Efficient Algorithms for Lipschitz Bandits

Anonymous Author(s)
Affiliation
Address
email

Abstract

Lipschitz bandits is a fundamental framework used to model sequential decision-making problems with large, structured action spaces. This framework has been applied in various areas. Previous algorithms, such as the Zooming algorithm, achieve near-optimal regret with $O(T^2)$ time complexity and $O(T)$ arms stored in memory, where T denotes the size of the time horizons. However, in practical scenarios, learners may face limitations regarding the storage of a large number of arms in memory. In this paper, we explore the bounded memory stochastic Lipschitz bandits problem, where the algorithm is limited to storing only a limited number of arms at any given time horizon. We propose algorithms that achieve near-optimal regret with $O(T)$ time complexity and $O(1)$ arms stored, both of which are almost optimal and state-of-the-art. Moreover, our numerical results demonstrate the efficiency of these algorithms.

1 Introduction

Multi-armed Bandits (MAB) is a powerful framework used to balance the exploration-exploitation trade-off in online decision-making problems. Within this framework, a learner sequentially selects arms (actions, decisions, or items) and learns from the associated feedback, aiming to maximize the expected total reward within finite time horizons. Some well-known algorithms, such as UCB1 and Exp3, have achieved near-optimal regret by storing records of all arms in memory. In many bandit problems, algorithms can access information about the similarity between arms, suggesting that arms with similar characteristics often yield similar expected rewards. The Lipschitz bandits framework is a prominent variant that addresses decision-making in large, structured action spaces, where the expected reward of the arms follows a Lipschitz function. For instance, in recommendation systems, the arms correspond to items represented by feature vectors. Items with similar feature vectors are likely to result in similar outcomes or conversions.

Recently, a series of works in the field of online learning have been dedicated to managing scenarios with large action spaces while maintaining sub-linear memory usage. This direction is driven by the need to effectively tackle extensive real-world applications such as recommendation systems, search ranking, and crowdsourcing. In these applications, arms correspond to items, solutions, or models, which leads to significant memory demands. For instance, in recommendation systems, the learner faces the challenge of choosing from millions of items, like music and movies, to present to users, especially in scenarios characterized by limited space or an infinite number of arms. Therefore, the development of memory-efficient algorithms has become crucial for these applications. In recent years, substantial efforts have been made to address the challenge of bandits with limited memory (Assadi & Wang, 2020; Jin et al., 2021; Maiti et al., 2020; Agarwal et al., 2022; Assadi & Wang, 2022; Wang, 2023; Assadi & Wang, 2023a). However, previous research has mainly focused on unstructured action spaces, often overlooking the fact that in these applications, arms with similar characteristics tend to yield similar expected rewards.

Submitted to 38th Conference on Neural Information Processing Systems (NeurIPS 2024). Do not distribute.

Table 1: Comparison with State-of-the-art Lipschitz Bandits Algorithms

Algorithm	Regret	Time complexity	Space complexity
Zooming(Kleinberg et al., 2019)	$\widetilde{O}\left(T^{\frac{d_z+1}{d_z+2}}\right)$	$O\left(T^2\right)$	$O(T)$
HOO(Bubeck et al., 2011a)	$\widetilde{O}\left(T^{\frac{d_z+1}{d_z+2}}\right)$	$O\left(T\log T\right)$	$O(T)$
MBAD(Ours)	$\widetilde{O}\left(T^{\frac{d_z+1}{d_z+2}}\right)$	$O\left(T\right)$	$O\left(1\right)$

One general approach to solving Lipschitz bandits is through discretizing the structured action space. Algorithms based on uniform discretization have been shown to achieve optimal worst-case regret up to a logarithmic factor (Kleinberg, 2004). Another strategy, adaptive discretization, progressively 'zooms in' on more promising regions of the action space, yielding near-optimal problem-dependent regret (Kleinberg et al., 2019). However, existing algorithms like the Zooming algorithm necessitate $O(T)$ stored arms in memory and $O(T^2)$ time complexity for stochastic Lipschitz bandits (Kleinberg et al., 2019; Feng et al., 2022), which may be impractical for many real-world applications. In this paper, we consider a typical scenario where the learner operates within the stochastic bandits framework over a Lipschitz action space while facing constraints on the number of arms that can be stored in memory.

The limited memory constraint and large structured action space present several challenges, necessitating a nuanced approach to effectively balance exploration and exploitation under uncertainty. One key challenge is the propensity to over-exploit suboptimal arms retained in memory, leading to high regret. Conversely, reading new arms into memory risks discarding potentially valuable arms. In scenarios with infinite actions, the vast search space requires numerous samples to ensure adequate exploration. The structured nature of the action space demands that algorithms focus on zooming in on more promising regions, but space constraints limit the learner's capacity to acquire comprehensive knowledge about the metric space. Traditional full-memory algorithms start by dividing the action space into many small subcubes, a process known as discretization. Each cube is treated as an arm, and in each round, the algorithm updates the average estimate of the selected cube's reward based on feedback. It then compares this estimate against all other cubes in the storage space through various computational methods.

1.1 Our Contributions

Our primary insight revolves around two key aspects: metric embedding and pairwise comparisons. Metric embedding involves mapping elements from one metric space to another while preserving distance relationships as closely as possible. Our algorithm effectively maps the metric space to a tree, where each node represents a cube. Traversing this tree is analogous to navigating the entire metric space. Pairwise comparisons of arms reduce memory complexity. Instead of constantly covering the entire space, our approach considers all subcubes as a stream. From this stream, we continuously select cubes for pairwise comparisons, gradually converging to the optimal region.

Based on this insight, we introduce two algorithms: the Memory Bounded Uniform Discretization (MBUD) algorithm and the Memory Bounded Adaptive Discretization (MBAD) algorithm. The MBUD algorithm employs a uniform discretization strategy combined with an Explore-First approach. In this method, all cubes are of the same size. The algorithm prioritizes selecting a near-optimal arm following an exploration phase and allocates the remaining rounds to exploitation, achieving near-optimal worst-case regret. The exploration phase consists of "cross exploration phases" and the "summarize phase". During the cross exploration phases, exploration is confined to a subset of cubes to gather information about the optimal arm while minimizing regret. The summarize phase explores all cubes to pinpoint the optimal arm's location.

The MBAD algorithm utilizes an adaptive discretization strategy, incorporating a round-robin playing approach. This allows for subcubes within subcubes, organizing the entire action space into a tree structure. The algorithm selectively focuses on more promising regions of the action space, thereby attaining near-optimal instance-dependent regret. Each node in this structure represents a subcube, with parent and child nodes corresponding to subcubes and their subdivisions, respectively. Traversal involves transitioning from a node to its child and navigating through a parent node's children to the next subcube. Pruning prevents over-zooming through two conditions: discarding inferior cubes with high confidence and establishing a lower bound on cube edge length, which decreases as exploration progresses. These conditions ensure efficient exploration without over-zooming.

Overall, our contribution lies in pioneering memory-efficient algorithms for large structured action spaces, particularly within Lipschitz metric spaces. We introduce the MBUD and MBAD algorithms, which achieve near-optimal regret while requiring storage for only the best-estimate arm for exploitation and one additional arm for exploration. This means only two arms need to be stored in memory, regardless of the problem's scale. Furthermore, each algorithm exhibits $O(T)$ time complexity, indicating that their execution time scales linearly with the number of rounds.

1.2 Related Work

Lipschitz bandits. Multi-armed bandits is one of the most classical frameworks to model the trade-off between exploration and exploitation in online decision problems. The Lipschitz bandits framework considers the large, structured action space in which the algorithm has information on similarities between arms. The model was first introduced by Agrawal (1995) with interval $[0, 1]$. The near-optimal upper and lower bounds for the worst case were provided in Kleinberg (2004) via the uniform discretization strategy. Subsequent work (Kleinberg et al., 2019) proposed the zooming algorithm, achieving near-optimal instance-dependent regret for the problem and studying the extension for the general metric action space. Several other works have established regret bounds for the stochastic reward feedback setting (Bubeck et al., 2011a; Magureanu et al., 2014; Lazaric et al., 2014). Other works have also extended the results to the adversarial version (Podimata & Slivkins, 2021; Kang et al., 2023), contextual setting (Slivkins, 2014; Krishnamurthy et al., 2019; Lee et al., 2022), ranked setting (Slivkins et al., 2013), contract design (Ho et al., 2014), federated X-armed bandit (Li et al., 2024a,b), and other settings (Bubeck et al., 2011b; Lu et al., 2019; Wang et al., 2020; Grant & Leslie, 2020; Feng et al., 2022; Xue et al., 2024).

Memory-efficient learning. Another line relevant to this paper is online learning with memory constraints. Liau et al. (2018) considered stochastic bandits with constant arm memory and proposed an algorithm achieving an $O(\log 1/\Delta)$ factor of optimal instance-dependent regret, where Δ is the gap between the best arm and the second-best arm. Chaudhuri & Kalyanakrishnan (2020) studied stochastic bandits with M stored arms and showed there is an algorithm with regret $\tilde{O}(KM + (K^{3/2}\sqrt{T})/M)$. Subsequent work (Agarwal et al., 2022) provided an algorithm achieving regret $O(\sqrt{KT \log T \log\log T})$. In addition to the bandits problem, there are also many works about other online learning problems. Srinivas et al. (2022); Peng & Zhang (2022) showed the trade-off between regret and memory for the expert problem. More pure exploration models with memory constraints were considered in Assadi & Wang (2020), including the coin tossing problem, noisy comparisons problem, and Top-K arms identification. Previous works on bandits with limited memory have not considered structured action spaces and could not deal with infinite actions. There are some other works on memory-efficient online learning (Peng & Rubinstein, 2023; Assadi & Wang, 2023b). Beyond the online learning setting, the memory-efficient learning problem was solved in different situations, including statistical learning (Steinhardt et al., 2016; Garg et al., 2017; Raz, 2017; Garg et al., 2019; Sharan et al., 2019; Lyu et al., 2023), convex optimization (Marsden et al., 2022; Blanchard et al., 2023a,b; Chen & Peng, 2023), estimation problems (Acharya et al., 2019; Diakonikolas et al., 2022; Berg et al., 2022), parity learning (Raz, 2019; Kol et al., 2017), and other learning problems (Hopkins et al., 2021; Brown et al., 2022; Chen et al., 2022).

2 Problem Setup and Preliminaries

Notations. In this paper, we use bold fonts to represent vectors and matrices. For a positive integer T, we use $[T]$ to denote the set $\{1, 2, \ldots, T\}$. For a set $\mathcal{X}$, we use $|\mathcal{X}|$ to denote its cardinality. For a random variable Z, we use $\mathbb{E}[Z]$ to denote its expectation. For an event $\mathcal{E}$, we use $\mathbb{P}[\mathcal{E}]$ to denote its probability.

2.1 Problem Setup

We formally define the Lipschitz bandits problem below. Given T rounds, dimension d, and arm space $\mathcal{X} = [0, 1]^d$, each arm $x \in \mathcal{X}$ is associated with an unknown reward distribution $\mathcal{D}_x$. In each round $t \in [T]$, the algorithm selects an arm $x_t \in \mathcal{X}$ and obtains a scalar-valued reward feedback $r_t \in [0, 1]$, which is a sample from the reward distribution $\mathcal{D}_{x_t}$. The expected reward $\mu(\cdot)$ of the reward distribution satisfy the Lipschitz condition: $|\mu(x) - \mu(y)| \leq L \cdot |x - y| \quad \forall x, y \in \mathcal{X}$. And we call L the Lipschitz constant. Then a problem instance is specified by the known number of time horizons T, known Lipschitz constant L, and unknown mean reward $\mu(\cdot)$. For the purposes of simplification in our proofs, we assume $L = 1$. The algorithm aims to maximize the expected total reward $\mathbb{E}[\sum_{t\in[T]} r_t]$.

We use regret to measure the performance of the algorithm compared with the expected total reward of the best-fixed arm in action space $\mathcal{X}$: $\mathbb{R}_{\mathcal{X}}(T) = T \cdot \sup_{x \in \mathcal{X}} \mu(x) - \mathbb{E}\left[\sum_{t \in [T]} r_t\right]$.

Then we present the memory model employed in the paper. The algorithm operates by selecting arms from the memory and pulling them. When the memory reaches the capacity and the algorithm attempts to choose a new arm, it becomes necessary to discard at least one arm from the memory. Consequently, any statistical information associated with the discarded arm, including its index, mean reward, and number of pulls, is forgotten and will not be retained thereafter. We measure the space complexity of the algorithm by the hard constraint for the number of arms stored in the memory. This constraint aligns with the assumption of having oracle access to the input arm, as commonly defined in streaming problems.

2.2 Covering Dimension and Zooming Dimension

Then we provide some technical tools that are used in this paper and introduce the covering dimension and zooming dimension for one action space $\mathcal{X}$. We use the definitions in (Slivkins, 2019) and provide them below. Notice that the Lipschitz bandits problem is defined in an infinite-action space. We select a fixed, finite discretization actions space $\mathcal{S} \subset \mathcal{X}$. Let $\{\mathcal{X}_1, \ldots, \mathcal{X}_N\}[\mathcal{X}_i \subset \mathcal{X}]$ be an cover of the action space $\mathcal{X}$. Let ϵ denote the maximum diameter of $\mathcal{X}_i$ for all $i \in [N]$. Then the arm set $\mathcal{S} = \{x_i | x_i \in \mathcal{X}_i, i \in [N]\}$ is an ϵ-mesh. The covering dimension d of the action space $\mathcal{X}$ is defined as $d = \inf_{\alpha \geq 0} \{|\mathcal{S}| \leq \epsilon^{-\alpha}, \forall \epsilon > 0\}$. Let $\mu^*_{\mathcal{X}} := \sup_{x \in \mathcal{X}} \mu(x)$ denote the expected per-round reward of the optimal arm in space $\mathcal{X}$ and $\Delta(x) := \mu^*_{\mathcal{X}} - \mu(x)$ denote the gap between arm x and the optimal arm. Define $\mathcal{Y}_j = \{x \in \mathcal{X} : 2^{-j} \leq \Delta(x) < 2^{1-j}, j \in \mathbb{N}\}$, then set $\mathcal{Y}_j$ contains all arms whose gap is between 2^{-j} and 2^{1-j}. Consider the ϵ-mesh $\mathcal{S}_j$ for space $\mathcal{Y}_j$. Then the zooming dimension d_z for the action space $\mathcal{X}$ is $d_z = \inf_{\beta \geq 0} \left\{|\mathcal{S}_j| \leq \epsilon^{\beta}, \epsilon = 2^{-j}, \forall j \in \mathbb{N}\right\}$.

Covering dimension is a property of the action space while the zooming dimension is a property of the instance. Notice that we always have $d_z \leq d$. This is because the covering dimension considers the ϵ-mesh of the entire action space $\mathcal{X}$, whereas the zooming dimension focuses only on the set $\mathcal{Y}_j$. The covering dimension is closely related to other notions of dimensionality in a metric space, such as the Hausdorff dimension, capacity dimension, and box-counting dimension, all of which characterize the covering properties in fractal geometry. Similarly, the zooming dimension is another measure used to evaluate the structure of a metric space. Both of these dimensions are widely utilized in the field of Lipschitz bandits. For further details and alternative formulations regarding the covering dimension and zooming dimension, refer to (Kleinberg et al., 2019).

3 Warm Up: Uniform Discretization Algorithm

This section provides the intuition, specification, and theoretical analysis of the Memory Bounded Uniform Discretization (MBUD) algorithm (shown in Algorithm 1) for the stochastic Lipschitz bandits problem.

Algorithm overview. To facilitate our discussion, we begin by outlining the core idea behind the algorithm. This algorithm employs a uniform discretization strategy and adopts an Explore-First methodology, which endeavors to identify a near-optimal arm following the exploration phase and dedicates the remaining rounds to exploitation. Throughout the exploration stage, the algorithm allocates two units of memory space: one for storing the best-estimated arm and another for temporarily holding a newly read arm. Note that the best-estimated arm serves a dual purpose: it is not only crucial for the exploitation phase but also enables the swift identification of sub-optimal arms.

The exploration phase in Algorithm 1 is divided into $\lceil \log \log T \rceil$ phases, further structured into two main segments: the 'cross exploration phases' and the 'summarize phase'. During the initial $\lceil \log \log T \rceil - 1$ phases, the algorithm iterates over the arms within the discretized action space to minimize regret. Exploration is limited to a subset of cubes at a time, allowing the algorithm to gather information about the optimal arm while minimizing regret. In the final phase, termed the 'summarize phase', the algorithm revisits all arms within the uniform discretization space. Overall, each arm is read into memory twice to ensure thorough evaluation. Furthermore, we implement a budgeting strategy for each phase, wherein the total number of pulls across all arms is constrained by a predefined budget. The goal is to select the optimal arm with high probability after accumulating sufficient information during the previous phases. This structured approach balances exploration and

Algorithm 1 Memory Bounded Uniform Discretization (MBUD)

Input: arm space $\mathcal{X} = [0,1]^d$, time horizon T, parameter c.

$\boldsymbol{y} \leftarrow \mathbf{0}$, $\bar{r}_y \leftarrow 0$, $n_y \leftarrow 0$, $B_{-1} \leftarrow 1$, $\epsilon = \left(\frac{\log T}{T}\right)^{1/(d+2)}$, $\phi \leftarrow \lceil \log\log T \rceil - 1$.
for $p = 0, \cdots, \phi - 1$ **do**
  $B_p \leftarrow \sqrt{T B_{p-1}}$.
  for $q = 1, \cdots, \lfloor \phi \epsilon^{-d} \rfloor$ **do**
    Generate a new cube $C \leftarrow$ CROSSCUBE(ϕ, ϵ, p, q), and select a arm $\boldsymbol{x}$ from C.
    $(\boldsymbol{y}, \bar{r}_y, n_y) \leftarrow$ COMPARE$(c, \boldsymbol{x}, \boldsymbol{y}, \bar{r}_y, n_y, \epsilon B_p)$.
  end for
end for
for $q = 1, \cdots, \lfloor \epsilon^{-d} \rfloor$ **do**
  Generate a new cube $C \leftarrow$ GENERATECUBE(ϵ, q), and select a arm $\boldsymbol{x}$ from C.
  $(\boldsymbol{y}, \bar{r}_y, n_y) \leftarrow$ COMPARE$(c, \boldsymbol{x}, \boldsymbol{y}, \bar{r}_y, n_y, \epsilon B_{\phi-1})$.
end for
Play arm y until the end of the game.

Algorithm 2 CROSSCUBE

Input: number of phases ϕ, edge-length ϵ, parameters q.

$\kappa_1 \leftarrow \max_{k \in \mathbb{N}}\{k^d \le \phi\}$, $\kappa_2 \leftarrow \max_{k \in \mathbb{N}}\{k^d \le \lfloor \phi \epsilon^{-d} \rfloor\}$.
Let `node` $\leftarrow \epsilon \mathcal{G}_d(p, \kappa_1) + \frac{\epsilon\phi}{\sqrt{d}} \mathcal{G}_d(q, \kappa_2)$, then the cube could be determined by `node` and ϵ.

exploitation under memory constraints, aiming to quickly identify the optimal arm while minimizing the sampling of suboptimal arms. The specifics of this approach will be detailed subsequently.

Exploration strategies. For the cross exploration phases, the gap between neighboring arms is $\epsilon\phi$ (ϕ defined in Algorithm 1). There are $O(\epsilon^{-d})$ cubes (arms) in the discretization action set, which is an ϵ-mesh of $\mathcal{X}$. Each cross exploration phase will only explore $O\left(\frac{1}{\log\log T}\right)$ of them. We generate a new cube by using the function $\mathcal{G}_d(a, b), a, b \in \mathbb{N}$ which converts the integer a to a d-dimension vector. And the i-th entry of the vector is the i-th right-most digit in base b. To aid understanding, we offer several examples: $\mathcal{G}_3(3,2) = (0,1,1)$, $\mathcal{G}_3(1208, 26) = (1, 20, 12)$, and $\mathcal{G}_2(1208, 26) = (20, 12)$. The function could be done by a succession of Euclidean divisions by b. For the summarize phase, the gap is ϵ and all cubes in the discretization set are explored.

The CROSSCUBE function generates cubes for the cross exploration phases by calculating parameters based on the number of phases and the edge-length of the cubes. Specifically, CROSSCUBE generates a new cube using a combination of two geometric sequences. It first calculates the parameters κ_1 and κ_2 as the maximum integers such that $k^d \le \phi$ and $k^d \le \lfloor \phi \epsilon^{-d} \rfloor$, respectively. The function then determines the cube's position using these parameters and the edge-length ϵ. The cube is defined by a node position generated by $\epsilon \mathcal{G}_d(p, \kappa_1)$ and $\frac{\epsilon\phi}{\sqrt{d}} \mathcal{G}_d(q, \kappa_2)$, where $\mathcal{G}_d$ is a geometric sequence generator that converts an integer to a d-dimensional vector. The GENERATECUBE function is similar to CROSSCUBE but is used during the summarize phase to generate cubes without considering the phases. It calculates the parameter κ as the maximum integer such that $k^d \le \lfloor \epsilon^{-d} \rfloor$. The cube is then determined by the edge-length ϵ and a node position generated by $\epsilon \mathcal{G}_d(q, \kappa)$.

Compare strategy. Then we introduce the compare strategy, which is also useful for the MBAD algorithm described in the following section. The algorithm always selects the arm with the fewest pulls in the memory. After sufficient samples, it will eliminate one sub-optimal arm based on its upper confidence bound and then generate a new arm (i.e., read a new arm into the memory). Notice that the algorithm may prioritize two sub-optimal arms with a small gap. Therefore, there is a cap on the number of pulls each phase for any arm. It helps the algorithm in striking a balance between exploration (read a new arm) and exploitation (play arms in memory).

The algorithm maintains three statistics for one arm in memory: the index x, the mean reward estimator $\bar{r}_x$, and the number of pulls n_x. The constant c is an exploration and exploitation balancing parameter. In the exploration part, there are $\lceil \log\log T \rceil$ phases. Let B_p be the budget of samples for the p-th phase. We use y and x to denote the best-estimated arm and the new arm in the algorithm,

Algorithm 3 GENERATECUBE

Input: edge-length ϵ, parameters q.

Let $\kappa \leftarrow \max_{k \in \mathbb{N}}\{k^d \leq \lfloor \epsilon^{-d} \rfloor\}$.

Let node $\leftarrow \epsilon \mathcal{G}_d(q, \kappa)$, then the cube could be determined by node and ϵ.

Algorithm 4 COMPARE

Input: constant c, arm x and y, $\bar{r}_y$, n_y, b.

$\bar{r}_x \leftarrow 0$, $n_x \leftarrow 0$.
while $n_x \leq b$ or $n_y \leq b$ **do**
  Pull the least played arm between x and y. If there is no single least played arm, select a random arm.
  Update $\bar{r}_x$, n_x, $\bar{r}_y$, n_y.
  if $\min\{\bar{r}_x + \sqrt{(c \log T)/n_x}, 1\} < \max\{\bar{r}_y - \sqrt{(c \log T)/n_y}, 0\}$ **then**
    Break and return $(y, \bar{r}_y, n_y)$.
  else if $\max\{\bar{r}_x - \sqrt{(c \log T)/n_x}, 0\} > \max\{\bar{r}_y - \sqrt{(c \log T)/n_y}, 0\}$ **then**
    Break and return $(x, \bar{r}_x, n_x)$.
  end if
end while
Return $(y, \bar{r}_y, n_y)$.

respectively. If the upper confidence bound (UCB) of arm x is less than the lower confidence bound (LCB) of arm y, then x is suboptimal with high probability. If the LCB of y is less than the LCB of arm x, then x is not too bad with high probability. For the remaining cases, we could choose either x or y, and we choose arm y at the end of the algorithm.

Flowchart. In Appendix A.1, we include a flowchart that illustrates the operation of the algorithm.

Theoretical result. The computational workload of the MBUD algorithm is characterized by a constant per-round operation, leading to a total time complexity of $O(T)$, where T represents the number of rounds. Regarding space complexity, the MBUD algorithm necessitates the storage of merely two arms in memory at any given time. Additionally, the space requirements for the GENERATECUBE and CROSSCUBE subroutines are minimal, each consuming $O(1)$ units of space in terms of arm storage. Consequently, the overall space complexity of the algorithm is $O(1)$.

We provide the theoretical result below and provide the details of the theoretical analysis in Appendix B. The result recovers the worst case regret in previous work and recovers the lower bound up to a logarithmic factor (Kleinberg, 2004).

Theorem 1. *For the stochastic Lipschitz bandits problem with metric $(\mathcal{X}, \mathcal{D})$ and time horizon T, where $\mathcal{X} = [0, 1]^d$ and $\mathcal{D}$ is a known metric function. Algorithm 1 uses $O(1)$ stored arms and achieves regret*

$$\mathbb{R}_{\mathcal{X}}(T) \leq \tilde{O}(T^{\frac{d+1}{d+2}}),$$

where d is the covering dimension of space $\mathcal{X}$.

The theoretical analysis is mainly based on the 'clean event', which holds that the observed mean average is a good estimator for the expectation with high probability. At a high level, the analysis shows that the deviation between the mean estimator of the best-estimated arm y and the optimal expected reward $\mu^*_{\mathcal{X}}$ is small enough when $p \geq 1$. Then the sub-optimal arms could be discarded quickly, which helps us to bound the incurred regret of sub-optimal arms and the exploitation phase. We bound the expected regret during all time horizons by considering the discretization error, the incurred regret of all sub-optimal arms during the exploration, and the sub-optimality of the selected arm before the exploitation together.

4 Adaptive Discretization Algorithm

This section provides the main idea, specification, and theoretical analysis of the Memory Bounded Adaptive Discretization (MBAD) algorithm (shown in Algorithm 5).

Algorithm 5 Memory Bounded Adaptive Discretization (MBAD)

Input: time horizon T, constant c.

1: $y \leftarrow 0, \bar{r}_y \leftarrow 0, n_y \leftarrow 0, B_1 \leftarrow \sqrt{T}$.
2: **for** $p = 1, 2, \ldots$ **do**
3: $\quad x \leftarrow 0, \bar{r}_x \leftarrow 0, n_x \leftarrow 0, b_p \leftarrow B_p \cdot \left(\frac{\log T}{T}\right)^{1/(d+2)}$.
4: $\quad$ AdaptiveCube(4, 1).
5: $\quad B_{p+1} \leftarrow B_p \log T$.
6: **end for**

Algorithm 6 AdaptiveCube

Input: parameters m, q, edge-length $\epsilon = 2^{-m}$.

1: $\kappa \leftarrow \max_{k \in \mathbb{N}}\{k^d \leq \lfloor \epsilon^{-d} \rfloor\}$.
2: `node` $\leftarrow \epsilon \mathcal{G}_d(q, \kappa)$, then the cube C could be determined by `node` and ϵ.
3: Select a arm $\boldsymbol{x}$ from C.
4: **if** $q + 1 \leq 2^m$ **and** the output of Compare$(c, \boldsymbol{x}, \boldsymbol{y}, \bar{r}_y, n_y, 20\epsilon^{-2})$ is arm $\boldsymbol{y}$ **then**
5: $\quad$ check the next cube with parameters m and $q + 1$.
6: **else if** $20\epsilon^{-2} \leq b_p$ **then**
7: $\quad (\boldsymbol{y}, \bar{r}_y, n_y) \leftarrow$ Compare$(c, \boldsymbol{x}, \boldsymbol{y}, \bar{r}_y, n_y, 20\epsilon^{-2})$.
8: $\quad$ Equally partition the cube C into 2^d subcubes and check the first subcube.
9: **end if**

Algorithm overview. We begin with some intuitions. The MBUD algorithm achieves near-optimal regret in the worst case but fails to leverage the beneficial structure of 'nice' problem instances. To address this, we present the MBAD algorithm, which is based on adaptive discretization, and establish a near-optimal instance-dependent upper bound. The idea behind adaptive discretization is straightforward: the algorithm should focus more on promising regions. For instance, the zooming algorithm approximates the expected rewards over the action space and explores more in regions with a high probability of yielding high rewards. However, due to memory constraints, the algorithm cannot obtain a comprehensive picture of the action space over time. To overcome this obstacle, the MBAD algorithm employs a "round robin" strategy, storing the best-estimated arm as the next read arm in memory. Unlike the MBUD method, which chooses predetermined steps, the MBAD algorithm selects the next read arm based on the confidence radius of the arms in memory. Consequently, steps are smaller and probes (newly picked arms) are more numerous in promising regions.

Exploration strategies. The AdaptiveCube subroutine is the cornerstone of the MBAD algorithm, functioning as a recursive mechanism to navigate and leverage a cubic region within the decision space. This procedure dynamically adjusts the exploration granularity based on observed rewards and predetermined sampling constraints. Initially, the algorithm selects a cube C for exploration. If this cube is deemed sub-optimal compared to the optimal estimated arm stored in memory (denoted as arm $\boldsymbol{y}$), the algorithm discards this cube in favor of exploring a subsequent cube, following the generation rules outlined in the GenerateCube subroutine described in the MBUD algorithm (Section 3). Conversely, if the cube shows promise, the algorithm proceeds to explore within it, subdividing it into smaller subcubes for more detailed exploration. Each exploration phase is governed by a specific sample budget, which regulates the granularity of exploration to prevent excessive sampling of sub-optimal arms in the early stages. This adaptive exploration process continues until the entire action space has been thoroughly explored. The decision-making process is inherently dynamic, constantly evolving based on past actions to enhance the efficiency of future exploration and exploitation efforts.

To prevent the MBAD algorithm from "over-zooming", we implement two stop conditions. The first condition discards the current cube in favor of a new one once we are highly confident that the current cube is inferior to the best cube we've explored (see lines 4-5 of Algorithm 6). The second condition sets a lower bound on the edge length of the cube to be explored in each round, which gradually decreases as exploration progresses (see line 6 of Algorithm 6). These conditions together ensure the algorithm avoids over-zooming. In the initial learning phase, our knowledge of the optimal cube is limited, making it challenging to effectively distinguish suboptimal cubes using only the first stop

condition. However, the second condition, with a larger initial lower bound on cube edge length, prevents over-zooming. As the learning process advances, the algorithm can more reliably eliminate suboptimal cubes, thus avoiding over-zooming on them.

Flowchart and algorithm description. Due to page limitations, Appendix A.2 contains a flowchart illustrating the operation of the algorithm along with its description.

Theoretical result. Analyzing the space complexity of the MBAD algorithm and its ADAPTIVECUBE subroutine requires careful consideration due to the subroutine's recursive nature. Specifically, the conditional logic that triggers further recursion or partitioning into 2^d subcubes adds layers of complexity. Within the ADAPTIVECUBE subroutine, each recursive invocation contributes to the call stack, with space consumption directly proportional to the recursion depth. The space required to sustain the state of each cube, alongside the recursive call stack within ADAPTIVECUBE, implies a complexity that scales linearly with recursion depth, complemented by constant overheads for variables preserved at each recursion level. Nonetheless, the algorithm's design allows for the direct computation of all parent and neighboring cube information from the current cube's coordinates and edge length, obviating the need for multiple cube storage in memory. Consequently, only a single cube needs to be maintained at any time during the ADAPTIVECUBE process, affirming a space complexity of $O(1)$ for the MBAD algorithm. This space complexity analysis directly informs the algorithm's time complexity. Similar to the MBUD algorithm, the overall time complexity of the MBAD algorithm remains linear with respect to the total number of rounds.

As a by-product of the MBAD algorithm, we introduce a simpler, more practical algorithm for scenarios where $d_z \leq 1$. Detailed descriptions and theoretical analyses of this algorithm can be found in Appendix D. We provide the theoretical result below and elaborate on the details of the theoretical analysis in Appendix C. The result establishes the optimal instance-dependent upper bound, up to a logarithmic factor, for the stochastic Lipschitz bandits problem. Previous works (Slivkins, 2014; Kleinberg et al., 2019) have already established related lower bounds, indicating that our work achieves near-optimal regret. While there are other forms of results, such as those presented in work (Magureanu et al., 2014), we believe that adopting one form is sufficient to demonstrate the near-optimal performance of our algorithm.

Theorem 2. *For Lipschitz bandits with time horizon T and Lipschitz constant L, Algorithm 5 with $c \geq 5$ achieves regret*

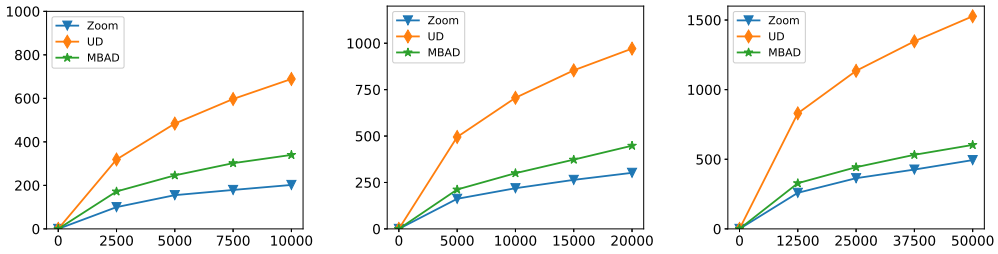
$$\mathbb{R}_{\mathcal{X}}(T) \leq \tilde{O}(T^{\frac{d_z+1}{d_z+2}}),$$

using $O(1)$ stored arms, where d_z is the zooming dimension of space $\mathcal{X}$.

We also mainly consider the clean event. The algorithm plays in a 'round-robin' manner. There are at most $O(\log T)$ phases because of the delicate design of the budget for each phase. For each phase, we show that the deviation between the mean reward of the best-estimated arm and optimal expected per-round reward $\mu^*_{\mathcal{X}}$ is small. Then the algorithm could approximately adjust the sub-optimality of arms and set more probes in more promising regions. Then we prove that the incurred regret could be bounded by $O(T^{\frac{z+1}{z+2}}(\log T)^{\frac{2}{z+2}})$ by bounding the pulls of bad arms according to the definition of zooming dimension.

5 Numerical Evaluations

In this section, we show the efficiency of our algorithms through a series of numerical simulations. The baseline consists of three algorithms: the uniform discretization with UCB1 algorithm (UD) and the zooming algorithm. For the uniform discretization, we pick a fixed ϵ-mesh of the action space and run the UCB1 algorithm only considering the finite uniform discretization action space. The UCB1 algorithm is a popular algorithm for achieving near-optimal regret with finite action space. Kleinberg (2004) prove that the uniform achieves optimal worst-case regret up to logarithm factors. The zooming algorithm (Kleinberg et al., 2019) is an implementation of the adaptive discretization strategy, which deploys more probes in regions deemed more 'promising'. Theoretical analysis shows that the zooming algorithm both achieves optimal worst-case regret and instance-dependent regret up to logarithm factors.



(a) $T = 1 \times 10^4$ (b) $T = 2 \times 10^4$ (c) $T = 5 \times 10^4$

Figure 1: The results obtained with different time horizons.

We set $\mathcal{X} = [0, 1]$ and choose the reward function $f(x) = 0.5 - |x - 0.5|$. In each round t, the algorithm plays one arm x_t and receives a stochastic reward y satisfying

$$y = \begin{cases} f(x) + \xi, & 0 \le f(x) + \xi \le 1 \\ 1, & f(x) + \xi > 1 \\ 0, & f(x) + \xi < 0 \end{cases}.$$

Specifically, $\xi \sim \mathcal{N}(0, 0.1^2)$ is the Gaussian noise. The MBAD algorithm are only allowed to store two arms in memory, while there is no memory constraint for the UD+UCB1 algorithm and the zooming algorithm. All results are averages over 50 runs. Figure 1 displays the results obtained across varying time horizons, where the horizontal axis denotes the time horizon and the vertical axis measures regret. From the figure, we have that MBAD algorithm significantly outperforms the UD strategy. Additional numerical results are detailed in Appendix E.

6 Conclusion and Discussion

We consider the Lipschitz bandits with limited memory problem. We introduce two novel algorithms: the Memory Bounded Uniform Discretization (MBUD) algorithm and the Memory Bounded Adaptive Discretization (MBAD) algorithm, which are predicated on the principles of uniform and adaptive discretization, respectively. Theoretical analyses reveal that the MBAD algorithm achieves near-optimal performance with $O(1)$ stored arms and $O(T)$ time complexity, highlighting its efficiency and practical applicability. Moreover, numerical results show the efficiency of our algorithms.

The Lipschitz bandit problem in higher dimensions is often perceived as a 'needle in a haystack' problem. Intuitively, finding the optimal solution in such high-dimensional spaces seems extremely challenging, but this perception does not always hold in practice. Many scenarios reveal beneficial structures within Lipschitz bandits, which is why our research emphasizes not only worst-case regret but also instance-dependent regret. Our proposed algorithm achieves nearly optimal time and space complexity for both worst-case and instance-dependent regrets.

In practical applications, Lipschitz bandit problems are found in areas such as non-parametric estimation, model selection in machine learning tasks, and decision-making processes in robotics and games. Furthermore, research on Lipschitz bandits has inspired algorithmic advancements in other domains, such as decision trees and tree-based methods, where the principles from Lipschitz bandit algorithms guide the splitting and growth of trees. Despite these advancements, certain limitations remain. High-dimensional Lipschitz bandits can still pose significant computational challenges, especially in cases where the underlying structure is less apparent or more complex. Additionally, the requirement for sufficient exploration to accurately estimate the optimal arm can lead to increased computational overhead in large action spaces.

Our algorithm introduces a novel framework that efficiently addresses online decision-making and balances exploration and exploitation in Lipschitz action spaces. This framework leverages beneficial structures in the problem space to enhance performance while maintaining computational efficiency. We hope our approach could make a substantial contribution to the community, especially in areas that require efficient and effective decision-making under uncertainty.

References

Acharya, J., Bhadane, S., Indyk, P., and Sun, Z. Estimating entropy of distributions in constant space. In *Advances in Neural Information Processing Systems 32*, pp. 5163–5174, 2019.

Agarwal, A., Khanna, S., and Patil, P. A sharp memory-regret trade-off for multi-pass streaming bandits. In *Conference on Learning Theory (COLT)*, volume 178 of *Proceedings of Machine Learning Research*, pp. 1423–1462. PMLR, 2022.

Agrawal, R. The continuum-armed bandit problem. *SIAM journal on control and optimization*, 33 (6):1926–1951, 1995.

Assadi, S. and Wang, C. Exploration with limited memory: streaming algorithms for coin tossing, noisy comparisons, and multi-armed bandits. In *Symposium on Theory of Computing (STOC)*, pp. 1237–1250. ACM, 2020.

Assadi, S. and Wang, C. Single-pass streaming lower bounds for multi-armed bandits exploration with instance-sensitive sample complexity. In *Advances in Neural Information Processing Systems 35*, 2022.

Assadi, S. and Wang, C. The best arm evades: Near-optimal multi-pass streaming lower bounds for pure exploration in multi-armed bandits. *CoRR*, abs/2309.03145, 2023a.

Assadi, S. and Wang, C. The best arm evades: Near-optimal multi-pass streaming lower bounds for pure exploration in multi-armed bandits. *arXiv preprint arXiv:2309.03145*, 2023b.

Berg, T., Ordentlich, O., and Shayevitz, O. On the memory complexity of uniformity testing. In *Proceedings of the 35th Conference on Learning Theory*, pp. 3506–3523, 2022.

Blanchard, M., Zhang, J., and Jaillet, P. Memory-constrained algorithms for convex optimization. In *Advances in Neural Information Processing Systems 36*, 2023a.

Blanchard, M., Zhang, J., and Jaillet, P. Quadratic memory is necessary for optimal query complexity in convex optimization: Center-of-mass is pareto-optimal. In *Proceedings of 36th Conference on Learning Theory*, pp. 4696–4736, 2023b.

Brown, G., Bun, M., and Smith, A. D. Strong memory lower bounds for learning natural models. In *Proceedings of the 35th Conference on Learning Theory*, pp. 4989–5029, 2022.

Bubeck, S., Munos, R., Stoltz, G., and Szepesvári, C. X-armed bandits. *Journal of Machine Learning Research*, 12(5), 2011a.

Bubeck, S., Stoltz, G., and Yu, J. Y. Lipschitz bandits without the lipschitz constant. In *Algorithmic Learning Theory*, volume 6925, pp. 144–158. Springer, 2011b.

Chaudhuri, A. R. and Kalyanakrishnan, S. Regret minimisation in multi-armed bandits using bounded arm memory. In *AAAI*, pp. 10085–10092. AAAI Press, 2020.

Chen, X. and Peng, B. Memory-query tradeoffs for randomized convex optimization. In *Proceedings of the 64th Symposium on Foundations of Computer Science*, 2023.

Chen, X., Papadimitriou, C. H., and Peng, B. Memory bounds for continual learning. In *Proceedings of the 63rd Symposium on Foundations of Computer Science*, pp. 519–530, 2022.

Diakonikolas, I., Kane, D. M., Pensia, A., and Pittas, T. Streaming algorithms for high-dimensional robust statistics. In *Proceedings of the 39th International Conference on Machine Learning*, pp. 5061–5117, 2022.

Feng, Y., Huang, Z., and Wang, T. Lipschitz bandits with batched feedback. In *NeurIPS*, 2022.

Garg, S., Raz, R., and Tal, A. Extractor-based time-space tradeoffs for learning. *Manuscript. July*, 2017.

Garg, S., Raz, R., and Tal, A. Time-space lower bounds for two-pass learning. In *Proceedings of the 34th Computational Complexity Conference*, pp. 22:1–22:39, 2019.

Grant, J. A. and Leslie, D. S. On thompson sampling for smoother-than-lipschitz bandits. In *The 23rd International Conference on Artificial Intelligence and Statistics*, volume 108 of *Proceedings of Machine Learning Research*, pp. 2612–2622. PMLR, 2020.

Ho, C., Slivkins, A., and Vaughan, J. W. Adaptive contract design for crowdsourcing markets: bandit algorithms for repeated principal-agent problems. In *Conference on Economics and Computation (EC)*, pp. 359–376. ACM, 2014.

Hopkins, M., Kane, D., Lovett, S., and Moshkovitz, M. Bounded memory active learning through enriched queries. In *Proceedings of the 34th Conference on Learning Theory*, pp. 2358–2387, 2021.

Jin, T., Huang, K., Tang, J., and Xiao, X. Optimal streaming algorithms for multi-armed bandits. In *Proceedings of the 38th International Conference on Machine Learning*, pp. 5045–5054, 2021.

Kang, Y., Hsieh, C., and Lee, T. C. M. Robust lipschitz bandits to adversarial corruptions. In *Advances in Neural Information Processing Systems 36*, 2023.

Kleinberg, R., Slivkins, A., and Upfal, E. Bandits and experts in metric spaces. *J. ACM*, 66(4): 30:1–30:77, 2019.

Kleinberg, R. D. Nearly tight bounds for the continuum-armed bandit problem. In *Advances in Neural Information Processing Systems (NIPS)*, pp. 697–704, 2004.

Kol, G., Raz, R., and Tal, A. Time-space hardness of learning sparse parities. In *Proceedings of the 49th Symposium on Theory of Computing*, pp. 1067–1080, 2017.

Krishnamurthy, A., Langford, J., Slivkins, A., and Zhang, C. Contextual bandits with continuous actions: Smoothing, zooming, and adapting. In *Conference on Learning Theory*, volume 99 of *Proceedings of Machine Learning Research*, pp. 2025–2027. PMLR, 2019.

Lazaric, A., Brunskill, E., et al. Online stochastic optimization under correlated bandit feedback. In *Proceedings of the 31st International Conference on Machine Learning*, pp. 1557–1565, 2014.

Lee, H., Lee, J., Choi, Y., Jeon, W., Lee, B., Noh, Y., and Kim, K. Local metric learning for off-policy evaluation in contextual bandits with continuous actions. In *NeurIPS*, 2022.

Li, W., Song, Q., and Honorio, J. Personalized federated x-armed bandit. In *International Conference on Artificial Intelligence and Statistics*, pp. 37–45, 2024a.

Li, W., Song, Q., Honorio, J., and Lin, G. Federated x-armed bandit. In *Proceedings of the 38th Conference on Artificial Intelligence*, pp. 13628–13636, 2024b.

Liau, D., Song, Z., Price, E., and Yang, G. Stochastic multi-armed bandits in constant space. In *International Conference on Artificial Intelligence and Statistics (AISTATS)*, volume 84, pp. 386–394. PMLR, 2018.

Lu, S., Wang, G., Hu, Y., and Zhang, L. Optimal algorithms for lipschitz bandits with heavy-tailed rewards. In *Proceedings of the 36th International Conference on Machine Learning*, volume 97, pp. 4154–4163. PMLR, 2019.

Lyu, X., Tal, A., Wu, H., and Yang, J. Tight time-space lower bounds for constant-pass learning. In *Proceedings of the 64th Symposium on Foundations of Computer Science*, 2023.

Magureanu, S., Combes, R., and Proutière, A. Lipschitz bandits: Regret lower bound and optimal algorithms. In *Proceedings of The 27th Conference on Learning Theory*, pp. 975–999, 2014.

Maiti, A., Patil, V., and Khan, A. Streaming algorithms for stochastic multi-armed bandits. *CoRR*, abs/2012.05142, 2020.

Marsden, A., Sharan, V., Sidford, A., and Valiant, G. Efficient convex optimization requires superlinear memory. In *Proceedings of the 35th Conference on Learning Theory*, pp. 2390–2430, 2022.

Peng, B. and Rubinstein, A. Near optimal memory-regret tradeoff for online learning. In *IEEE 64th Annual Symposium on Foundations of Computer Science*, pp. 1171–1194, 2023.

Peng, B. and Zhang, F. Online prediction in sub-linear space. *CoRR*, abs/2207.07974, 2022.

Podimata, C. and Slivkins, A. Adaptive discretization for adversarial lipschitz bandits. In *Conference on Learning Theory (COLT)*, volume 134 of *Proceedings of Machine Learning Research*, pp. 3788–3805. PMLR, 2021.

Raz, R. A time-space lower bound for a large class of learning problems. In *Proceedings of the 58th Symposium on Foundations of Computer Science*, pp. 732–742, 2017.

Raz, R. Fast learning requires good memory: A time-space lower bound for parity learning. *J. ACM*, 66(1):3:1–3:18, 2019.

Sharan, V., Sidford, A., and Valiant, G. Memory-sample tradeoffs for linear regression with small error. In *Proceedings of the 51st Symposium on Theory of Computing*, pp. 890–901, 2019.

Slivkins, A. Contextual bandits with similarity information. *J. Mach. Learn. Res.*, 15(1):2533–2568, 2014.

Slivkins, A. Introduction to multi-armed bandits. *Found. Trends Mach. Learn.*, 12(1-2):1–286, 2019.

Slivkins, A., Radlinski, F., and Gollapudi, S. Ranked bandits in metric spaces: learning diverse rankings over large document collections. *J. Mach. Learn. Res.*, 14(1):399–436, 2013.

Srinivas, V., Woodruff, D. P., Xu, Z., and Zhou, S. Memory bounds for the experts problem. In *Symposium on Theory of Computing (STOC)*, pp. 1158–1171. ACM, 2022.

Steinhardt, J., Valiant, G., and Wager, S. Memory, communication, and statistical queries. In *Proceedings of the 29th Conference on Learning Theory*, pp. 1490–1516, 2016.

Wang, C. Tight regret bounds for single-pass streaming multi-armed bandits. *CoRR*, abs/2306.02208, 2023.

Wang, T., Ye, W., Geng, D., and Rudin, C. Towards practical lipschitz bandits. In *FODS '20: ACM-IMS Foundations of Data Science Conference*, pp. 129–138. ACM, 2020.

Xue, B., Cheng, J., Liu, F., Wang, Y., and Zhang, Q. Multiobjective lipschitz bandits under lexicographic ordering. In *Proceedings of the 38th Conference on Artificial Intelligence*, pp. 16238–16246, 2024.

A Algorithm Flowchart

A.1 Flowchart for the MBUD Algorithm

The flowchart illustrates the process of the Memory Bounded Uniform Discretization (MBUD) algorithm, showcasing its core steps and transitions. The algorithm begins by dividing the exploration phase into $\lceil \log \log T \rceil$ phases, further segmented into cross exploration phases and the summarize phase.

At the start of the algorithm, the arm space $\mathcal{X} = [0, 1]^d$, time horizon T, and parameter c are initialized. The initial values for variables such as the best-estimated arm $\boldsymbol{y}$, its average reward $\bar{r}_y$, and the number of pulls n_y are set to zero. The budget parameter B_{-1} is initialized to 1, and the discretization parameter ϵ is calculated. Rather than covering the entire space continuously, the MBUD algorithm treats subcubes as a stream, selecting cubes for pairwise comparisons and gradually converging to the optimal region.

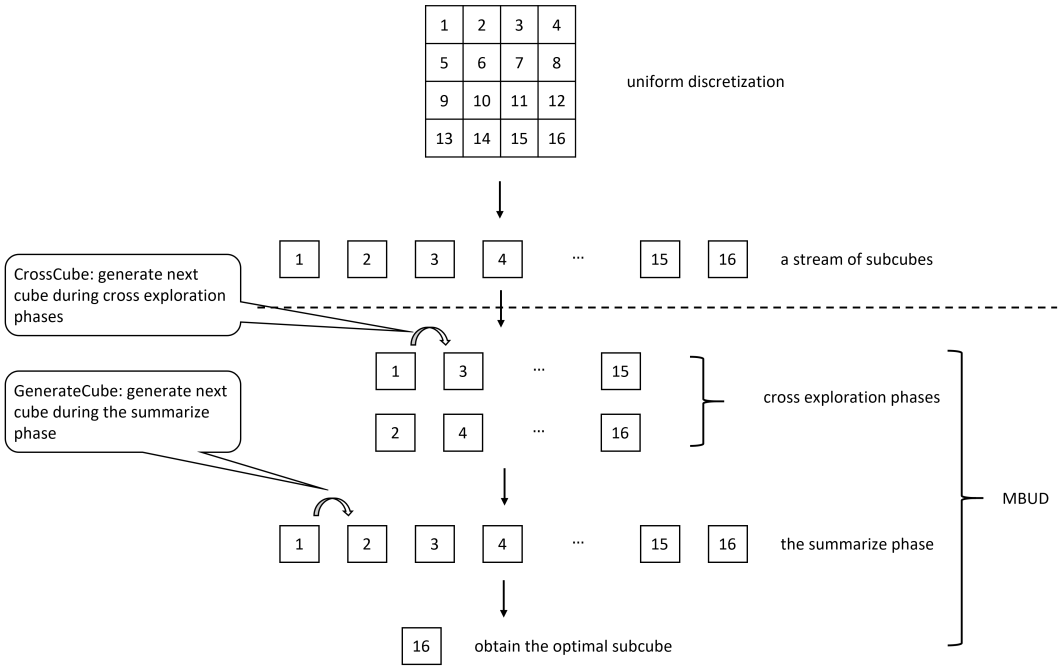


Figure 2: Flowchart for the MBUD algorithm

During the cross exploration phases, which encompass the first $\lceil \log \log T \rceil - 1$ phases, the algorithm iterates over arms within the discretized action space. In each phase p, the budget parameter B_p is updated to $\sqrt{TB_{p-1}}$. For each q from 1 to $\lfloor \phi \epsilon^{-d} \rfloor$, the CROSSCUBE function generates a new cube C by calculating parameters κ_1 and κ_2 and determining the cube's position using the edge-length ϵ. An arm $\boldsymbol{x}$ is then selected from the cube C. The COMPARE function evaluates the selected arm against the current best-estimated arm $\boldsymbol{y}$, updating $\boldsymbol{y}$ if necessary based on the comparison of their upper and lower confidence bounds. In the final phase, known as the summarize phase, the algorithm revisits all arms within the uniform discretization space. For each q from 1 to $\lfloor \epsilon^{-d} \rfloor$, the GENERATECUBE function generates a new cube C without considering the phases, using a parameter κ to determine the cube's position. An arm $\boldsymbol{x}$ is selected from this cube and compared against the current best-estimated arm $\boldsymbol{y}$ using the COMPARE function, ensuring thorough evaluation. The algorithm culminates by selecting the best-estimated arm $\boldsymbol{y}$ and playing it for the remaining rounds until the end of the time horizon.

A.2 Flowchart for the MBAD Algorithm

The MBAD algorithm dynamically adapts its discretization of the action space, focusing more on promising regions to identify the optimal arm with high probability. The flowchart effectively demonstrates how the algorithm narrows down the search space through adaptive discretization. Initially, the algorithm sets up the necessary parameters and variables. During the cross-exploration phases, the ADAPTIVECUBE function generates and selects arms from cubes. Arrows indicate the process of moving to the next cube if $\boldsymbol{y}$ remains the best arm, and the selection and evaluation of subcubes when the comparison budget condition is met.

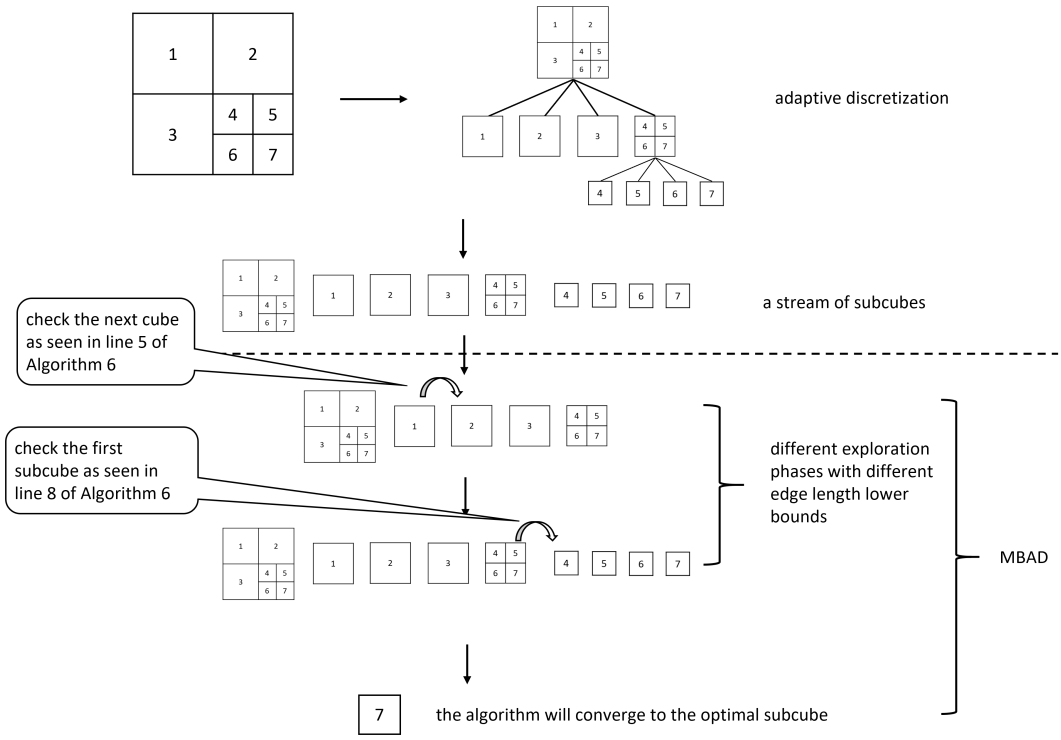


Figure 3: Flowchart for the MBAD algorithm

We present the pseudocode. The algorithm begins by initializing key parameters: the time horizon T and a constant c. Initial values for essential variables include the best-estimated arm $\boldsymbol{y}$, its average reward $\bar{r}_y$, and the number of pulls n_y, all set to zero. The initial budget B_1 is set to $\sqrt{T}$. In each phase p, the algorithm initializes a new arm $\boldsymbol{x}$ with zero values for its index, average reward $\bar{r}_x$, and the number of pulls n_x. The budget for the current phase b_p is calculated as $B_p \cdot \left(\frac{\log T}{T}\right)^{1/(d+2)}$. The ADAPTIVECUBE function is then called with parameters $m = 4$ and $q = 1$, and the budget for the next phase B_{p+1} is updated to $B_p \log T$.

The ADAPTIVECUBE function is crucial for refining the discretization of the action space and selecting promising arms. It begins by setting the edge-length $\epsilon = 2^{-m}$. The function calculates the parameter κ as the largest integer such that $k^d \leq \lfloor \epsilon^{-d} \rfloor$. A node is generated using the geometric sequence $\mathcal{G}_d(q, \kappa)$, and the cube C is defined by this node and ϵ. An arm $\boldsymbol{x}$ is selected from the cube C. If $q + 1 \leq 2^m$ and the COMPARE function indicates that $\boldsymbol{y}$ remains the best arm after comparison, the algorithm proceeds to the next cube with parameters m and $q + 1$. If the comparison budget $20\epsilon^{-2}$ does not exceed b_p, the algorithm updates $\boldsymbol{y}$ using the COMPARE function, partitions the cube C into 2^d subcubes, and checks the first subcube.

B Proof of Theorem 1

Let $\mathcal{S}$ denote the ϵ-mesh of the action space where $\epsilon = \left(\frac{\log T}{T}\right)^{1/(d+2)}$. Similarly, the discretization error is the gap of the best fixed arm benchmarks between two spaces:

$$\mathbb{D}_{\mathcal{X}}(\mathcal{S}) = T \cdot \sup_{x\in\mathcal{X}} \mu(x) - T \cdot \sup_{x\in\mathcal{S}} \mu(x).$$

Then the regret could be rewrote as

$$\mathbb{R}_{\mathcal{X}}(T) = \mathbb{R}_{\mathcal{X}}(T) + \mathbb{D}_{\mathcal{X}}(\mathcal{S}).$$

We call $\mathbb{D}_{\mathcal{X}}(\mathcal{S})$ the discretization error and it could be bounded by

$$\mathbb{D}_{\mathcal{X}}(\mathcal{S}) \le T\epsilon \le T^{\frac{d+1}{d+2}} (\log T)^{\frac{1}{d+2}}.$$

In the rest of this subsection, we shall prove

$$\mathbb{R}_{\mathcal{S}}(T) \le \tilde{O}\left(T^{\frac{d+1}{d+2}}\right).$$

For any fixed arm $x \in \mathcal{S}$, with probability $1 - T^{-c}$:

$$|\mu(x) - \bar{r}_x| \le \sqrt{\frac{c \log T}{n_x}}. \tag{1}$$

By a union bound for all arms and all rounds, (1) holds for all arm $x_t \in \mathcal{S}, t \in [T]$ with probability at least $1 - T^{4-c}$. To ease the reading, we assume $c = 5$. We call this 'clean event' and let $\mathcal{E}$ denote it. Then we analyze the regret based on the clean event. Let $\mathbb{R}^p_{\mathcal{S}}$ denote the regret for the p-th phase. Consider $\mathbb{R}^0_{\mathcal{S}}$, because the number of pulls of all arms in phase 0 is bounded by ϵB_0, we have

$$\mathbb{R}^0_{\mathcal{S}} \le 2\sqrt{T}/\log\log T.$$

Then we consider $\mathbb{R}^p_{\mathcal{S}}, p \in [\phi - 1]$. Let $\mu^*_{\mathcal{S}} := \sup_{x\in\mathcal{S}} \mu(x)$ denote the expected per-round reward of the optimal arm in space $\mathcal{S}$. Let x^*_p and μ^*_p denote the optimal selected arm during phase p and its expected per-round reward, respectively. From the definition of uniform discretization and Lipschitz condition, we have

$$\mu^*_{\mathcal{S}} - \mu^*_p \le \left(\frac{\log T}{T}\right)^{1/(d+2)} \log\log T. \tag{2}$$

for all phase $p \in [\phi - 1]$. To ease the reading, define

$$\Phi := \left(\frac{\log T}{T}\right)^{1/(d+2)} \log\log T.$$

For phase p, we consider the best estimate arm y at the start of the p-th phase. If x^*_{p-1} is discarded in phase $p-1$, according to the stop condition of compare strategy, we have

$$\bar{r}_y \ge \mu^*_{p-1} - \sqrt{\frac{5\log T}{n_y}} - \sqrt{\frac{5\log T}{n_{x^*_{p-1}}}} \ge \mu^*_{p-1} - 2\sqrt{\frac{5\log T}{\epsilon B_{p-1}}}. \tag{3}$$

For arbitrary discarded arm x, let R^p_x and N^p_x denote the accumulated reward and total number of pulls during phase p, respectively. Notice that the value of $\bar{r}_y - \sqrt{(5\log T)/n_y}$ is non-decreasing, so we have

$$\frac{R^p_x}{N^p_x - 1} + \sqrt{\frac{5\log T}{N^p_x - 1}} \ge \mu^*_{p-1} - 2\sqrt{\frac{5\log T}{\epsilon B_{p-1}}}.$$

Combine (2) and (3) together, we get

$$R^p_x \ge 2N^p_x \left(\mu^*_{p-1} - \sqrt{\frac{5\log T}{N^p_x - 1}} - \sqrt{\frac{5\log T}{\epsilon B_{p-1}}}\right).$$

Let $\mathbb{R}_x^p$ denote the cumulative regret of playing arm x during phase p, we have

$$\begin{aligned}\mathbb{R}_x^p &\leq 2N_x^p\left(\sqrt{\frac{5\log T}{N_x^p-1}}+\sqrt{\frac{5\log T}{\epsilon B_{p-1}}}+\Phi\right)\\ &\leq 2\left(\sqrt{6N_x^p\log T}+N_x^p\sqrt{\frac{5\log T}{\epsilon B_{p-1}}}+N_x^p\Phi\right).\end{aligned}$$

The first term from the gap between the expected reward of best estimated arm and the selected sub-optimal arm. The second term from the deviation between the best estimated arm and optimal expected per-round reward of the $(p-1)$-th phase. And the last is the discretization error during phase $p-1$. Let $\mathcal{S}_p$ denote the set of arms in phase p. According to Jensen's inequality

$$\frac{1}{|\mathcal{S}_p|}\sum_{x\in\mathcal{S}_p}\sqrt{N_x^p}\leq\sqrt{\frac{1}{|\mathcal{S}_p|}\sum_{x\in\mathcal{S}_p}N_x^p}\leq\sqrt{\epsilon B_p}.$$

Then we obtain

$$\sum_{x\in\mathcal{S}_p}\sqrt{N_x^p}\leq|\mathcal{S}_p|\sqrt{\epsilon B_p}\leq\sqrt{\frac{B_p}{\epsilon\log\log T}}.$$

Consider all selected arms during phase p and the stop condition of the compare strategy, we have

$$\begin{aligned}\mathbb{R}_{\mathcal{S}}^p &\leq 2\sum_{x\in\mathcal{S}_p}\left(\sqrt{6N_x^p\log T}+N_x^p\sqrt{\frac{5\log T}{\epsilon B_{p-1}}}+N_x^p\Phi\right)\\ &\leq\frac{3B_p}{\log\log T}\left(\sqrt{\frac{5\log T}{\epsilon B_{p-1}}}+\Phi\right)+2\sqrt{6\log T}\sum_{x\in\mathcal{S}_p}\sqrt{N_x^p}\\ &\leq\frac{3B_p}{\log\log T}\left(\sqrt{\frac{5\log T}{\epsilon B_{p-1}}}+\Phi\right)+3\sqrt{\frac{6B_p\log T}{\epsilon\log\log T}}.\end{aligned}$$

For the incurred regret by the deviation between the expected reward of best estimated arm and the selected sub-optimal arm, we have

$$\sum_{p=1}^{\phi-1}3\sqrt{\frac{6B_p\log T}{\epsilon\log\log T}}\leq 6\sqrt{\frac{6B_{\phi-1}\log T}{\epsilon\log\log T}}\leq 6\sqrt{\frac{6T\log T}{\epsilon\log\log T}}\leq\tilde{O}\left(T^{\frac{d+1}{d+2}}\right).$$

For the incurred regret the deviation between the best estimated arm and optimal expected per-round reward, we have

$$\sum_{p=1}^{\phi-1}\frac{3B_p}{\log\log T}\sqrt{\frac{5\log T}{\epsilon B_{p-1}}}\leq 6B_{\phi-1}\sqrt{\frac{5\log T}{\epsilon B_{\phi-2}}}\leq 6\sqrt{T}\sqrt{\frac{5\log T}{\epsilon}}.$$

For the incurred regret of the discretization error during one phase, we obtain

$$\sum_{p=1}^{\phi-1}\frac{3B_p\Phi}{\log\log T}\leq\frac{6B_{\phi-1}\Phi}{\log\log T}\leq\tilde{O}\left(T^{\frac{d+1}{d+2}}\right).$$

Combine them together, we get

$$\sum_{p=1}^{\phi-1}\mathbb{R}_{\mathcal{S}}^p\leq\tilde{O}\left(T^{\frac{d+1}{d+2}}\right).$$

Then we consider the total cumulative regret during the exploration part. Let $\mathbb{R}_{\mathcal{S}}^{\phi}$ denote the regret incurred in the last part. According to that the value of $\bar{r}_y-\sqrt{(5\log T)/n_y}$ is non-decreasing and the relationship between B_ϕ and $B_{\phi-1}$, we have

$$\mathbb{R}_{\mathcal{S}}^{\phi}\leq\mathbb{R}_{\mathcal{S}}^{\phi-1}.$$

Then the regret incurred by the exploration is

$$\sum_{p=0}^{\phi} \mathbb{R}_{\mathcal{S}}^{p} \leq 2 \sum_{p=0}^{\phi-1} \mathbb{R}_{\mathcal{S}}^{p} \leq \tilde{O}\left(T^{\frac{d+1}{d+2}}\right).$$

Consider the selected arm y after the exploration and let $\mathbb{R}_{\mathcal{S}}^{y}$ denote the regret due to selecting it. According to the stop condition of compare strategy, we have

$$\bar{r}_y \geq \mu_{\mathcal{S}}^{*} - 2\sqrt{\frac{5\log T}{\epsilon B_{\phi-1}}}.$$

Then for the regret

$$\mathbb{R}_{\mathcal{S}}^{y} \leq 2T\sqrt{\frac{5\log T}{\epsilon B_{\phi-1}}} \leq \tilde{O}\left(T^{\frac{d+1}{d+2}}\right).$$

Based on the clean event, we have

$$\mathbb{E}[\mathbb{R}_{\mathcal{S}}(T)|\mathcal{E}] = \mathbb{R}_{\mathcal{S}}^{y} + \sum_{p=0}^{\phi} \mathbb{R}_{\mathcal{S}}^{p} \leq \tilde{O}\left(T^{\frac{d+1}{d+2}}\right).$$

Then the regret is

$$\begin{aligned}
\mathbb{R}_{\mathcal{S}}(T) &= \mathbb{E}[\mathbb{R}_{\mathcal{S}}(T)|\mathcal{E}] \cdot \mathbb{P}(\mathcal{E}) + \mathbb{E}[\mathbb{R}_{\mathcal{S}}(T)|\neg\mathcal{E}] \cdot \mathbb{P}(\neg\mathcal{E}) \\
&\leq [\tilde{O}\left(T^{\frac{d+1}{d+2}}\right)](1 - 1/T) + 1 \\
&\leq \tilde{O}\left(T^{\frac{d+1}{d+2}}\right).
\end{aligned}$$

Combine it with the discretization error, then we complete the proof.

C Proof of Theorem 2

To ease the reading, let $c = 5$. For all arms $x_t \in \mathcal{X}$ and all rounds $t \in [T]$, the gap between the mean reward and the expectation could be bounded with probability $1 - T^{-1}$:

$$|\mu(x_t) - \bar{r}_{x_t}| \leq \sqrt{\frac{5\log T}{n_{x_t}}}, \forall t \in [T].$$

We call this 'clean event' $\mathcal{E}$ and mainly analyze the regret based on $\mathcal{E}$. Assume the MBAD algorithm consume all time horizons during the ϕ-th phase. For the stochastic Lipschitz instance, we always have $\phi \leq O\left(\frac{\log T}{\log\log T}\right)$. Let $\mathbb{R}_{\mathcal{S}}^{p}$ denote the regret for the p-th phase. For the first phase, we have $\mathbb{R}_{\mathcal{X}}^{1} \leq N^1 \leq B_1 \leq \sqrt{T}$. Then we consider $\mathbb{R}_{\mathcal{S}}^{p}, 1 < p \leq \phi$. For phase p, we consider the best estimate arm y at the start of the p-th phase. If x_{p-1}^{*} is discarded in phase $p-1$, according to the stop condition of compare strategy, we have

$$\bar{r}_y \geq \mu_{p-1}^{*} - \sqrt{\frac{5\log T}{n_y}} - \sqrt{\frac{5\log T}{n_{x_{p-1}^{*}}}} \geq \mu_{p-1}^{*} - 2\sqrt{\frac{5\log T}{b_{p-1}}}.$$

For arbitrary discarded arm x, let R_x^p and N_x^p denote the accumulated reward and total number of pulls during phase p, respectively. Notice that the value of $\bar{r}_y - \sqrt{(5\log T)/n_y}$ is non-decreasing, so we have

$$\frac{R_x^p}{N_x^p - 1} + \sqrt{\frac{5\log T}{N_x^p - 1}} \geq \mu_{p-1}^{*} - 2\sqrt{\frac{5\log T}{b_{p-1}}}.$$

Then we get

$$R_x^p \geq 2N_x^p\left(\mu_{p-1}^{*} - \sqrt{\frac{5\log T}{N_x^p - 1}} - \sqrt{\frac{5\log T}{b_{p-1}}}\right).$$

Let $\mathbb{R}_x^p$ denote the cumulative regret of playing arm x during phase p, we have

$$\mathbb{R}_x^p \leq 2N_x^p \left(\sqrt{\frac{5\log T}{N_x^p - 1}} + \sqrt{\frac{5\log T}{b_{p-1}}} \right).$$

Similarly, the first term from the gap between the expected reward of best estimated arm and the selected sub-optimal arm. The second term from the deviation between the best estimated arm and optimal expected per-round reward of the $(p-1)$-th phase. Recall the set

$$\mathcal{Y}_i = \{x \in X : 2^{-i} \leq \Delta(x) < 2^{1-i}, i \in \mathbb{N}\},$$

and the definition of zooming dimension

$$d_z = \inf_{\beta \geq 0} \left\{ |\mathcal{S}_j| \leq O(\epsilon^\beta), \epsilon = O(2^{-j}), \forall j \in \mathbb{N} \right\}.$$

Pick $\delta = \left(\frac{\log^2 T}{T}\right)^{\frac{1}{d_z+2}}$, if $\sqrt{\frac{5\log T}{N_x^p - 1}} + \sqrt{\frac{5\log T}{b_{p-1}}} \leq O(\delta)$, then $\mathbb{R}_x^p \leq O(\delta N_x^p)$. If $\sqrt{\frac{5\log T}{b_{p-1}}} > \Omega(\delta)$, then $b_{p-1} = O(\log T)\Delta^{-2}(x)$. If $\sqrt{\frac{5\log T}{N_x^p - 1}} > \Omega(\delta)$, then $N_x^p = O(\log T)\Delta^{-2}(x)$. According the stop condition of the compare strategy and the definition of zooming dimension, we have

$$\begin{aligned}
\mathbb{R}_{\mathcal{X}}^p &\leq \delta N^p + \sum_{i:2^{-i} > \delta} \sum_{x \in Y_i} \mathbb{R}_x^p \\
&\leq \delta N^p + O((\log T)^2)\delta^{d_z+1} \leq \delta T + O((\log T)^2)\delta^{d_z+1} \\
&\leq O(T^{\frac{d_z+1}{d_z+2}}(\log T)^{\frac{2}{d_z+2}}).
\end{aligned}$$

Then we have

$$\sum_{p=1}^{\phi} \mathbb{R}_{\mathcal{X}}^p \leq \sum_{p=1}^{\phi} O(T^{\frac{d_z+1}{d_z+2}}(\log T)^{\frac{2}{d_z+2}}) \leq \tilde{O}(T^{\frac{d_z+1}{d_z+2}}).$$

Based on the clean event, we have

$$\mathbb{E}[\mathbb{R}_{\mathcal{X}}(T)|\mathcal{E}] \leq \sum_{p=1}^{\phi} \mathbb{R}_{\mathcal{X}}^p \leq \tilde{O}(T^{\frac{d_z+1}{d_z+2}}).$$

The regret is

$$\mathbb{R}_{\mathcal{X}}(T) \leq \tilde{O}(T^{\frac{d_z+1}{d_z+2}})(1 - 1/T) + 1 \leq \tilde{O}(T^{\frac{d_z+1}{d_z+2}}).$$

Then we complete the proof.

D A Simple Algorithm

We present the pseudocode. The algorithm also maintains the index x, the mean reward estimator $\bar{r}_x$, and the number of pulls n_x for one arm in memory. The constants c and η are two parameters to balance the exploration and exploitation. For each phase p, the algorithm determines the budget b_p of samples for each probe, the number of total pulls N^p during the phase, and the usage factor λ_p obtained after the phase. Once the arm x is discarded, the algorithm chooses the next probe by adding $\eta\sqrt{(c\log T)/(n_x L^2)}$, which is the step size and has the same order as the confidence radius of arm x.

We have the following theoretical result.

Theorem 3. *For Lipschitz bandits with time horizon T and Lipschitz constant L, Algorithm 7 with $c \geq 5$ and $\eta = 1/3$ achieves regret*

$$\mathbb{R}_{\mathcal{X}}(T) \leq \tilde{O}(T^{\frac{d_z+1}{d_z+2}}),$$

using $O(1)$ stored arms, where $d_z \leq 1$ is the zooming dimension of space $\mathcal{X}$.

Algorithm 7 A Simple Algorithm

Input: rounds T, Lipschitz constant L, constant c and η

$y \leftarrow 0, \bar{r}_y \leftarrow 0, n_y \leftarrow 0, B_1 \leftarrow \sqrt{T}$.
for $p = 1, 2, \cdots$ **do**
  $x \leftarrow 0, N^p \leftarrow 0$.
  while $x \leq 1$ **do**
    $\bar{r}_x \leftarrow 0, n_x \leftarrow 0, b_p \leftarrow (B_p/3)^{2/3}(c \log T)^{1/3} L^{-2/3}$.
    while $n_x \leq b_p$ or $n_y \leq b_p$ **do**
      $N^p \leftarrow N^p + 1$.
      Pull the least played arm between x and y, and select a random arm if there not exists a least played arm.
      Update $\bar{r}_x, n_x, \bar{r}_y, n_y$.
      if $\min\{\bar{r}_x + \sqrt{(c \log T)/n_x}, 1\} < \max\{\bar{r}_y - \sqrt{(c \log T)/n_y}, 0\}$ **then**
        Break.
      else if $\max\{\bar{r}_x - \sqrt{(c \log T)/n_x}, 0\} > \max\{\bar{r}_y - \sqrt{(c \log T)/n_y}, 0\}$ **then**
        $y \leftarrow x, \bar{r}_y \leftarrow \bar{r}_x, n_y \leftarrow n_x$.
        Break.
      end if
    end while
    $x \leftarrow x + \eta \sqrt{(c \log T)/(n_x L^2)}$.
  end while
  $B_{p+1} \leftarrow B_p \log T$.
end for

D.1 Proof of Theorem 3

The proof closely mirrors that of Theorem 2. To ease the reading, let $c = 5$ and $\eta = 1/3$. For all arms $x_t \in \mathcal{X}$ and all rounds $t \in [T]$, the gap between the mean reward and the expectation could be bounded with probability $1 - T^{-1}$:

$$|\mu(x_t) - \bar{r}_{x_t}| \leq \sqrt{\frac{5 \log T}{n_{x_t}}}, \forall t \in [T].$$

We call this 'clean event' $\mathcal{E}$ and mainly analyze the regret based on $\mathcal{E}$. For the number of phases ϕ, we always have $\phi \leq O\left(\frac{\log T}{\log \log T}\right)$. Notice that the number of total pulls during the phase p,

$$N^p \leq 3 b_p L \sqrt{\frac{b_p}{5 \log T}} \leq 3 (B_p/3)^{2/3} (5 \log T)^{1/3} \sqrt{\frac{(B_p/3)^{2/3} (5 \log T)^{1/3}}{5 \log T}} = B_p.$$

Let $\mathbb{R}_{\mathcal{S}}^p$ denote the regret for the p-th phase. For the first phase, we have $\mathbb{R}_{\mathcal{X}}^1 \leq N^1 \leq B_1 \leq \sqrt{T}$. Then we consider $\mathbb{R}_{\mathcal{S}}^p, 1 < p \leq \phi$. For phase p, we consider the best estimate arm y at the start of the p-th phase. If x_{p-1}^* is discarded in phase $p-1$, according to the stop condition of compare strategy, we have

$$\bar{r}_y \geq \mu_{p-1}^* - \sqrt{\frac{5 \log T}{n_y}} - \sqrt{\frac{5 \log T}{n_{x_{p-1}^*}}} \geq \mu_{p-1}^* - 2\sqrt{\frac{5 \log T}{b_{p-1}}}.$$

For arbitrary discarded arm x, let R_x^p and N_x^p denote the accumulated reward and total number of pulls during phase p, respectively. Notice that the value of $\bar{r}_y - \sqrt{(5 \log T)/n_y}$ is non-decreasing, so we have

$$\frac{R_x^p}{N_x^p - 1} + \sqrt{\frac{5 \log T}{N_x^p - 1}} \geq \mu_{p-1}^* - 2\sqrt{\frac{5 \log T}{b_{p-1}}}.$$

Then we get

$$R_x^p \geq 2 N_x^p \left(\mu_{p-1}^* - \sqrt{\frac{5 \log T}{N_x^p - 1}} - \sqrt{\frac{5 \log T}{b_{p-1}}}\right).$$

Let $\mathbb{R}_x^p$ denote the cumulative regret of playing arm x during phase p, we have

$$\mathbb{R}_x^p \leq 2N_x^p \left(\sqrt{\frac{5\log T}{N_x^p - 1}} + \sqrt{\frac{5\log T}{b_{p-1}}} \right).$$

Similarly, the first term from the gap between the expected reward of best estimated arm and the selected sub-optimal arm. The second term from the deviation between the best estimated arm and optimal expected per-round reward of the $(p-1)$-th phase. Recall the set

$$\mathcal{Y}_i = \{x \in X : 2^{-i} \leq \Delta(x) < 2^{1-i}, i \in \mathbb{N}\},$$

and the definition of zooming dimension

$$d_z = \inf_{\beta \geq 0} \left\{ |\mathcal{S}_j| \leq O(\epsilon^\beta), \epsilon = O(2^{-j}), \forall j \in \mathbb{N} \right\}.$$

Pick $\delta = \left(\frac{\log^2 T}{T}\right)^{\frac{1}{d_z+2}}$, if $\sqrt{\frac{5\log T}{N_x^p - 1}} + \sqrt{\frac{5\log T}{b_{p-1}}} \leq O(\delta)$, then $\mathbb{R}_x^p \leq O(\delta N_x^p)$. If $\sqrt{\frac{5\log T}{b_{p-1}}} > \Omega(\delta)$, then $b_{p-1} = O(\log T)\Delta^{-2}(x)$. If $\sqrt{\frac{5\log T}{N_x^p - 1}} > \Omega(\delta)$, then $N_x^p = O(\log T)\Delta^{-2}(x)$. According the stop condition of the compare strategy and the definition of zooming dimension, we have

$$\begin{aligned}
\mathbb{R}_{\mathcal{X}}^p &\leq \delta N^p + \sum_{i:2^{-i} > \delta} \sum_{x \in Y_i} \mathbb{R}_x^p \\
&\leq \delta N^p + O((\log T)^2)\delta^{d_z+1} \leq \delta T + O((\log T)^2)\delta^{d_z+1} \\
&\leq O(T^{\frac{d_z+1}{d_z+2}} (\log T)^{\frac{2}{d_z+2}}).
\end{aligned}$$

Then we have

$$\sum_{p=1}^{\phi} \mathbb{R}_{\mathcal{X}}^p \leq \sum_{p=1}^{\phi} O(T^{\frac{d_z+1}{d_z+2}} (\log T)^{\frac{2}{d_z+2}}) \leq \tilde{O}(T^{\frac{d_z+1}{d_z+2}}).$$

Based on the clean event, we have

$$\mathbb{E}[\mathbb{R}_{\mathcal{X}}(T)|\mathcal{E}] \leq \sum_{p=1}^{\phi} \mathbb{R}_{\mathcal{X}}^p \leq \tilde{O}(T^{\frac{d_z+1}{d_z+2}}).$$

The regret is

$$\mathbb{R}_{\mathcal{X}}(T) \leq \tilde{O}(T^{\frac{d_z+1}{d_z+2}})(1 - 1/T) + 1 \leq \tilde{O}(T^{\frac{d_z+1}{d_z+2}}).$$

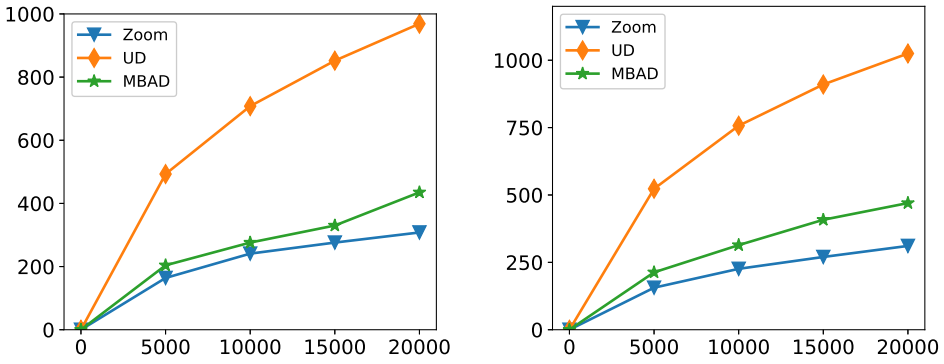
Then we complete the proof.

E Numerical Results

Different variances. Keeping other setting of Figure 1(b) unchanged, Figure 4(a-b) present the results with different variances. For $\xi \sim \mathcal{N}(0, 0.05^2)$ (Figure 4(a)), the MBAD algorithm achieves 140.2% regret of the zooming algorithm. For $\xi \sim \mathcal{N}(0, 0.2^2)$ (Figure 4(b)), the MBAD algorithm achieves 149.5% regret of the zooming algorithm. Overall, our algorithm performs better when the variance is small. Note that the algorithm is based on the 'successive elimination-style' strategy and smaller variances make the algorithm select better arms during comparisons with higher probability.

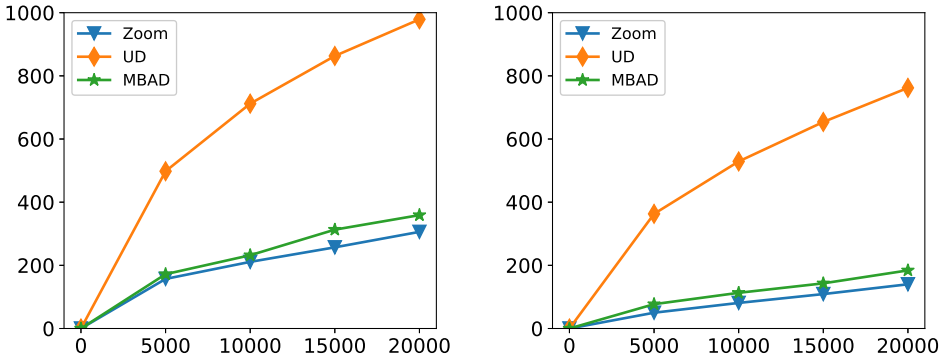
Uniform noise distribution. Keeping other setting of Figure 1(b) unchanged, Figure 5(a) presents the results with uniform noise distribution. For $\xi \sim \mathcal{U}(-0.2, 0.2)$ (Figure 5(a)), the MBAD algorithm achieves 118.1% regret of the zooming algorithm. The results show that our algorithms work robustly for different noise distributions.

Quadratic reward function. We also provide the numerical results for different reward functions. Keeping other setting of Figure 1(b) unchanged, Figure 5(b) presents the results with uniform noise distribution. For $f(x) = 1 - 4 \times (0.5 - x)^2$ (Figure 5(b)), the MBAD algorithm achieves 132.3% regret of the zooming algorithm. The results show that our algorithms work robustly for different reward functions.



(a) $\xi \sim \mathcal{N}(0, 0.05^2)$ (b) $\xi \sim \mathcal{N}(0, 0.2^2)$

Figure 4: Performance comparisons for Gaussian distribution with different variances.



(a) $\xi \sim \mathcal{U}(-0.2, 0.2)$ (b) $f(x) = 1 - 4 \times (0.5 - x)^2$

Figure 5: Performance comparisons for (a) Uniform distribution; (b) Quadratic reward function.

NeurIPS Paper Checklist

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes]

   Justification: The main claims in the abstract and introduction accurately reflect the paper's contributions and scope as they succinctly outline the problem addressed, the approach taken, and the novel insights or advancements achieved within the specified research domain.

   Guidelines:

   - The answer NA means that the abstract and introduction do not include the claims made in the paper.
   - The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A No or NA answer to this question will not be perceived well by the reviewers.
   - The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
   - It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

Answer: [Yes]

Justification: The paper discusses the limitations of the work in the Conclusion and Discussion section.

Guidelines:

- The answer NA means that the paper has no limitation while the answer No means that the paper has limitations, but those are not discussed in the paper.
- The authors are encouraged to create a separate "Limitations" section in their paper.
- The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
- The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
- The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
- The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
- If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
- While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory Assumptions and Proofs**

Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

Answer: [Yes]

Justification: Yes, the paper provides the full set of assumptions and delivers complete and correct proofs for each theoretical result, ensuring rigor and thoroughness in the presentation of mathematical or theoretical findings.

Guidelines:

- The answer NA means that the paper does not include theoretical results.
- All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
- All assumptions should be clearly stated or referenced in the statement of any theorems.
- The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
- Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
- Theorems and Lemmas that the proof relies upon should be properly referenced.

4. **Experimental Result Reproducibility**

Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

Answer: [Yes]

Justification: Yes, the paper provides all the necessary details for anyone to reproduce the main experimental results, ensuring transparency and allowing others to validate the claims and conclusions independently.

Guidelines:

- The answer NA means that the paper does not include experiments.
- If the paper includes experiments, a No answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
- If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
- Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
- While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
  (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
  (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
  (c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).
  (d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. **Open access to data and code**

Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

Answer: [No]

Justification: We plan to release this information after the paper's publication.

Guidelines:

- The answer NA means that paper does not include experiments requiring code.
- Please see the NeurIPS code and data submission guidelines (https://nips.cc/public/guides/CodeSubmissionPolicy) for more details.
- While we encourage the release of code and data, we understand that this might not be possible, so "No" is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
- The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (https://nips.cc/public/guides/CodeSubmissionPolicy) for more details.
- The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.

- The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
- At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
- Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental Setting/Details**

   Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer, etc.) necessary to understand the results?

   Answer: [Yes]

   Justification: The paper clearly outlines essential information needed to understand the results.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
   - The full details can be provided either with the code, in appendix, or as supplemental material.

7. **Experiment Statistical Significance**

   Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

   Answer: [Yes]

   Justification: Yes, the paper appropriately reports error bars and provides correct definitions or other relevant information about the statistical significance of the experiments, ensuring clarity and accuracy in the interpretation of results.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - The authors should answer "Yes" if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.
   - The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
   - The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
   - The assumptions made should be given (e.g., Normally distributed errors).
   - It should be clear whether the error bar is the standard deviation or the standard error of the mean.
   - It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
   - For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g. negative error rates).
   - If error bars are reported in tables or plots, The authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments Compute Resources**

   Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

Answer: [Yes]

Justification: The paper gives clear details on the computer resources needed for each experiment.

Guidelines:

- The answer NA means that the paper does not include experiments.
- The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
- The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
- The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. **Code Of Ethics**

Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics https://neurips.cc/public/EthicsGuidelines?

Answer: [Yes]

Justification: The research conducted in the paper adheres to the NeurIPS Code of Ethics in all respects, ensuring ethical standards are met throughout the research process.

Guidelines:

- The answer NA means that the authors have not reviewed the NeurIPS Code of Ethics.
- If the authors answer No, they should explain the special circumstances that require a deviation from the Code of Ethics.
- The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader Impacts**

Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

Answer: [Yes]

Justification: The paper discusses societal impacts of the work in the Conclusion and Discussion section.

Guidelines:

- The answer NA means that there is no societal impact of the work performed.
- If the authors answer NA or No, they should explain why their work has no societal impact or why the paper does not address societal impact.
- Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.
- The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
- The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
- If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pretrained language models, image generators, or scraped datasets)?

Answer: [NA]

Justification: The paper is a theoretical paper and poses no such risks.

Guidelines:

- The answer NA means that the paper poses no such risks.
- Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
- Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
- We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

Answer: [NA]

Justification: The paper does not use existing assets.

Guidelines:

- The answer NA means that the paper does not use existing assets.
- The authors should cite the original paper that produced the code package or dataset.
- The authors should state which version of the asset is used and, if possible, include a URL.
- The name of the license (e.g., CC-BY 4.0) should be included for each asset.
- For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.
- If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
- For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.
- If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New Assets**

Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

Answer: [NA]

Justification: The paper does not release new assets.

Guidelines:

- The answer NA means that the paper does not release new assets.
- Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
- The paper should discuss whether and how consent was obtained from people whose asset is used.

- At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and Research with Human Subjects**

    Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

    Answer: [NA]

    Justification: The paper does not involve crowdsourcing nor research with human subjects.

    Guidelines:

    - The answer NA means that the paper does not involve crowdsourcing nor research with human subjects.
    - Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
    - According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional Review Board (IRB) Approvals or Equivalent for Research with Human Subjects**

    Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

    Answer: [NA]

    Justification: The paper does not involve crowdsourcing nor research with human subjects.

    Guidelines:

    - The answer NA means that the paper does not involve crowdsourcing nor research with human subjects.
    - Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
    - We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
    - For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.