Dueling Bandits Black Boxes

Amit Wolfenfeld

Dueling Bandits Black Boxes

Research Thesis

Submitted in partial fulfillment of the requirements
for the degree of Master of Science in Electrical Engineering

Amit Wolfenfeld

Submitted to the Senate of
the Technion — Israel Institute of Technology
Tishrei 5776 Haifa September 2015

The research thesis was done under the supervision of Prof. Nir Ailon in the Computer Science Department.

Contents

List of Figures

Abstract

In machine learning, the notion of multi-armed bandits refers to a class of online learning problems, in which a learner (also called decision maker or agent) explores and exploits a given set of choice alternatives in the course of a sequential decision process. In the standard setting, the learner learns from stochastic feedback in the form of real-valued rewards.

We study a partial information online learning problem where actions are restricted to noisy comparisons between pairs of alternatives - The Dueling Bandits Problem. As opposed to conventional approaches which requires the real valued reward of the chosen arms to be numerical and observable, the Dueling Bandits setting only assumes that the (noisy) stochastic feedback about the relative reward of two chosen arms is readily available. This type of relative feedback, or pairwise preference, is particularly fitting in cases where numerical or absolute rewards are difficult to quantify (for instance, user preference of a set of search engine results, preference of taste), but where pairwise preference are easy to make.

In this paper we propose several new methods for the Dueling Bandit Problem. Our approach extends the Doubler and Sparring algorithm proposed on [4]. We show empirical results using real data from Microsoft Research's LETOR project.

Abbreviations and Notations

For the convenience of the reader we have included a table of all the definitions:

T	—	Horizon
t	—	Round
X	—	Arms space
$K = \|X\|$	—	Total Number of Arms
$x_t \in X$	—	Left arm
$y_t \in X$	—	Right arm
$u_t \in \mu_t$	—	Left utility
$v_t \in \mu_t$	—	Right utility
b_t	—	Observed feedback
$R_U(T)$	—	Total Utility Based Regret till T
$R_P(T)$	—	Total Preference Based Regret till T
$\mathcal{P}$	—	Set of potential arms
$\hat{P}_{x,y}$	—	Estimate of $P(x > y)$
$\hat{C}_{x,y}$	—	Confidence interval of - $(\hat{P}_{x,y} - \sqrt{\log(1/\delta)/t}, \hat{P}_{x,y} + \sqrt{\log(1/\delta)/t})$
n_x	—	The number of times arm x has been played
w_x	—	The number of times arm x has won
$\hat{P}_x$	—	w_x/n_x
$W = [w_{x,y}]$	—	Number of wins of arm x over arm y
$G = [g_{x,y}]$	—	Utility function
$\Theta_{x,y}$	—	Random variable with Beta distribution.

MAB	—	Multi Armed Bandit
PBDB	—	Preference Based Dueling Bandits
UBDB	—	Utility Based Dueling Bandits
UCB	—	Upper Confidence Bound
IF	—	Interleaved Filter
BTM	—	Beat The Mean
RUCB	—	Relative Upper Confidence Bound
RCS	—	Relative confidence Sampling
SAVAGE	—	Sensitivity Analysis of Variables for the Generic Exploration
SBM	—	Singleton Bandits Machine
IR	—	Information Retrieval
LETOR	—	Learning to Rank

Chapter 1

Introduction

Multi-armed bandit (MAB) algorithms have received considerable attention and have been studied quite intensely in machine learning since 1985 when Lai and Robbins released their paper [12]. The huge interest in this topic is not very surprising, due to the fact that this MAB setting is not only theoretically challenging but also extremely useful, as can be seen from its use in a wide range of applications. MAB algorithms are used today for solving many problems such as in search engines [16, 1, 10, 19], online advertisement [18, 22, 8], and recommendation systems [13, 14].

The multi-armed bandit problem, or bandit problem for short, is one of the simplest instances of the sequential decision making problem, in which a learner needs to select options from a given set of alternatives repeatedly in an online manner. The name comes from the gambling world in which a gambler decides from a row of slot machines (sometimes known as "one-armed bandits") and decides which machines to play, how many times to play each machine, and in which order to play them. When played, each machine provides a random reward from a distribution specific to that machine. The objective of the gambler is to maximize the sum of rewards earned through a sequence of lever pulls. To be more precise, the learner selects one option at a time and observes a numerical (and typically stochastic) reward, providing information on the quality of that arm. The goal of the learner is to optimize an evaluation criterion such as the error rate (the expected percentage of playing a suboptimal arm) or the cumulative regret (the expected difference between the sum of the rewards actually obtained and the sum of rewards that could have been obtained by playing the best

arm in each round).

In order to minimize the regret, the learner has to face the crucial tradeoff at each trial between "exploitation" of the machine that has the apparent highest expected payoff and "exploration" to get more information about the expected payoffs of the other machines. The learner has to find the best "ratio" between playing the arms that produced high rewards in the past (exploitation) and trying other, possibly even better arms the (expected) reward of which is not precisely known so far (exploration).

There are many variations of the MAB problem and in most of them we assume a numerical reward such as "arm number 1 has the value of 0.7"; however there are many applications where such assumption does not hold. In these applications the feedback is a pairwise comparison in the form of "arm number 1 seems better than arm number 2".

There are many cases in the world of machine learning where precise feedback is not available, and only preference feedback is available. In these cases preference learning must be used, and this is what we study here. to be more precise, in preference learning feedback is typically represented in a purely qualitative way, namely in terms of pairwise comparisons or rankings.

Web search and Internet marketing are two examples that show the importance for the Dueling Bandits setting. A search engine needs to give the user the best result for his or her query. For every query the search engine lets the user select from several options of search results and receives the feedback according to the users choice. This feedback comes in the form of "the first result is preferred over the other results". Another example is when an advertiser aims to sell products online. The advertiser will direct users to his sale page. Every advertiser wants to sell as much as possible; therefore they will want to improve their sale page. In order to improve their sale page the advertiser will create several versions of the pages, split the users between them, and see which one is the top performer. This process is called A/B testing, and each page version is represented by an arm in the Multi Armed Bandit setting. The problem with standard bandits is that there are trends in the market that temporarily decrease or increase the overall performance (Christmas time for instance). Assuming that the arm's order of performance stays the same, meaning the best arm, performance-wise, stays first, the second best arm stays second and so on - Dueling Bandits Algorithms can be used to increase the advertiser's sale performance

while keeping the regret to a minimum. Extending the multi-armed bandit setting to the case of preference-based feedback, i.e., the case in which the online learner is allowed to compare arms in a qualitative way, is therefore a promising idea. Indeed, extensions of that kind have received increasing attention in the recent years. The aim of this thesis is to provide a survey of the state-of-the-art in the field of Dueling Bandits Algorithms and present several new algorithms. In Chapter 2 we provide a scientific background for the Dueling Bandits Problem. In Chapter 3 we survey the state-of-the-art algorithms. In Chapter 4 we present a new algorithm that out performs the algorithm described in Chapter 3. In Chapter 5 we present empirical results.

Chapter 2

Scientific Background

In this chapter we will go into more detail of what the MAB problem is and more importantly the definition of the Dueling Bandits Problem. We discuss two types of settings, the first - Utility Based Dueling Bandit (UBDB) setting and the second Preference Based Dueling Bandit (PBDB) setting.

2.1 Multi Armed Bandits

As described in the previous section, the multi-armed bandit problem is a sequential decision making problem, where a learner explores and exploits a stochastic environment. In this setting, the learner performs actions, referred to as arm pulls. The arms belong to an infinite or finite set X. If the set is finite we denote $|X|$ by K. Each arm $x \in X$ is associated with a probability distribution over $[0, 1]$, with expectation μ_x. Throughout this paper we assume the existence of a unique best arm:

$$x^* = argmax_{x \in X}(\mu_x) \tag{2.1}$$

At each round $t > 0$ the learner "pulls" an arm $x_t \in X$ and acquires a stochastic reward or utility u_t, independently of all previous rewards (i.i.d). For each arm x and for all rounds $t \geq 1$, $n_x(t)$ denotes the number of times arm x has been ''played". In this setup the cumulative regret is defined as follows:

$$R(T) = \sum_{t=1}^{T} \mu_{x^*} - u_t \ . \tag{2.2}$$

The cumulative regret measures the difference between the utility the player could have acquired if he played the best arm and the sum of utilities actually acquired.

2.1.1 UCB Algorithm

The most commonly studied algorithm for the MAB setting [5, 6] is known as UCB. It relies on finding a high confidence upper bound for all arms, as seen in line 4, according to the samples so far, called Upper Confidence Bound (hence the name). The bound for arm x decreases as the number of times it had been sampled so far, n_x, grows. At each round, the sampled arm is the arm that has the highest confidence bound. Typically, an arm can have a high confidence bound either because it hadn't been sampled enough, or because the arm's average reward has been promising.
The UCB algorithm gives the following guarantees:

Theorem 1 *[From [5]]*

Running the UCB algorithm with $|X| = K$, with a finite time horizon of $T > K$, the expected regret is bounded by $R(T) = \mathcal{O}\left(\left(\sum_{x \in X \setminus x^*} \frac{1}{\Delta_x}\right) \log T\right)$, where Δ_x is the gap between the best arm x^* and arm x.

Algorithm 1: UCB

Input: $X, K = |X|$

1 $t \leftarrow 1$
2 $\forall x \in X : \hat{\mu_x} \leftarrow 0$
3 **while** $t \leq T$ **do**
4 $\quad x_t \leftarrow argmax_{x \in X}\left(\hat{\mu}_x + \sqrt{\frac{2 \ln t}{n_x}}\right)$
5 $\quad$ play arm x_t and acquire u_t
6 $\quad$ update $\hat{\mu}_x$

Figure 2.1: UCB Algorithm

2.1.2 Thompson Sampling Algorithm

One of the earliest algorithms, given by W. R. Thompson, dates back to 1933 [20]. The basic idea is to choose an arm to play according to its perceived probability of being the best arm, based on a Beta prior.

The Thompson Sampling algorithm proceeds as follows. The original analysis for this algorithm is based on binary feedback, meaning that the utility of an arm is either 1 (a "success" henceforth) or 0 (a "failure" henceforth).

The algorithm maintains the number of successes and failures for each arm, and holds a random variable with β-distribution for each arm $\Theta_x \sim Beta(Success_x + 1, Failures_x + 1)$. At each round all the random variables $\Theta_{x,t}$ are sampled. The chosen arm is then given by $x_t \in argmax_{x \in X} \Theta_{x,t}$. While the theoretical behaviour of Thompson sampling has remained elusive for a long time, fairly good understanding of its theoretical properties was achieved by Agrawal and Goyal [2, 3], proving the first logarithmic regret bound:

Theorem 2 *[From [2]]*

Running the Thompson Sampling algorithm with $|X| = K$, with a finite time horizon of $T > K$, the expected regret is bounded by $R(T) = \mathcal{O}\left(\left(\sum_{x \in X \setminus x^*} \frac{1}{\Delta_x^2}\right)^2 \log T\right)$, where Δ_x is the gap between the best arm x^* and arm x.

The pseudo-code of the Thompson Sampling algorithm can be seen in Figure 2.2.

Algorithm 2: Thompson Sampling

Input: $X, K = |X|$

$t \leftarrow 1$

$\forall x \in X : Success_x \leftarrow 0$

$\forall x \in X : Fails_x \leftarrow 0$

while $t \leq T$ **do**

  $\forall x \in X : \Theta_{x,t} \sim Beta(Success_x + 1, Fails_x + 1)$

  $x_t \leftarrow argmax(\Theta_{x,t})$

  play arm x_t and acquire u_t

  $Success_{x_t} \leftarrow Success_{x_t} + u_t$

  $Fails_{x_t} \leftarrow Fails_{x_t} + (1 - u_t)$

Figure 2.2: Thompson Sampling Algorithm

2.2 Dueling Bandits

To formalize the problem of learning from preferences, we consider the following interactive online learning model for the K-armed dueling bandit problem [23, 24]. At each iteration t, the learning system chooses not one but two arms $x_t, y_t \in X$, where as before, X is the set (either finite or infinite) of possible actions. The two arms are compared by a stochastic process, and the feedback is returned in the form of a binary random variable b_t, encoding which arm beat the other.

There are various ways to define the stochastic comparison process. In this thesis we will study two different settings.

In the first, Utility Based Dueling Bandits **UBDB**, each arm $x \in X$ is (just as in standard MAB) equipped with a latent expected utility. When two arms x_t, y_t are played, just as in MAB random utilities are drawn independently for each arm. However, unlike MAB these utilities are *not* viewed. As stated above, only a noisy comparison is viewed. The noisy compaison "respects" the drawn utilities in the sense that the arm with higher utility has a better chance of beating the other. Pictorially, this can be shown as a match between two boxers both of whom have a latent expected skill level. The actual performance of each boxer during the match is a random variable distributed around his corresponding expectation, and the match outcome

is a coin flip biased toward the boxer with the higher performance.

In the second - **PBDB** – Preference Based Dueling Bandits, the outcome of the comparison between any two arms behaves like a biased coin flip, where the bias depends on the identity of the two arms only. The latent matrix defining these random coins, which we denote by P, does not necessarily have any structure, although some of the algorithms we study below for PBDB will assume some social choice theoretical properties that will be explained below. Note that in PBDB there is no underlying utility for each arm. This makes the definition of regret challenging. We will borrow some standard definitions from the literature.

It is important to note that UBDB is a special case of PBDB in which the matrix P is induced by a value function assigned to the different arms. Although in a sense UBDB is the simplest form of dueling bandits, we will show below that the theory behind it is nevertheless rich.

2.2.1 UBDB

At iteration t, let u_t and v_t be the drawn (hidden) utilities of the chosen arms x_t and y_t, respectively. We assume, as always, these utilities are in $[0, 1]$. The learner is rewarded with the average utility

$$U_{av}(t) = (u_t + v_t)/2$$

of the two actions it presents, but it does not observe this reward. As explained above, instead, it only observes a binary choice among the two alternative arms x_t, y_t, which depends on the respective utilities u_t and v_t. In particular, we model the observed choice as a binary random variable b_t, which takes the value 1 if x_t (of utility u_t) wins, otherwise 0. A link function $\phi : [0, 1] \times [0, 1] \to [0, 1]$ posits the connection between the two utilities and the distribution of b_t, as follows:

$$\begin{cases} P(b_t = 1|u_t, v_t) = \phi(u_t, v_t) \\ P(b_t = 0|u_t, v_t) = \phi(v_t, u_t) \ . \end{cases} \tag{2.3}$$

Clearly, the link function has to satisfy $\phi(A, B) + \phi(B, A) = 1$. Henceforth we will use the notation $x_t \succ y_t$ for the even ''x_t beats y_t in round t". The link function ϕ, which is assumed to be known, quantitatively determines

how to translate the utilities u_t, v_t to winning probabilities. The linear link function ϕ_{lin} is defined by

$$P(b_t = 1|u_t, v_t) = \phi_{lin}(u_t, v_t) = \frac{1 + v_t - u_t}{2} \in [0, 1] \ . \tag{2.4}$$

For the UBDB case the definition of the regret is very natural and straightforward:

$$R_U(T) = \sum_{t=1}^{T} \mu_{x^*} - U_{av}(t) \ . \tag{2.5}$$

Where, as always, $x^* = argmax_{x \in X} \mu_x$. This implies that the expected zero regret is achievable by setting $(x_t, y_t) = (x^*, x^*)$. It should be also clear that playing (x^*, x^*) is pure exploitation, because the feedback is then an unbiased coin with zero exploratory information.

2.2.2 PBDB

Consider a fixed set of arms $X = \{x_1, .., x_k\}$. As actions, the learner performs a comparison between any pair of arms x_t and y_t, meaning the action space is identified with the set of index pairs $(x, y) \in X \times X$. In this work we characterize the feedback of the comparison by an unknown preference matrix P, which is not necessarily devised from a latent utility. More precisely

$$P = [p_{x,y}] \in [0, 1]^{K \times K} \ . \tag{2.6}$$

To be more precise - for each pair of arms (x, y), this relation specifies the probability of the event

$$Pr(x \succeq y) = p_{x,y} \tag{2.7}$$

of observing a preference for x in a direct comparison with y. Meaning, each $p_{x,y}$ defines a Bernoulli distribution.

When two arms (x_t, y_t) are played and compared at time t, the outcome bit b_t is distributed as an independent Bernoulli , without any dependencies on the previous iterations. The relation matrix P is reciprocal in the sense that $p_{x,y} = 1 - p_{x,y}$ for all $x, y \in X$. Arm x is said to outperform arm y if $p_{x,y} > 1/2$, meaning the probability of winning in a pairwise comparison

is larger for x than it is for y. The closer $p_{x,y}$ is to $1/2$, the harder it is to distinguish between arm x and arm y based on a finite sample set from $Pr(x \succeq y)$. This resembles the case in the standard MAB problem where the gap $\Delta_{x,y}$ is very small.

Defining a regret is more tricky in PBDB. In particular, it is necessary to make assumptions on P for the definition to make sense. In [25] "Relaxed Stochastic Transitivity" is assumed, defined by: For any triplet of arms $x \succ y \succ z$ (with respect to some latent underlying order) and some $\gamma \geq 1$, we assume

$$\gamma\Delta_{x,z} \geq \max(\Delta_{x,y}, \Delta_{y,z}) \ .$$

Where $\Delta_{x,y}$ is defined

$$\Delta_{x,y} = p_{x,y} - 1/2 \tag{2.8}$$

In a later paper [21] this assumption was relaxed, and only a Condorcet winner is assumed, where a Condercet winner is defined as an arm x, such that $\forall y, p_{x,y} > 1/2$. Given the existence of a Condorcet winner which we denote x^*, we define regret for each time-step as follows [25]: if arms x and y were chosen for comparison at time t, then the regret at that time is $\frac{\Delta_{x^*,x_t}+\Delta_{x^*,y_t}}{2}$ and the cumulative regret is

$$R_P(T) = \sum_{t=1}^{T} \frac{\Delta_{x*,x_t} + \Delta_{x*,y_t}}{2} \ . \tag{2.9}$$

2.2.3 The Relation Between Preference and Utility based Regrets

In the extreme case where the preference matrix P is induced by **UBDB**, we argue that the regret defined in section 2.2.1 (using the linear link function) is the same as in section 2.2.2. More precisely, we will show that using the definition of the linear link function both utility based regret and preference based regret are the same (up to a factor of 2):

$$p_{x^*,y} = \phi_{lin}(\mu_{x^*}, \mu_y) = \frac{1 + \mu_y - \mu_{x^*}}{2} \tag{2.10}$$

Incorporating (2.10) in the definition of $\Delta_{x,y}$ we get

$$\Delta_{x^*,y} = p_{x^*,y} - \frac{1}{2} = \frac{\mu_{x^*} - \mu_y}{2}$$

And so the total regret is defined:

$$R_P(T) = \sum_{t=1}^{T} \frac{\Delta_{x^*,x_t} + \Delta_{x^*,y_t}}{2} = \sum_{t=1}^{T} \frac{\frac{\mu_{x^*} - \mu_{x_t}}{2} + \frac{\mu_{x^*} - \mu_{y_t}}{2}}{2} =$$

$$\frac{1}{2}\sum_{t=1}^{T} \mu_{x^*} - \frac{\mu_{x_t} + \mu_{y_t}}{2} = \frac{1}{2}\sum_{t=1}^{T} \mu_{x^*} - U_{av} = \frac{1}{2} R_U(T) \ .$$

2.3 Probability Estimation

The Dueling Bandit game is played in discrete rounds, either through a finite time horizon or an infinite horizon. As described in the previous section, the learner compares between two arms in each round $t > 0$. And so, in each round t, the learner selects a pair of arms x_t, y_t and observes

$$\begin{cases} x_t \succeq y_t & \text{with probablity } p_{x_t,y_t} \\ y_t \succeq x_t & \text{with probablity } p_{y_t,x_t} \end{cases} \tag{2.11}$$

In this thesis the pairwise probabilities $p_{x,y}$ can be estimated according to the finite sample sets. We consider the set of rounds among the first t iterations, in which the learner decides to compare arms x and y, and denote the size of this set by $n_{x,y}$, or the number of times x and y have been compared. We denote the number of times x "beat" over y by $w_{x,y}$ and $w_{y,x}$ the number of times y "beat" x. It is easy to see that $n_{x,y} = n_{y,x} = w_{x,y} + w_{y,x}$ and so the unbiased estimation of $p_{x,y}$ up to iteration t is then given by

$$\hat{p}_{x,y} = \frac{w_{x,y}}{n_{x,y}} = \frac{w_{x,y}}{w_{x,y} + w_{y,x}} \tag{2.12}$$

As mentioned above, in this thesis we assume that the samples are independent and identically distributed. The quantity $\hat{p}_{i,j}$ is an inbiased estimator of $p_{x,y}$. As in most MAB algorithms a high probability confidence interval is obtained by the Hoeffding bound which we remind the reader in Theorem 3

below. The confidence intervals may differ from one algorithm to another, but usually they are of the form

$$[\hat{p}_{x,y} - c(x,y), \hat{p}_{x,y} + c_{x,y}] \ .$$

We will henceforth say that x outperforms arm y with high likelihood if the upper end of the corresponding confidence interval is above $1/2$, namely if $p_{x,y} + c_{x,y} > 1/2$. Similarly, we say that x is beaten by arm y with high confidence if , if $p_{x,y} + c_{x,y} < 1/2$.

2.3.1 Hoeffding Inequality

As a preliminary step, we review Hoeffding's inequality.

Theorem 3 *[9] Suppose $\{X_1, .., X_N\}$ are independent random variables with values in the interval $[a, b]$. We denote $\bar{X}_N = \sum_{i=1}^{N} \frac{X_i}{N}$.*
If $E[\bar{X}_N] = \mu_X$, then for any $d > 0$:

$$Pr(\bar{X}_N \geq \mu_X + d) \leq e^{-\frac{2d^2}{(b-a)^2} \cdot \frac{1}{N}} \tag{2.13a}$$

$$Pr(\bar{X}_N \leq \mu_X - d) \leq e^{-\frac{2d^2}{(b-a)^2} \cdot \frac{1}{N}} \tag{2.13b}$$

Chapter 3

Survey of Algorithms for Dueling Bandits

We start by surveying dueling bandit algorithms. In this section we have included Interleaved Filtering [23], Beat the Mean Bandit [25], RUCB [26], RCS [27], SAVAGE [21] and the Sparring and Doubler algorithms [4] for they served as a foundation for our new approach. We categorize these algorithm based on the following criteria - Explore then Exploit algorithms and Pure Exploitation algorithms.

3.1 Explore then Exploit Algorithms

In the finite horizon case, most PBDB algorithms are based on the idea of splitting the rounds into two phases, exploration and then exploitation. In the first phase the algorithm identifies the best arm with high probability. In the second phase the algorithm repeatedly compares the chosen arm to itself.

The main drawback for these algorithms is that T, the horizon, needs to be known in advance. The horizon is needed for the algorithm to control the trade-off between exploration and exploitation and to control the regret accumulated in the event of failure to identify the best arm in the exploration phase. To be more specific, let's assume the algorithm can identify the best arm with probability of at least $1 - \delta$. We will set δ to $1/T$ and so the algorithm can guarantee that the best arm is chosen within T rounds with

probability greater than $1 - 1/T$. Let's define $R_{explore}(T), R_{exploit}(T)$ as the regrets for the exploration phase and exploitation phase that define the total regret:

$$R(T) = R_{explore}(T) + R_{exploit}(T) \ .$$

Assuming the algorithm made a mistake and did not choose the best arm at the end of exploration phase, the regret accumulated, $R_{explore}(T) + R_{exploit}(T)$ is at most $\mathcal{O}(T)$, since the regret accumulated in each round is bounded by 1. Assuming the algorithm did choose the best arm, $R_{exploit}(T)$ is identically 0. Consequently, the expected regret of an explore-then-exploit algorithm is

$$E[R(T)] \leq (1 - 1/T)E[R_{explore}(T)] + (1/T)\mathcal{O}(T) = \mathcal{O}(E[R_{explore}(T)] + 1) \tag{3.1}$$

The same argument holds for the case of high probability regret bounds in the explore-then-exploit framework. In summary, the performance of an explore-then-exploit algorithm is bounded by the performance of the exploration algorithm. Although Explore and Exploit algorithms need to know the Horizon T in advance, by using a simple workaround, the "Squaring Trick", these algorithm can be used in the infinite horizon setting. An estimated horizon $\hat{T}$ is submitted to the algorithm and once the algorithm reaches $t = \hat{T}$ the horizon is set to $\hat{T}^2$ and the algorithm restarts. This process continues until the algorithm terminates, and hence the name "Squaring Trick".

3.2 Pure Exploration Algorithms

In the infinite time horizon case no horizon is specified and the game continues indefinitely. The algorithm must minimize regret by constantly decreasing the frequency of comparisons involving suboptimal arms. The algorithm continues until certain stopping condition is satisfied unlike Explore then Exploit algorithms that terminate after a predefined horizon T.

3.3 The Algorithms

3.3.1 Interleaved Filter

Yue et al. [23] propose an algorithm for PBDB in a finite horizon setting. In their setting, they assume existence of an underlying latent complete linear ordering over the set X. Note that this ordering is not induced by utilities, and is completely nonparameteric. We denote this ordering using $\succ$.

Their algorithm works for the scenario in which the preference matrix satisfies the following properties:

1. Strong stochastic transitivity.

2. Stochastic triangle inequality.

Strong stochastic transitivity means that for any triplet of arms $x \succ y \succ z$ we assume $\Delta_{x,z} \geq \max(\Delta_{x,y}, \Delta_{y,z})$. Stochastic triangle inequality means that for any triplet of arms $x \succ y \succ z$ we assume $\Delta_{x,z} \leq \Delta_{x,y} + \Delta_{y,z}$. The regret is defined as in (2.9). The algorithm implements an explore-then-exploit strategy. This means that the time steps $1..T$ are divided into two phases, the first serving for exploration only (learning) and the second for exploitation only. In particular, in the exploitation phase, the algorithm plays only the apparent optimal action consistently. The exploration phase works by repeated elimination as follows. At each step the algorithm maintains a working set of arms that are potentially optimal. It is shown that with high probability the true optimal arm is consistently in the working set. The elimination is done using a round robin tournament strategy, as shown in line 5 of algorithm 3.1. Each arm in the working set is compared against all the arms in the working set, until sufficient confidence is achieved to determine whether the arm must be eliminated (removed from the working set). This is encoded in the following condition (line 10):

$$\hat{p}_{x,y} + c_{x,y} < 1/2 \ .$$

Apart from the round robin elimination, a pruning process is executed, as we now explain. As can be seen in line 12, an element y is removed from the working set $\mathcal{P}$ if

$$\hat{p}_{x,y} - c_{x,y} < 1/2$$

In words, this condition implies with high confidence that y is inferior compared to the current candidate in the round robin. Once a single arm is left in the working set, the algorithm enters the exploitation phase. The algorithm pseudo-code is presented in Figure 3.1. The guarantee provided is as follows:

Theorem 4 *[From [23]]*

Running the Interleaved Filter algorithm with $|X| = K$, with a known finite time horizon of $T > K$, the expected regret is bounded by $R_P(T) = \mathcal{O}\left(\frac{K \log K}{\Delta_{x^*,y}} \log T\right)$, where y is an arm that is only beaten by x^*.

From the analysis of the algorithm it can be seen that the algorithm returns the best arm with probability of $1 - \frac{1}{T}$. Correspondingly, a suboptimal arm is returned by the algorithm with probability of $\frac{1}{T}$ hence , similar to a PAC type argument: With high probability of at least $1 - \frac{1}{T}$, the estimated regret after T steps is at most $\mathcal{O}\left(\frac{K \log K}{\Delta^*} \log T\right)$. The regret is defined as in (2.9).

Algorithm 3: Interleaved Filter

Input: T,X, $K = |X|$, δ

$t \leftarrow 1$

$\mathcal{P} \leftarrow X$ Choose $x_t \in \mathcal{P}$ randomly

$\mathcal{P} \leftarrow \mathcal{P} \backslash x_t$

while $|\mathcal{P}| > 1 \wedge t \leq T$ **do**

  for $x \in \mathcal{P}$ **do**

    compare(y, x_t)

    update$(\hat{p}_{x_t,y})$

    $t \leftarrow t + 1$

  $c_t \leftarrow \sqrt{log(1/\delta)/t}$

  while $\exists y \in s.t.\ (\hat{p}_{x_t,y} + c_t < 1/2)$ **do**

    $\mathcal{P} \leftarrow \mathcal{P} \backslash \{y\}$

  if $\exists y \in s.t.\ (\hat{p}_{x_t,y} - c_t < 1/2)$ **then**

    $x_{t+1} \leftarrow y$

    $\mathcal{P} \leftarrow \mathcal{P} \backslash \{y\}$

    $\forall x \in \mathcal{P}$ reset $\hat{p}_{y,x}$

return $x_t \in \mathcal{P}$

Figure 3.1: Interleaved Filter Algorithm

3.3.2 Beat The Mean Bandit

Yue and Joachims [25] proposed an explore-then-exploit strategy for a finite horizon setting, very similar to IF. Beat the Mean (BTM) is a preference-based online learning algorithm. This algorithm is based on relaxed assumptions with regards to the IF algorithm:

1. Relaxed stochastic transitivity.

2. Stochastic triangle inequality

Relaxed stochastic transitivity means that for any triplet of arms $x \succ y \succ z$ and for some $\gamma \geq 1$ we assume $\gamma\Delta_{x,z} \geq \max(\Delta_{x,y}, \Delta_{y,z})$. The main idea of BTM is to maintain a score of the "Mean Bandit" (arm in our case) and eliminating arms that are inferior to it with enough confidence. Similar to the IF algorithm, BTM maintains a working set of potentially optimal

arms. At each round the algorithm picks an arm that has the least plays in the corresponding current history, n_x, and compares it with a randomly chosen arm from the working set. Comparing an arm against random arms in the working set is functionally identical to comparing an arm against the "Mean Bandit" sampled uniformly from the working set. The arm, x, that is empirically the worst arm and is separated enough from the best arm with enough confidence is defined using:

$$min_{x\in\mathcal{P}}(\hat{P}_x) + c^* \leq max_{y\in\mathcal{P}}(\hat{P}_y) - c^* \tag{3.2}$$

Once an arm reaches (3.2) (see line 15) it is eliminated from the working set. The confidence is defined as $c^* = c_{\delta,\gamma}(n^*)$, where $c_{\delta,\gamma}(n) = 3\gamma^2\sqrt{\frac{1}{n}\log\frac{1}{\delta}}$. The algorithm pseudo-code is presented in Figure 3.2. The guarantee provided is as follows:

Theorem 5 *[From [25]] Running the Beat the Mean Bandit algorithm with $|X| = K$, with a known finite time horizon of $T > K$, the expected regret is bounded by $R_P(T) = \mathcal{O}\left(\frac{\gamma^7 K}{\Delta_{x^*,y}}\log T\right)$, where y is an arm that is only beaten by x^*.*

Similarly to the analysis of the IF algorithm, it can be seen that the algorithm returns the best arm with probability of $1 - \frac{1}{T}$.

Algorithm 4: Beat the Mean Bandit

Input: T,X, $K = |X|$, $c_{\delta,\gamma}(\cdot)$

$\mathcal{P} \leftarrow X$
$W = [w_{x,y}]$
$\forall x \in \mathcal{P}, n_x \leftarrow 0$
$\forall x \in \mathcal{P}, w_x \leftarrow 0$
$\forall x \in \mathcal{P}, \hat{P}_x \leftarrow w_x/n_x$ or 0 if $n_x = 0$
$t \leftarrow 0$
while $|\mathcal{P}| > 1 \wedge t \leq T$ **do**
    $x_t \leftarrow argmin_{x \in \mathcal{P}} n_x$
    select $y_t \in \mathcal{P}$ at random
    compare(x_t, y_t)
    update(W)
    $n_{x_t} \leftarrow n_{x_t} + 1$
    $n^* \leftarrow \min_{x \in \mathcal{P}} n_x$
    $c^* \leftarrow c_{\delta,\gamma}(n^*)$ or 1 if $n^* = 0$
    if $min_{y \in \mathcal{P}}(\hat{P}_y) + c^* \leq max_{x \in \mathcal{P}}(\hat{P}_x) - c^*$ **then**
        $y \leftarrow argmin_{x \in \mathcal{P}} \hat{P}_x$
        $\forall x \in \mathcal{P}$ delete comparison with y from w_y, n_y
        $\mathcal{P} \leftarrow \mathcal{P} \backslash \{y\}$
    $t \leftarrow t + 1$
return $x_t \in \mathcal{P}$

Figure 3.2: Beat the Mean Bandit Algorithm

3.3.3 RUCB

Relative Upper Confidence Bound by Zoghi et al. [26], adapts the most commonly used algorithm UCB [Algorithm 2.1] in the standard MAB setting to the Dueling Bandit setting (PBDB version). The algorithm assumes that a Condorcet winner exists. A Condorcet winner is a definition borrowed from social choice [7]. Roughly speaking, a Condorcet winner beats all the other arms on average. Mathematically, a Condorcet winner (if exists) is defined by: x is the Condorcet winner if $p_{x,y} > 1/2, \forall y$. The algorithm pseudo-code is presented in Figure 3.3. Similarly, the UCB algorithm RUCB maintains an upper confidence bound g_{x_t,y_t} for each pair of arms, as seen in line 3 of

Algorithm 3.3. On each round, RUCB selects the arms to compare from the set of potential Condorcet winners, meaning the set of arms for which all are potentially Condorcet winners with sufficiently high probability. After selecting the set of potential Condercet winners the algorithm selects the pair of arms with the highest upper confidence bound g_{x_t,y_t}; this can be seen as a Dueling Bandit version of the ''optimism in the face of uncertainty" principle from UCB. Note that in line 5 the comparisons are based on g_{x_t,y_t} and in line 7 of Algorithm 3.3 the comparisons are based on g_{y_t,x_t}, making it more difficult for the algorithm to compare an arm against itself. Thus, RUCB strives to avoid auto-comparisons (comparisons of an item with itself) until there is great certainty that x_t is indeed the Condorcet winner. Roughly speaking, if the algorithm would pick the arm y_t according to g_{x_t,y_t} and not g_{y_t,x_t}, the algorithm will pick the arm according to the impact of x_t and not y_t and is more likely to pick the pair $x_t, y_t = x_t, x_t$. The RUCB algorithm guarantees the following:

Theorem 6 *[From [26]]*

Running the RUCB algorithm with $|X| = K$, with a finite time horizon of $T > K$, the expected regret is bounded by $R_P(T) = \mathcal{O}\left(\frac{KlogT}{\Delta_{x^*,y}}\right)$, where y is an arm that is only beaten by x^*.

Algorithm 5: RUCB

Input: T,$X, K = |X|, \alpha > 1/2, T \in \{1, 2, ...\}$

1 $W = [w_{x,y}]$

2 **while** $t \leq T$ **do**

3 $\quad G = [g_{x,y}] \leftarrow \frac{w_{x,y}}{w_{x,y}+w_{y,x}} + \sqrt{\frac{\alpha \cdot \ln(t)}{w_{x,y}+w_{y,x}}}$

4 $\quad \forall x \in X : u_{x,x} \leftarrow 0$

5 $\quad$ Pick any x_t that satisfies $g_{x_t,x} \geq 1/2, \forall x$.

6 $\quad$ If no x_t exists pick x_t randomly from X.

7 $\quad y_t \leftarrow argmax_y g_{y,x_t}$

8 $\quad$ compare(x_t , y_t)

9 $\quad$ update(W)

10 $\quad t \leftarrow t + 1$

Figure 3.3: RUCB Algorithm

3.3.4 RCS

(Munos et al., 2014) Relative Confidence Sampling [27] is very similar to RUCB in the sense that it aims to minimize cumulative regret and does not eliminate arms from a working set. The algorithm pseudo-code is presented in Figure 3.5. Similarly to the RUCB algorithm, the RCS algorithm's only assumption is the existence of a Condorcet winner.

On each round the algorithm simulates a tournament between all the arms, against all the arms. The simulation is conducted, as seen in line 5, according to a random variable $\Theta_{x,y}$ with $Beta(\alpha, \beta)$ distribution. The hyper-parameters, α, β, are $w_{x,y} + 1, w_{y,x} + 1$, respectively, where $w_{x,y}$ is the number of times arm x beat arm y. The main idea behind this simulation technique is to use the superior performance of Thompson Sampling [2] in the same manner it is used in the K-armed bandit setting. Once the simulation is over, similarly to the RUCB algorithm, the RCS algorithm select a set of potential Condorcet winners, as seen in line 8, and by this the algorithm selects arm x_t. In line 12 the algorithm selects the arm y_t according to the "optimism in the face of uncertainty" principle, the same as the UCB algorithm. After both arms are selected and compared, the algorithm updates $w_{x,y}$ for all the arms and continues to the next round. Although no theoretical guarantees were presented in [27], according to the experiments there, RCS outperforms the RUCB algorithm for a small number of arms (less than 40). In Figure 3.4 we can see the results of the experiments Munos et al. performed on the MSLR-WEB30K dataset. To evaluate the RCS algorithm, Munos et al. applied it to the problem of scorer (which was inferred as ranker in the original paper) evaluation from the field of information retrieval (IR). A ranker is a function that takes a user's search query and ranks the documents in a collection according to their relevance to that query. Ranker evaluation aims to determine which set of rankers performs best. Given a set of K rankers, the problem of finding the best ranker can then be modelled as a K-armed Dueling Bandit problem, with each arm relating to a ranker. These experiments are built on real IR data from the MSLR-WEB30K dataset. We will go into more detail on the experiments in chapter 5. Using this data set, 20, 30, and 40 rankers each relate to a ranking feature provided in the data set, e.g., PageRank. The ranker evaluation task then continues to conclude which single feature is the

best ranker (Hofmann et al., 2013)[11].

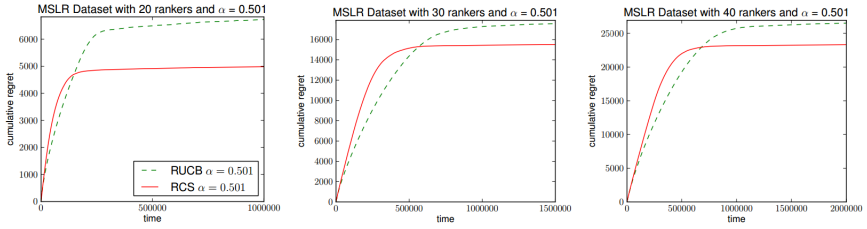

Figure 3.4: Cumulative regret averaged over 90 runs

Algorithm 6: RCS

Input: T,$X, K = |X|, \alpha > 1/2, T \in \{1, 2, ...\}$

$W = [w_{x,y}]$

while $t \leq T$ **do**

  assume an arbitrary injective mapping index: $X \to [K]$,

  inducing an arbitrary canonical order on X.

  for $x, y \in X$ *s.t.* $index(x) < index(y)$ **do**

    $\Theta_{x,y}(t) \sim Beta(w_{x,y} + 1, w_{y,x} + 1)$

    $\Theta_{y,x}(t) = 1 - \Theta_{x,y}(t)$

  Pick any x_t that satisfies $\Theta_{x_t,x} \geq 1/2, \forall x$.

  If no x_t exists pick x_t that was chosen least often ($argmin_x n_x$).

  $G = [g_{x,y}] \leftarrow \frac{w_{x,y}}{w_{x,y}+w_{y,x}} + \sqrt{\frac{\alpha \cdot \ln(t)}{w_{x,y}+w_{y,x}}}$

  $G_{z,z} \leftarrow 0$ for each $z \in X$

  $y_t \leftarrow argmax_y g_{y,x_t}$

  compare(x_t, y_t)

  update(W)

  $t \leftarrow t + 1$

Figure 3.5: RCS Algorithm

3.3.5 SAVAGE

(Uryoy et al., 2013)[21] Sensitivity Analysis of Variables for the Generic Exploration (SAVAGE) algorithm is another explore-then-exploit algorithm, as are BTM and IF. As in RUCB and RCS this algorithm runs under the assumption that a Condorcet winner exists. The algorithm pseudo-code is presented in Figure 3.6. In general, the algorithm compares pair of arms until there exists a pair for which one of the arms beats the other by a wide margin, in which case the loser is removed from the working set. More precisely, in each round two arms, with the least amount of plays, are chosen, as seen in line 4. The pair of arms are compared and given a score, u_{x_t,y_t} (Line 9). An arm x that does not uphold the condition

$$\sum_y \mathbb{1}(u_{x,y} > 0.5) < K$$

is eliminated from the working set. This condition is very similar to the condition in the RUCB algorithm. Although this algorithm seems like "more of the same", this algorithm eliminates arms from the working set (as seen in IF and BTM) according to the likelihood of the arm being the Condorcet winner, while RUCB and RCS only pick the most likely pair of arms to hold the Condorcet winner. This algorithm guarantees the following:

Theorem 7 *[From [21]]*

Running the SAVAGE algorithm with $|X| = K$, with an unknown infinite time horizon of $T > K$, the expected regret is bounded by

$$R_P(T) = \mathcal{O}\left(K^2 log(T) \sum_{x \in X \setminus x^*} \frac{1}{\Delta^2_{x,x^*}}\right)$$

Algorithm 7: SAVAGE

Input: $X, K = |X|, \alpha$

$\mathcal{P} \leftarrow X$

$\forall x \in \mathcal{P}, n_x \leftarrow 0$

while $|\mathcal{P}| > 1 \wedge t \leq T$ **do**

  $x_t, y_t \leftarrow argmin_x n_x$

  Compare(x_t, y_t)

  $n_{x_t} \leftarrow n_{x_t} + 1$

  $n_{y_t} \leftarrow n_{y_t} + 1$

  update(W)

  $G = [g_{x_t,y_t}] \leftarrow \frac{w_{x,y}}{w_{x,y}+w_{y,x}} + \sqrt{\frac{\alpha}{w_{x,y}+w_{y,x}}}$

  $\mathcal{P} \leftarrow \mathcal{P} \setminus \left\{x \mid \sum_y \mathbb{1}(g_{x,y} > 0.5) < K)\right\}$

  $t \leftarrow t + 1$

return x_t

Figure 3.6: SAVAGE Algorithm

3.3.6 Black-Box Algorithms

In [4], Ailon et al. tried to tackle the dueling bandits problem (UBDB-version) using standard bandit algorithms. In their work, Ailon et al. use MAB algorithm black boxes, defined as Singleton Bandits Machine (SBM) - closed computational units with internal timer and memory. A SBM S supports three operations: reset, advance, and feedback. The reset operation simply clears the SBM's state. The advance operation returns the next chosen arm, and feedback is used for simulating a feedback (the utility).

Doubler (Ailon et al., 2014) handles a large or possibly infinite set of arms X for the infinite horizon case. The algorithms pseudo-code is presented in Figure 3.7. This algorithm is best explained by thinking of a competition between two players; the first controls the choice of the left arm and the second player controls the right arm. The objective of each player is to win as many rounds as possible. This algorithm divides the time axis to exponentially growing epochs p (first epoch is 2 rounds, second epoch is 4, third epoch is 8, and so forth). In each epoch, the left side plays according to a fixed stochastic strategy, which will be explained shortly, while the right one plays adaptively according to a strategy provided by a SBM black-box. In the beginning of a new epoch (line 4) the distribution governing the left arm changes in a way that mimics the actions of the right arm in the previous epoch. More precisely, the left side plays according to a distribution defined by the multi-set $\mathcal{L}$ that holds all the plays the right side performed in the previous epoch. The algorithm guarantees the following:

Theorem 8 *[From [4]]*

Running the Doubler algorithm with $|X| = K$, with an infinite time horizon, the expected regret is bounded by $R_U(T) = \mathcal{O}\left(Hlog^2T\right)$ for all $T > 0$, where $H = \sum_{x \in X \setminus x^*}^{K} \Delta_x^{-1}$ and the regret as in (2.5), and Δ_x is the gap between arm x and the optimal arm.

As can be seen in line 4 of Algorithm 3.7, $\mathcal{S}$ is reset every time a new epoch begins, by removing all the information gained in the previous rounds. This procedure inflicts the extra $\log(T)$ factor to the accumulated regret.

Algorithm 8: Doubler

Input: X, $\mathcal{S}$

```
p ← 1
L ← X
while True do
    reset(S)
    for t = 2^(p−1) to 2^p do
        choose x_t uniformly from L
        y_t ← advance(S)
        compare (x_t, y_t), observe choice b_t
        feedback (S, b_t)
    L ←the multi-set of arms played as y t in the last for-loop.
    p → p + 1
```

Figure 3.7: Doubler Algorithm

Sparring (Ailon et al., 2014)[4] The algorithm pseudo-code is presented in Figure 3.8. The idea for the Sparring algorithm comes from analysis of an adversarial version of UDBD, in which it can be easily shown that the resulting expected regret of Sparring is at most a constant times the regret of the two SBM black-boxes that implement an algorithm for adversarial MAB (i.e., $EXP3$). We do not elaborate on these arguments and on adversarial games in these thesis.

Conjecture 1 *In the paper Ailon et al., 2014 conjectured that the utility based regret of the Sparring algorithm is bounded by the combined regret of the SBM black-boxes, with a possibility of a small additive overhead.*

Proving the conjecture appears to be tricky due to the fact that the left (resp. right) SBM black-box does not see a stochastic environment, because its feedback depends on non-stochastic actions of the two SBM black-boxes.

Algorithm 9: Sparring

Input: $X, \mathcal{L} = \{x_1\}$, $\mathcal{S}_R, \mathcal{S}_L$

$t \leftarrow 1$
while *True* **do**
  $x_t \leftarrow \text{advance}(\mathcal{S}_L)$, $y_t \leftarrow \text{advance}(\mathcal{S}_R)$
  $\text{compare}(x_t, y_t)$
  observe b_t
  $\text{feedback}(\mathcal{S}_L, \mathbb{1}_{b_t=0})$
  $\text{feedback}(\mathcal{S}_R, \mathbb{1}_{b_t=1})$
  $r \leftarrow t + 1$

Figure 3.8: Sparring Algorithm

Chapter 4

Our Approach

We start with two algorithms that try to fix a major shortcoming of the Doubler algorithm. In every epoch, the Doubler algorithm resets the SBM black-box, meaning that it will need to re-learn the environment thus incurring an extra $\log(T)$ factor on the cumulative regret. We present two algorithms, Improved Doubler and Balanced Doubler, that tackle this shortcoming.

4.1 Improved Doubler

The Improved Doubler algorithm's pseudo-code is presented in Figure 4.1. Instead of initializing the right arm's SBM at each epoch and losing all the information that was learnt, we will store an invariant of this data. Let D_p denote the distribution of the left arm in the p'th epoch. Had we known $f_p := E_{x \in D_p} \mu_x / 2$, we could have used $b_t + f_p$ as feedback on the right side. This would have offset the change in the expected utility of the left arm, from epoch to epoch. As a result, we would not have had to reset the SBM at each epoch. We fix this by changing line 9 in the Doubler algorithm to feedback$(\mathcal{S}, b_t + f_p)$. Assuming the link function $\phi = \phi_{lin}$, meaning $\phi(u, v) = \frac{1+v-u}{2}$, it is easy to see that $E(b_t + f_p)$ is exactly $E(\frac{u(y_t)+1}{2})$, meaning we have decoupled the arms' utilities. Of course we do not know f_p, and can only estimate it. Luckily, an unbiased estimate of f_p, $\hat{f}_p$ can be obtained using a simple inductive formula. First we notice $E[f_p]$

is defined by

$$E[f_{p+1}] = \frac{1}{|T_p|} \sum_{t \in T_p} \mu_{y_t}/2 \tag{4.1}$$

thus:

$$E[f_{p+1}] = \frac{1}{|T_p|} \sum_{t \in T_p} E[b_t - \frac{1-\mu_{x_t}}{2}] = \frac{1}{|T_p|} \sum_{t \in T_p} E[b_t + f_{p-1} - \frac{1}{2}] = E[f_{p-1}] + E[b_t - \frac{1}{2}]$$

The estimate for f_p is given by a recursive formula:

$$\hat{f}_p = \hat{f}_{p-1} + \frac{\sum\limits_{t \in T_p} (2b_t - 1)}{|T_{p-1}|} \tag{4.2}$$

We wish to bound $\hat{f}_p$:

$$\hat{f}_{p+1} = \hat{f}_p + \frac{\sum\limits_{t \in T_p} (2b_t - 1)}{|T_p|} = \hat{f}_p + \frac{\sum\limits_{t \in T_p} (2b_t - 1)}{2^p} = \hat{f}_p + \frac{\sum\limits_{t \in T_i} 2b_t}{2^p} - 1 =$$

$$\hat{f}_p + B_p - 1 = \hat{f}_{p-1} + B_{p-1} - 1 + B_p - 1 = ... = \hat{f}_0 + \sum_{j=1}^{p} B_j + p \ ,$$

where $B_p = \frac{\sum\limits_{t \in T_p} 2b_t}{2^p}$ is the average of b_t in epoch p, which is bound by one, meaning $\sum\limits_{j=1}^{p} B_j$ is bound by p thus bounding $\hat{f}$ by $\log(T)$. This is a very rough bound of $\hat{f}_p$, with it we cannot prove that the total regret achieve a better bound than $\mathcal{O}(\log^2(T))$. In our experiments we have witnessed that $\hat{f}_p$ is far smaller than the rough bound. A high probability bound of $\mathcal{O}(1)$ would prove that the total regret is bound by $\mathcal{O}(\log(T))$. We have yet to prove this and leave this for further research.

Algorithm 10: Improved Doubler

Input: X, $\hat{f}_0 = 0$, $\mathcal{S}$

$\mathcal{L} \leftarrow X$
$p \leftarrow 1$
while *True* **do**
  for $t = 2^{p-1}$ **to** 2^p **do**
    choose x_t uniformly from $\mathcal{L}$
    $y_t \leftarrow advance(\mathcal{S})$
    compare (x_t, y_t), observe choice b_t
    feedback $\left(\mathcal{S}, b_t + \hat{f}_{p-1}\right)$
  $\mathcal{L} \leftarrow$the multi-set of arms played as y t in the last for-loop.
  $\hat{f}_p \leftarrow \hat{f}_p + \sum_{s \in T_p} b_s / 2^{p-1} - 1/2$
  $p \rightarrow p + 1$

Figure 4.1: Improved Doubler Algorithm

4.2 Balanced Doubler

Another improvement of the Doubler algorithm is the Balanced Doubler algorithm. In this algorithm we assume we know $min_{x \in X \setminus x^*}(\Delta_x)$. Instead of resetting the SBM on each epoch as in the Doubler algorithm, we will use an estimator as defined in the Balanced Doubler. In order to cancel the offset in the estimator we will use average of averages:

$$\tilde{\mu}(y) = E\left[\frac{1}{P}\sum_{p=1}^{P}\frac{1}{n_{y,p}}\sum_{t \in T_{y,p}} b_t\right] \tag{4.3}$$

where $n_{y,p}$ is the number of times arm y was chosen in epoch p. We will prove that this estimate is an unbiased estimate of μ_y up to a constant that is invariant of y:

$$E\left[\frac{1}{P}\sum_{p=1}^{P}\frac{1}{n_{y,p}}\sum_{t \in T}\frac{\mu_y - \mu_{x_t} + 1}{2}\right] = \mu_y + \frac{1}{2} + E\left[\frac{1}{P}\sum_{p=1}^{P}\frac{1}{n_{y,p}}\sum_{t \in T_{y,p}}\frac{\mu_{x_t}}{2}\right]$$

In short:

$$\tilde{\mu}(y) = \mu_y + C(P) \ . \tag{4.4}$$

We will now show that in order to achieve $\mathcal{O}(\log T)$ regret bound we need a warm up phase. We define s to be the number of times each arm is played in the warm-up phase. Due to the fact that $\Delta_x \in [0,1]$, the total regret can be bound by:

$$R = R_{warm-up} + R_{rest} \leq s \cdot pK + \sum_{t=1}^{\infty} Pr\left(\tilde{\mu}(x_t) > \tilde{\mu}(x^*) \mid x_t \in X \backslash x^*\right)$$

We wish to bound $Pr(\tilde{\mu}(x) > \tilde{\mu}(x^*))$. In the worst case scenario (standard deviation-wise) $n_{x,p} \equiv s$, meaning that arm x was chosen in each epoch only s times, $s \cdot p$ in total. The probability of choosing the wrong arm can be bound by:

$$Pr(\tilde{\mu}(x) > \tilde{\mu}(x^*)) < Pr(\tilde{\mu}(x) > \mu_x + \frac{\Delta_x}{2}) + Pr(\tilde{\mu}(x^*) < \mu_{x^*} - \frac{\Delta_x}{2}) \tag{4.5}$$

Meaning that at epoch p suboptimal arm x was played $s \cdot P$ times and so we can bound the probability of that arm being played again by:

$$Pr(\tilde{\mu}(x) > \mu_x + \frac{\Delta_x}{2}) + Pr(\tilde{\mu}(x^*) < \mu_{x^*} - \frac{\Delta_x}{2}) \leq \sum_{p=1}^{P} 2e^{-\frac{\Delta_x^2}{2} s \cdot p} \tag{4.6}$$

And the regret incurred by arm x can be bound by:

$$R_x < s \cdot P\Delta_x + 2\Delta_x \sum_{p=1}^{P} \left(2^p e^{-\frac{\Delta_x^2}{2} s \cdot p}\right) = s \cdot P\Delta_x + 2\Delta_x \sum_{p=1}^{P} \left(2e^{-\frac{\Delta_x^2}{2} s}\right)^p \tag{4.7}$$

By using the sum of geometric series we can bound the regret by:

$$R_x < s \cdot P\Delta_x + 2\Delta_x \frac{2e^{-\frac{\Delta_x^2}{2} s} - (2e^{-\frac{\Delta_x^2}{2} s})^P}{1 - 2e^{-\frac{\Delta_x^2}{2} s}} \tag{4.8}$$

We wish to find s that will yield the minimum regret. We will split the calculation of s :

$$s < \frac{2\log(2)}{\Delta_x^2}$$

Assuming the above, we get:

$$R_x < s \cdot P\Delta_x + 2\Delta_x \frac{2e^{-\frac{\Delta_x^2}{2}s} - (2e^{-\frac{\Delta_x^2}{2}s})^P}{1 - 2e^{-\frac{\Delta_x^2}{2}s}} \sim s \cdot P\Delta_x - (2e^{-\frac{\Delta_x^2}{2}s})^P \quad (4.9)$$

We will derive and get

$$\begin{aligned}
&P\Delta_x + \Delta_x \frac{P\Delta_x^2}{2}(2e^{-\frac{\Delta_x^2}{2}s})^P = 0 \\
&1 + \frac{\Delta_x^2}{2}(2e^{-\frac{\Delta_x^2}{2}s})^P = 0 \\
&-\frac{2}{\Delta_x^2} = 2^P e^{-\frac{P\Delta_x^2}{2}s} \\
&\log(\frac{\Delta_x^2}{2}) = -\frac{P\Delta_x^2}{2}s + P\log(2) \\
&\log(\frac{\Delta_x^2}{2}) - P\log(2) = -\frac{P\Delta_x^2}{2}s \\
&\frac{P\Delta_x^2}{2}s = P\log(2) - \log(\frac{\Delta_x^2}{2}) \\
&s = \frac{2\log(2)}{\Delta_x^2} - \frac{\log(\frac{\Delta_x^2}{2})}{P\Delta_x^2}
\end{aligned}$$

Taking $P \to \infty$ gives us $s = \frac{2\log(2)}{\Delta_x^2}$. In the second case:

$$s > \frac{2\log(2)}{\Delta_x^2}$$

Assuming the above we get:

$$R_x < s \cdot P\Delta_x + 2\Delta_x \frac{2e^{-\frac{\Delta_x^2}{2}s} - (2e^{-\frac{\Delta_x^2}{2}s})^P}{1 - 2e^{-\frac{\Delta_x^2}{2}s}} \sim s \cdot P\Delta_x + 2\Delta_x \frac{2e^{-\frac{\Delta_x^2}{2}s}}{1 - 2e^{-\frac{\Delta_x^2}{2}s}} \quad (4.10)$$

And from here we can see that the minimum of this function (assuming $s > \frac{2\log(2)}{\Delta_x^2}$) is $\frac{2\log(2)}{\Delta_x^2}$.

Now that we have the number of rounds each arm is played in the warm

start phase we calculate the regret. For a single arm x:

$$R_x < \log(4)P\frac{1}{\Delta_x} + 2\Delta_x \sum_{p=1}^{P} \left(2e^{-\frac{\Delta_x^2}{2}\frac{2\log(2)}{\Delta_x^2}} \right)^p = P(\frac{\log(4)}{\Delta_x} + 2\Delta_x) \quad (4.11)$$

And for all the arms we get:

$$R(P) = P \sum_{x \in X \setminus x^*} (\frac{\log(4)}{\Delta_x} + 2\Delta_x) \quad (4.12)$$

And where P is of $\mathcal{O}(\log(T))$ we get the regret is of $\mathcal{O}(\log(T))$. And we conclude with the following theorem:

Theorem 9 *Running the Balanced Doubler algorithm with $|X| = K$, with an infinite time horizon, the expected regret is bounded by $R(T) = \mathcal{O}(logT)$ for all $T > 0$.*

Algorithm 11: Balanced Doubler

Input: X, $\mathcal{S}$

$\mathcal{L} \leftarrow X$
$p \leftarrow 1$
while *True* **do**
  Play each arm $\frac{k}{min_{x \in X \setminus x^*}(\Delta_x)}$ times against all other arms
  for $t = 2^{p-1}$ **to** 2^p **do**
    choose x_t uniformly from $\mathcal{L}$
    $y_t \leftarrow argmax_i \left(\frac{1}{p} \sum_{j=1}^{p} \frac{1}{n_{i,j}} \sum_{t \in T_{i,j}} b_t + \sqrt{\frac{\alpha}{p^2} \cdot \log(t) \cdot (\sum_{j=1}^{p} \frac{1}{n_{i,j}})} \right)$
    compare (x_t, y_t)
    observe choice b_t
  $\mathcal{L} \leftarrow$ the multi-set of arms played as y_t in the last for-loop.
  $p \rightarrow p + 1$

Figure 4.2: Balanced Doubler

4.3 Sparring with Thompson Sampling

The main drawback of the Balanced Doubler algorithm is the warm-start phase. One of the benefits of the Thompson Sampling algorithm is that it does not need any warm start phase. As mentioned in section 2.1.2, the Thompson Sampling algorithm samples the random variables $\Theta_x, x \in X$ with Beta distribution and picks the arm with the highest value. Until an arm x is played, the random variable holds a uniform distribution, this roughly explains why the warm-start phase is not needed. The Sparring with Thompson Sampling algorithm pseudo-code is presented in Figure 4.3. The algorithm maintains the number of times each arm won and lost a comparison. In each round the algorithm samples all the random variables for each of the SBMs: $\Theta_{x,R}, \Theta_{x,L}$ (for each arm $x \in X$) with Beta distribution. The Beta distribution is defined by the hyper-parameters $Success_{L,x}$, $Fails_{L,x}$ for the left set of random variables and $Success_{R,x}, Fails_{R,x}$ for the right set (lines 3, 4). After the arms are compared and the outcome is observed all the relevant hyper-parameters are updated with the results.

Algorithm 12: Sparring Thompson Sampling

Input: X, $\mathcal{S}_R, \mathcal{S}_L$

1 $t \leftarrow 1$
2 **while** *True* **do**
3 $\quad \forall x \in X \Theta_{L,x} \sim Beta(Success_{L,x} + 1, Fails_{L,x} + 1)$
4 $\quad \forall x \in X \Theta_{R,x} \sim Beta(Success_{R,x} + 1, Fails_{R,x} + 1)$
5 $\quad x_t \leftarrow argmax_{x \in X}(\Theta_{L,x})$
6 $\quad y_t \leftarrow argmax_{x \in X}(\Theta_{R,x})$
7 $\quad$ compare(x_t, y_t)
8 $\quad$ observe b_t
9 $\quad Success_{L,x_t} \leftarrow Success_{L,x_t} + \mathbb{1}_{b_t=0}$
10 $\quad Fails_{L,x_t} \leftarrow Fails_{L,x_t} + \mathbb{1}_{b_t=1}$
11 $\quad Success_{R,y_t} \leftarrow Success_{R,y_t} + \mathbb{1}_{b_t=1}$
12 $\quad Fails_{R,y_t} \leftarrow Fails_{R,y_t} + \mathbb{1}_{b_t=0}$
13 $\quad t \rightarrow t + 1$

Figure 4.3: Sparring Thompson Sampling Algorithm

4.4 Sparring with Thompson Sampling Turbo

From the experiments conducted on the Sparring with Thompson Sampling algorithm we witnessed that in the initial phases the algorithm explores more than needed. In order to give the algorithm a more "aggressive behaviour" in the initial rounds and keep the exploration of the regular Thompson Sampling in advanced rounds, we added a time dependent factor of $\log(1 + t \cdot 2^{-t/k^2})$ to the Fails hyper parameter as seen in lines 3 and 4 in the Sparring with Thompson Sampling Turbo algorithm. By increasing the effective number of fails the arms with successes "stand out". We do not wish to keep this "aggressive behaviour" for the advanced rounds and hence $2^{-t/k^2}$ in $\log(1 + t \cdot 2^{-t/k^2})$ in order to remove this factor from advanced rounds. The Sparring with Thompson Sampling algorithms pseudo-code is presented in Figure 4.4.

Algorithm 13: Sparring Thompson Sampling Turbo

Input: X, $\mathcal{S}_R, \mathcal{S}_L$

1 $t \leftarrow 1$
2 **while** *True* **do**
3 $\quad \forall x \in X \Theta_{L,x} \sim$
$\quad\quad Beta(Success_{L,x} + 1, Fails_{L,x} + 1 + \log(1 + t \cdot 2^{-t/k^2}))$
4 $\quad \forall x \in X \Theta_{R,x} \sim$
$\quad\quad Beta(Success_{R,x} + 1, Fails_{R,x} + 1 + \log(1 + t \cdot 2^{-t/k^2}))$
5 $\quad x_t \leftarrow argmax_{x \in X}(\Theta_{L,x})$
6 $\quad y_t \leftarrow argmax_{x \in X}(\Theta_{R,x})$
7 $\quad$ compare(x_t, y_t)
8 $\quad$ observe b_t
9 $\quad Success_{L,x_t} \leftarrow Success_{L,x_t} + \mathbb{1}_{b_t=0}$
10 $\quad Fails_{L,x_t} \leftarrow Fails_{L,x_t} + \mathbb{1}_{b_t=1}$
11 $\quad Success_{R,y_t} \leftarrow Success_{R,y_t} + \mathbb{1}_{b_t=1}$
12 $\quad Fails_{R,y_t} \leftarrow Fails_{R,y_t} + \mathbb{1}_{b_t=0}$
13 $\quad t \rightarrow t + 1$

Figure 4.4: Sparring Thompson Sampling Algorithm Turbo

4.5 Unforgetful Thompson Sampling Doubler

Another improvement of the Doubler algorithm is the Unforgetful Thompson Sampling Doubler algorithm. Similarly to the Improved Doubler algorithm, we do not reset the SBM in each epoch and use the Thompson Sampling method as the SBM black-box. The idea for this algorithm came while trying to prove the logarithmic bound of the Improved Doubler algorithm. This approach outperformed all the other algorithms apart from the Thompson Sampling Sparring. The algorithm's pseudo-code is presented in Figure 4.5.

Algorithm 14: Unforgetful Thompson Sampling Doubler

Input: X, $\mathcal{S}$

$\mathcal{L} \leftarrow X$
$p \leftarrow 1$
while *True* **do**
  for $t = 2^{p-1}$ **to** 2^p **do**
    choose x_t uniformly from $\mathcal{L}$
    $\forall x \in X \Theta_{R,x} \sim Beta(Success_{R,x} + 1, Fails_{R,x} + 1)$
    $y_t \leftarrow argmax_{x \in X}(\Theta_{R,x}))$
    compare (x_t, y_t)
    observe choice b_t
    $Success_{R,y_t} \leftarrow Success_{R,y_t} + \mathbb{1}_{b_t=1}$
    $Fails_{R,y_t} \leftarrow Fails_{R,y_t} + \mathbb{1}_{b_t=0}$
  $\mathcal{L} \leftarrow$the multi-set of arms played as y_t in the last for-loop.
  $p \rightarrow p + 1$

Figure 4.5: Unforgetful Thompson Sampling Doubler

Chapter 5

Experiments

To evaluate our algorithms, we apply them on random synthetic data, special test cases (will be explained shortly), and also to the problem of scorer evaluation from the field of information retrieval (IR) (Manning et al., 2008) [15]. A scorer is a function that takes a user's search query and ranks the documents in a collection according to their relevance to that query. Scorer evaluation aims to determine which set of scorers performs best.

Given a set of K scorers, the problem of finding the best scorer can then be modelled as a K-armed Dueling Bandit problem, with each arm relating to a scorer. This approach follows an experimental paradigm that has previously been used for assessing the performance of scorers [17].

5.1 Real Data

Our experiments is built on standard IR datasets, namely the LETOR MQ2007 and MSLR-WEB30K datasets (Liu et al., 2007).[1] MQ2007 holds about 65 million rows and MSLR-WEB30K holds 30K rows. Each row in the data is a query-URL pair, where the number in the beginning of the row signifies the relevance of this result and each of the following pairs is a scorer-score pair. Here is an example of a single row from MQ2007:

$$2 \text{ qid} : 10032, 1 : 0.0565, 2 : 0.0000, 3 : 0.6667...45 : 0.0000, 46 : 0.0769 \quad (5.1)$$

[1] These data sets can be retrieved from http://research.microsoft.com/en-us/um/beijing/projects/letor/

This data set holds 46 scorers, each relating to a ranking feature provided in the data set. These scores are used in order to rank the relevance of a search query to a document. A good example of a feature is the number of times the query shows up in the body of the document. The MSLR-WEB30K is built in the same manner; the only difference is the number of scorers (136 in total). We use two different settings, BPDP and UBDB; for each of these settings we use the data sets in a different manner.

5.1.1 Converting LETOR data to a PBDB matrix

We used the following method to convert the method to a PBDB matrix P over the set of scorers. The method is based on a combination of standard IR techniques for simulating noisy comparisons between lists. We show the method in the pseudo-code in Figure 5.1. In order to set the preference matrix we need to compare between the scorers/arms; in our setting we use the Interleaved Comparison method (Radlinski et al., 2008) [16]. The main idea behind the Interleaved Comparison is deciding which of the two given alternatives is preferred. In particular, Radlinski proposed a method for presenting the results from two rankers, where a ranker is a function that when given a query, returns a list of elements (i.e., websites, documents, actions) ordered according to the ranker's functionality (i.e., number of times the query showed up in the text of the document). The two rankers provide a rank, for instance ranker A provides (a_1, a_2, a_3) and ranker B provides the rank (b_1, b_2, b_3). The interleaved rank is a "merge" of the ranks (i.e., (a_1, b_1, a_2)). For each query an interleaved rank is presented to the user. If the user clicks a_1 from the given example we assume that ranker A is preferred over ranker B. In order to create a ranking several query-URL pairs are taken, with the same query but different URLs, ranked them according to the relevant scorers, and then the two ranks are interleaved into an interleaved rank. Sets of rows with the same query and different URL's are sampled randomly, interleaved into an interleaved rank and clicks are generated probabilistically, according to the score of the relevant scorer for the sampled query. More precisely, for each document in the list, starting from the top of the list, a click is generated according the probability $P(\text{click}|\text{scorer's score for the pair})$, meaning the higher the scorer's score the higher the probability of the document being clicked. In order to set

$P = [p_{x,y}]$ each arm is compared to every other arm, using the interleaved comparison method, and the empirical probability is set.

Algorithm 15: Converting LETOR Data to the Preference Matrix

Input: LETOR Data Set, $X, K = |X|$

$W = [w_{x,y}]$

$P = [p_{x,y}]$

$\mathcal{S} \leftarrow N$ sets of queries-url with the same query from data-set.

for $t = 1$ **to** $80,000$ **do**

  forall $x \in X$ **do**

    forall $y \in X$ **do**

      $s_x \leftarrow$ set from $\mathcal{S}$ sorted according to scorer x

      $s_y \leftarrow$ set from $\mathcal{S}$ sorted according to scorer y

      $I \leftarrow$ interleaved set of s_x, s_y.

      Generate a click.

      If a row from set s_x was clicked $w_{x,y} \leftarrow w_{x,y} + 1$

      If a row from set s_y was clicked $w_{y,x} \leftarrow w_{y,x} + 1$

$P = [p_{x,y}] \leftarrow \left[\frac{w_{x,y}}{w_{x,y}+w_{y,y}}\right]$

Figure 5.1: Converting LETOR Data Procedure

In order to evaluate our algorithms in this setting we use the PBDB regret as defined in (2.9)

5.1.2 Converting LETOR data to UBDB utilities

In the UBDB setting, in each round t, a query is sampled from the MQ2007 data-set, the unobserved utilities u_t, v_t are set according to the chosen arms, and finally the observed feedback b_t is defined by a Bernoulli trail with $p = \frac{u_t - v_t + 1}{2}$.

If we use the above example (5.1) and set the chosen arms to $x_t = 1, y_t = 3$ then $b_t \sim Bernoulli\left(\frac{0.0565 - 0.6667 + 1}{2}\right)$. In order to evaluate our algorithm in this setting we use the UBDB regret as defined in (2.5) and μ_x, the average utility of arm x (that is needed in order to calculate the regret) is set to be the average of scores in the data set for scorer x.

5.2 Synthetic Data

In order to evaluate how our algorithm reacts to special cases of input we use synthetic data that is predefined. In our experiments we let our algorithm handle the following scenarios:

1. Linear Mean Utilities
2. Small Gap
3. Big Gap
4. Random Mean Utilities

The first special case we evaluate is Linear Mean Utilities, for example with 5 arms $(\mu_1, \mu_2, \mu_3, \mu_4, \mu_5) = (0, 0.25, 0.5, 0.75, 1)$. Two other special cases are when there is one mean utility that is notably higher than all the other arms, and the last case is where the mean utilities of all the arms a fairly close . The last case we simulate is random mean utilities. The regret is defined similarly to the real data experiments.

5.3 Results

In this section we will be showing simulation results for PBDB and UBDB setting, with various algorithms and several values of K for $T = 2^{17}$ time steps. Each graph shows the empirical mean of the Cumulative Regret, over $M = 200$ independent runs. The mean of the Cumulative Regret for each time step is simply:

$$\hat{R}(t) = \frac{1}{M} \sum_{m=1}^{M} R_m(t) \tag{5.2}$$

where $R_m(t)$ is cumulative regret as defined in (2.5) and (2.9). For the convenience of the reader we provide a table of all the simulations and relevant figures:

Simulation		
Figure	K	Test Case
16	8	Small Gap
17	16	Small Gap
18	46	Small Gap
19	8	Linear Mean Utilities
20	16	Linear Mean Utilities
21	46	Linear Mean Utilities
22	8	Big Gap
23	16	Big Gap
24	46	Big Gap
25	8	Random Mean Utilities
26	16	Random Mean Utilities
27	46	Random Mean Utilities
28	8	MQ2007 Data Set
29	16	MQ2007 Data Set
30	46	MQ2007 Data Set
31	46	WEB30K Data Set

In some test cases, some algorithms are extremely out-performed by the other algorithms; in those cases we leave out the plot of the cumulative regret of those algorithms.

5.3.1 Small Gap

In these experiments we have 3 arms whos mean utilities are $1, 0.995, 0.99$, where the rest of the arm's utilities are spread uniformly between 0.5 and 1.

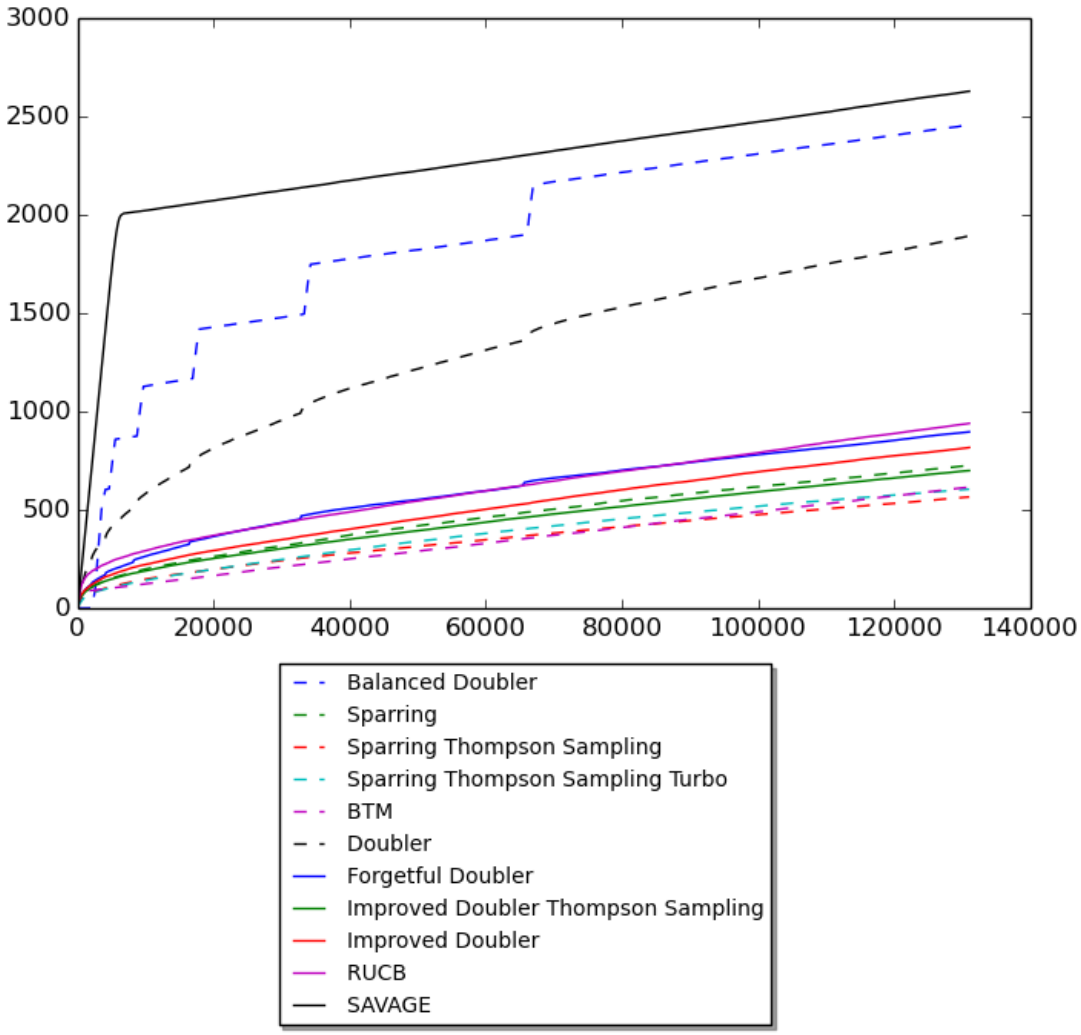

Figure 5.2: Small Gap - 8 Arms

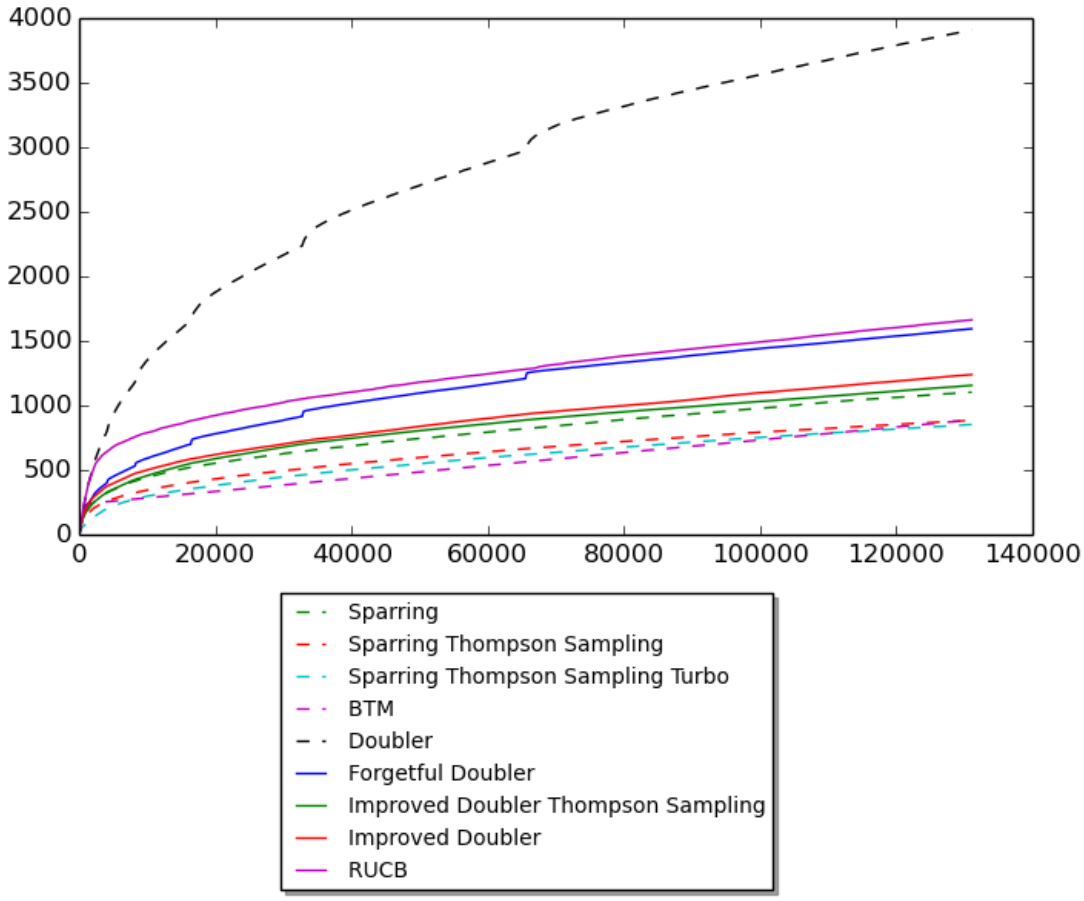

Figure 5.3: Small Gap - 16 Arms

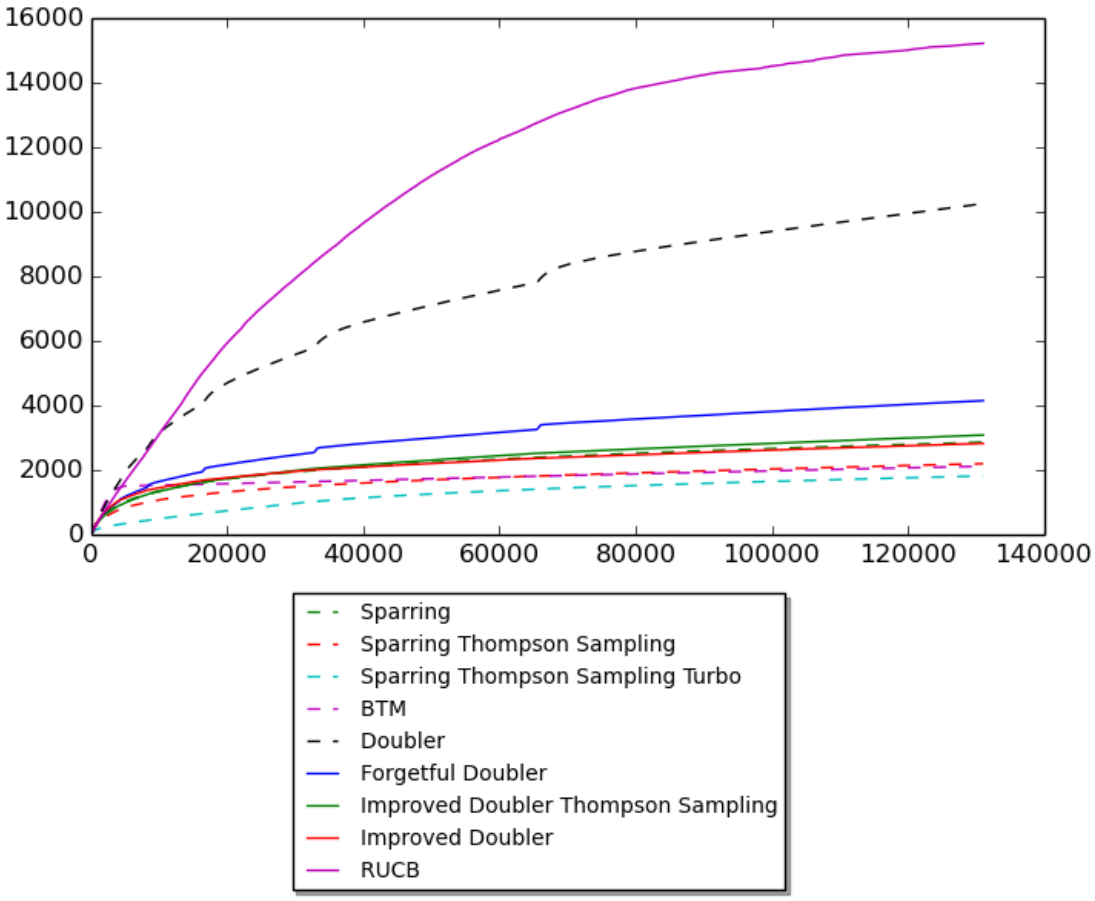

Figure 5.4: Small Gap - 46 Arms

5.3.2 Linear Mean Utilities

In these experiments we the mean utilities are spread uniformly between 1 and 0.

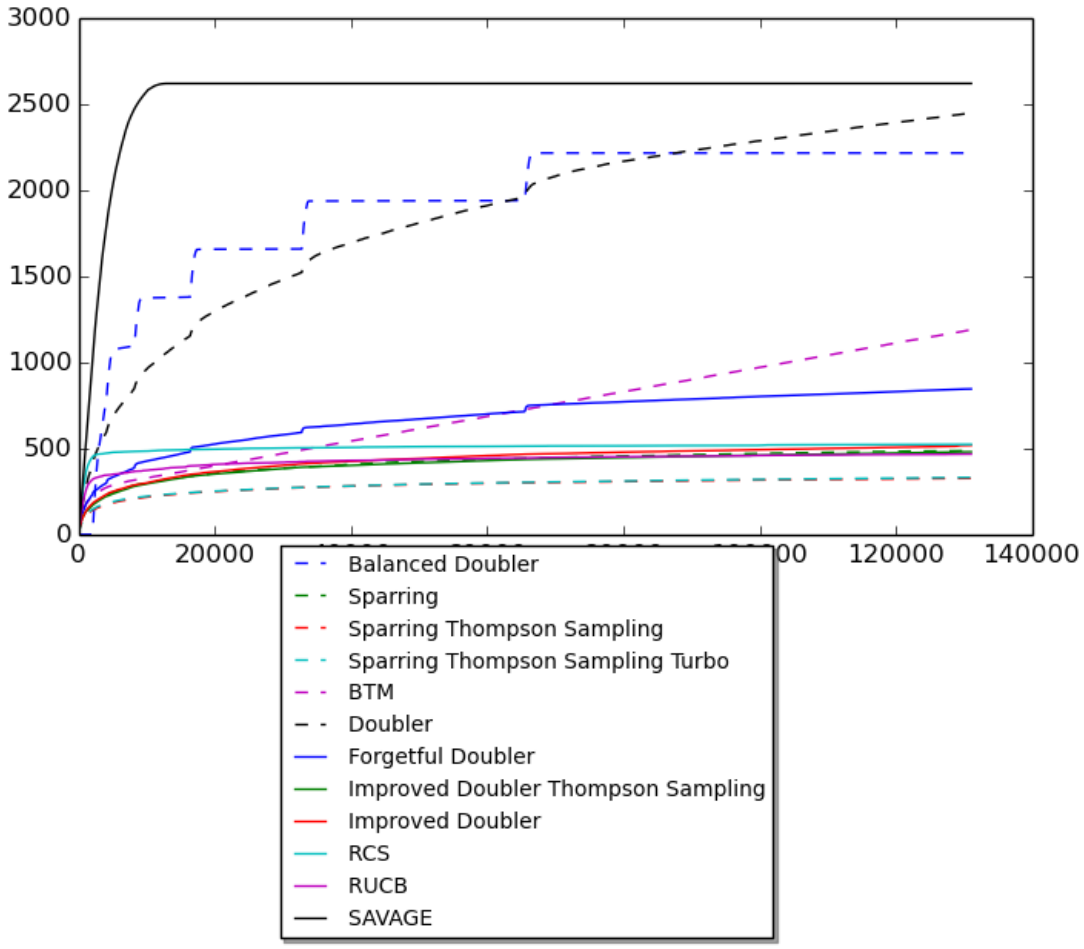

Figure 5.5: Linear Mean Utilities - 8 Arms

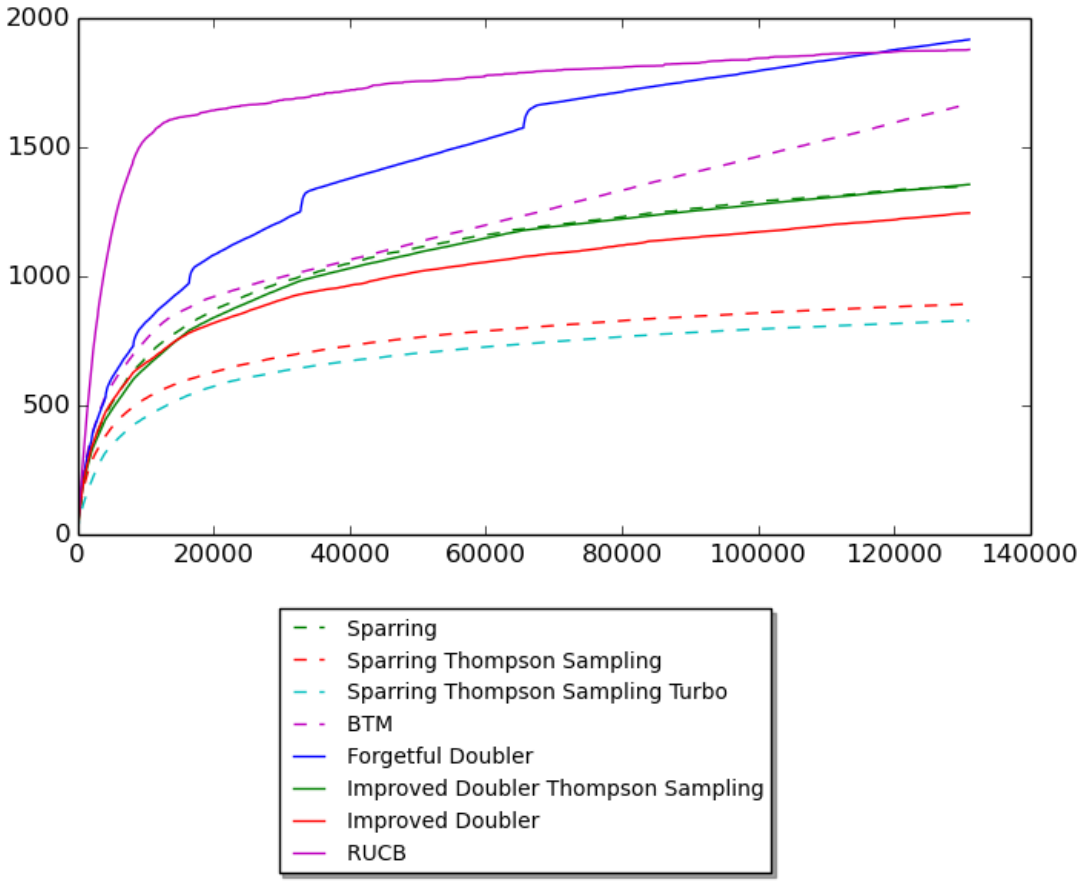

Figure 5.6: Linear Mean Utilities - 16 Arms

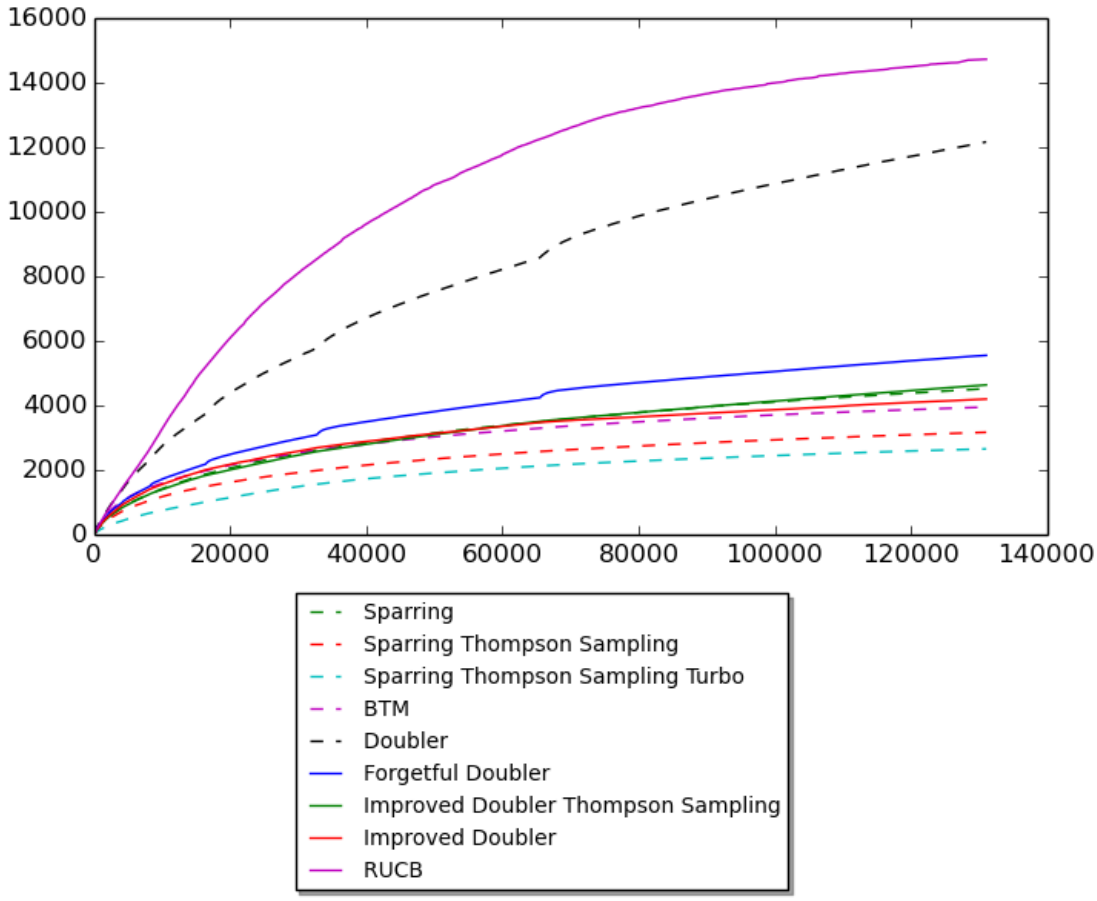

Figure 5.7: Linear Mean Utilities - 46 Arms

5.3.3 Big Gap

In these experiments there is one arm with a mean utility of 1 and the rest of the arms are spread uniformly between 0.5 and 0.

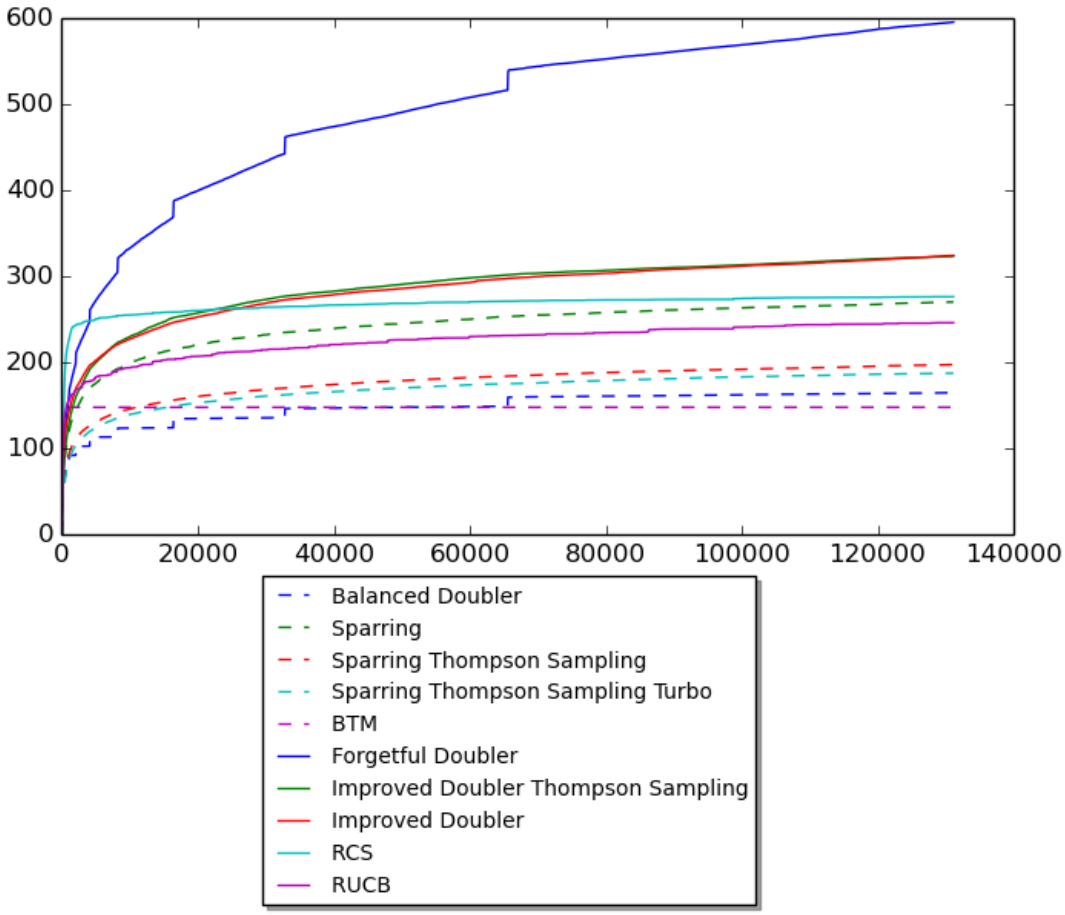

Figure 5.8: Big Gap - 8 Arms

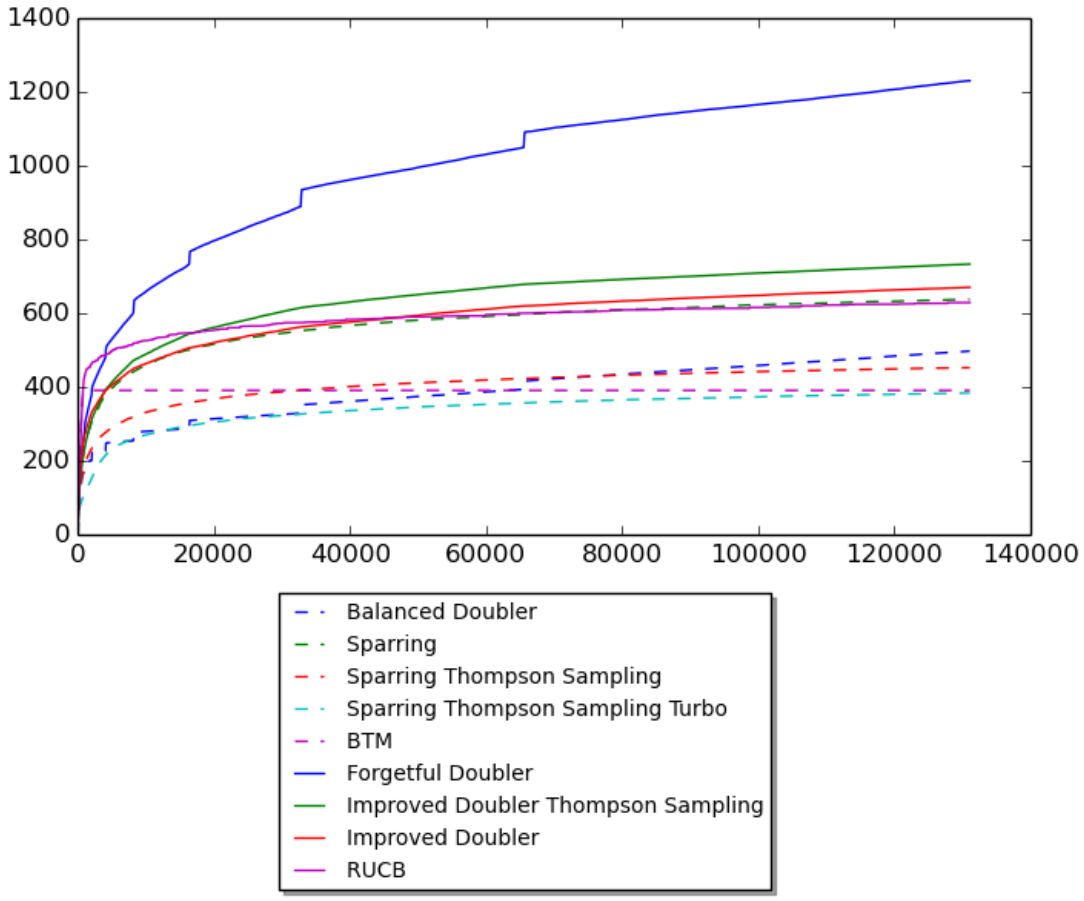

Figure 5.9: Big Gap - 16 Arms

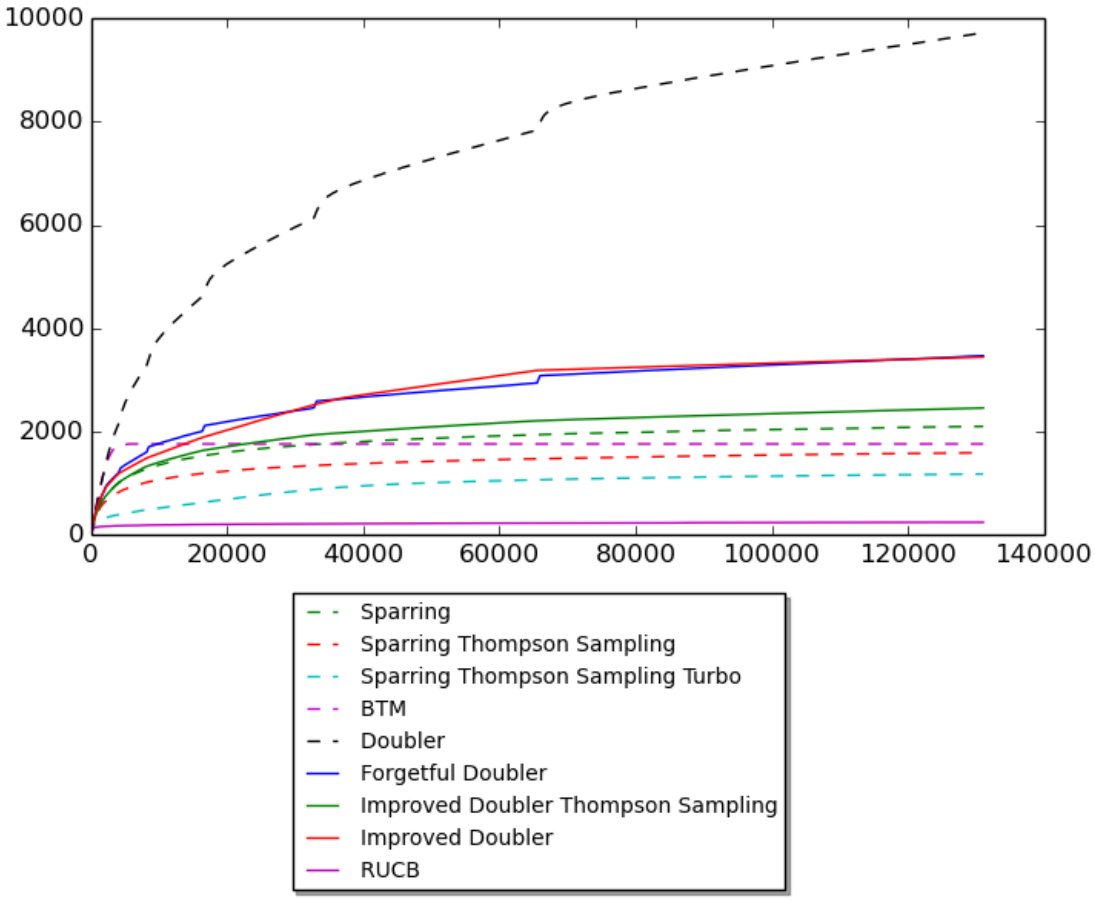

Figure 5.10: Big Gap - 46 Arms

5.3.4 Random Mean Utilities

In these experiments all the mean utilities are random.

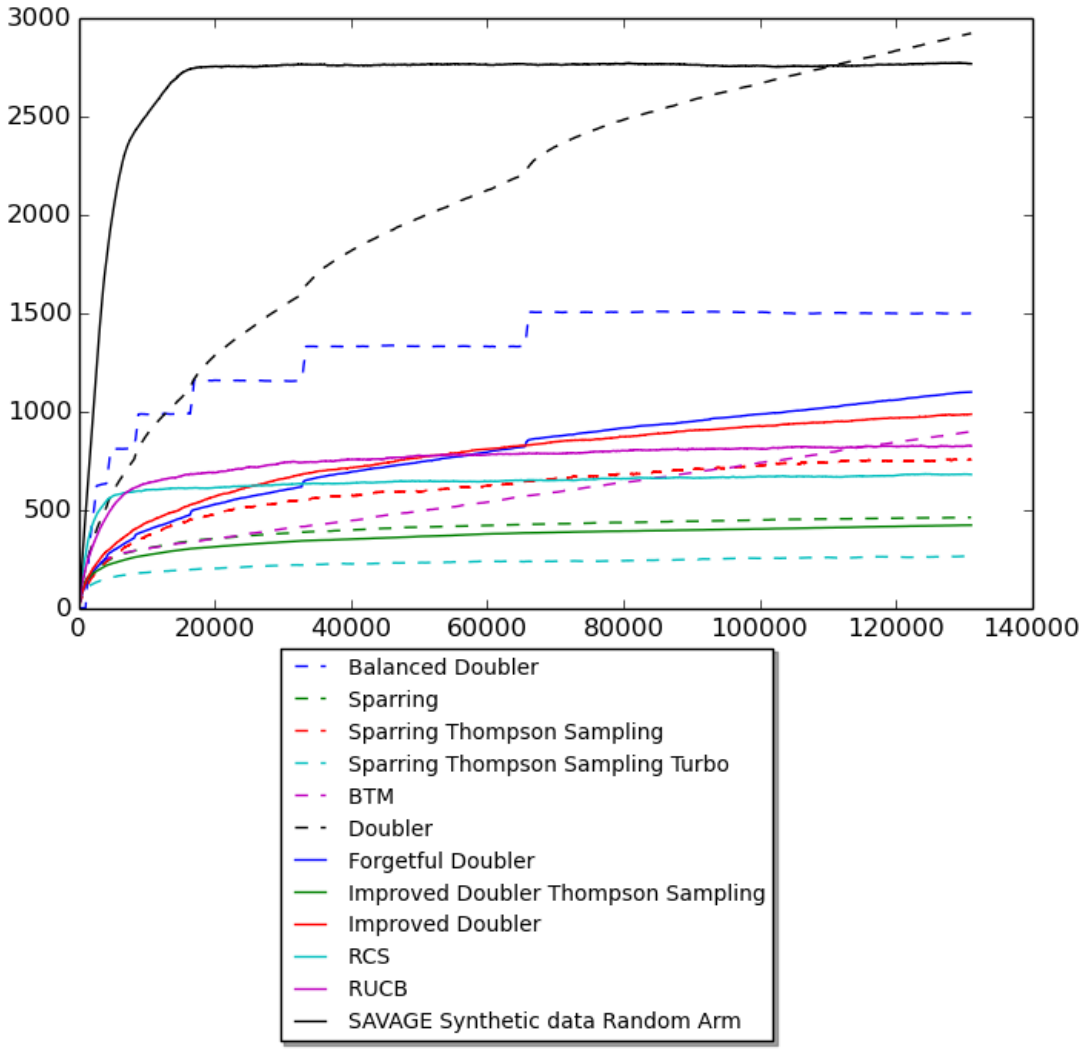

Figure 5.11: Random Mean Utilities - 8 Arms

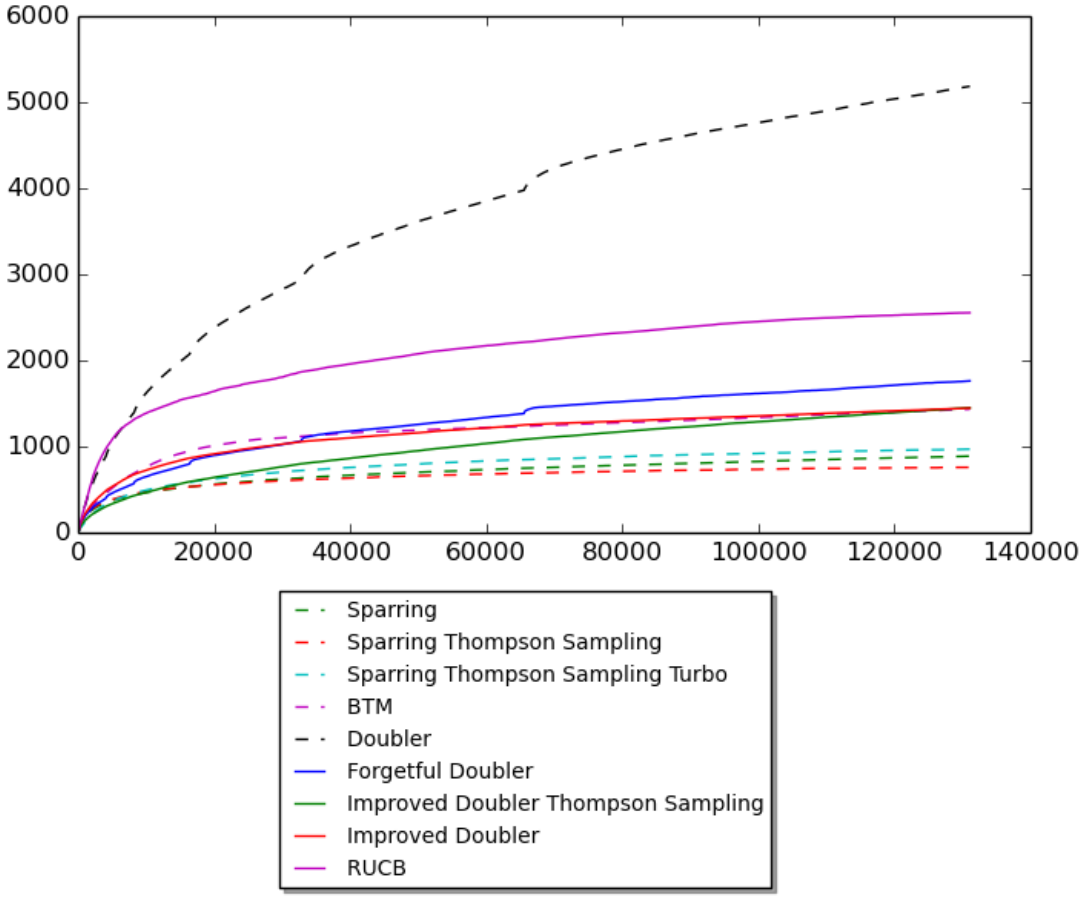

Figure 5.12: Random Mean Utilities - 16 Arms

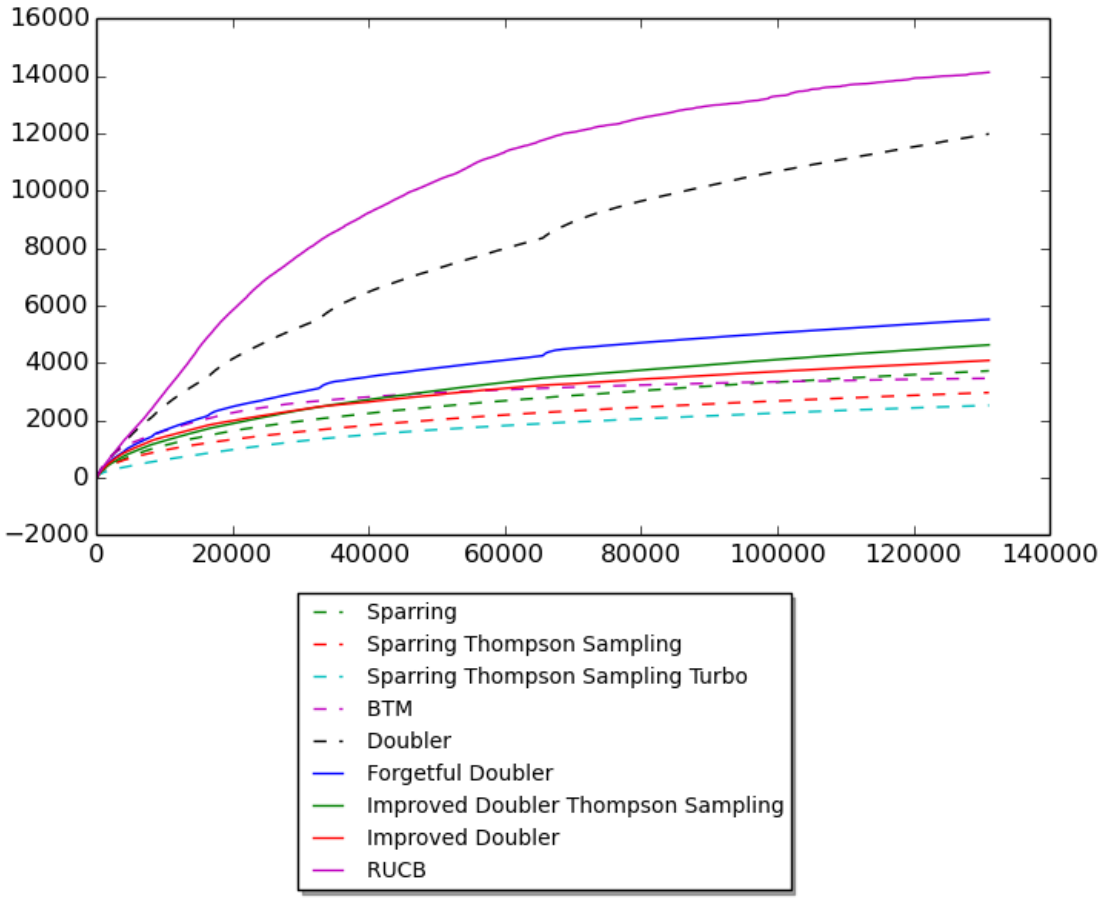

Figure 5.13: Random Mean Utilities - 46 Arms

5.3.5 MQ2007 Data Set

In these experiments the utilities of each arm are extracted from the data set MQ2007.

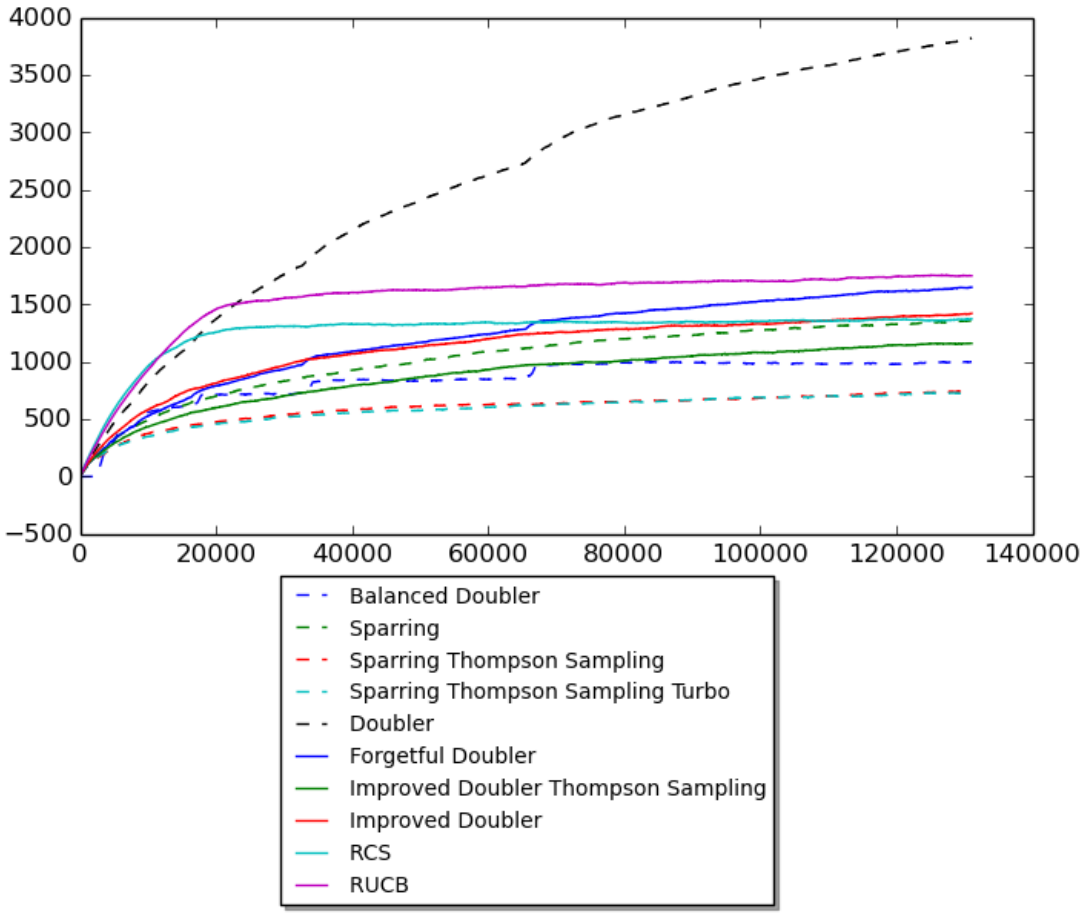

Figure 5.14: MQ2007 Data Set - 8 Arms

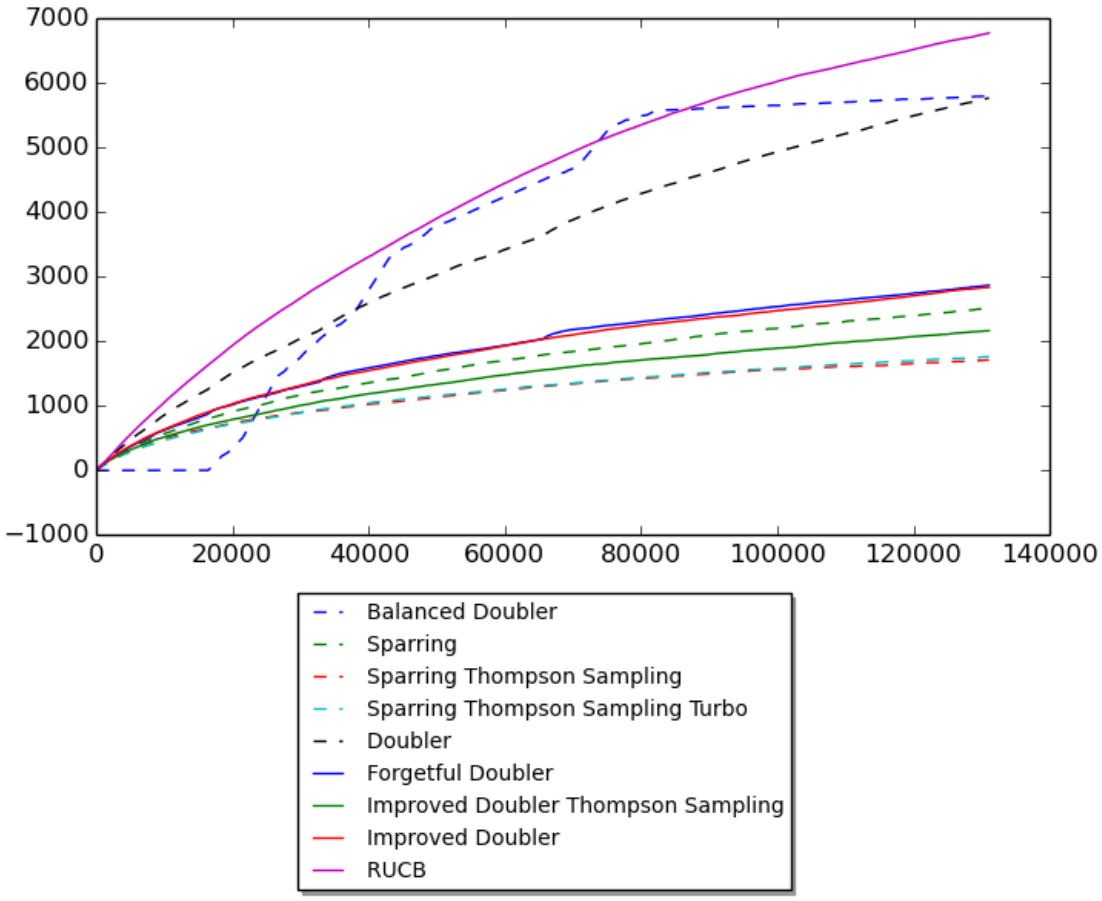

Figure 5.15: MQ2007 Data Set - 16 Arms

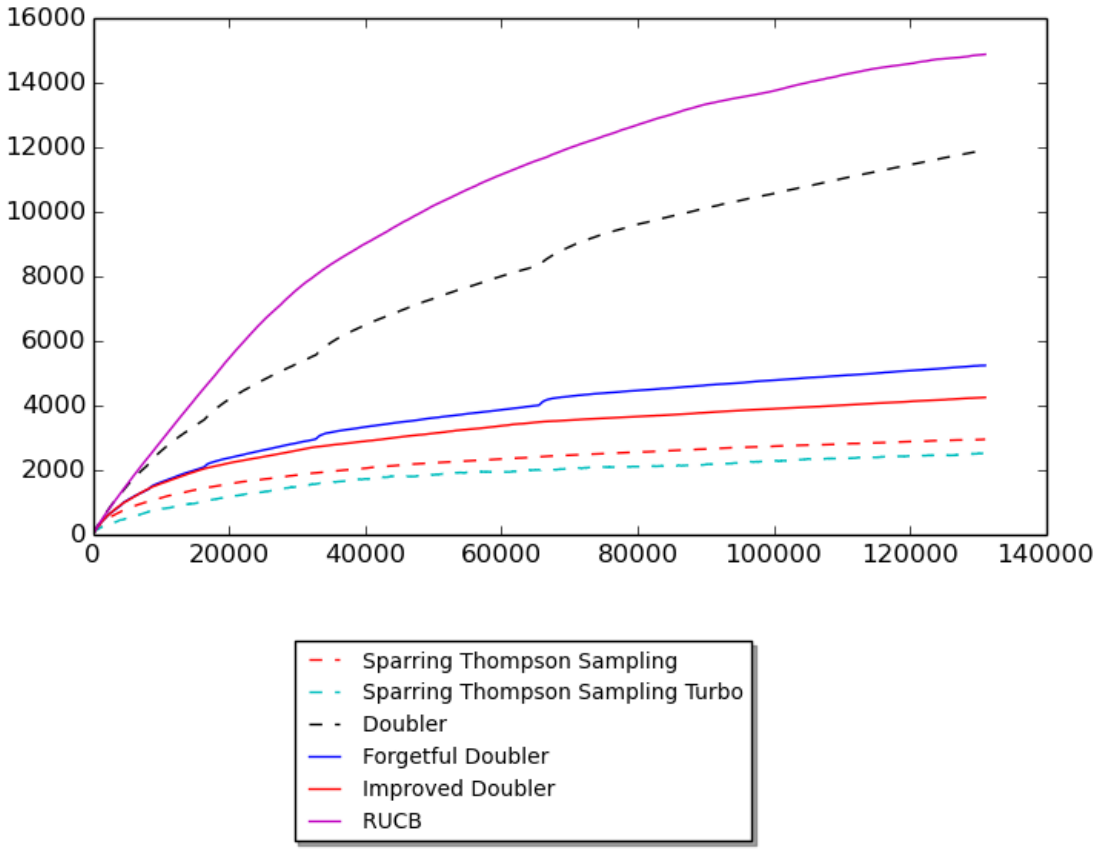

Figure 5.16: MQ2007 Data Set - 46 Arms

5.3.6 WEB30K Data Set

In this experiment the P acquired from the WEB30K data set as described in section 5.1.1.

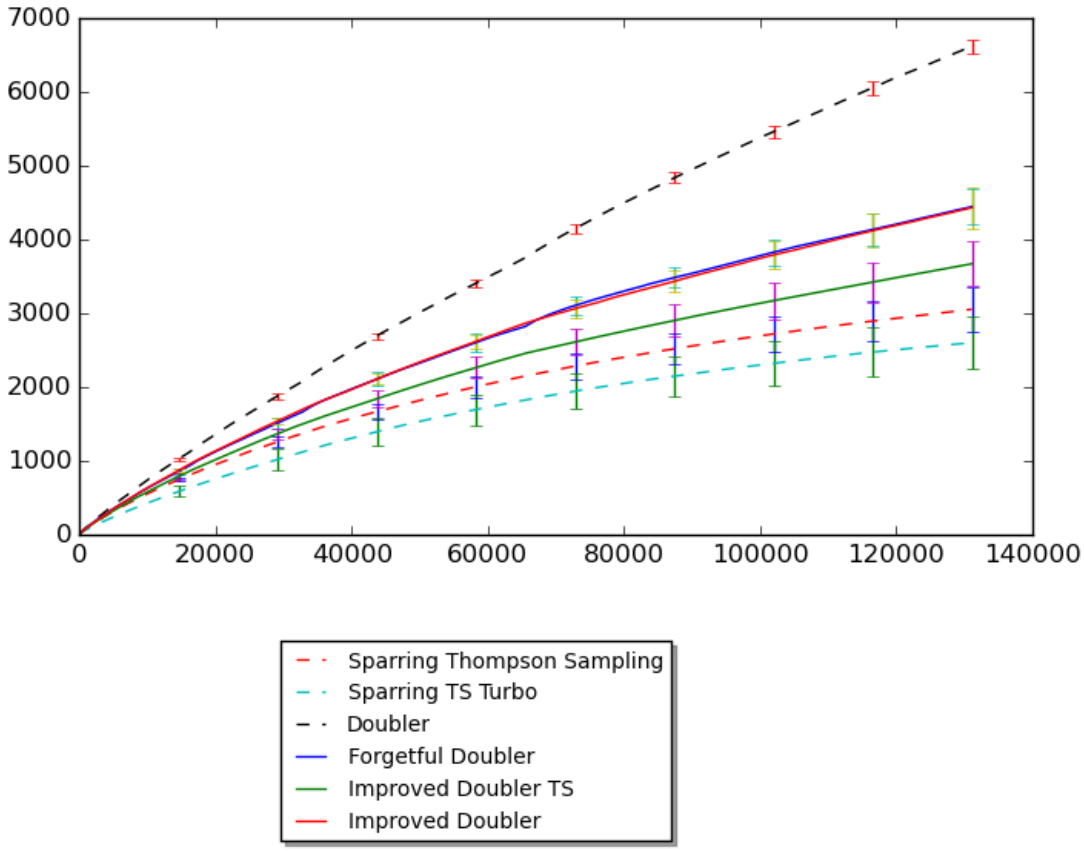

Figure 5.17: WEB30K Data Set - 46 Arms

5.4 Conclusions of Results

In the following table we have summarized the best performing algorithm for each of the test cases: From this table it is clear that the Sparring

Simulation			
Figure	K	Test Case	Best Performing Algorithm
16	8	Small Gap	Sparring Thompson Sampling
17	16	Small Gap	Sparring Thompson Sampling Turbo
18	46	Small Gap	Sparring Thompson Sampling Turbo
19	8	Linear Mean Utilities	Sparring Thompson Sampling
20	16	Linear Mean Utilities	Sparring Thompson Sampling Turbo
21	46	Linear Mean Utilities	Sparring Thompson Sampling Turbo
22	8	Big Gap	BTM
23	16	Big Gap	BTM
24	46	Big Gap	RUCB
25	8	Random Mean Utilities	Sparring Thompson Sampling Turbo
26	16	Random Mean Utilities	Sparring Thompson Sampling
27	46	Random Mean Utilities	Sparring Thompson Sampling Turbo
28	8	MQ2007 Data Set	Sparring Thompson Sampling Turbo
29	16	MQ2007 Data Set	Sparring Thompson Sampling
30	46	MQ2007 Data Set	Sparring Thompson Sampling Turbo
31	46	WEB30K Data Set	Sparring Thompson Sampling Turbo

Thompson Sampling Turbo and Sparring Thompson Sampling algorithm are best performing in most cases apart from the case of Big Gap where the elimination algorithm out-perform the black-box algorithms.

Chapter 6

Conclusions and Future Work

This thesis has covered a range of algorithms for the Dueling Bandits Problem. We have shown in our work a novel approach for tackling the Dueling Bandits Problem. We show promising empirical results, comparable to the existing algorithms designed for the Dueling Bandits Problem and far outperforming them. In future work, we will consider two extensions to this research. First, we dealt with choice in sets of size 2. Extending the algorithm to N choices would be an interesting direction. Second, we have yet to try more complex white boxes. Sparring Thompson Sampling Turbo shows great promise, and modifying the hyper parameters may apply better performing algorithms as well as tighter theoretical bounds.

Bibliography

[1] Rakesh Agrawal, Sreenivas Gollapudi, Alan Halverson, and Samuel Ieong. Diversifying search results. In *Proceedings of the Second ACM International Conference on Web Search and Data Mining*, pages 5–14. ACM, 2009.

[2] Shipra Agrawal and Navin Goyal. Analysis of thompson sampling for the multi-armed bandit problem. *arXiv preprint arXiv:1111.1797*, 2011.

[3] Shipra Agrawal and Navin Goyal. Further optimal regret bounds for thompson sampling. *arXiv preprint arXiv:1209.3353*, 2012.

[4] Nir Ailon, Thorsten Joachims, and Zohar Karnin. Reducing dueling bandits to cardinal bandits. *arXiv preprint arXiv:1405.3396*, 2014.

[5] Peter Auer, Nicolo Cesa-Bianchi, and Paul Fischer. Finite-time analysis of the multiarmed bandit problem. *Machine learning*, 47(2-3):235–256, 2002.

[6] Peter Auer and Ronald Ortner. Ucb revisited: Improved regret bounds for the stochastic multi-armed bandit problem. *Periodica Mathematica Hungarica*, 61(1-2):55–65, 2010.

[7] Duncan Black. On the rationale of group decision-making. *The Journal of Political Economy*, pages 23–34, 1948.

[8] Deepayan Chakrabarti, Ravi Kumar, Filip Radlinski, and Eli Upfal. Mortal multi-armed bandits. In *Advances in Neural Information Processing Systems*, pages 273–280, 2009.

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

[10] Katja Hofmann, Shimon Whiteson, and Maarten de Rijke. Contextual bandits for information retrieval. In *NIPS 2011 Workshop on Bayesian Optimization, Experimental Design, and Bandits, Granada*, volume 12, page 2011, 2011.

[11] Katja Hofmann, Shimon Whiteson, and Maarten De Rijke. Fidelity, soundness, and efficiency of interleaved comparison methods. *ACM Transactions on Information Systems (TOIS)*, 31(4):17, 2013.

[12] Tze Leung Lai and Herbert Robbins. Asymptotically efficient adaptive allocation rules. *Advances in applied mathematics*, 6(1):4–22, 1985.

[13] Lihong Li, Wei Chu, John Langford, and Robert E Schapire. A contextual-bandit approach to personalized news article recommendation. In *Proceedings of the 19th international conference on World wide web*, pages 661–670. ACM, 2010.

[14] Lihong Li, Wei Chu, John Langford, and Xuanhui Wang. Unbiased offline evaluation of contextual-bandit-based news article recommendation algorithms. In *Proceedings of the fourth ACM international conference on Web search and data mining*, pages 297–306. ACM, 2011.

[15] Christopher D Manning, Prabhakar Raghavan, Hinrich Schütze, et al. *Introduction to information retrieval*, volume 1. Cambridge university press Cambridge, 2008.

[16] Filip Radlinski, Robert Kleinberg, and Thorsten Joachims. Learning diverse rankings with multi-armed bandits. In *Proceedings of the 25th international conference on Machine learning*, pages 784–791. ACM, 2008.

[17] Filip Radlinski, Madhu Kurup, and Thorsten Joachims. How does clickthrough data reflect retrieval quality? In *Proceedings*

of the 17th ACM conference on Information and knowledge management, pages 43–52. ACM, 2008.

[18] Paat Rusmevichientong and David P Williamson. An adaptive algorithm for selecting profitable keywords for search-based advertising services. In *Proceedings of the 7th ACM Conference on Electronic Commerce*, pages 260–269. ACM, 2006.

[19] Aleksandrs Slivkins, Filip Radlinski, and Sreenivas Gollapudi. Ranked bandits in metric spaces: learning optimally diverse rankings over large document collections. *arXiv preprint arXiv:1005.5197*, 2010.

[20] William R Thompson. On the likelihood that one unknown probability exceeds another in view of the evidence of two samples. *Biometrika*, pages 285–294, 1933.

[21] Tanguy Urvoy, Fabrice Clerot, Raphael Féraud, and Sami Naamane. Generic exploration and k-armed voting bandits. In *Proceedings of the 30th International Conference on Machine Learning (ICML-13)*, pages 91–99, 2013.

[22] Shuai Yuan, Jun Wang, and Maurice van der Meer. Adaptive keywords extraction with contextual bandits for advertising on parked domains. *arXiv preprint arXiv:1307.3573*, 2013.

[23] Yisong Yue, Josef Broder, Robert Kleinberg, and Thorsten Joachims. The k-armed dueling bandits problem. *Journal of Computer and System Sciences*, 78(5):1538–1556, 2012.

[24] Yisong Yue and Thorsten Joachims. Interactively optimizing information retrieval systems as a dueling bandits problem. In *Proceedings of the 26th Annual International Conference on Machine Learning*, pages 1201–1208. ACM, 2009.

[25] Yisong Yue and Thorsten Joachims. Beat the mean bandit. In *Proceedings of the 28th International Conference on Machine Learning (ICML-11)*, pages 241–248, 2011.

[26] Masrour Zoghi, Shimon Whiteson, Remi Munos, and Maarten de Rijke. Relative upper confidence bound for the k-armed dueling bandit problem. *arXiv preprint arXiv:1312.3393*, 2013.

[27] Masrour Zoghi, Shimon A Whiteson, Maarten de Rijke, and Remi Munos. Relative confidence sampling for efficient on-line ranker evaluation. In *Proceedings of the 7th ACM international conference on Web search and data mining*, pages 73–82. ACM, 2014.

האופטימלי, זאת למרות שינוי המגמה של השוק בתקופת החגים. הרחבת בעית ה־ MAB הסטנדרטית למקרים של משוב מסוג העדפה, לדוגמא, מקרה שבו הסוכן משווה בין שתי זרועות ומקבל משוב איכותני, הינה בעיה מבטיחה. בעיה זו מקבלת תשומת לב מרובה בשנים האחרונות. בתזה זאת, אנו מציעים מספר אלוגוריתמים המבוססים על גישת "קופסאות שחורות" כפי שתואר במאמר[4]. מבנה התזה הוא כדלקמן: בפרק 2, אנו מסקרים את בעיית ה־ MAB, האלגוריתמים הרלוונטים עבורה ובעיית ה־ Dueling Bandits . בפרק 3, אנו מסקרים את האלגוריתמים המתקדמים ביותר כיום ומסבירים את גישת ה"קופסאות השחורות" בצורה מדויקת. בפרק 4, אנו דנים באלגוריתמים Thompson Sampling Doubler Sparring with Thompson Sampling, sparring with Thompson Sampling Turbo, forgetful Improved Doubler, Balanced Doubler,. בפרק 5, אנו מציגים את הניסויים, אשר משווים בין הביצועים של האלגוריתמים המוצגים בפרק 3 לבין האלגוריתמים המוצגים בפרק 4. הביצועים נמדדים באמצעות השוואה של החרטה המצטברת במשך הריצה של האלגוריתמים, כאשר סט הנתונים (Data Set) נלקח ממאגר הנתונים של LETOR ־ Microsoft . מאגר זה מכיל תוצאות חיפוש של משתמשים אשר שימשו אותנו להגדרת הניסויים והגדרת החרטה. בנוסף למאגר הנתונים, השתמשנו בסט נתונים סינתטי, אשר איפשר לנו להשוות בין ביצועיהם של האלגוריתמים שלנו לבין ביצועיהם של האלגוריתמים האחרים, אשר קיימים בספרות, עבור מקרים מיוחדים כגון: מספר זרועות בעל ערכי תגמול ממוצע סמוך, זרוע אחת בעלת תגמול ממוצע גבוה ביחס לשאר הזרועות וכן על נתונים אקראים. בניסויים אלה, ראינו כי התוצאות של האלגוריתמים שלנו מראות שיפור משמעותי,כך גם לפי הניסויים על בסיס מאגר הנתונים והמידע הסינתטי. אנו מציעים את יישום האלגוריתם ה־ with Thomson Sampling Turbo Sparring, המשיג תוצאות טובות עבור בעיית ה־ Dueling Bandits, לטובת תהליך Testing A/B ומציגים תוצאות הרצה התומכות בגישה זו.

1 הניבה תשואה של 0.7". יחד עם זאת, ישנן בעיות רבות בהן הנחה זו אינה מתקיימת, לדוגמא: בעיות העדפה, כאשר הסוכן אינו יכול לקבל ערך מספרי, אלא ערך איכותני: "זרוע מספר 1 יותר טובה מזרוע מספר 2".

ישנם מקרים רבים, בעולם הלמידה, אשר משוב מספרי אינו נגיש ואילו משוב מועדף או יחסי כן נגיש. במקרים אלו, נדרש להשתמש ב־Preference Learning , אשר נלמד כאן. כלומר ב־ Preference Learning , המשוב אינו מספרי, אלא איכותני, בדרך כלל, במושגים של השוואות או דירוגים. בעיית ה־ Dueling Bandits מוגבלת למשוב איכותני בלבד, החושף את העדפת זרוע אחת על פני השניה. בעיה זו ממדלת מקרים אלו, בהם משוב מספרי אינו קיים או קשה להשגה ומתאפשר השימוש רק במידע איכותני בביצועים של כל הזרועות.

מנועי חיפוש ופרסום מכוון, הינן שתי דוגמאות המראות את נחיצות הגדרת המבנה של ה־ Dueling Bandits. מנוע חיפוש, ישאף תמיד לתת למשתמש את התוצאה הרלונטית ביותר לשאילתא. כאשר משתמש מבצע חיפוש של ערך מסוים, הוא ייחשף למספר אפשרויות רלוונטיות, אותו משתמש יבחר מבין אפשרויות אלו. ברור כי במקרה זה, המשוב אשר המשתמש מספק למנוע החיפוש, אינו יכול להתקבל כערך מספרי, אלא כהעדפה של אפשרות אחת על יתר האפשרויות. כלומר, המשוב למנועי החיפוש יגיע בצורה של "אני מעדיף את התוצאה הראשונה על פני התוצאה השניה". דוגמא נוספת, הלקוחה מעולם השיווק באינטרנט, הינה כאשר מפרסם רוצה למכור את מוצריו בצורה מכוונת. בהינתן מספר משתמשים סופי או אינסופי, אשר ייחשפו למוצר של המפרסם, שאיפתו של המפרסם הינה למכור לכמה שיותר משתמשים את מוצריו. תהליך המכירה מתבצע כדלהלן: משתמש מגיע לדף המכירה של המוצר, במידה והשתכנע בעמוד זה, יבצע רכישה. על מנת למקסם את מספר המכירות, על המפרסם לבחון מספר גרסאות של דף המכירה, להסיק איזה דף מכירה הינו האופטימלי, כלומר המניב את כמות המכירות הרבה ביותר, לכל משתמש. ככל שהמפרסם יתכנס לדף המכירות האופטימלי, תוך שימוש במספר מצומצם של משתמשים, כך יוכל להפיק יותר מכירות. תהליך זה, נקרא "A/B Testing", כאשר כל גרסה הינה זרוע בבעיית ה־ MAB. נשאלת השאלה: מדוע לא להשתמש בגרסה הסטנדרטית של ה־ MAB ? זאת כיון שלגרסה זו אין מענה למקרים של שינוי כללי בערכי התגמול של הזרועות. לדוגמא: בתקופת החגים, ישנה עליה בכמות המכירות. במידה והמפרסם בוחן גרסה לא אופטימלית של דף המכירות, הוא יקבל משוב רועש ויכול להסיק מכך כי דף מכירות זה יותר טוב מהדף האופטימלי, זאת מפני שלא דגם את הדף האופטימלי באותה תקופה. כלומר אחד החסרונות במבנה ה־ MAB הסטנדרטי הוא חוסר היכולת שלו לתת מענה, במקרה של שינוי גורף בכלל הערכים של התגמול הממוצע של כל הזרועות.

בהנחה כי סדר הזרועות נשמר במהלך השינוי, כלומר זרוע "1" נשארת יותר טובה מזרוע "2" וכדומה, אלגוריתמי Dueling Bandits יכולים לשמש לטובת שיפור הביצועים של אותו מפרסם. כלומר, שימוש במשוב איכותני, היה מאפשר למפרסם להתכנס לדף המכירה

תקציר

אלגוריתמי (MAB) Multi Armed Bandits נחקרים ומקבלים תשומת לב נרחבת בתחום הלמידה, מאז שנת 1985, כאשר Lai and Robbins פירסמו את מאמרם [6]. העניין הרב בנושא זה אינו מפתיע, לאור העובדה כי מבנה בעיית ה־MAB , לא רק מעניין מבחינה תאורטית, אלא גם מאד שימושי. באלגוריתמי MAB משתמשים כיום, לפתירת בעיות רבות, כגון: מנועי חיפוש [19, 10, 1, 16], פרסום מכוון[8, 22, 18] ומערכות המלצה[14, 13].

בעיית ה־ MAB, הינה בעיית קבלת החלטות סדרתית, אשר בה הסוכן נדרש לבחור אפשרות אחת (מכונות או זרועות), מתוך סט האפשרויות, בצורה סדרתית. השם שאול מעולם ההימורים, אשר בו נאלץ המהמר, לבחור מבין סט של מכונות הימורים (לעיתים נקראות one armed bandit) ולהחליט באיזו מכונת מזל לשחק, כמה פעמים בכל מכונה ובאיזה סדר. כאשר משחק המהמר במכונה, מקבל פרס בעל ערך אקראי מההתפלגות האופיינית לאותה מכונה. שאיפתו של המהמר הינה להגיע לתגמול מצטבר מירבי על ידי משחק סדרתי בסט המכונות. כלומר, הסוכן דוגם אפשרות אחת בכל תור במשחק, מקבל את תגמולו ובהינתן המידע אשר צבר עד כה, מחליט מה תהיה בחירתו הבאה. מטרת המהמר הינה לשחק כמה שפחות במכונות שאינן מניבות את התגמול המקסימלי, כלומר, למזער את החרטה. מזעור החרטה הינו ההפרש בין סך כל התגמולים אשר צבר המהמר במהלך המשחק, לבין סך כל התגמולים שיכל הסוכן לקבל, זאת אילו היה יכול לשחק תמיד את הזרוע האופטימלית. חשוב לציין כי מזעור החרטה הינו שקול למקסום התגמול המצטבר. כלומר, לצורך מזעור החרטה, הסוכן צריך לבחור בכל תור, האם לשחק במכונת המזל, אשר לכאורה, תניב לו את התגמול הגבוה ביותר או לחפש זרוע אשר תניב תגמול יותר גבוה. מהות ההתלבטות של הסוכן בבעיית ה־ MAB, הינה השאיפה לאזן בין משחק במכונות המזל, אשר תוחלתן אינה ידועה לו בסבירות גבוהה, אם כי עשויות להיות בעלות תגמול גבוה, לבין ניצול מכונות המזל, אשר כבר הסתמנו כטובות בדגימות, לצורך קבלת תגמול מירבי. דילמה זו, בין הגלוי למצוי, נקראת Exploration Exploitation Dilemma.

קיימות גרסאות רבות לבעיית ה־ MAB. כאשר במרבית הגרסאות הללו, אנו מניחים כי כאשר הסוכן בוחר לשחק בזרוע מסויימת, הינו נחשף למשוב בעל ערך מספרי, לדוגמא: "זרוע מספר

המחקר נעשה בהנחיית פרופ' ניר אילון בפקולטה למדעי המחשב.

אני מודה לטכניון שנתן מלגה אם קבלת על התמיכה הכספית הנדיבה בהשתלמותי.

קופסאות שחורות של בנדיטים בדו־קרב

חיבור על מחקר

לשם מילוי חלקי של הדרישות לקבלת התואר

מגיסטר למדעים בהנדסת חשמל

עמית וולפנפלד

הוגש לסנט הטכניון – מכון טכנולוגי לישראל

תשרי כ"ט תשע"ו חיפה אוקטובר 2015

קופסאות שחורות של בנדיטים בדו־קרב

עמית וולפנפלד