Title

Amit Wolfenfeld[1]

Technion

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 agent learns from stochastic feedback in the form of real-valued rewards.
The Dueling Bandits setting is an online learning framework in which actions are restricted to stochastic comparisons between pairs of strategies. It models settings were absolute rewards are difficult to estimate but pairwise preferences are readily available.
In this paper we propose several new methods for the Dueling Bandit Problem. Our approach extends the Doubler and Sparring algorithm proposed on [???]. We show empirical results using real data from Microsoft Research's LETOR project.

1 Introduction

Multi-armed bandit (MAB) algorithms have received considerable attention and have been studied quite intensely in machine learning since the 50's when Lai and Robbins released their paper [?]. The huge interest in this top is not really surprising, sue to the fact that this (MAB) setting is not only theoretically challenging but also extremely useful, as can be seen from their use in a wide range of applications. MAB algorithms are used today for solving many problems such as - in search engines [?], online advertisement [?], and recommendation systems [?]. 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 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 option. 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 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 the we assume a numerical reward such as arm 1 has the value of 0.7, however there are many applications where such assumption does not hold, were the feedback is a pairwise comparison (arm 1 is better then 2) as opposed to standard bandits. There are many cases in the world of machine learning where where precise feedback is not available, and only preference feedback is available. In these cases weakly supervised learning and preference learning can be used.

In preference learning, feedback is typically represented in a purely qualitative way, namely in terms of pairwise comparisons or rankings. Feedback of this kind can be useful in online learning, too, as has been shown in online information retrieval. Web search and internet marketing are two examples that show the importance for the Dueling Bandits setting. For web-search, were there is no implicit feedback, only relative. When a web-search user prefers one choice over the other. As for internet marketing, when advertises aim to sell products online, they will direct users to their sale page. Every advertiser wants 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 (MAB) 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. And indeed, extensions of that kind have received increasing attention in the recent years. The aim of this paper is to provide a survey of the state-of-the-art in the field of Dueling Bandits Algorithms and present several new algorithms. In section 2 we provide a scientific background for the Dueling Bandits Problem. In section 3 we survey the state-of-the-art algorithms. In section 4 we present new algorithm that our perform the algorithm described in section 3 and In section 5 we present the empirical results.

2 Scientific Background

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

2.1 Multi Armed Bandit

As described in the previous section the multi armed bandit problem is a sequential decision problem, where a learner explores and exploits a stochastic environment. The name "bandit" stems from a well known type of gambling machines, where at each turn a player pulls a mechanic lever (an "arm") and receives a reward with some (usually small) probability. In this setting, the learner performs actions, referred to as arm pulls. The arms belong to some finite set $X = x_1, .., x_k$, and is denoted as $|X| = K$. Each arm $x_i \in x_1, .., x_2$ has some probability distribution over $[0, 1]$, with expectation μ_{x_i}. Throughout this paper we assume the existence of a unique best arm:

$$x^* = argmax_{x_i}(\mu_{x_i})$$

At each round $t \in 0, 1, .., T$ the learner "pulls" an arm $x_i \in x_1, .., x_2$ and acquires a stochastic reward or utility u_t, independently of all previous rewards (i.i.d). For each arm x_i and for all rounds $t \geq 1$, $n_{x_i}(t)$ is denotes the number of times arm x_i has been "played".

2.2 UBDB

To formalize the problem of learning from preferences, we consider the following interactive online learning model, which we call the Utility-Based Dueling Bandits Problem (UBDB) similar to (Yue et al., 2012, Yue and Joachims, 2011). At each iteration t, the learning system presents two actions $x_t, y_t \in X$ to the user, where X is the set (either finite or infinite) of possible actions. Each of the two actions has an associated random reward (or utility) for the user, which we denote by u_t and v_t, respectively. The quantity u_t (resp. v_t) is drawn from a distribution that depends on x_t (resp. y_t) only. We assume these utilities are in $[0, 1]$. The learning system is rewarded the average utility$U_a v(t) = (u_t + v_t)/2$ of the two actions it presents, but it does not observe this reward. Instead, it only observes the user's binary choice among the two alternative actions xt, y_t, which depends on the respective utilities u_t and v_t. In particular, we model the observed choice as a $\{0, 1\}$ valued random variable b_t distributed as

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

where $\phi : [0, 1] \times [0, 1] \rightarrow [0, 1]$ is a link function. Clearly, the link function has to satisfy $\phi(A, B) + \phi(B, A) = 1$. The binary choice is interpreted as a

stochastic preference response between the left alternative x_t (if $b_t = 0$) and the right alternative y_t (if $b_t = 1$). The utility U_{av} captures the overall latent user experience from the pair of alternatives. In the utility based dueling bandit game (UBDB), the algorithm chooses $(x_t, y_t) \in X \times X$ at each step, and a corresponding pair of random utilities $(u_t, v_t) \in [0, 1]$ are given rise to, but not observed by the algorithm. We assume u_t is drawn from a distribution of expectation $\mu(x_t)$ and v_t independently from a distribution of expectation $\mu(y_t)$. The algorithm observes a choice variable $b_t \in \{0, 1\}$ distributed according to the law (1.1). This random variable should be thought of as the outcome of a duel, or match between x_t and y_t. The outcome $b_t = 1$ (resp. $b_t = 0$) should be interpreted as "y_t is chosen' (resp. "x_t is chosen"). 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

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

The unobserved reward is $U_{av}(t) = (u_t+v_t)/2$, and the corresponding cumulative utility based regret after T steps is $R_U(T) = \sum_{t=1}^{T} \mu(x^*) - U_{av}(t)$, where $x^* = argmax_{x \in X} \mu(x)$. This implies that expected zero regret is achievable by setting $(x_t, y_t) = (x^*, x^*)$. In practice, these two identical alternatives would be displayed as one, as would naturally happen in interleaved retrieval evaluation (Chapelle et al., 2012). It should be also clear that playing (x^*, x^*) is pure exploitation, because the feedback is then an unbiased coin with zero exploratory information.

2.3 PBDB

The notion of "Dueling Bandits" refers to the stochastic MAB problem with pairwise comparisons as actions has been intensively studied studied[?]. In this subsection we will define the relevant terms that are used throughout this paper. Same as in the UBDB setting consider a fixed set of arms $X = x_1, .., x_k$. As actions, the learner performs a comparison between any pair of arms x_i and x_j , meaning the action space is identified with the set of index pairs (i, j) such that $1 \leq i \leq j \leq K$. In this paper we characterise the feedback of the comparison by an unknown Preference Matrix P.

$$P = [p_{x_i,x_j}] \in [0, 1]^{K \times K}$$

To be more precise - for each pair of arms (x_i, x_j), this relation specifies the probability

$$Pr(x_i \succeq x_j) = p_{x_i,x_j} \tag{2}$$

of observing a preference for x_i in a direct comparison with x_j. Meaning, each p_{x_i,x_j} defines a Bernoulli distribution. Throughout this paper we assume the these probabilities are independent and stationary during all rounds $t \in T$.

Meaning that whenever two arms is played (x_i, x_j) and compared, the outcome is distributed according to (1), without any dependencies on the previous

iterations. The relation P is reciprocal in the sense that $p_{i,j} = 1 - p_{j,i}$ for all $i, j \in [K] = \{1, ..., K\}$. In this setting we allow ties in the comparison of two arms. In several algorithm this is solved by a random winner or splitting the reward for each arm. Arm x_i is said to beat arm x_j if $p_{i,j} > 1/2$, meaning the probability of winning in a pairwise comparison is larger for x_i than it is for x_j . The closer $p_{i,j}$ is to $1/2$, the harder it is to distinguish between arm x_i and arm x_j based on a finite sample set from $Pr(x_i \succeq x_j)$. This resembles the case in the standard MAB problem where the gap $\Delta_{i,j}$ is very small. When $p_{i,j} = 1/2$, the learner cannot decide which arm is better based on a finite number of pairwise comparisons. In order to define the regret we need some kind of quantity.

$$\Delta_{i,j} = p_{i,j} - 1/2$$

seems to be a good quantity to characterize the "damage" that is incurred for each choice the learner makes during the Deuling Bandits game. As opposed to the standard bandit game $\Delta_{i,j}$ can be negative, in which the quantity used for the multi-armed bandit task is always positive and depends on the gap between the means of the best arm and the suboptimal arms.

2.4 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 $t \in T$. And so, in each round t, the learner selects a pair $1 \leq i(t) \leq j(t) \leq K$ and observes

$$\begin{cases} x_{i(t)} \succeq x_{j(t)} & \text{with probablity } p_{i,j} \\ x_{j(t)} \succeq x_{i(t)} & \text{with probablity } p_{j,i} \end{cases} \tag{3}$$

In this paper the pairwise probabilities $p_{i,j}$ can is 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_i and x_j , and denote the size of this set by $n_{i,j}$, or the number of times x_i and x_j have been compared. We denote the number of times x_i "won" over x_j by $w_{i,j}$ and $w_{j,i}$ the number of "wins" of x_j over x_i. It is easy to see that $n_{i,j} = n_{j,i} = w_{i,j} + w_{j,i}$ and so the estimation of $p_{i,j}$ up to iteration t is then given by

$$\hat{P}_{x_i,x_j} = \frac{w_{i,j}}{n_{i,j}} = \frac{w_{i,j}}{w_{i,j} + w_{j,i}}$$

As mention above, in this paper we assume that the samples are independent and identically distributed (i.i.d), $\hat{p}_{i,j}$ is a good estimate of the pairwise probability (1). This estimate might be biased, since $n_{i,j}$ depends on the choice of the learner, which in turn depends on the out come of the previous rounds and so $n_{i,j}$ itself is a random quantity. As in most MAB algorithms a high probability confidence interval is obtained by the Hoeffding bound. The confidence intervals may differ from one algorithm to another, but usually it is of the form $[p_{i,j} \pm c_{i,j}]$. According

to this definition arm x_i beats arm x_j with high probability if $p_{i,j} + c_{i,j} > 1/2]$, and, x_i is beaten by arm x_j with high probability, if $p_{i,j} + c_{i,j} < 1/2]$.

In the preference-based setting, defining a the regret is not as easy as in the standard MAB setting or utility-based setting (defined above), where the sub-optimality of an action can be expressed according to the utility or reward. Since the learner compares between two arms, both of them should be effect the regret, both of them should be compared to the optimal arm. Usually the following regret is used for the preference based setting. Assuming the learner selects arms $x_{i(t)}$ and $x_{j(t)}$ at time t. The cumulative regret incurred by the learner up to time T is:

$$R_P(T) = \sum_{t=1}^{T} \frac{\Delta_{i^*,i(t)} + \Delta_{i^*,j(t)}}{2} \tag{4}$$

We will show that using the definition of the linear link function we both utility based regret and preference based regret are the same (till a factor of 2):

$$p_{i^*,k} = \phi_{lin}(\mu(x^*), \mu(x_k)) = \frac{1 + \mu(x_k) - \mu(x^*)}{2} \tag{5}$$

Incorporating (5) in the definition of $\Delta_{i^*,k}$ we get

$$\Delta_{i^*,k} = p_{i^*,k} - \frac{1}{2} = \frac{\mu(x^*) - \mu(x_k)}{2}$$

And so the total regret is defined:

$$R_P(T) = \sum_{t=1}^{T} \frac{\Delta_{i^*,i(t)} + \Delta_{i^*,j(t)}}{2} = \sum_{t=1}^{T} \frac{\frac{\mu(x^*)-\mu(x_{i(t)})}{2} + \frac{\mu(x^*)-\mu(x_{j(t)})}{2}}{2} =$$

$$\frac{1}{2}\sum_{t=1}^{T} \mu(x^*) - \frac{\mu(x_{i(t)}) + \mu(x_{j(t)})}{2} = \frac{1}{2}\sum_{t=1}^{T} \mu(x^*) - U_{av} = \frac{1}{2}R_U(T)$$

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
$U = [u_{x,y}]$	Utility function
$\Theta_{x,y}$	Random variable

3 Related Work

We start by surveying dueling bandit algorithms. In this section we have included Interleaved Filtering, Beat the Mean Bandit, RUCB, RCS, SAVAGE and went into greater detail for the Sparring and Doubler algorithms they served as a foundation for our new approach.

Interleaved Filter Yue et al.[?] propose an explore-then-exploit algorithm. Starts with all the arms in the potential arms set - $\mathcal{P}$, it chooses a random arm as a candidate. The exploration step consists of a simple elimination strategy,called Interleaved Filtering (IF), which identifies the best arm with probability at least $1-\delta$. The IF algorithm successively selects an arm and compares it to other arms (Bernoulli trial). More specifically, the currently selected arm x_i is compared to the rest of the arms in the potential arms set $\mathcal{P}$. IF maintains a mean and confidence interval for each pair of arms being compared. If an arm x_j beats x_i , that is, $q_{i,j} + c_{i,j} < 1/2$, then x_i is eliminated, and x_j is compared to the rest of the arms in the potential arms set $\mathcal{P}$. In addition, a simple pruning technique can be applied: if $q_{i,j} - c_{i,j} < 1/2$ for an arm x_j at any time, then x_j can be eliminated, as it cannot be the best arm any-more (with high probability). After the exploration phase, the exploitation phase simply takes the most promising arm x^* found by IF and repeatedly compares x^* to itself.

Algorithm 1: Interleaved Filter

input : T,$X = \{x_1, ..., x_K\}, \delta$
$t \leftarrow 1$
$\mathcal{P} \leftarrow X$ Choose $x_t \in \mathcal{P}$ randomly
$\mathcal{P} \leftarrow \mathcal{P} \backslash x_t$
while $|\mathcal{P}| > 1$ **do**
  for $x \in \mathcal{P}$ **do**
    compare x, x_t
    update $\hat{C}_{x_t,x}$, $\hat{P}_{x_t,x}$
  while $\exists x \in s.t.$ $\left(\hat{P}_{x_t,x} > 1/2 \wedge 1/2 \notin \hat{C}_{x_t,x}\right)$ **do**
    $\mathcal{P} \leftarrow \mathcal{P} \backslash \{x\}$
  if $\exists x_{t+1} \in s.t.$ $\left(\hat{P}_{x_t,x_{t+1}} < 1/2 \wedge 1/2 \notin \hat{C}_{x_t,x_{t+1}}\right)$ **then**
    $x_t \leftarrow x_{t+1}$,
    $\mathcal{P} \leftarrow \mathcal{P} \backslash \{x_{t+1}\}$
    $\forall x \in \mathcal{P}$ reset $\hat{C}_{x_t,x}$, $\hat{P}_{x_t,x}$
  $t \leftarrow t + 1$

Beat The Mean Bandit Yue and Joachims [?] propoed an elimination strategy very similar to IF. Beat the Mean (BTM) - a preference-based online learning algorithm. The difference is where IF compares each arm to all the other arms in the potential arm set, BTM picks an arm that has the least plays n_x and compares it with a randomly chosen arm from the set of potential arms. BTM maintains a score for each arm $\hat{P}_x = \frac{w_x}{n_x}$, where w_x is the number of times arm x has won the comparison (not taking into account which arm it was compared to. The name of this algorithm comes from the idea of comparing the arm against the "mean" arm (bandit in this case). The idea is beating half of the arms when comparing to the "mean" arm. BTM also maintains a confidence interval c^* around the score $\hat{P}_{x_i}$, using this interval arms can be removed from the potential arm set $\mathcal{P}$ if an arm is significantly low in score from "strongest" arm (the arm with the highest score).

Algorithm 2: Beat The Mean Bandit

input : T,$X = \{x_1, ..., x_K\}, N, c_{\delta,\gamma}(\cdot)$
$\mathcal{P} = X$
$\forall x \in \mathcal{P}, n_x \leftarrow 0$
$\forall x \in \mathcal{P}, w_x \leftarrow 0$
$\forall x \in \mathcal{P}, \hat{P}_x = w_x/n_x$ or 0 if $n_x = 0$
$n^* = \min_{x \in \mathcal{P}} n_x$
$c^* = c_{\delta,\gamma}(n^*)$ or 1 if $n^* = 0$
$t \leftarrow 0$

while $|\mathcal{P}| > 1 \wedge t < T \wedge n < N$ **do**
  $x_t \leftarrow argmin_{x \in \mathcal{P}} n_x$
  select $y_t \in \mathcal{P}$ at random, compare x_t with y_t
  if x_t *wins* **then**
    $w_{x_t} \leftarrow w_{x_t} + 1$
  $n_{x_t} \leftarrow n_{x_t} + 1$
  if $min_{y_t \in W}(\hat{P}_{y_t}) + c^* \leq max_{y_t \in \mathcal{P}}(\hat{P}_{y_t}) - c^*$ **then**
    $y_t \leftarrow argmin_{x \in \mathcal{P}} \hat{P}_x$
    $\forall x \in \mathcal{P}$ delete comparison with y_t from w_x, n_x
    $\mathcal{P} \leftarrow \mathcal{P} \backslash \{y_t\}$
  $t \leftarrow t + 1$

RUCB Relative Upper Confidence Bound by Zoghi et al. [?], adapts the most commonly used algorithm UCB [?] in the standard MAB setting to the Dueling Bandit setting. This only assumption this algorithm holds on the arms is that exists a Condorcet winner - meaning an arm that beats all the other arms on average. Similar the the UCB algorithm the RUCB algorithm, this algorithm is based on the "optimism in the face of uncertainty" principle, meaning that the arms that are selected hold the highest pairwise upper bounds - u_{x_i,x_j}. In each round, RUCB selects the arms to compare from the set of potential Condorcet winners, meaning the set of arms for which all u_{x_i,x_j} values are above $1/2$. Then x_i is compared to the arm $x_j = argmax_x u_{x,x_j}$, in order to minimize regret, taking into account the optimistic estimates. In the analysis of the infinite horizon RUCB algorithm, both expected and high probability regret bounds are provided and both bounds are $O(KlogT)$.

Algorithm 3: RUCB

input : $X = \{x_1, ..., x_K\}, \alpha > 1/2, T \in \{1, 2, ...\} \cap \{\infty\}$
$W = [w_{x,y}] \leftarrow 0_{K \times K}$

for $t = 1, 2.., T$ **do**
- $U = [u_{x,y}] \leftarrow \frac{W}{W+W^T} + \sqrt{\frac{\alpha \cdot \ln(t)}{W+W^T}}$
- $u_{x,x} \leftarrow 0$ for each $x \in \{x_1, ..., x_K\}$
- Pick any x_t that satisfies $u_{x_t,x} \geq 1/2, \forall x$.
- If no x_t exists pic x_t randomly from X.
- $y_t \leftarrow argmax_x u_{x,x_t}$
- Compare arms x_t and y_t and increment w_{x_t,y_t} or w_{y_t,x_t} depending on which arm won.

RCS (Munos et al., 2014) Relative Confidence Sampling is very similar to RUCB in the manner that it aims to minimize cumulative regret and it does not eliminate arms from a potential set of arms. Although RCS is very similar to the RUCB algorithm it differs from it by sampling arms from a set of arms instead of picking an arm from the set of potential Condorcet winners. The main idea behind this sampling technique is to use the superior performance of Thompson Sampling [?], in the same manner it is used in the K-armed bandit setting.

Algorithm 4: RCS

input : $X = \{x_1, ..., x_K\}, \alpha > 1/2, T \in \{1, 2, ...\} \cap \{\infty\}$
$W = [w_{x,y}] \leftarrow 0_{K \times K}$
for $t = 1, 2.., T$ **do**
  $\Theta(t) = \frac{\mathbb{1}_{K \times K}}{2}$
  for $x_i, x_j \in X$ *s.t.* $i < j$ **do**
    $\Theta_{x_i,x_j}(t) \sim Beta(w_{x_i,x_j} + 1, w_{x_j,x_i} + 1)$
    $\Theta_{x_j,x_i}(t) = 1 - \Theta_{x_i,x_j}(t)$
  Pick any x_t that satisfies $\Theta_{x_t,x} \geq 1/2, \forall x$.
  If no x_t exists pic x_t that was chosen list often.
  $U = [u_{x_i,x_j}] \leftarrow \frac{W}{W+W^T} + \sqrt{\frac{\alpha \cdot \ln(t)}{W+W^T}}$
  $U_{x_i,x_i} \leftarrow 0$ for each $x_i \in X$
  $y_t \leftarrow argmax_y u_{y,x_t}$
  Compare arms x_t and y_t and increment w_{x_t,y_t} or w_{y_t,x_t} depending on which arm won.
,

SAVAGE (Uryoy et al., 2013) works by reducing a box-shaped confidence set until a single arm remains. Sensitivity analysis of variables for generic exploration (SAVAGE) is a more recent algorithm that performs better than both IF and BTM by a wide margin when the number of arms is small. SAVAGE compares pairs of arms uniformly randomly 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 pool of arms under consideration.

Sparring (Ailon et al., 2014) works by drawing a pair of arms from two SBM's in each round and comparing them as all the previous algorithms. Once observing the preference the algorithm updates the preferred (according to the observation) SBM.

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's, with a possibility of a small overhead [?].

Algorithm 5: SAVAGE

input : $X = \{x_1, ..x_K\}, \alpha$
$\mathcal{P} \leftarrow X$
$\forall x \in \mathcal{P}, n_x \leftarrow 0$

while $|\mathcal{P}| > 1 \wedge t < T$ **do**
  $x_t, y_t \leftarrow= argmin_{x_t} n_{x_t}$
  Compare arms x_t, y_t
  $n_{x_t} \leftarrow n_{x_t} + 1$
  $n_{y_t} \leftarrow n_{y_t} + 1$
  Update W
  $U = [u_{x_i,x_j}] \leftarrow \frac{W}{W+W^T} + \sqrt{\frac{\alpha}{W+W^T}}$
  $\mathcal{P} \leftarrow \mathcal{P} \backslash \left\{ x \mid \sum_y (u_{x,y} > 0.5) < K) \right\}$
  $t \leftarrow t + 1$

Algorithm 6: Sparring

input : $X, \mathcal{L} = \{x_1\}$, $\hat{f}_0 = 0, \mathcal{S}_R, \mathcal{S}_L$

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

Sparring with Thompson Sampling In our work we have experimented with both SBMs black and white boxes. The configuration that showed the best result with minimum regret, is the following:

Algorithm 7: Sparring with Thompson Sampling

input : $X, \mathcal{L} = \{x_1\}$, $\hat{f}_0 = 0, \mathcal{S}_R, \mathcal{S}_L$

$t \leftarrow 1$
while *True* **do**
  $\forall i \in [1..K] \Theta_{L,i} \sim Beta(Success_{L,i} + 1, Fails_{L,i} + 1)$
  $\forall i \in [1..K] \Theta_{R,i} \sim Beta(Success_{R,i} + 1, Fails_{R,i} + 1)$
  $x_t \leftarrow argmax(\Theta_{L,i})$, $y_t \leftarrow argmax(\Theta_{R,i})$
  play(x_t, y_t)
  observe b_t
  $Success_{L,x_t} \leftarrow Success_{L,x_t} + \mathbb{1}_{b_t=0}$
  $Fails_{L,x_t} \leftarrow Fails_{L,x_t} + \mathbb{1}_{b_t=1}$
  $Success_{R,y_t} \leftarrow Success_{R,y_t} + \mathbb{1}_{b_t=1}$
  $Fails_{R,y_t} \leftarrow Fails_{R,y_t} + \mathbb{1}_{b_t=0}$
  $t \rightarrow t + 1$

Doubler (Ailon et al., 2014) handles a large or possibly infinite set of arms X. 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 possible. This algorithm divides the time axis to exponentially growing epochs (First epoch is 2 rounds, second epoch is 4, third epoch is 8 and so forth). In each epoch, the left player plays according to some fixed (stochastic) strategy, while the right one plays adaptively according to a strategy provided by a SBM. At the beginning of a new epoch the distribution governing the left arm changes in a way that mimics the actions of the right arm in the previous epoch.

In the finite case, one may set the SBM S to the standard UCB, and obtain according to Corollary 3.3 the regret of Doubler is at most $O(H \log^2(T))$ where $H = \sum_{i=1}^{K} \Delta_i^{-1}$.

Algorithm 8: Doubler

input : $X, \mathcal{L} = \{x_1\}$, $\hat{f}_0 = 0, \mathcal{S}$

Initialization
$p \leftarrow 1$
while *True* **do**
  For each epoch
  reset($\mathcal{S}$)
  for $t = 2^{p-1}$ **to** 2^p **do**
    choose x_t uniformly from $\mathcal{L}$
    $y_t \leftarrow advance(\mathcal{S})$
    play (x_t, y_t), observe choice b_t
    feedback $(\mathcal{S}, b_t)$
  $\mathcal{L} \leftarrow$the multi-set of arms played as y t in the last for-loop.
  $p \rightarrow p + 1$

4 Our Approach

Improved Doubler The main drawback of the Doubler algorithm is that in each epoch the SBM is reset and as a result all the acquired data is lost. Therefore adding an extra $log(T)$ factor to the total regret.
Instead of initializing the right arm's SBM at each epoch and losing all the data, we will store this data. Let D_p denote the distribution of the left arm in the p'th epoch. By knowing $f_p := E_{x \in D_p} \mu(x)/2$, and by using $b_t + f_p$ as feedback on the right side we could separate y_t from x_t in the feedback.

Algorithm 9: Improved Doubler

input : x_1 fixed in $X, \mathcal{L} = \{x_1\}$, $\hat{f}_0 = 0$,$\mathcal{S}$

Initialization
$p \leftarrow 1$
while *True* **do**
  for $t = 2^{p-1}$ **to** 2^p **do**
    choose x_t uniformly from $\mathcal{L}$
    $y_t \leftarrow advance(\mathcal{S})$
    play (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.
  $\hat{f}_p \leftarrow \hat{f}_p + \sum_{s \in T_p} b_s / 2^{p-1} - 1/2$
  $p \rightarrow p + 1$

f_p is defined by $f_{p+1} = \frac{1}{|T_p|} \sum_{t \in T_p} \mu(y_t)/2$

and so: $f_{p+1} = \frac{1}{|T_p|} \sum_{t \in T_p} E[b_t - \frac{1-\mu(x_t)}{2}] = \frac{1}{|T_p|} \sum_{t \in T_p} E[b_t - f_{p-1} - \frac{1}{2}]$

We need to estimate f_p with $\hat{f}_p$ and so we get

$$\hat{f}_i = \hat{f}_{i-1} + \frac{\sum_{t \in T_i} (2b_t - 1)}{|T_{i-1}|}$$

Let's break it down a bit:

$\hat{f}_{i+1} = \hat{f}_i + \frac{\sum_{t \in T_i} (2b_t - 1)}{|T_i|} = \hat{f}_i + \frac{\sum_{t \in T_i} (2b_t - 1)}{2^i} = \hat{f}_i + \frac{\sum_{t \in T_i} 2b_t}{2^i} - 1 = \hat{f}_i + B_i - 1 =$

$\hat{f}_{i-1} + B_{i-1} - 1 + B_i - 1 = ... = \hat{f}_0 + \sum_{j=1}^{i} B_j + i$

From her we can bound $\hat{f}_i$. First lets look at $\sum_{j=1}^{i} B_j$, this equals $E[b]$ which is bounded by $[0, 1]$ and i which is $log(T)$. And so we can bound $\hat{f}$ by $\log(T)$.

Unforgetful Thompson Sampling Doubler Another improvement of the Doubler algorithm is the Unforgetful Thompson Sampling Doubler algorithm - here we do not reset the SBM in each epoch and use the Thompson Sampling method. This approach outperformed all the other algorithms apart from the Thompson Sampling Sparring approach mentioned in section 3.

Algorithm 10: Unforgetful Thompson Sampling Doubler

input : x_1 fixed in $X, \mathcal{L} = \{x_1\}$, $\hat{f}_0 = 0$,$\mathcal{S}$

Initialization
$p \leftarrow 1$
while *True* **do**
  for $t = 2^{p-1}$ **to** 2^p **do**
    choose x_t uniformly from $\mathcal{L}$
    $\forall i \in [1..K] \Theta_{R,i} \sim Beta(Success_{R,i} + 1, Fails_{R,i} + 1)$
    $y_t \leftarrow argmax(\Theta_{R,i}))$
    play (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.
  $p \rightarrow p + 1$

Balanced Doubler Another improvement of the Doubler algorithm is the Balanced Doubler algorithm.In this algorithm we assume we know Δ ($\Delta = \mu(x_1) - \mu(x_2)$ were x_1 is the arm with the highest estimated reward and x_2 is the arm with the second highest estimated reward.

Algorithm 11: Balanced Doubler

input : x_1 fixed in $X, \mathcal{L} = \{x_1\}$

Initialization

$p \leftarrow 1$

while *True* **do**

  Play each arm $\frac{k}{\Delta}$ times

  for $t = 2^{p-1}$ **to** 2^p **do**

    choose x_t uniformly from $\mathcal{L}$

    $y_t \leftarrow argmax_i \left(\frac{1}{p} \sum_{j=1}^{p} \frac{1}{n_{i,j}} \sum_{t \in T_{i,j}} b_t + \sqrt{\alpha \cdot \log(t) \cdot (\sum_{j=1}^{p} \frac{1}{n_{i,j}})/p^2} \right)$

    play (x_t, y_t), observe choice b_t

    update (y_t)

  $\mathcal{L} \leftarrow$the multi-set of arms played as y t in the last for-loop.

  $p \rightarrow p + 1$

Proof of $log(T)$ regret:

Lets define : $Y_i(t) = \frac{1}{p} \sum_{j=1}^{p} \frac{1}{n_{i,j}} \sum_{t \in T_{i,j}} b_t$

Where $n_{i,j}$ is the number of time arm i is played in epoch j.

$X_i = Y_i(b_t = b_t - f_j) = \frac{1}{p} \sum_{j=1}^{p} \frac{1}{n_{i,j}} \sum_{t \in T_{i,j}} b_t - f_j$

Now let's look at $\mathbb{E}(e^{\lambda X_i})$

$$\mathbb{E}(e^{\lambda X_i}) = \mathbb{E}(e^{\lambda \cdot (\frac{1}{p} \sum_{j=1}^{p} \frac{1}{n_{i,j}} \sum_{t \in T_{i,j}} (b_t - f_j)}) = \prod_{j=1}^{p} \left(\mathbb{E} \left[e^{\frac{\lambda}{p} \cdot \left(\sum_{t \in T_{i,j}} \frac{(b_t - f_j)}{n_{i,j}} \right)} \right] \right) =$$

$= \prod_{j=1}^{p} \prod_{t \in T_{i,j}} \left(\mathbb{E} \left[e^{\frac{\lambda}{p} \cdot \left(\frac{(b_t - f_j)}{n_{i,j}} \right)} \right] \right)$ Assuming that $\frac{1}{p} \cdot \left(\frac{b_t - f_j}{n_{i,j}} \right)$ is bounded by 1 and has an average of 0, we can say that

$$\prod_{j=1}^{p} \prod_{t \in T_{i,j}} \left(\mathbb{E} \left[e^{\frac{\lambda}{p} \cdot \left(\frac{b_t - f_j}{n_{i,j}} \right)} \right] \right) \leq \prod_{j=1}^{p} \prod_{t \in T_{i,j}} \left(e^{\frac{\lambda^2}{p^2 \cdot n_{i,j}^2}} \right)$$

There are $n_{i,j}$ plays in $T_{i,j}$, and so we get:

$$\prod_{j=1}^{p} \prod_{t \in T_{i,j}} \left(e^{\frac{\lambda^2}{p^2 \cdot n_{i,j}^2}} \right) = \prod_{j=1}^{p} \left(e^{\frac{\lambda^2}{p^2 \cdot n_{i,j}}} \right) = e^{\frac{\lambda^2}{p^2} \sum_{j=1}^{p} \frac{1}{n_{i,j}}}$$

And now for the main point, we wish to show that $Pr[X_j > a] < \frac{1}{t}$

$Pr[X_j > a] = Pr[e^{X_j} < e^a] \leq e^{\frac{\lambda^2}{p^2} \sum_{j=1}^{p} \frac{1}{n_{i,j}} - \lambda \cdot a}$

$\lambda = \frac{a}{\frac{2}{p^2}\left(\sum_{j=1}^{p} \frac{1}{n_{i,j}}\right)}$

with some reverse engineering and in order to get $Pr[X_j > a] < \frac{1}{t}$ we get that

$a = \sqrt{\frac{\log(t) \sum_{j=1}^{p} \frac{1}{n_{i,j}}}{p^2}}$

5 Experiments

To evaluate our algorithms, we apply them to the problem of ranker evaluation from the field of information retrieval (IR) (Manning et al., 2008). 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.

An effective way to achieve this is to use interleaved comparisons (Radlinski et al., 2008), which interleave the documents proposed by two different rankers and presents the resulting list to the user, whose resulting click feedback is used to infer a stochastic preference for one of the rankers. 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.

Our setup is built on real IR data, namely the LETOR MQ2007 dataset (Liu et al., 2007). Using this data set, we create a set of 46 rankers, each relating to a ranking feature provided in the data set, e.g., PageRank. The ranker evaluation task then corresponds to conclude which single feature is the best ranker (Hofmann et al., 2013).

We evaluated our algorithms and RCS using randomly chosen subsets from the pool of 46 rankers, yielding K-armed dueling bandit problems with $K \in \{7, 16, 46\}$.

We will present the utility based regret and the general regret (as define in section 2.1 and 2.2) on the MQ2007.

5.1 Results

We will start with displaying the results for 7 arms. From Figure 1 we can clearly see that the SAVAGE and Doubler algorithms are outperformed by the other algorithms and so will be left out from the next figures.

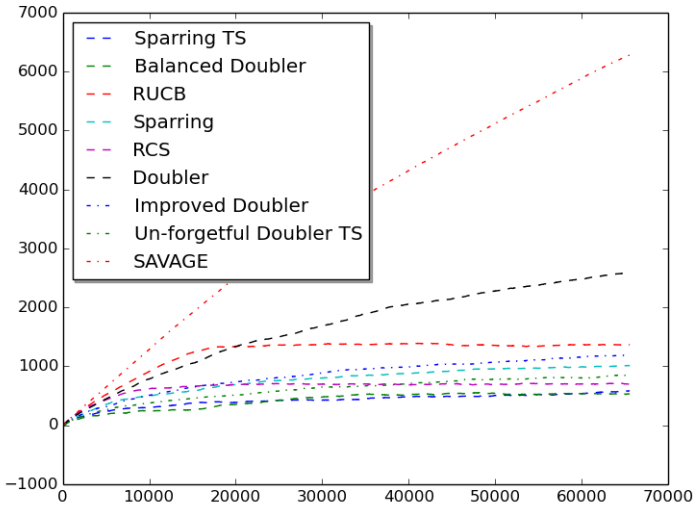


Fig. 1: Utility Based Regret with 7 Arms

For a larger number of arms we can see that RCS and RUCB are outperformed by the other algorithms.

And for 46 arms we can see that Sparring with Thompson Sampling methods performs better than all the other algorithms.

5.2 Sparring

We can see the clear advantage of using the Sparring algorithm over the RCS algorithm when dealing with a large number of arms.

With a small number of arms (Fig 1.) we can observe that the RCS algorithm quickly converges to the optimal arm.

With a larger number of arms we can see that Sparring out-performs the RCS algorithm.

Now lets look at the general regret (as defined section 2.2). Here we used 46 arms:

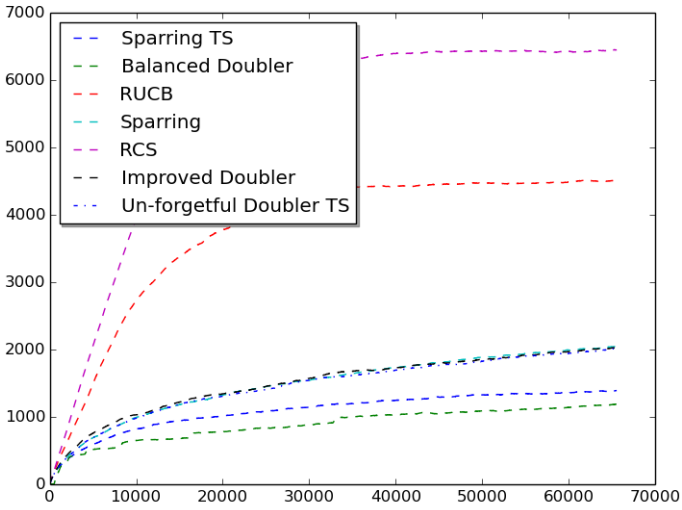
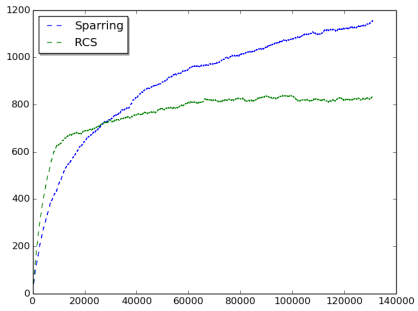


Fig. 2: Utility Based Regret with 24 Arms

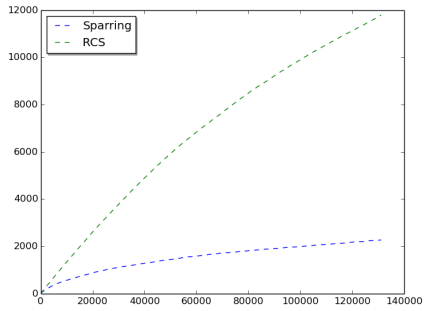
5.3 Doubler

Now lets have a look at the Doubler algorithm using the same data set.

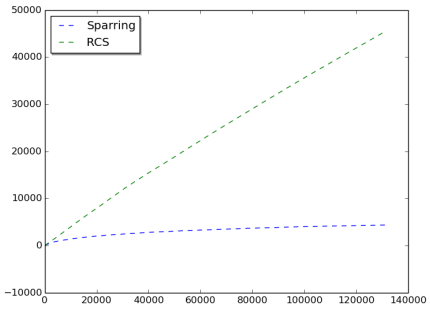
The general regret:



(a) 7 Arms



(b) 16 Arms



(c) 46 Arms

Fig. 3: RCS Versus Sparring with Utility Based Regret

5.4 Improved Doubler

Now lets have a look at the Improved Doubler algorithm using the same data set.

The general regret:

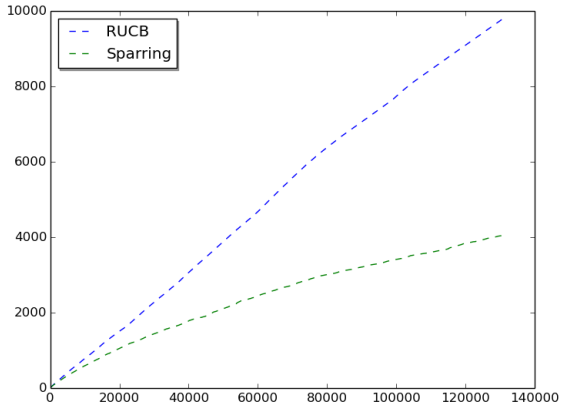


Fig. 4: RCS Versus Sparring with Preference Based Regret with 46 Arms

5.5 Balanced Doubler

Now lets have a look at the Balanced Doubler algorithm using the same data set.

5.6 Thompson Sampling Doubler

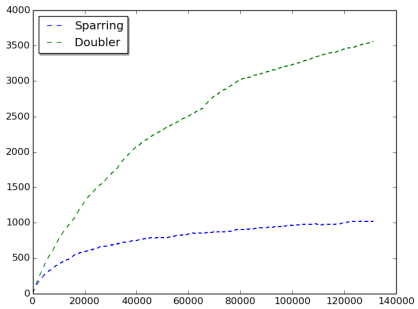
Now lets have a look at the Balanced Doubler algorithm using the same data set.

The general regret:

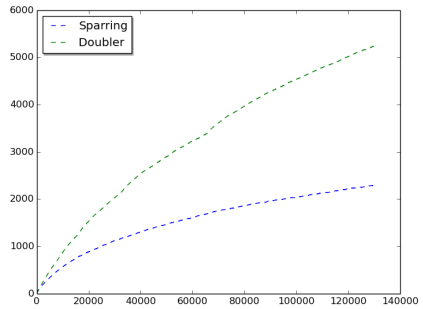
5.7 Thompson Sampling Sparring

And finally, lets show the results of the sparring algorithm when using Thompson Sampling black boxes.

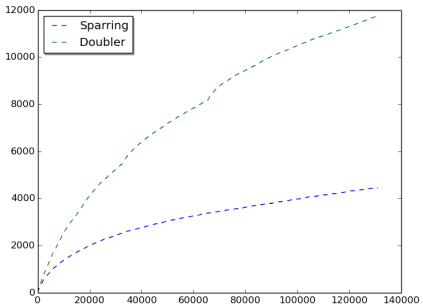
The general regret:



(a) 7 Arms



(b) 16 Arms



(c) 46 Arms

Fig. 5: Doubler Versus Sparring with Utility Based Regret

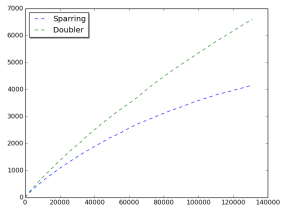
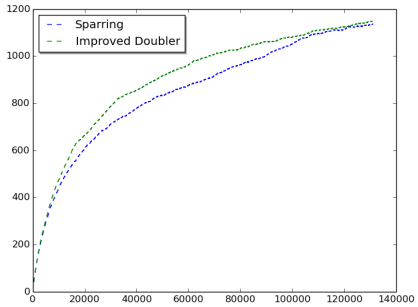
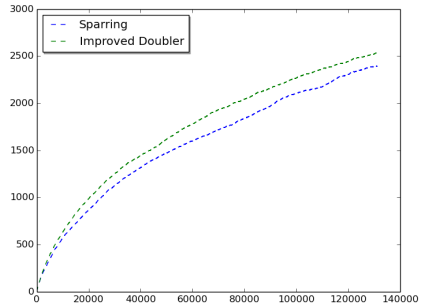


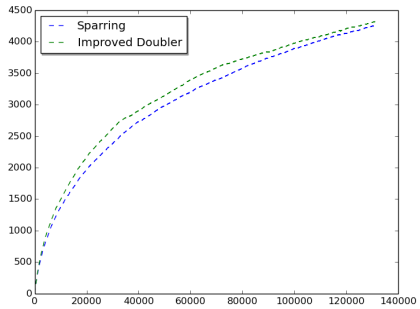
Fig. 6: Doubler Versus Sparring with Preference Based Regret with 46 Arms



(a) 7 Arms



(b) 16 Arms



(c) 46 Arms

Fig. 7: Improved Doubler Versus Sparring with Utility Based Regret

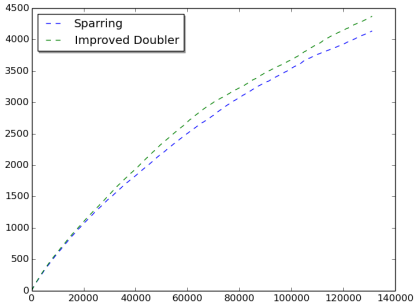


Fig. 8: Improved Doubler Versus Sparring with Preference Based Regret with 46 Arms

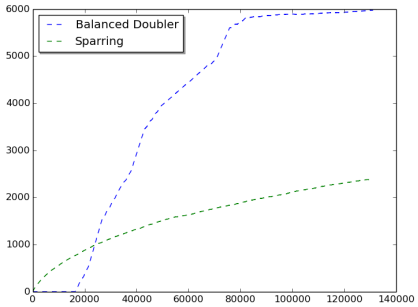
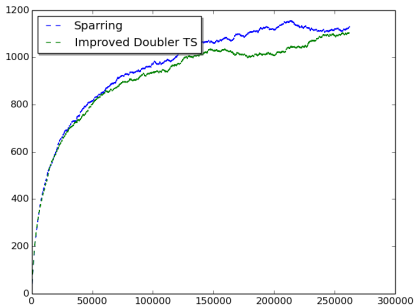
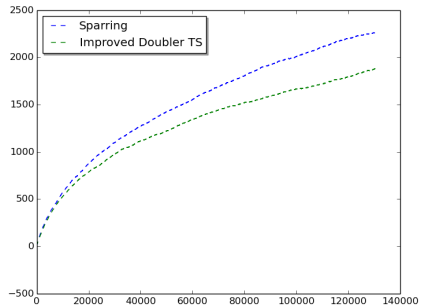


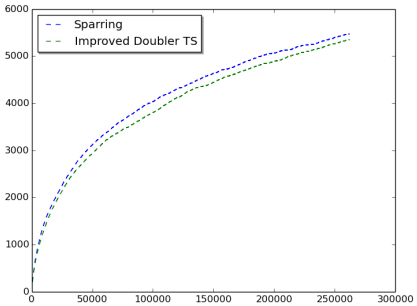
Fig. 9: Balanced Doubler Versus Sparring with Utility Based Regret with 16 Arms



(a) 7 Arms



(b) 16 Arms



(c) 46 Arms

Fig. 10: Improved Doubler Versus Sparring with Utility Based Regret

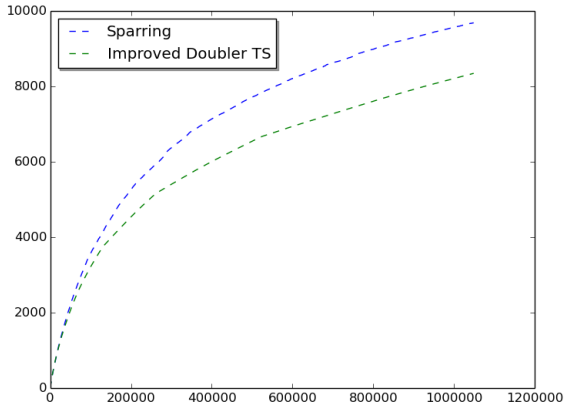


Fig. 11: Improved Doubler Versus Sparring with Preference Based Regret with 46 Arms

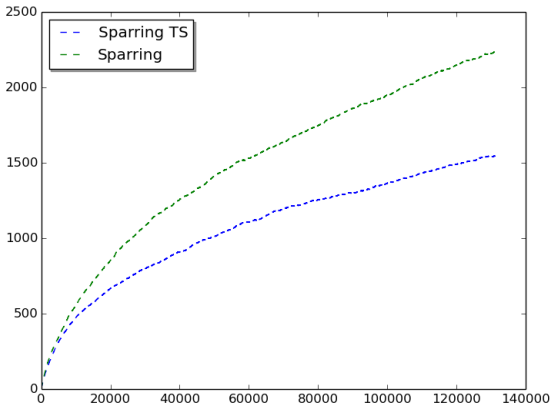


Fig. 12: Thompson Sampling Sparring Versus Sparring with Utility Based Regret with 16 Arms

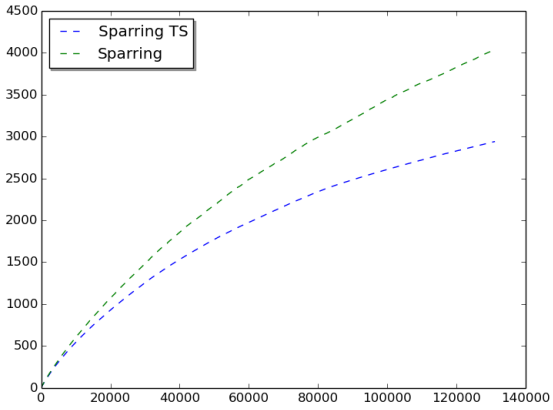


Fig. 13: Thompson Sampling Sparring Versus Sparring with Preference Based Regret with 46 Arms