CS6046 Multi Armed Bandits

Project Report: Dueling Bandits

Group Members:

Vishwajit Prakash Hegde	**ME17B039**
Akash Reddy A	**EE17B001**
Dhruv Gopalakrishnan	**AE17B004**

Paper1: The K-armed Dueling Bandits Problem

1 Introduction

- In Dueling Bandits Setting, (noisy) binary feedback about the relative reward of the two chosen strategies is available.
- Applicable in situations where users compare two objects and provide a feedback of which one is better.
- There are totally K arms and in each round two arms are selected and the algorithm receives a feedback of which one is better.
- Goal is to find the best arm and minimize the regret (defined later).
- There are two phases, Exploration followed by Exploitation.
- Two algorithms are presented, Interleaved Filter 1 (IF1) and Interleaved Filter 2 (IF2) for exploration.

2 Problem Formulation

- $\mathcal{B} = \{b_1, b_2, ..., b_k\}$ - K bandits
- $P(b > b') = \epsilon(b, b') + 1/2$ - stationary over time, $\epsilon(b, b') \in (-1/2, 1/2)$
- $b \succ b'$ implies $\epsilon(b, b') > 0$, $\epsilon_{i,j} \equiv \epsilon(b_i, b_j)$
- **Strong Regret:** $R_T = \sum_{t=1}^{T}(\epsilon(b^*, b_1^{(t)}) + \epsilon(b^*, b_2^{(t)}))$
- **Weak Regret:** $\tilde{R_T} = \sum_{t=1}^{T} min\{\epsilon(b^*, b_1^{(t)}), \epsilon(b^*, b_2^{(t)})\}$, $(b_1^{(t)}, b_2^{(t)})$ bandits chosen at iteration t, b^* is the best overall bandit.
- **Strong Stochastic Transitivity:** if $b_i \succ b_j \succ b_k$ then, $\epsilon_{i,k} \geq max\{\epsilon_{i,j}, \epsilon_{j,k}\}$
- **Stochastic Triangle Inequality:** if $b_i \succ b_j \succ b_k$ then, $\epsilon_{i,k} \leq \epsilon_{i,j} + \epsilon_{j,k}$
- **Bradley-Terry Model:** $P(b_i > b_j) = \frac{\mu_i}{\mu_i + \mu_j}$, b_i is assigned a positive real value μ_i
- **Gaussian Model:** $P(b_i > b_j) = P(X_i - X_j > 0)$, where $X_i - X_j \sim N(\mu_i - \mu_j, 2)$
  Both models satisfy Strong Stochastic Transitivity and Stochastic Triangle Inequality.

3 Algorithms and Analysis

Algorithm 1 Explore Then Exploit Solution

Input: $T, \mathcal{B} = \{b_1, \ldots, b_K\}, EXPLORE$
$(\hat{b}, \hat{T}) \leftarrow EXPLORE(T, \mathcal{B})$
for $t = \hat{T} + 1, \ldots, T$ **do**
    compare $\hat{b}$ and $\hat{b}$
end for

Figure 1: Explore Then Exploit Solution Algorithm

Algorithm 2 Interleaved Filter 1 (IF1)

Input: $T, \mathcal{B} = \{b_1, \ldots, b_K\}$
$\delta \leftarrow 1/(TK^2)$
Choose $\hat{b} \in \mathcal{B}$ randomly
$W \leftarrow \{b_1, \ldots, b_K\} \setminus \{\hat{b}\}$
$\forall b \in W$, maintain estimate $\hat{P}_{\hat{b},b}$ of $P(\hat{b} > b)$
$\forall b \in W$, maintain $1 - \delta$ confidence interval $\hat{C}_{\hat{b},b}$ of $\hat{P}_{\hat{b},b}$
while $W \neq \emptyset$ **do**
    for $b \in W$ **do**
        compare $\hat{b}$ and b
        update $\hat{P}_{\hat{b},b}, \hat{C}_{\hat{b},b}$
    end for
    while $\exists b \in W$ s.t. $\left(\hat{P}_{\hat{b},b} > 1/2 \wedge 1/2 \notin \hat{C}_{\hat{b},b}\right)$ **do**
        $W \leftarrow W \setminus \{b\}$
    end while
    if $\exists b' \in W$ s.t. $\left(\hat{P}_{\hat{b},b'} < 1/2 \wedge 1/2 \notin \hat{C}_{\hat{b},b'}\right)$ **then**
        $\hat{b} \leftarrow b'$, $W \leftarrow W \setminus \{b'\}$ *//new round*
        $\forall b \in W$, reset $\hat{P}_{\hat{b},b}$ and $\hat{C}_{\hat{b},b}$
    end if
end while
$\hat{T} \leftarrow$ Total Comparisons Made
return $(\hat{b}, \hat{T})$

Figure 2: Interleaved Filter 1 Algorithm

IF1 and IF2 return the best bandit with probability at most $1 - 1/T$.

$$\begin{aligned} \mathbf{E}[R_T] &\leq (1 - 1/T)\mathbf{E}[R_T^{IF}] + 1/T\mathcal{O}(T) \\ &= \mathcal{O}(\mathbf{E}[R_T^{IF}] + 1) \end{aligned}$$

Terminology: IF makes a **mistake** if it draws a false conclusion regarding a pair of bandits. **Match** is defined as all the comparisons IF makes between two bandits. **Round** is defined as all the matches played by one candidate $\hat{b}$. Additional step of IF2 (improvement over IF1) is called **pruning**.

Confidence Intervals: $\hat{C}_t = (\hat{P}_t - c_t, \hat{P}_t + c_t)$, where $c_t = \sqrt{log(1/\delta)/t}$, $\delta = 1/(TK^2)$

Lemma 1. *For $\delta = 1/(TK^2)$, the number of comparisons in a match between b_i and b_j is with high probability at most $\mathcal{O}(\frac{1}{\epsilon_{i,j}^2} log(TK))$. Moreover, the winner is identified with probability at least $1 - \delta$, provided $\delta \in (0, 1/2]$.*

Lemma 2. *Assuming b_1 has not been removed and $T \geq K$, then with high probability the accumulated weak regret and also (assuming stochastic triangle inequality) strong regret from any match is at most $\mathcal{O}(\frac{1}{\epsilon_{1,2}} logT)$.*

Lemma 3. *For $\delta \leq 1/(TK^2)$, IF1 makes a mistake with probability at most $1/T$.*

Random Walk Model: The sequence of candidate bandits chosen by IF can be modeled as a random walk on a graph with bandits as nodes. The bandits are sorted in preferential order such that $b_1 \succ b_2 \succ .... \succ b_K$. Let $X_i (1 \leq i < K)$ be an indicator random variable corresponding to whether a random walk starting as b_K visits b_i in the Random Walk Model.

Lemma 4. *For X_i as defined above with $1 \leq i < K, P(X_i = 1) = 1/i$, and furthermore, for all $S \subseteq \{X_1, ..., X_{K-1}\}, P(S) = \prod_{X_i \in S} P(X_i)$, meaning $X_1, ..., X_{K-1}$ are mutually independent.*

Lemma 5. *Assuming IF1 is mistake-free then it runs for $\mathcal{O}(logK)$ rounds with high probability.*

Regret Bound for Interleaved Filter 1
Theorem 1. *Running Algorithm 1 with $\mathcal{B} = \{b_1, ..., b_K\}$, time horizon $T (T \geq K)$, and IF1 incurs expected regret (both weak and strong) bounded by $\mathbb{E}[R_T] \leq \mathcal{O}(\mathbb{E}[R_T^{IF1}]) = \mathcal{O}(\frac{K log K}{\epsilon_{1,2}} logT)$*

Regret Bound for Interleaved Filter 2
Theorem 2. *Running Algorithm 1 with $\mathcal{B} = \{b_1, ..., b_K\}$, time horizon $T (T \geq K)$, and IF2 incurs expected regret (both weak and strong) bounded by $\mathbb{E}[R_T] \leq \mathcal{O}(\mathbb{E}[R_T^{IF2}]) = \mathcal{O}(\frac{K}{\epsilon_{1,2}} logT)$.*

Theorem 3. *For $\delta \leq 1/(TK^2)$, when the incumbent bandit $\hat{b}$ is defeated with $1 - \delta$ confidence by some other bandit b', then for all bandits b'' found to be empirically inferior (but lacking $1 - \delta$ confidence) to $\hat{b}$, we can conclude that b' is superior to b'' with $1 - \delta$ confidence.*

Lemma 6. *For $\delta \leq 1/(TK^2)$, IF2 makes a mistake with probability at most $1/T$.*

Lemma 7. *Assuming IF2 is mistake-free, then it plays $\mathcal{O}(K)$ matches in expectation.*

Lower Bounds:
Theorem 4. *Any algorithm ϕ for the dueling bandits problem has $R_T^{\phi} = \Omega(\frac{K}{\epsilon} logT)$, where $\epsilon = min_{b \neq b^*} P(b^*, b)$.*

Algorithm 3 Interleaved Filter 2 (IF2)

Input: $T, \mathcal{B} = \{b_1, \ldots, b_K\}$
$\delta \leftarrow 1/(TK^2)$
Choose $\hat{b} \in \mathcal{B}$ randomly
$W \leftarrow \{b_1, \ldots, b_K\} \setminus \{\hat{b}\}$
$\forall b \in W$, maintain estimate $\hat{P}_{\hat{b},b}$ of $P(\hat{b} > b)$
$\forall b \in W$, maintain $1-\delta$ confidence interval $\hat{C}_{\hat{b},b}$ of $\hat{P}_{\hat{b},b}$
while $W \neq \emptyset$ **do**
  for $b \in W$ **do**
    compare $\hat{b}$ and b
    update $\hat{P}_{\hat{b},b}, \hat{C}_{\hat{b},b}$
  end for
  while $\exists b \in W$ s.t. $\left(\hat{P}_{\hat{b},b} > 1/2 \wedge 1/2 \notin \hat{C}_{\hat{b},b}\right)$ **do**
    $W \leftarrow W \setminus \{b\}$
  end while
  if $\exists b' \in W$ s.t. $\left(\hat{P}_{\hat{b},b'} < 1/2 \wedge 1/2 \notin \hat{C}_{\hat{b},b'}\right)$ **then**
    while $\exists b \in W$ s.t. $\hat{P}_{\hat{b},b} > 1/2$ **do**
      $W \leftarrow W \setminus \{b\}$ *//pruning*
    end while
    $\hat{b} \leftarrow b'$, $W \leftarrow W \setminus \{b'\}$ *//new round*
    $\forall b \in W$, reset $\hat{P}_{\hat{b},b}$ and $\hat{C}_{\hat{b},b}$
  end if
end while
$\hat{T} \leftarrow$ Total Comparisons Made
return $(\hat{b}, \hat{T})$

Figure 3: Interleaved Filter 2 Algorithm

4 Experiments

Parameters: $T = 100000, K = 10$.
The algorithm is run for two sets of μ values with logistic model.
Set 1: $\mu = [0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9, 1]^T$
Set 2: $\mu = [0.586, 0.537, 0.497, 0.983, 0.392, 0.412, 0.005, 0.657, 0.940, 0.242]^T$
The second set is random and the first set follows arithmetic progression. The algorithm is executed 50 times and the mean regret is plotted along with 1 standard deviation error bar.

4.1 Interleaved Filter 1

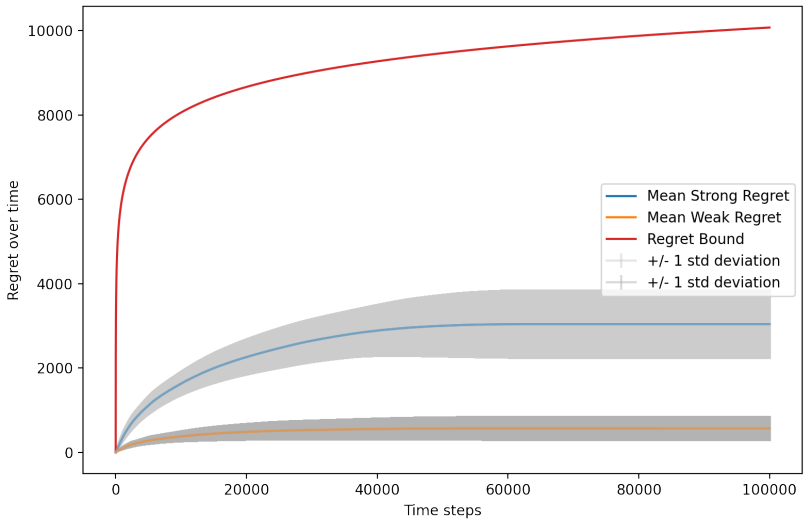

Figure 4: Regret Plot for IF1 with set 1 μ values

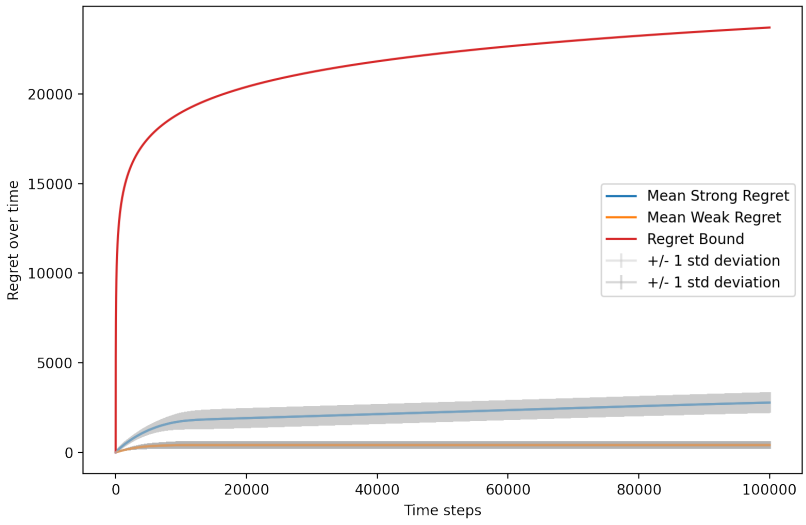

Figure 5: Regret Plot for IF1 with set 2 μ values

4.2 Interleaved Filter 2

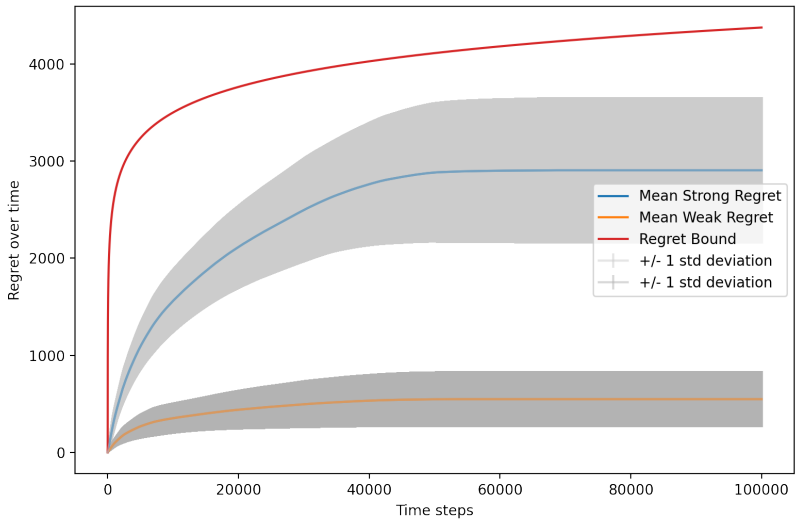

Figure 6: Regret Plot for IF2 with set 1 μ values

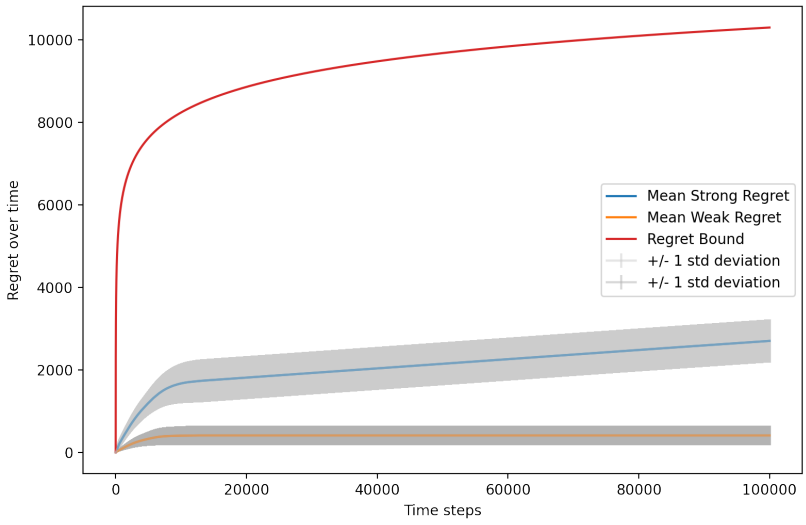

Figure 7: Regret Plot for IF2 with set 2 μ values

For the second set of μ values, the regret is increasing after 100000 iterations. This means the algorithm hasn't found the best bandits after 100000 iterations. This is because the best two bandits have μ values very close to each other in set 2. Both strong and weak regrets are within the bounds for both IF1 and IF2 algorithms.

Paper 2: Reducing Dueling Bandits to Cardinal Bandits

1 Introduction

- As opposed to the conventional Cardinal Bandits where we receive cardinal feedback of value of arms (e.g., "arm A = 2.5"), Dueling Bandits provide ordinal binary feedback (e.g., "arm A is better than arm B").
- In contrast to Paper 1 which describes *specific* algorithms for Dueling Bandits, this paper uses reduction methods utilising the *conventional* MAB methods for the Dueling Bandits setting.
- The three algorithms described: **Doubler, MultiSBM,** and **Sparring** are crafted using black-box MAB algorithms formulated as Singleton Bandit Machines (SBMs). They are shown to have regret bounds close to those of the "constituent" MAB methods encapsulated in these SBMs.

2 Utility-Based Dueling Bandits (UBDB) Formulation

- $\mathbf{x_t}, \mathbf{y_t} \in \mathbf{X}$: Two actions presented to the user by the learner at time step t.
- $\mathbf{u_t}, \mathbf{v_t} \in [\mathbf{0}, \mathbf{1}]$: Random independent rewards or "utilities" returned at time step t (of x_t, y_t respectively).
- **Average utility** $\mathbf{U_t^{av}} = \frac{\mathbf{1}}{\mathbf{2}}(\mathbf{u_t} + \mathbf{v_t})$: the (unobserved) reward that the learner is actually awarded. Neither u_t nor v_t is awarded individually to the learner.
- **Corresponding Regret** $\mathbf{R_T^{av}} \coloneqq \sum_{\mathbf{t=1}}^{\mathbf{T}}(\mu(\mathbf{x}^*) - \mathbf{U_t^{av}})$ where x^* is the arm with the highest expected utility.
- **Link functions** $\phi(\mathbf{u_t}, \mathbf{v_t})$ that map the pair (u_t, v_t) to a number in $[0, 1]$, which will represent the probability with which the user will make a choice.
- **Choice Variable** $\mathbf{b_t} \in [\mathbf{0}, \mathbf{1}]$: $b_t = 0$ (or 1) for user's action $= x_t$ (or y_t). Probabilities will depend on the pair of utilities (u_t, v_t) through the link function ϕ (reasonable assumption, as user choice usually depends on relative utility of the two options) as:

$$\Pr[b_t = 0|(u_t, v_t)] = \phi(u_t, v_t)$$
$$\Pr[b_t = 1|(u_t, v_t)] = \phi(v_t, u_t)$$

  In this paper, the regret bounds for the algorithms are established assuming the **linear** link function, according to which:

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

  If $u_t > v_t$, then $\Pr[b_t = 0] > 1/2 > \Pr[b_t = 1]$ as expected, and vice versa.

- (Another form of utility) Choice-Based Utility $U_t^{choice} \coloneqq (1 - b_t)u_t + b_t v_t$, and the corresponding regret $R_T^{av} \coloneqq \sum_{t=1}^{T}(\mu(x^*) - U_t^{choice}$. The utility is then based on the choice - if $b_t = 1$, $U_t^{choice} = v_t$ and if $b_t = 0$, $U_t^{choice} = u_t$ (corresponding to y_t and x_t respectively). This form of utility captures the user's actual experience after choosing a result.

  However, regret bounds w.r.t U_t^{av} **imply similar regret bounds w.r.t.** U_t^{choice} **as explained below.**

Observation 2.1. *Assuming a link function where $u \geq v$ implies $\phi(u, v) > 1/2$, for any x_t, y_t, $\mathbb{E}[R_t^{choice}|(x_t, y_t)] \leq \mathbb{E}[R_t^{av}|(x_t, y_t)]$.*

This observation therefore tells us that any regret bounds derived on R_t^{av} will hold for R_t^{choice}. In words, this observation can be explained as: *given two choices, the one with larger utility is more likely to be chosen. Hence, the value of the choice-based utility U_t^{choice} is higher than the average utility U_t^{av} in expectation. In turn, this means that the choice-based regret R_t^{choice} is lower than the average-based regret R_t^{av} in expectation.*

Therefore, we can work with U_t^{av} and R_t^{av}.

3 Singleton Bandit Machines (SBMs)

- Defined as a unit with an internal timer and memory that will run an instance of the black-box MAB algorithm.
- An SBM S has three operations: *reset, advance, feedback*
- *reset* - Clears state of SBM.
- *advance* - returns the next arm/bandit to play
- *feedback* - simulate utility, use choice b_t and not u_t or v_t in this case ($\because$dueling bandits) to update $\hat{\mu}_x$ estimates within black-box MAB algorithm
- In general, an SBM S will thereby undergo the following steps, say, in a conventional MAB game:
  - First, we invoke $reset(S)$ to initialise the algorithm.
  - We invoke $advance(S)$ and set the arm that it returns as the arm x that is returned by the current iteration of the algorithm.
  - We then play arm x against nature and observe a utility u, and the $feedback(S, u)$ from this observed utility will be used to obtain feedback and update the state of the SBM S for the next iteration.

  After the initial *reset*, the *advance* and $feedback$ operations therefore tend to be applied in an alternating fashion.

4 Classic Stochastic MAB: Conventional Setting

4.1 UCB

Algorithm 1 UCB algorithm for MAB with $|X| = K$ arms. Parameter α affects tail of regret per action in X.

$\forall x \in X$, set $\hat{\mu}_x = \infty$
$\forall x \in X$, set $t_x = 0$
set $t = 1$
while true **do**
  let x be the index maximizing $\hat{\mu}_x + \sqrt{\frac{(\alpha+2)\ln(t)}{2t_x}}$
  play x and update $\hat{\mu}_x$ as the average of rewards so far on action x; increment t_x by 1.
  $t \leftarrow t + 1$
end while

Figure 8: UCB Algorithm

- UCB is an algorithm for the classical MAB problem in the finite X setting.
- It has a regret bound of $O(\alpha H \ln(T))$ where $H = \sum_{x \in X \setminus \{x*\}} 1/\Delta_x$ (this is the gap-dependent UCB regret bound as opposed to the gap-independent $O((T \log T)^{0.5})$ bound).

Lemma 2.2. *Assume X is finite. Fix a parameter $\alpha > 0$. Let $H := \sum_{x \in X \setminus \{x*\}} 1/\Delta_x$. When running the UCB policy with parameter α for T rounds, the expected regret is bounded by*

$$2(\alpha + 2)H \ln T + K\frac{\alpha + 2}{\alpha} = O(\alpha H \ln T)$$

Furthermore, let $x \in X$ denote some suboptimal arm and let $s \geq 4\alpha \ln(T)/\Delta_x^2$. Denote by $\rho_x(T)$ the random variable counting the number of times arm x was chosen up to time T. Then $Pr[\rho_x(T) \geq s] \leq \frac{2}{\alpha} \cdot (s/2)^{-\alpha}$.

This last condition is also known as $\alpha-$robustness, and it is a condition that UCB satisfies that allows it to achieve the regret bound in the **MultiSBM** algorithm later on.

4.2 ConfidenceBall

ConfidenceBall1 and ConfidenceBall2 are algorithms given by Dani et al. (2008) for the infinite arm setting, where the set of arms X is a convex set in $\mathbb{R}^d$. The player observes a stochastic reward $\langle \mu, x \rangle$ for some vector $\mu \in \mathbb{R}^d$.

These algorithms have roughly $\sqrt{T}$ regret for general convex sets, and *polylog*(T) for polytopes where the Δ-gap case is assumed for the vertices of the polytope. **Since our implementation is done on a finite set of arms X, and UCB satisfies all the assumptions needed for the SBM algorithms, we stick to using UCB in the SBMs.**

5 The Algorithms

5.1 Doubler

Algorithm 2 (Doubler): Reduction for finite and infinite X with internal structure.

```
S ← new SBM over X
ℒ ← an arbitrary singleton in X
i ← 1, t ← 1
while true do
    reset(S)
    for j = 1...2^i do
        choose x_t uniformly from ℒ
        y_t ← advance(S)
        play (x_t, y_t), observe choice b_t
        feedback(S, b_t)
        t ← t + 1
    end for
    ℒ ← the multi-set of arms played as y_t in the last for-loop
    i ← i + 1
end while
```

Figure 9: Doubler

- This algorithm is presented for the case where X is large or structured.
- A two-player game, one controlling the left arm and the other, the right arm (in the dueling bandits setup).
- To ensure a low regret match, one of the players is not adapting their strategy to every round. Instead, this player maintains a fixed stochastic strategy for a while, that is updated infrequently.
- The time horizon is split into exponentially growing epochs.
- One SBM S is defined to follow the strategy of the MAD algorithm (UCB) and is reset at the beginning of each epoch.
- The right player with actions y_t is the one that is chosen to adapt strategy every time step (by playing according to the SBM).
- The left one changes strategy at the start of every epoch to that of the right player in the previous epoch, and maintains this strategy through the current epoch by producing actions x_t from the previous epoch.
- $feedback(S, b_t)$ is called by playing (x_t, y_t) and observing b_t. This is then used by the right player to produce the next y_t by invoking $advance(S)$. This process repeats until the epoch is done.
- These steps are repeated until a termination condition such as convergence is satisfied.

Theorem 3.1. Consider a UBDB game over a set X. Assume the SBM S in Line 1 of **Doubler** has an expected regret of $c \log^{\alpha} T$ after T steps, for all T. Then the expected regret of **Doubler** is at most $2c\frac{\alpha}{\alpha+1} \log^{\alpha+1} T$. If the expected regret of the SBM is bounded by some function $f(T) = \Omega(T^{\alpha})$(with $\alpha > 0$), then the expected regret of **Doubler** is at most $O(f(T))$.

- We use UCB as the SBM algorithm, which has a regret bound of $O(\alpha_{UCB} H \log T)$. Therefore, $\alpha = 1$ (in the notation of Theorem 3.1).

Corollary 3.3. ... Assume SBM S in Line 1 of **Doubler** is UCB. Then the expected regret of **Doubler** reduces to at most $O(H \log^2(T))$.

- Extra $O(\log T)$ factor over the UCB SBM regret has appeared.

Note: The distinction between "set" and "multi-set" is essential to note in the **Doubler** algorithm as using the "set" of arms from the last for-loop gives rise to a linear regret. However, sampling x_t from the "multi-set" instead that allows for repeated instances of arms, gives us the regret required as the good actions are repeated with higher frequency in each epoch.

5.2 MultiSBM

Algorithm 3 (MultiSBM): Reduction for unstructured finite X by using K SBMs in parallel.

```
For all x ∈ X: S_x ← new SBM over X, reset(S_x)
y_0 ← arbitrary element of X
t ← 1
while true do
    x_t ← y_{t−1}
    y_t ← advance(S_{x_t})
    play (x_t, y_t), observe choice b_t
    feedback(S_{x_t}, b_t)
    t ← t + 1
end while
```

Figure 10: MultiSBM

- This algorithm is presented for the case where X is finite and unstructured.
- $|X| = K$ SBMs are defined, one for each arm, and they are all reset before the algorithm begins.
- y_0 is initialised as an arbitrary action.
- At each time step t, x_t is set to be y_{t-1}, and $advance(S_{x_t})$ is invoked. (The SBM corresponding to x_t.)
- y_t is set to be the arm that is returned by $advance(S_{x_t})$.
- $feedback(S_{x_t}, b_t)$ is called by playing (x_t, y_t) and observing b_t. After this SBM is updated using the feedback, the new x_t is set to be the previous y_t, and the new y_t by invoking $advance(.)$ on the new S_{x_t}.
- These steps are repeated until a termination condition such as convergence is satisfied.

Naively, we feel regret is K times that of 1 SBM. However, total regret is shown to be dominated by that of SBM with maximum utility, if the SBM algorithm has the property of robustness.

Definition 4.1. Let T_x be the number of times a (suboptimal) arm $x \in X$ is played when running the policy T rounds. An MAB policy is said to be $\alpha - robust$ when it has the following property: for all $s \geq \alpha\Delta_x^{-2}\ln(T)$, it holds that $Pr[T_x > s] < \frac{2}{\alpha}(s/2)^{\alpha}$

As mentioned in the UCB section, the UCB algorithm satisfies this property.

Theorem 4.2. The total expected regret of MultiSBM is $O(H\alpha \ln T) + H\alpha(K \ln K + K \ln \ln T - \sum_{x \neq x^*} \ln \Delta_x)$ assuming the policy of the SBMs defined in Line 1 is α-robust for $\alpha = max(3, \ln(K)/\ln\ln(T))$. The robustness can be ensured by choosing UCB for the SBM with parameter α.

- The idea behind proof is the demonstration of the existence of a positive feedback loop that drives the step-wise regret to small numbers rapidly.
- If the right arm's regret is low at t, there is a higher chance that x^* is played at $t+1$.
- Conversely, if any fixed arm is played as the left arm often (especially x^*), the right arm regret decreases rapidly.

5.3 Sparring: A Heuristic Approach

Algorithm 4 (Sparring): Reduction to two SBMs.

1: $S_L, S_R \leftarrow$ two new SBMs over X
2: reset(S_L), reset(S_R), $t \leftarrow 1$
3: **while** true **do**
4: $\quad x_t \leftarrow$ advance(S_L); $y_t \leftarrow$ advance(S_R)
5: $\quad$ play (x_t, y_t), observe choice $b_t \in \{0, 1\}$
6: $\quad$ feedback$(S_L, \mathbf{1}_{b_t=0})$; feedback$(S_R, \mathbf{1}_{b_t=1})$
7: $\quad t \leftarrow t + 1$
8: **end while**

Figure 11: Sparring

- The **Sparring** algorithm shows very good performance. However, the authors have been able to prove performance bounds.
- **Sparring** begins with two SBMs S_L (left) and S_R (right). The first step is to invoke $reset(S_L)$ and $reset(S_R)$.
- Then, both the SBMs are *advance*d, and the arms are played as x_t, y_t. After the u_t, v_t are returned, they are used to calculate the average utility and therefore the regret in the time step.
- However only b_t is visible. **Now, depending on whether $b_t = 0$ or 1, the corresponding SBM S_L or S_R is updated with a feedback of 1, i.e., the $feedback$ operation uses 1 to update the "empirical mean reward" value $(\hat{\mu}_x)$ of that arm in the execution of the UCB algorithm within that SBM. The other SBM is updated with a feedback of 0.**

- This repeats until a termination condition is met, such as convergence for example.

The regret of **Sparring** is conjectured to be asymptotically bounded by a constant factor times the combined regret of the algorithms hidden in the individual SBMs, plus (possibly) a small overhead. The intuition behind this conjecture comes from an adversarial version of UBDB (the exact analysis has been omitted by the authors). They declare that the proof appears to be tricky because no SBM sees a stochastic environment, as its feedback depends on non-stochastic choices made by the other SBM.

6 Experimental Results

We used the following link functions and expected-value distributions which were also used in the paper for experimentation. A, B, C, D, E, F are the arms of the multi-armed bandit, and $\mu(.)$ represents the expected reward of the arm.

linear	$\phi(x, y) = (1 + x - y)/2$
natural	$\phi(x, y) = x/(x + y)$
logit	$\phi(x, y) = (1 + \exp\{y - x\})^{-1}$

Name	$\mu(A)$	$\mu(B)$	$\mu(C)$	$\mu(D)$	$\mu(E)$	$\mu(F)$
1good	0.8	0.2	0.2	0.2	0.2	0.2
2good	0.8	0.7	0.2	0.2	0.2	0.2
3good	0.8	0.7	0.7	0.2	0.2	0.2
arith	0.8	0.7	0.575	0.45	0.325	0.2
geom	0.8	0.7	0.512	0.374	0.274	0.2

Figure 12: Link Functions and Expected-Value Distributions

For 7 combinations **linear/1good, natural/1good, logit/1good, linear/2good, linear/3good, linear/arith, linear/geom** of arm values and link functions, we ran 4 algorithms- **Doubler, MultiSBM, Sparring, and Interleaved Filter 2 (IF2)**. The last IF2 algorithm is used as in Paper 1, as a baseline for state-of-the-art dueling-bandit-specific algorithms (comparison of algorithms across papers).

We used 32000 time steps for each run of any algorithm, and averaged over 50 runs to obtain the mean plots. The errorbars are ± 1 standard deviation errorbars across the 50 runs. Below are the regret bounds for the three algorithms excluding IF2, on linear/1good.

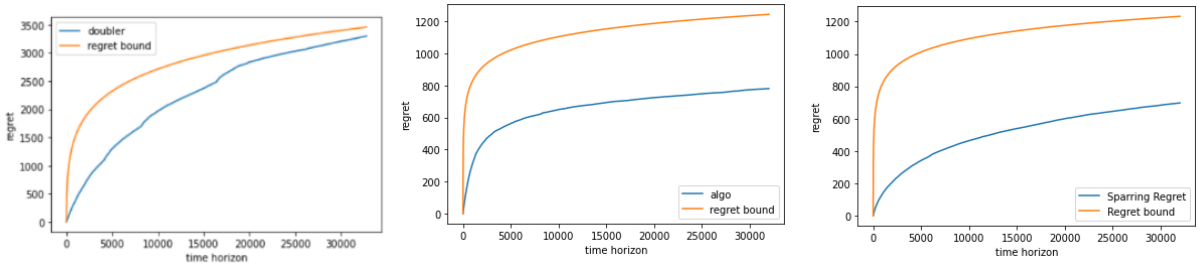

Figure 13: Demonstrating regret bounds using linear/1good on each of Doubler($O(\log^2 T)$), MultiSBM ($O(\log T)$), and Sparring ($O(\log T)$) respectively

Next, we plot regret plots for all the 7 combinations mentioned above with a `semilogx` scale going from 2^{10} to 2^{15}, similar to the plots in the paper.

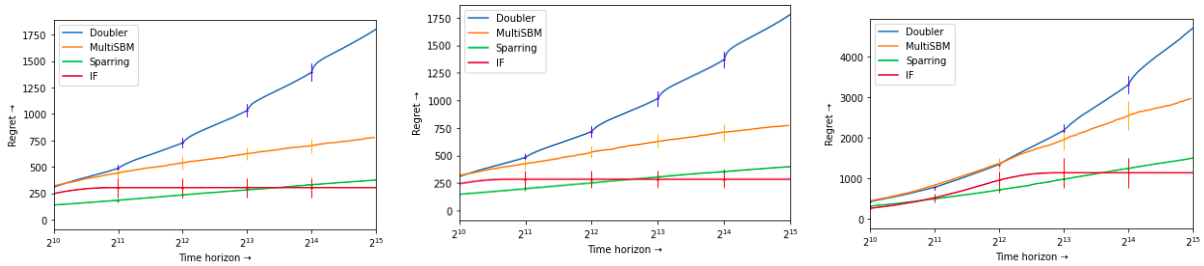

Figure 14: linear/1good, linear/2good, linear/3good

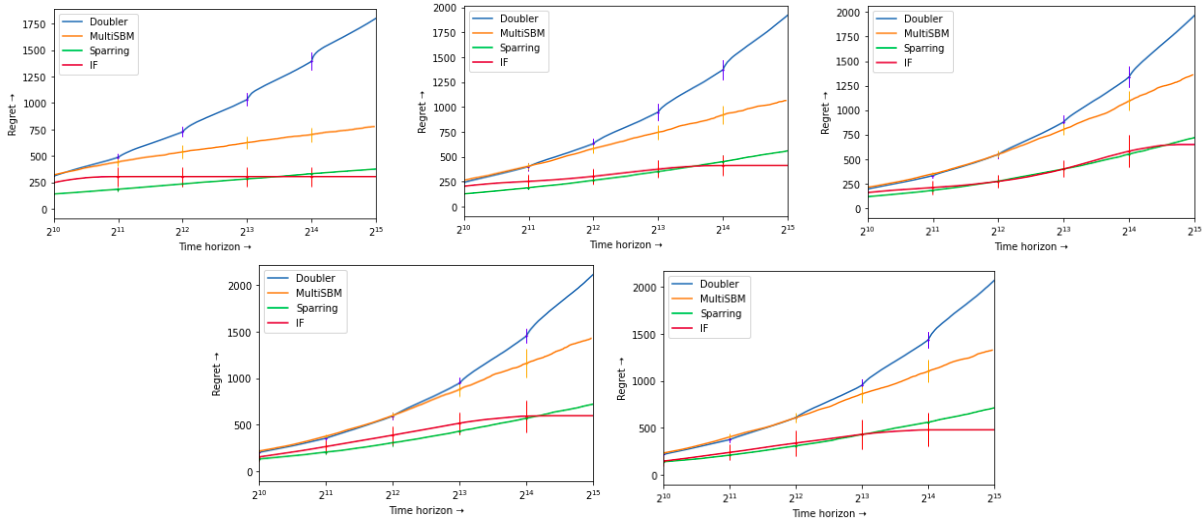

Figure 15: linear/1good, linear/2good, linear/3good, linear/arith, linear/geom

Observations

- It has been shown, ignoring constants, that the regret bound for **Doubler** is $O(\log^2(T))$ and for **MultiSBM** is $O(log(T))$. The conjectured bound for **Sparring** is also $O(\log(T))$. Therefore, in the above `semilogx` plots, the curves should look like $e^{\log^2(T)} = T^{\log T}$, $e^{\log(T)} = T$ (linear) and $e^{\log(T)} = T$ (linear) respectively. We see the regret plots looking to be of this form in all cases.
- The **Doubler** algorithm's strategy of splitting the time horizon into epochs between timesteps 2^i and 2^{i+1} for $i \in Z$ and re-initializing the SBM before every epoch clearly shows in the regret plots with the slopes on either side of the $2^i th$ time-step being markedly different from each other.
- The **IF2** algorithm incurs regret as it explores and finds the best arm. The exploitation that is done after the **IF2** exploration therefore incurs no regret. As a result, the curve becomes flat after the actual **IF2** exploration phase, and the **Sparring** regret curve goes above it at a later time step.
- However, while comparing regret during the exploration phase of **IF2**, we can see that **Sparring** is the best algorithm is almost all cases. **IF2** tends to incur lower regret than **MultiSBM** and **Doubler**, though.

7 Contributions

- **Vishwajit:** Interleaved Filter 1 and 2 implementation and slides

- **Akash and Dhruv:** first implemented Doubler together, then:
  - **Akash:** Sparring implementation, SBM/UCB implementation using OOPS class (gives easy black-box transferability to other reduction-based UBDB methods), Paper 2 slides on Introduction, UBDB Formalism, Sparring, Results
  - **Dhruv:** MultiSBM implementation, independent non-OOPS UCB implementation, Paper 2 slides on UCB, Doubler, MultiSBM