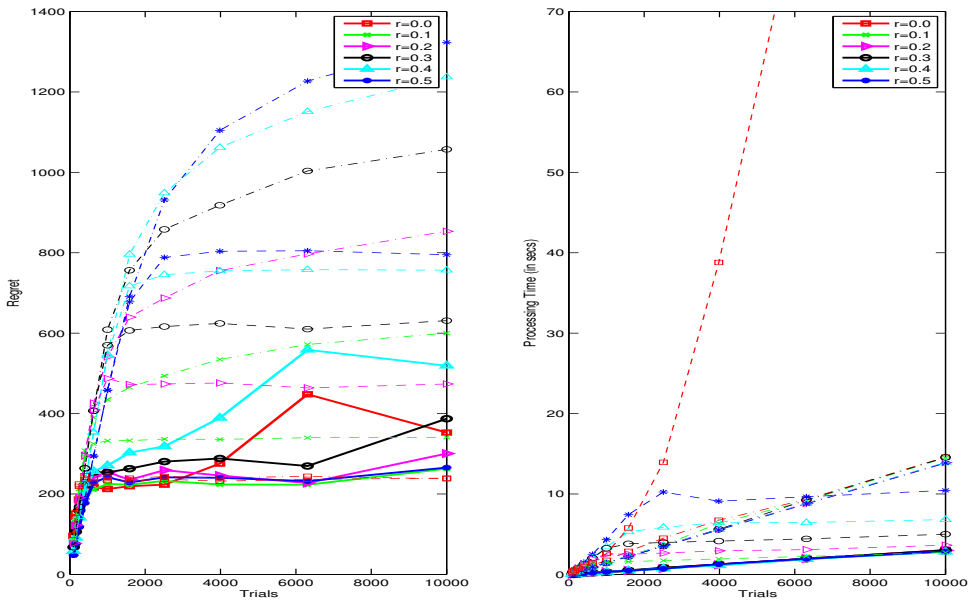

Figure 8.3: Regret and Processing time for the case of 14×14 arms for TS, DeTS, SR algorithms. (-.) line represents TS algorithm, (- -) line represents SR algorithm and (-) solid line represents DeTS algorithm.

8.3 Comparative Analysis

In the section below, we report the experiments done to study and compare Decentralized Thompson Sampling with Successive Reduction and Pure Thompson Sampling algorithms. Though we performed significant experiments over several values of the reward distributions, we only report the most important and relevant experiments in this paper. Each experiment is repeated 100 times, and average values are reported in the results.

The performance of the estimation technique is measured in terms of the *Regret* which is defined as,

$$Regret = \Sigma_{n=0}^{N}(r_*^n - r_k^n)$$

where N is the total number of trials, r_*^n is the Bernoulli output received after playing an arm with the highest θ_n^k at trial n and r_k^n is the reward obtained after sampling the k^{th} arm according to the applied algorithm. Hence, regret is a measure of the loss occurred when using an algorithm as compared to the optimal algorithm. We also report the processing times of the experiments which becomes an important factor when applying the algorithms to real-time applications.

The processing time is the measure of the time taken in seconds for a single iteration of the process to complete a given number of trials. All the experiments are performed on the same machine running Mac OSX 2.8 Ghz, 8 GB memory.

8.3.1 Stochastic Exponential Game Matrix

To show the performance results of our algorithm for a number of different scenarios occurring due to the presence of two different Samplers, we take a generalized $\sqrt{K} \times \sqrt{K}$ game matrix G as shown in Table 8.2. We systematically vary the Game matrix to change from a dominant strategy game with Pure Nash Equilibrium to multiple Nash Equilibria game to see the effect of this change on the performance of different algorithms.

G in its current form is a dominant strategy game with a_1 being the dominant strategy for agent A and $b_{\sqrt{K}}$ being the dominant strategy for agent B.

We slowly modify the matrix G to G' by applying the function $G' = r \times G + (1 - r) \times transpose(G)$ where $r \in \{0.0, 0.1, 0.2, 0.3, 0.4, 0.5\}$ and $q = 1$. The corresponding values of the probabilities for arms $K = 16$ are shown in Table 8.3. We see that the matrix G'_0 and G'_1 have dominant strategies at (a_4, b_1) with highest reward probabilities being 1.0 and 0.9. But the matrices G'_2, G'_3, G'_4 and G'_5 have two Nash equilibria at (a_1, b_4) and (a_4, b_1). (a_4, b_1) is the Pareto Optimal solution for G'_2, G'_3 and G'_4 while both (a_1, b_4) and (a_4, b_1) are both Pareto Optimal solutions for G'_5. We start our first experiment with small number of arms at $K = 16$ (Table 8.3) and then we increase the number of arms to $K = 100, 196$ to show the effect of increased number of arms on the results.

8.3.1.1 Experiment 1: $K = 16$

We perform the experiments for each algorithm - DeTS, SR and TS when the number of arms are $K = 16$. We see that the regret values are minimum for DeTS algorithms for all values of r except when $r = 0.4$ as shown in Fig. 8.1. For the case of $r = 0.4$ in matrix G'_4, the two Nash equilibria are very close to each other at 0.606 and 0.409, which leads to the agents converging to either in the case of decentralized decision making leading to lower performance.

We also notice that the processing time taken by the DeTS algorithm is the minimum while the time taken by the SR algorithm is the maximum. The reason of which is that in the DeTS algorithm the dimensionality of the state space reduces to $\sqrt{K}$ and hence the time taken to select the next arm to play becomes sublinear.

Although in the case of SR algorithm the number of arms are getting successively reduced, it still takes time to compute the probability $P(\theta_i > \theta_k)$ for each arm k, which is the basis for elimination of arms. We did not show the complete graph of the processing time for SR algorithm as it shoots beyond 7 seconds for a single processing of 6,000 trials, which is much higher as compared to the DeTS and TS algorithms.

8.3.1.2 Experiment 2: $K = 100$

In experiment 2, we use the same matrix G' but increase the number of arms to 100, thus each agent has 10 "virtual" arms. For the case of $K = 100$, Matrix G'_0 for $r = 0$ has a single Nash equilibrium at $(a_{10}, b_1) = 1$ and G'_1 for $r = 0.1$

has a Nash equilibria at $(a_{10}, b_1) = 0.90$. G'_2, G'_3, G'_4, G'_5 have two Nash equilibria at (a_{10}, b_1) and (a_1, b_{10}) with values $(0.80, 0.20), (0.70, 0.30), (0.60, 0.40), (0.50, 0.50)$. The results for this experiment are shown in Fig. 8.2, the decentralized solution consistently outperforms the other two solutions both in terms of regret and processing times as the number of trials increase to 10,000 similar to the previous result.

8.3.1.3 Experiment 3: $K = 196$

Since in the real world scenarios, it is common to have large number of arms, we do the third experiment with $K = 196$. The matrix is similarly formulated but as we can see the total number of arms is increased to almost double to that of matrix in Experiment 2. The results are shown in Fig. 8.3. The performance increase is much more as the number of arms have increased, thus making the sub-linear DeTS algorithm the best performer for large scale MABs.

8.4 Conclusion

In this chapter, we studied centralized and decentralized decision making for Multi-Armed bandit problems for Thompson Sampling methods. Although, in centralized systems the agents make joint decisions, the performance seem to deteriorate significantly as the number of arms increase. The Decentralized systems converge very quickly to the optimal arm in case of Pure Nash Equilibrium. In case of multiple Nash Equilibria, their performance may deteriorate, but decentralized algorithms are very scalable and perform much better when the number of arms increase. The

presence of two agents independently pushes the system towards higher performance in the decomposed arm space, thereby accelerating the learning process.

Chapter 9

Conclusion and Future Work

9.1 Conclusion

Multi-Armed bandit problem, a classic dilemma of "exploration vs. exploitation", has intrigued researchers for about fifty years now. The solution to the multi-armed bandit problem can be broadly divided into two categories - 1) Non-Bayesian 2) Bayesian Techniques. In this dissertation, we have explored multiple aspects of the Bayesian techniques, which is also known as "Thompson Method" for Multi-Armed bandit problems. We introduced several different algorithms – Beta Geometric Probabilistic, Beta/ Thompson Sampling, Beta Geometric Deterministic, Successive Reduction Hoeffding, Successive Reduction Order Statistics, Dynamic Thompson Sampling, and Decentralized Thompson Sampling which handle different aspects of the Multi-Armed Bandit problem and have performed thorough empirical analysis of the algorithms.

We start with a case of Two-armed bandit and present Beta Geometric Probabilistic, Beta/ Thompson Sampling, Beta Geometric Deterministic algorithms as solutions in the Bayesian framework. We then propose, Beta Geometric Probabilistic and Beta/Thompson Sampling, to handle the generalized case of k-armed bandit and perform empirical analysis for the case of rare as well as non-rare events.

With the emergence of online recommendation systems in the Internet domain,

it is not uncommon to have a few hundred arms, hence algorithms need to be designed for large scale Multi-Armed Bandits. In this dissertation, we introduced a rather radical strategy the Successive Reduction (SR) strategy that addresses the above weakness directly. Successive Reduction algorithms use Thompson Method for arm selection, however, concurrently, successively eliminate bandit arms from further consideration. We investigated two elimination criteria, one based on so-called Hoeffding Bounds and another one based on Order Statistics. The purpose was to increase reward probability by not considering inferior arms, and at the same time reduce information storage and management needs.

In the next part of the dissertation, we removed the assumption of θ being static and introduced a Dynamic Thompson Sampling method which modifies the estimate of the $\hat{\theta}$ based on exponential filtering. The only other solution which exists in this field is the UCB_f solution. The Dynamic Thompson Sampling method tracks θ very closely and is able to provide better solutions as compared to the UCB_f.

The presence of multiple decision makers is very common in current day systems, these decentralized systems with multiple agents are typically autonomous, maintain local view of information, and have a finite cost of communication. In last part of this dissertation, we compare centralized vs. decentralized systems in the Multi-Armed Bandit setting. The Decentralized Decision Making can be modeled as a *Game Theory* problem. Our results show that the Decentralized systems perform well for both the cases of Pure as well Mixed Nash equilibria and their performance scales well with the increase in the number of arms due to reduced dimensionality of the space.

9.2 Future Work

There are immense possibilities of extending the Bayesian Multi-Armed Bandits in the future.

One of the directions in which the bandits can be extended is the case of mortal and finite horizon bandits. The knowledge of finite horizon N can be helpful in deciding the online strategy as exploration and exploitation can be done dependent upon N. Moreover, it is also possible to have a non permanent set of arms which can become alive and dead in the middle of the trials. For example, in the case of Online advertising, the budget of an Internet advertiser can be limited which determines the total number of impressions, if the advertiser pays per impression. Hence, this problem can be directly mapped to the problem of finite horizon. Moreover, new advertisements can be introduced in the middle of the campaign and advertisements may be removed during the course of the campaign.

It would be interesting to see how Bayesian algorithms can be modified to incorporate finite- horizon mortal bandits. In literature, we have seen papers in this area using variants of confidence bound based and epsilon greedy algorithms [6]. We would like to extend *Bayesian Algorithms* towards this direction and perform empirical analysis to see how they perform.

Another interesting direction of future work will be to try the Decentralized Thompson Algorithm for other types of games such as common-interest games, zero-sum games, to name a few.

Bibliography

[1] Rajeev Agrawal. Sample mean based index policies with o(log n) regret for multi-armed bandit problem. *Advances in Applied Probability.*, 27:1054–1078, November 1995.

[2] Peter Auer, Nicolo Cesa-Bianchi, and Paul Fischer. Finite time analysis of multi-armed bandit problem. *Machine Learning*, 27(2-3):235–256, 2002.

[3] Peter Auer, Nicolo Cesa-Bianchi, Yoav Freund, and Robert E. Schapire. Gambling in a rigged casino: The adversarial multi-armed bandit problem. In *36th Annual Symposium on Foundations of Computer Science*. IEEE, 1995.

[4] Avraham Bab and Ronen I. Brafman. Multi-agent reinforcement learning in common interest and fixed sum stochastic games: An experimental study. *Journal of Machine Learning Research.*, pages 2635–2675, 2008.

[5] George E. P. Box and George C. Tao. *Bayesian Inference in Statistical Analysis*. John Wiley Sons, Inc., Hoboken, New Jersey, 1992.

[6] Deepayan Chakrabarti, Ravi Kumar, Filip Radlinski, and Eli Upfal. Mortal multi-armed bandits. In *NIPS*, 2008.

[7] J C Gittins. Bandit processes and dynamic allocation indices. *Journal of Royal Statistical Society. Series B*, 41(2):148–177, 1979.

[8] Thore Graepel, Joaquin Quinonero Candela, Thomas Borchert, and Ralf Herbrich. Web-scale bayesian click-through rate prediction for sponsored search advertising in microsofts bing search engine. In *ICML*, 2010.

[9] Ole-Christoffer Granmo. The bayesian learning automaton empirical evaluation with two-armed bernoulli bandit problems. *Research and Development in Intelligent Systems XXV*, pages 235–248, 2009.

[10] Ole-Christoffer Granmo. Solving two-armed bernoulli bandit problems using a bayesian learning automaton. *International Journal of Intelligent Computing and Cybernetics*, 2(3):207–234, 2010.

[11] Ole-Christoffer Granmo and Stian Berg. Solving non-stationary bandit problems by random sampling from sibling kalman filters. In *Twenty Third International Conference on Industrial, Engineering, and Other Applications of Applied Intelligent Systems (IEA-AIE 2010)*, 2010.

[12] Sudipto Guha, Kamesh Munagala, and Peng Shi. Approximation algorithms for restless bandit problems. *CoRR*, abs/0711.3861, 2007.

[13] Arjun K. Gupta and Saralees Nadarajah. *Handbook of Beta Distribution and its applications*. Marcer Dekker Inc., New York, 2004.

[14] Neha Gupta, Ole-Christoffer Granmo, and Ashok Agrawala. Successive reduction of arms in multi-armed bandits. *Research and Development in Intelligent Systems XXVII*, November 2011.

[15] Neha Gupta, Ole-Christoffer Granmo, and Ashok Agrawala. Thompson sampling for dynamic multi-armed bandits. In *ICMLA*, 2011.

[16] Shaun Hargreaves-Heap and Yanis Varoufakis. *Game Theory: A Critical Introduction*, chapter 2, pages 40–79. Second edition, 1995.

[17] Wassily Hoeffding. Probability inequalities for sums of bounded random variables. *Journal of the American Statistical Association*, 58(301):13–30, 1963.

[18] Jean-Yves, Sebastien Bubeck, and Remi Munos. Best arm identification in multi-armed bandits. In *COLT*, 2010.

[19] Leslie Pack Kaelbling, Michael L Littman, and Andew W Moore. Reinforcement learning: A survey. *Journal of Artificial Intelligence Research*, 4:237–285, 1996.

[20] T. L. Lai and H. Robbins. Asymptotically efficient adaptive bandit rules. *Advances in Applied Mathemetics*, 1985.

[21] Lihong Li, Wei Chu, John Langford, and Robert E. Schapire. A contextual-bandit approach to personalized news article recommendation. In *WWW*, 2010.

[22] Spyros Makridakis, Steven C Wheelwright, and Rob J Hyndman. *Forecasting Methods and Applications*, chapter 4, pages 135–179. John Wiley and Sons, Inc., third edition, 1998.

[23] Oded Maron and Andrew W. Moore. Hoeffding races: Accelerating model selection search for classification and function approximation. In *NIPS*, 1994.

[24] Benedict C. May, Nathan Korda, Anthony Lee, and David S. Leslie. Optimistic bayesian sampling in contextual-bandit problems. *Submitted to the Annals of Applied Probability*.

[25] Sandeep Pandey and Christopher Olston. Handling advertisements of unknown quality in search advertising. In *NIPS*, 2006.

[26] Aleksandrs Slivkins and Eli Upfal. Adapting to changing environment: the brownian restless bandits. In *COLT*, 2008.

[27] Ming Tan. Multi-agent reinforcement learning: Independent vs. cooperative agents. In *ICML*, 1993.

[28] William R. Thompson. On the likelihood that one unknown probability exceeds another in view of the evidence of two samples. *Biometrika*, 1933.

[29] Katja Verbeeck, Ann Now, Maarten Peeters, and Karl Tuyls. Multi-agent reinforcement learning in stochastic single and multi-stage games. *Adaptive Agents and Multi-Agent Systems II*, 3394(324):275–294, 2005.

[30] Katja Verbeeck, Ann Nowe, Tom Lenaerts, and Johan Parent. Learning to reach the pareto optimal nash equilibrium as a team. *AI 2002: Advances in Artificial Intelligence*, 2557(2002):407–418, 2002.

[31] Joannes Vermorel and Mehryar Mohri. Multi-armed bandit algorithms and empirical evaluation. In *ECML*, 2005.

[32] P Whittle. Restless bandits: Activity allocation in a changing world. *Journal of Applied Probability*, 25:pp. 287–298, 1988.

[33] J. Wyatt. Exploration and inference in learning from reinforcement. *Ph.D. thesis, University of Edinburgh*, 1997.