Linear Bandit Algorithms with Sublinear Time Complexity

Shuo Yang [1] **Tongzheng Ren** [1] **Sanjay Shakkottai** [2] **Eric Price** [1] **Inderjit S. Dhillon** [1] **Sujay Sanghavi** [2]

Abstract

We propose two linear bandits algorithms with per-step complexity sublinear in the number of arms K. The algorithms are designed for applications where the arm set is extremely large and slowly changing. Our key realization is that choosing an arm reduces to a maximum inner product search (MIPS) problem, which can be solved approximately without breaking regret guarantees. Existing approximate MIPS solvers run in sublinear time. We extend those solvers and present theoretical guarantees for online learning problems, where adaptivity (i.e., a later step depends on the feedback in previous steps) becomes a unique challenge. We then explicitly characterize the tradeoff between the per-step complexity and regret. For sufficiently large K, our algorithms have sublinear per-step complexity and $\widetilde{O}(\sqrt{T})$ regret. Empirically, we evaluate our proposed algorithms in a synthetic environment and a real-world online movie recommendation problem. Our proposed algorithms can deliver a more than 72 times speedup compared to the linear time baselines while retaining similar regret.

1. Introduction

Linear bandits problem is one of the most fundamental online learning problems, with wide applications in recommender systems, online advertisements, etc. (Deshpande & Montanari, 2012). Such applications usually have an extremely large set of items (e.g., millions of products to be recommended), which also changes over time. Specifically, we focus on two types of changes: (1) some new arms are added from time to time (e.g., new movies added to the database); and more generally (2) some new arms are added, and some old arms deleted (e.g., some new advertisements to be shown and some old ones expired). Such an extremely large arm set typically changes slowly, in the sense that a relatively small number of arms are added or deleted at every time step.

A linear scan is slow for an extremely large arm set. It is thus demanding to design linear bandit algorithms that have per-step time complexity sublinear in the number of arms K, for an extremely large and slowly changing arm set.

Common algorithms for linear bandits have per-step time complexity linear in K. For instance, Thompson Sampling (TS) draws a random parameter estimate and selects the best arm accordingly (Abeille & Lazaric, 2017). It needs to scan the entire set of arms to choose the most promising arm, which leads to time complexity linear in K.

In this paper, we propose two algorithms with per-step time complexity sublinear in K, based on the observation below:

Key observation: *The arm selection step in many linear bandits algorithms reduces to an (exact) maximum inner product search (MIPS) problem. The right way to approximately solve the MIPS problem, coupled with careful analysis, allows us to achieve sublinear per-step complexity and desired regret guarantees.*

Formally, given a set $P \in \mathbb{R}^d, |P| = K$, and a query $q \in \mathbb{R}^d$, the MIPS problem aims to find the point $p \in P$ that maximizes $p^\top q$. The TS algorithm is an immediate example of selecting arms by solving a MIPS problem. For arms a with embedding x_a, TS algorithm chooses the arm that maximizes $x_a^\top \widetilde{\theta}$, for the random $\widetilde{\theta}$ drawn by TS.

More importantly, the exact solution of the MIPS problem is not necessary for obtaining an $O(\sqrt{T})$ regret bound. Take TS algorithm again as an example, the estimate $\widetilde{\theta}$ has an estimation error (i.e., $\widetilde{\theta} \neq \theta^*$, where θ^* is the true environment parameter that determines reward expectation). By properly controlling the approximate MIPS accuracy, the error of approximately solving MIPS can be smaller than the estimation error of $\widetilde{\theta}$. The regret will therefore stay in the same order as solving the MIPS exactly.

Many approaches were previously established to approximately solve MIPS with time complexity sublinear in K. While it seems promising to adopt those approximate MIPS

[1]Department of CS, The University of Texas at Austin, TX, USA. [2]Department of ECE, The University of Texas at Austin, TX, USA.. Correspondence to: Shuo Yang <yangshuo_ut@utexas.edu>.

Accepted at the International Conference on Machine Learning (ICML) 2022.

solvers, there are still two challenges remaining:

Challenge 1. *How to design (and analyze) approximate MIPS solvers for a sequence of adaptive queries?* Queries are adaptive (i.e., later queries depend on the results of previous ones) for online learning problems. Existing probabilistic guarantees for approximate MIPS solvers do not allow the queries to be adaptive. In this paper, we provide an alternative scheme where a query is first rounded to the nearest point in an ϵ-net before sending to the MIPS solver. While this scheme is less accurate for a single query, it allows for a better success guarantee when applied to an adaptive sequence of T queries.

Challenge 2. *How to characterize the connection between per-step time complexity and regret?* Intuitively, a faster approximate MIPS solver is less accurate and thus leads to larger regret, while an exact MIPS solver enjoys an optimal regret but spends much more time. This tradeoff has not been characterized. For the two algorithms in this paper, we characterize this tradeoff, and furthermore, show that it allows for $O(K^{1-\alpha(T)})$ per-step complexity for some $\alpha(T) > 0$ while retaining $\widetilde{O}(\sqrt{T})$ regret.

As a summary, our **main contributions** are

1. We formally define the (c, r, ϵ)-MIPS problem (Definition 3.1), and propose a scheme to approximately solve MIPS for a sequence of adaptive queries (Algorithm 1). In Theorem 3.3, we show that our proposed algorithm has $K^{1+o(1)}$ preprocessing time complexity, $K^{\rho_q + o(\log^{-0.45} K)}$ query time complexity, with $\rho_q < 1$, and $K^{o(1)}$ time complexity for adding a new arm.
2. Building upon Algorithm 1, we propose a sublinear time elimination-based algorithm (Algorithm 3) and a sublinear time TS-based algorithm (Algorithm 4). We characterize the tradeoff between the time complexity and regret (Theorems 5.2 and 6.1). With a proper choice of parameters and sufficiently large K, one can obtain $\widetilde{O}(\sqrt{T})$ regret and sublinear per-step time complexity.
3. We evaluate our algorithms in a synthetic environment and a real-world movie recommendation problem. Compared with the linear time complexity baselines, our algorithms can offer a 72 times speedup when there are 100,000 arms while obtaining similar regret.

2. Related Work

Linear bandits. Two popular lines of approaches have been proposed for linear bandits: UCB-based and TS-based algorithms. The UCB-based algorithm chooses the arm with the largest plausible (according to the upper confidence bound) expected reward. The first algorithm was proposed by Auer (2002) under the name SupLinRel, and extended by Chu et al. (2011) to be SupLinUCB. The algorithms maintain a confidence interval estimation, and eliminate the arms stage-by-stage. Subsequently, Abbasi-Yadkori et al. (2011) presented an improved confidence bound construction and proposed the OFUL algorithm. It achieves $O(d\sqrt{T}\log T)$ regret bound, which nearly matches the information-theoretic lower bound $\Omega(d\sqrt{T})$ (Dani et al., 2008) up-to logarithmic factors.

TS algorithms maintain a posterior distribution of the environment parameter, and sampling from the posterior to determine the best arm. There is now a rich literature on both Bayesian (Russo & Van Roy, 2014; 2016) and frequentist (Kaufmann et al., 2012; Agrawal & Goyal, 2013; Gopalan et al., 2014; Abeille & Lazaric, 2017) regret bounds. Our work is based on the frequentist analysis for linear Thomson Sampling, introduced in (Abeille & Lazaric, 2017). For an arm set $\mathcal{A}$ with K arms, all previously mentioned algorithms have a $\Theta(K)$ per-step time complexity.

There are previous algorithms that achieve sublinear in K complexity, but do not fit into our setting. (Todd, 2016; Lattimore et al., 2020) show that the "optimal design" approach has constant per-step complexity, but does not work for a changing arm set. (Liau et al., 2018) solves the multi-arm bandits problem with constant per-step complexity and constant space complexity, but the approach does not extend to the linear bandits problem.

Jun et al. (2017) considered accelerating a TS and a modified UCB algorithm to have $\widetilde{O}(K^{\rho})$ per-step time complexity, with $\rho = 1 - o(1)$. Their proposed algorithms, however, need $\Omega(K^{1+\rho}T)$ time in preprocessing, as they need to build a MIPS solver for each of the steps in T to deal with adaptive queries. There is much room to improve on the near quadratic dependency on K.

Max inner product search (MIPS). There has been a large volume of work on (approximately) solving MIPS (Teflioudi et al., 2015; Shen et al., 2015; Guo et al., 2016; Li et al., 2017; Yu et al., 2017; Morozov & Babenko, 2018; Abuzaid et al., 2019; Ding et al., 2019; Tan et al., 2019; Zhou et al., 2019). It has also been demonstrated that MIPS can be applied to various problems for acceleration, e.g., quadratic regression (Yang et al., 2019), conditional gradient methods (Xu et al., 2021), sparsification problems (Song et al., 2022), reinforcement learning (Shrivastava et al., 2021), and deep learning (Spring & Shrivastava, 2017; Chen et al., 2019a;b; Kitaev et al., 2020; Chen et al., 2020; Song et al., 2021a;b).

For our theoretical analysis, we focus on reducing MIPS to the nearest neighbor search (NNS) problem, where various reductions have been previously proposed (Shrivastava & Li, 2014; Bachrach et al., 2014; Neyshabur & Srebro, 2015; Keivani et al., 2018). We then solve the NNS by Local-

ity Sensitive Hashing (LSH) (Andoni & Indyk, 2006; Har-Peled et al., 2012; Andoni et al., 2018; Yan et al., 2018), for its rigorous theoretical guarantee on sublinear query time. For our experiments, we use HNSW (Malkov & Yashunin, 2018) for its outstanding empirical performance.

3. MIPS Solver for Adaptive Queries

We start by formally defining the Maximum Inner Product Search (MIPS) problem. Subsequently, we define adaptive queries and show how it breaks existing MIPS solvers. We then propose our solution to adaptive queries, which can convert existing MIPS solvers to work for adaptive queries.

3.1. MIPS Problem and Sublinear Time Solver

Definition 3.1 ((c, r, ϵ)-MIPS problem)**.** Let $P \subseteq \mathbb{R}^d$ be a finite set of points with $\|p\|_2 \leq 1, \forall p \in P$. Let $q \in \mathbb{R}^d$ be the query with $\|q\|_2 \leq 1$. The (c, r, ϵ)-approximated max inner product search ((c, r, ϵ)-MIPS) aims to find $p \in P$ such that $\langle q, p\rangle \geq cr - \epsilon$ if there exists $p^* \in P$ with $\langle q, p^*\rangle \geq r + \epsilon$.

The definition is valid with $r > 0, c \leq 1, \epsilon \geq 0$. Intuitively, for any query q with unit norm, the (c, r, ϵ)-MIPS problem defined above looks for a point $p \in P$ with $\langle p, q\rangle \geq r$, allowing for $(1-c)$ multiplicative error and ϵ additive error. See Figure 1 for illustration.

Approximately solving the MIPS problem with sublinear time has been well studied. The next result is adapted from (Andoni et al., 2017), which solves $(c, r, 0)$-MIPS in sublinear time with a success probability of at least 0.9.

Proposition 3.2 (Single Query MIPS solver $\mathcal{S}(c, r, 0)$)**.** *For a point set $P \subseteq \mathbb{R}^d$ with K points, there exists a data structure $\mathcal{S}(c, r, 0)$ that solves $(c, r, 0)$-MIPS problem for an arbitrary query q with at least 0.9 probability. It has the following time complexity:* ***Preprocessing:*** *$K^{1+o(1)}$;* ***Add a Point to*** *P:* *$K^{o(1)}$;* ***Query:*** *$K^{\rho_q + o\left(\log^{-0.45} K\right)}$, where $\rho_q = \frac{4c'^2}{(1+c'^2)^2}$ and $c' = \sqrt{\frac{3-cr}{3-r}}$.*

Notice that for $c < 1$, we have $c' > 1$ and $\rho_q < 1$.

The online nature of linear bandits calls for a MIPS algorithm that can deal with a sequence of ***adaptive queries***, *where the later queries* ***depend on previous query results***.

Such adaptive queries naturally arise when applying a MIPS solver $\mathcal{S}$ to online learning problems - as will be discussed in later sections, one can query $\mathcal{S}$ with the current parameter estimate $\widetilde{\theta}_t$ and $\mathcal{S}$ returns an arm a_t that should be played. The query $\widetilde{\theta}_t$ depends on all previously played arms $a_\tau, \tau < t$, which are the results of previous queries.

As we illustrate in the next subsection, the adaptive queries introduce a fundamental challenge that one can not apply union bound to extend the probabilistic guarantee for one query to a sequence of adaptive queries.

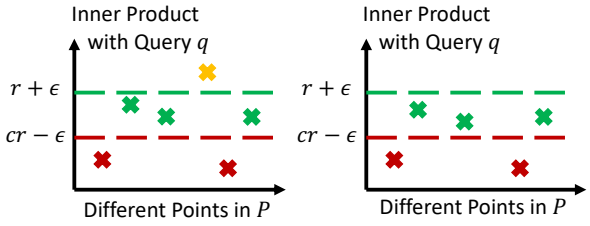

Figure 1: For query q, if there exists $p^* \in P$ that has inner product $\langle p, q\rangle \geq r + \epsilon$ (i.e., the yellow point), then the algorithm should return a point $p \in P$ with $\langle q, p\rangle \geq cr - \epsilon$ (i.e. green or yellow points in the left figure). Otherwise, no point needs to be returned (i.e., no point needs to be returned for the right-hand side figure).

3.2. Hardness of Adaptive Queries

To see how adaptive queries break union bound, consider the following example.

A Thought Experiment: A black-box $\mathcal{B}$ has a unit norm vector $p \in \mathbb{R}^{10}$, drawn uniformly at random when $\mathcal{B}$ is initialized. An agent $\mathcal{C}$ can send unit norm query $q \in \mathbb{R}^{10}$ to $\mathcal{B}$ and $\mathcal{B}$ returns a scalar $\langle q, p\rangle$. Suppose that the agent $\mathcal{C}$ can send 11 queries $q_1, \cdots, q_{11}$ and its goal is to send a query q^* with $\langle q^*, p\rangle = 1$.

For a single query q, it is probability 0 that $q = q^*$, as p is drawn uniformly at random. What is the probability that $\mathcal{C}$ can send such a query q^* within the 11 queries?

Adaptive v.s. Non-adaptive: Consider the two settings - *(1) Non-adaptive queries:* $q_1, \cdots, q_{11}$ can have arbitrary dependency on other queries, but can not depend on any of the results that $\mathcal{B}$ returns; and *(2) Adaptive queries:* a later query q_i can be constructed based on previous queries' result: $\langle q_j, p\rangle, j < i$.

For non-adaptive queries, each q_i has probability 0 to be q^*, and thus by union bound, it is probability 0 that $\mathcal{C}$ sends q^* within the 11 queries.

For adaptive queries, $\mathcal{C}$ can first send 10 linearly independent queries $q_1, \cdots, q_{10}$. With the results returned from $\mathcal{B}$, it can solve for p exactly, and send $q_{11} = p$ which gives $\langle q_{11}, p\rangle = 1$. Therefore, by allowing the queries to be adaptive, $\mathcal{C}$ can send q^* with probability 1. The drastic difference between probability 0 and probability 1 demonstrates the unique challenge of adaptive queries.

The thought experiment above shows that the probabilistic guarantee for one query cannot be extended to a sequence of adaptive queries via union bound. In the next subsection, we propose a scheme that builds upon $\mathcal{S}(c, r, 0)$ and solves MIPS for adaptive queries.

3.3. MIPS for Adaptive Queries

The key to solving MIPS with adaptive queries is to discretize the unit ℓ_2 ball Q (which contains all possible queries) into an ϵ-net $\widehat{Q}$ and use multiple independent $\mathcal{S}(c, r, 0)$ to give correct answers for all queries in $\widehat{Q}$.

For a query $q \in Q$, we first round q to its nearest neighbor $\widehat{q} \in \widehat{Q}$, which is at most ϵ away. We then query $\widehat{q}$ to multiple $\mathcal{S}_i(c, r, 0)$ with $i \in [\kappa]$, and return a correct result from any of the $\mathcal{S}_i(c, r, 0)$ as the result for query q. Figure 2 is an illustration for such process, and shows that it solves (c, r, ϵ)-MIPS problem. Algorithm 1 presents the pseudocode, which we will later refer as $\mathcal{M}(c, r, \epsilon, \delta)$.

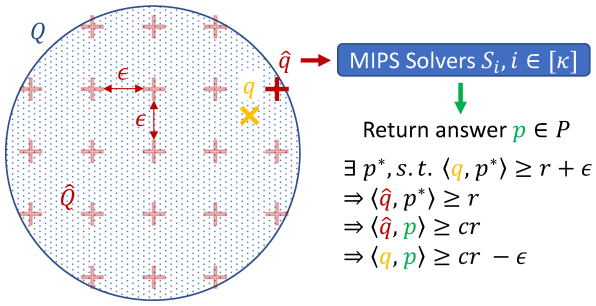

Figure 2: Illustration for Algorithm 1. The blue circle represents the continuous set Q which contains all possible queries, and $\widehat{Q}$ is an ϵ-net in Q. For a query $q \in Q$, it is first rounded to $\widehat{q} \in \widehat{Q}$. Then the MIPS solvers $\mathcal{S}_i(c, r, 0), i \in [\kappa]$ are invoked to answer $\widehat{q}$. Suppose $\exists p^* \in P, s.t. \langle q, p^* \rangle \geq r + \epsilon$ and a point $p \in P$ is returned by some $\mathcal{S}_i$. We have $\langle q, p \rangle \geq cr - \epsilon$ as indicated by the figure. Thus Algorithm 1 solves (c, r, ϵ)-MIPS.

Our next result shows that with $\kappa \triangleq d \log\left(\frac{Kd}{\epsilon\delta}\right)$ independent single query MIPS solvers $\mathcal{S}(c, r, 0)$, we can construct a MIPS solver $\mathcal{M}(c, r, \epsilon, \delta)$ that gives correct answer for $(c, r, 0)$-MIPS problem for all queries in $\widehat{Q}$ with probability at least δ. It therefore solves (c, r, ϵ)-MIPS problem for an arbitrary sequence of queries (possibly adaptive) from Q. Coupled with Proposition 3.2, our next result presents the sublinear in K time complexity of Algorithm 1.

Theorem 3.3 (Adaptive MIPS solver $\mathcal{M}(c, r, \epsilon, \delta)$)**.** *For a point set $P \in \mathbb{R}^d$ with K points, there exists a data structure $\mathcal{M}(c, r, \epsilon, \delta)$ (Algorithm 1) that sovles (c, r, ϵ)-MIPS problem correctly for arbitrary (possibly adaptive) queries with at least $(1 - \delta)$ probability, for any $\delta \in (0, 1)$. It has the following time complexity:* ***Preprocessing:*** *$\kappa K^{1+o(1)}$;* ***Add a Point to*** *P:* *$\kappa K^{o(1)}$;* ***Query:*** *$\kappa K^{\rho_q + o\left(\log^{-0.45} K\right)}$, where $\rho_q = \frac{4c'^2}{(1+c'^2)^2}$ and $c' = \sqrt{\frac{3-cr}{3-r}}$.*

Here we illustrate how one can use multiple instances of $\mathcal{S}_i(c, r, 0)$ to answer all queries in $\widehat{Q}$ correctly. Note that the only failure case of $\mathcal{S}_i(c, r, 0)$ is when there exists $p^* \in P$ such that $\langle q, p^* \rangle \geq r$ and it fails to return any p. This is because we can avoid spurious answer p with a simple sanity check of $\langle q, p \rangle \geq cr$. Therefore, outputting a point p is an indicator of success, which allows for using multiple $\mathcal{S}_i$ to construct another one with a higher success probability.

Algorithm 1 ADAPTIVE MIPS SOLVER $\mathcal{M}(c, r, \epsilon, \delta)$

Preprocess
Input: set of points $P \subseteq \mathbb{R}^d$, parameter (c, r, ϵ) of the MIPS problem (see Definition 3.1), desired failure probability bound δ
Set $\kappa = d \log\left(\frac{Kd}{\epsilon\delta}\right)$
Construct κ non-adaptive $(c, r, 0)$-MIPS solvers $\mathcal{S}_i, i \in [\kappa]$ for the set of points P (Proposition 3.2)

Add a new point p **to** P
Input: a new point $p \in \mathbb{R}^d$
For all $\mathcal{S}_i, i \in [\kappa]$, add p to $\mathcal{S}_i$

Query
Input: query $q \in \mathbb{R}^d$
Round the query q to the nearest point $\widehat{q}$, whose coordinates are all multiples of $\frac{\epsilon}{d}$
Query all non-adaptive MIPS solvers $\{\mathcal{S}_i\}_{i \in [\kappa]}$ with $\widehat{q}$
Return: any point $p \in P$ returned by any of $\{\mathcal{S}_i\}_{i \in [\kappa]}$, otherwise return null

4. Linear Bandits Problem Setup

We first introduce the ***extremely large and slowly changing*** linear bandits problem setting. Let $\mathcal{A}$ be the set of all arms, where each of the arm $a \in \mathcal{A}$ has a feature vector $x_a \in \mathbb{R}^d$.

The setting is called *extremely large* as we focus on the regime where $\mathcal{A}$ is extremely large while time horizon T is moderate (e.g., $T = \Theta(\log^\gamma K)$ for some constant γ).

The arm set $\mathcal{A}$ can *change* in two ways: at each time step t (1) there is a set of new arms $\mathcal{A}_{new}$ included into the arm set $\mathcal{A}$, but no deletions from $\mathcal{A}$; or more generally (2) there are some new arms $\mathcal{A}_{new}$ added, and some old arms in $\mathcal{A}$ deleted. We use K to denote the maximum size of $\mathcal{A}$, and our goal is to achieve per-step complexity sublinear in K.

Further, the arm set $\mathcal{A}$ changes *slowly* in the sense that, at every time step, there is at most C_{change} additions and deletions. For simplicity, we assume C_{change} to be a constant in the rest of our paper. Our results, however, are valid for any $C_{change} = O(K^\gamma)$ for any constant $\gamma < 1$.

At time step t, the online learner plays an arm $a_t \in \mathcal{A}$, and observes the reward r_t. We adopt the following commonly used assumptions:

Assumption 4.1 (Linear Realizability)**.** $\exists \theta^* \in \mathbb{R}^d$, such

that $r_t = \langle \theta^*, x_{a_t} \rangle + \eta_t$, where η_t is a mean 0 noise.

Assumption 4.2 (Subgaussian Noise)**.** The noise satisfies,

$$\mathbb{E}\left[\exp\left(\alpha\eta_t\right) \,\middle|\, \mathcal{F}_{t-1}\right] \leq \exp\left(\frac{\alpha^2}{2}\right), \forall \alpha \in \mathbb{R}, \forall t \in [T],$$

with the filtration $\mathcal{F}_{t-1} = \sigma\left(a_1, r_1, \cdots, a_{t-1}, r_{t-1}, a_t\right)$.

Assumption 4.3 (Bounded Parameters)**.** We assume that $\|\theta^*\|_2 \leq 1$ and $\|x_a\|_2 \leq 1, \forall a \in \mathcal{A}$.

The regret is defined as $R(T) := \sum_{t=1}^{T} x_{a_t^*}^\top \theta^* - x_{a_t}^\top \theta^*$, where $a_t^* := \operatorname{argmax}_{a \in \mathcal{A}} x_a^\top \theta^*$ is the optimal arm at time step t. The goal of the online learner is to minimize the regret $R(T)$.

5. Sublinear Time Elimination Algorithm

In this section, we focus on an arm set $\mathcal{A}$ that keeps growing and no arm is deleted. We present an elimination-based algorithm that achieves sublinear per-step complexity. Intuitively, we adopt the MIPS solver to choose the arm with approximately the highest uncertainty in $o(K)$ time. The elimination-based algorithm is additionally faster in later stages, as many arms are eliminated.

We can estimate θ^* with an online ridge regression,

$$\widehat{\theta}_{t+1} = \left(X_{1:t}^\top X_{1:t} + I\right)^{-1} X_{1:t}^\top Y_{1:t}, \tag{1}$$

where $X_{1:t}$ is the matrix whose rows are $x_{a_1}^\top, \cdots, x_{a_t}^\top$ and $Y_{1:t} = (r_1, \cdots, r_t)$. As established in (Abbasi-Yadkori et al., 2011), for any $\delta \in (0,1)$, with probability at least $(1-\delta)$, for all $t \geq 1$, we have $\|\widehat{\theta}_t - \theta^*\|_{V_t} \leq \beta(\delta)$, with $V_t = I + \sum_{s=1}^{t-1} x_s x_s^\top$ and $\beta(\delta) = 1 + \sqrt{2\log\left(\frac{1}{\delta}\right) + d\log\left(1 + \frac{T}{d}\right)}$.

In the standard linear bandits setting, the arm set $\mathcal{A}$ is fixed and does not grow over time. An elimination-based algorithm typically selects the arm a with the highest uncertainty, measured by $\|x_a\|_{V_t^{-1}}$, and periodically eliminates the bad arms (i.e. the arms with $x_a^\top \widehat{\theta}_t + \beta(\delta)\|x_a\|_{V_t^{-1}}$ smaller than $\underline{r} \triangleq \max_a x_a^\top \widehat{\theta}_t - \beta(\delta)\|x_a\|_{V_t^{-1}}$). After elimination, any remaining arm a costs at most $C \cdot \beta(\delta) \max \|x_a\|_{V_t^{-1}}$ regret, whose summation over T can be controlled by existing results.

Notice that the elimination requires a scan through all the arms. It is thus an $\Theta(K)$ time operation, which we do not hope to pay per-step. A common choice is to adopt stage-wise elimination – initializing $s = 1$ and eliminating when the uncertainty $\beta(\delta)\|x_a\|_{V_t^{-1}}$ of all arms falls below 2^{-s}, then increment s by 1. The elimination therefore only happens $\log T$ times. In the next subsection, however, we show that such a simple strategy fails when $\mathcal{A}$ keeps growing.

5.1. Efficient Elimination with Heap

Elimination is necessary every time when $\mathcal{A}$ grows. As new arms $\mathcal{A}_{new}$ coming, the elimination threshold $\underline{r} \triangleq \max_a x_a^\top \widehat{\theta}_t - \beta(\delta)\|x_a\|_{V_t^{-1}}$ might significantly increase. This typically happens when $\mathcal{A}_{new}$ contains an arm that is much better than the previously optimal arm. When $\underline{r}$ increases, some of arms that were not previously eliminated should be eliminated – otherwise they might still be selected according to the criterion $\operatorname{argmax}_a \|x_a\|_{V_t^{-1}}$ but incurring a regret much larger than $C \cdot \beta(\delta)\|x_a\|_{V_t^{-1}}$, which possibly leads to an unbounded regret.

The necessity to eliminate arms according to the newly added arms calls for a more carefully designed data structure, which supports incremental elimination but avoids linear scanning through all arms $\mathcal{A}$.

Our solution is presented in Algorithm 2, which partitions the arm set $\mathcal{A}$ into sets Ψ_s. The arms reside in Ψ_s all have uncertainty $\beta(\delta)\|x_a\|_{V_t^{-1}}$ smaller than 2^{-s}.

More importantly, the arm set Ψ_s is augmented with a min heap $\mathcal{H}_s$, which stores arm a indexed by $x_a^\top \widehat{\theta} + 2^{-s}$. Whenever a larger $\underline{r}$ appears, Ψ_s can quickly compare the heap top $x_a^\top \widehat{\theta} + 2^{-s}$ with $\underline{r}$ and eliminates the arm a as necessary. This avoids the linear scan for $\mathcal{A}$ when the elimination threshold $\underline{r}$ changes with the newly added arms $\mathcal{A}_{new}$.

An important implication is that after elimination (line 9 – 14 of Algorithm 2), playing an arm a with the (approximately) largest uncertainty, the regret is again bounded by $C \cdot \beta(\delta)\|x_a\|_{V_t^{-1}}$. Formally, at time step t, let s_t be the minimum s such that Ψ_s is not empty, we have:

Lemma 5.1. *For all* $a \in \Psi_{s_t}$, $x_{a_t^*}^\top \theta^* - x_a^\top \theta^* \leq 4 \cdot 2^{-s_t}$.

The approximate MIPS query step (line 15 – 18 of Algorithm 2) can upper bound 2^{-s_t} by $16 \cdot \beta(\delta)\|x_{a_t}\|_{V_t^{-1}}$, up to some approximation error. It, therefore, retains the original regret guarantee (by following existing bounds on the summation of $\|x_{a_t}\|_{V_t^{-1}}$ over t), without linearly scanning the arm set $\mathcal{A}$ at every step.

5.2. Algorithm and Its Regret, Time Complexity

Here we present the sublinear time elimination-based algorithm, and its regret and time complexity.

The crux to achieve per-step $o(K)$ time complexity is twofold: (1) Selecting an arm that approximately has maximum uncertainty $\|x_a\|_{V_t^{-1}} = \left\langle V_t^{-1}, x_a x_a^\top \right\rangle$ is a MIPS problem. Algorithm 1 can solve it with sublinear time complexity; (2) The elimination (line 11 and line 23) uses Algorithm 2 as a sub-routine, and in total causes $K^{1+o(1)}$ complexity, which the algorithm does not need to pay per-step.

Running Algorithm 3 for a linear bandits problem that sat-

Algorithm 2 HEAP AUGMENTED ARM SET Ψ_s

Initialize
Input: stage index s, parameters $d, \beta, \eta, \delta_\Psi$
Initialize an adaptive MIPS solver $\mathcal{M}_s$ with $\left(\frac{1}{4}, \frac{2^{-2s-2}(1-\eta^2)}{d^2\beta^2}, \frac{2^{-2s-2}\eta^2}{d^2\beta^2}, \delta_\Psi\right)$
Initialize an empty min heap $\mathcal{H}_s$

Add
Input: arm a, parameter estimate $\widehat{\theta}$
Add point $vec\left(x_a x_a^\top/d\right)$ to $\mathcal{M}_s$
Push $(x_a^\top\widehat{\theta} + 2^{-s}, a)$ to heap $\mathcal{H}_s$, using scalar $(x_a^\top\widehat{\theta} + 2^{-s})$ for ordering

Eliminate
Input: new elimination threshold $\underline{r}$
while Heap $\mathcal{H}_s$ top is smaller than $\underline{r}$ **do**
  $v, a = \mathcal{H}_s$.pop()
  Delete arm a from $\mathcal{M}_s$
end while

Query
Input: $V \in \mathbb{R}^{d\times d}$
Query $\mathcal{M}_s$ with $vec(V/d)$, denote the $\mathcal{M}_s$ output as a

Return: a if a is not null; otherwise return null

isfies Assumptions 4.1 to 4.3, we have the following result for the regret and time complexity.

Theorem 5.2 (Regret and time complexity of Algorithm 3, formal version see Theorem B.4)**.** *For any $\delta \in (0,1)$, with probability at least $1-\delta$, the regret is bounded by*

$$R(T) = \widetilde{O}\left(d\sqrt{T} + \eta(T)\cdot T\right),$$

with $\eta(T)$ controlling the approximate MIPS accuracy.

The per-step time complexity is $K^{1-\Theta(\frac{\eta(T)^4}{\log^2 T})+o(\log^{-0.45} K)}$. The overall time complexity overhead (e.g., initialization) is $K^{1+o(1)}$.

$\eta(T)$ offers a trade-off between complexity and regret. The following corollaries show examples of choosing $\eta(T)$.

Corollary 5.3. *Given any T that does not scale with K, one can choose $\eta(T) = \frac{1}{\sqrt{T}}$. The regret bound is $\widetilde{O}(d\sqrt{T})$, while the per-step complexity is $K^{1-\Theta(\frac{1}{T^2\log^2 T})}$ for sufficiently large K. Note that this achieves per-step complexity sublinear in K and retains the regret of $O(\sqrt{T})$.*

Corollary 5.4. *Consider the regime where K is extremely large and $T = \Theta\left(\log^\gamma K\right)$ for some constant γ. Choosing $\eta(T) = T^{-\frac{0.1}{\gamma}}$, the regret bound is $\widetilde{O}(T^{\frac{1}{2}} + T^{1-\frac{0.1}{\gamma}})$, while the per-step complexity is $o(K)$. It shows that it is possible to achieve both sublinear regret and sublinear time complexity, for any large K and moderate T.*

Algorithm 3 SUBLINEAR TIME ELIMINATION

Input: arm set $\mathcal{A}$, time horizon T, desired failure probability bound δ, desired accuracy $\eta(T)$
Initialize $V_1 = I, s = 1, \mathcal{A}_1 = \mathcal{A}$
Set $\beta(\frac{\delta}{2}) = 1 + \sqrt{2\log\left(\frac{2}{\delta}\right) + d\log\left(1+\frac{T}{d}\right)}$
Set $s_{max} = \left\lceil \log\frac{1}{8\eta(T)} \right\rceil$, initialize Ψ_s for $s \in [s_{max}]$ with $\left(s, d, \beta(\frac{\delta}{2}), \eta(T), \frac{\delta}{2s_{max}}\right)$
Add all arms $a \in \mathcal{A}$ to Ψ_0
for $t = 1, 2, \cdots, T$ **do**
  /* Add new arms $\mathcal{A}_{new}$ */
  For all $a \in \mathcal{A}_{new}$, add a to Ψ_s with $s = \min\left(\left\lfloor -\log\left(\beta(\frac{\delta}{2})\|x_a\|_{V_t^{-1}}\right)\right\rfloor, s_{max}\right)$
  Set $\underline{r}' = \max_{a\in\mathcal{A}_{new}}\left(x_{a_t}^\top\widehat{\theta}_t - \beta(\frac{\delta}{2})\|x_a\|_{V_t^{-1}}\right)$
  if $\underline{r}' > \underline{r}$ **then**
    Set $\underline{r} = \underline{r}'$. For all s, Ψ_s eliminates arms with $\underline{r}$
  end if
  /* Choose an arm in $o(K)$ time */
  Let $s_t = \text{argmin}_s\,|\Psi_s| > 0$
  if $s_t = \left\lceil \log\frac{1}{8\eta(T)} \right\rceil$ **then**
    Let a_t be an random arm in Ψ_{s_t}
  else
    Let a_t be the result of querying Ψ_{s_t} with V_t^{-1}
    while a_t is null **do**
      Set $\underline{r}' = \max_{a\in\Psi_{s_t}}\left(x_{a_t}^\top\widehat{\theta}_t - \beta(\frac{\delta}{2})\|x_a\|_{V_t^{-1}}\right)$
      For all $a \in \Psi_{s_t}$, add a to new set $\Psi_{s'}$ with $s' = \min\left(\left\lfloor -\log\left(\beta(\frac{\delta}{2})\|x_a\|_{V_t^{-1}}\right)\right\rfloor, s_{max}\right)$, and remove a from Ψ_{s_t}
      if $\underline{r}' > \underline{r}$ **then**
        $\underline{r} = \underline{r}'$. For all s, Ψ_s eliminate arms with $\underline{r}$
      end if
      Let $s_t = \text{argmin}_s\,|\Psi_s| > 0$
      Let a_t be the result of querying Ψ_{s_t} with V_t^{-1}
    end while
  end if
  Play arm a_t, observe reward r_t
  Update $V_{t+1} = V_t + x_{a_t}x_{a_t}^\top$
  Update $\widehat{\theta}_{t+1}$ according to Equation (1)
end for

One additional benefit of Algorithm 3 is that the elimination typically removes many arms, which provides further speedup. Such speedup does not show up in the theoretical analysis as it depends on the distribution of arms. The acceleration brought by elimination is clearly presented in our empirical evaluation (Section 7).

Such additional speedup, however, comes with the price

that the elimination-based algorithm can not handle deletions - as the remaining arms after elimination might get deleted from $\mathcal{A}$. In the next section, we present a sublinear time TS that allows for both additions and deletions.

6. Sublinear Time TS-based Algorithm

In this section, we present a Thompson Sampling (TS) based algorithm with sublinear per-step time complexity. It works for the general arm set changing, where arms can be added to or deleted from $\mathcal{A}$. The TS-based algorithm also avoids paying the overhead for elimination (as required by Algorithm 3), and therefore after initialization, the time complexity for every time step is sublinear in K.

6.1. Algorithm and its Regret, Time Complexity

The linear TS algorithm (Abeille & Lazaric, 2017) maintains the estimation $\widehat{\theta}_t$ as Equation (1). At each time step t, a random $\widetilde{\theta}_t$ is constructed as $\widetilde{\theta}_t = \widehat{\theta}_t + \beta(\frac{\delta}{4T})V_t^{-1/2}\xi_t$, with ξ_t drawn from distribution $\mathcal{D}^{TS}$, which satisfies *concentration* and *anti-concentration* properties (see Definition B.1 in Appendix). For instance, $\mathcal{D}^{TS}$ can simply be a spherical Gaussian distribution.

After $\widetilde{\theta}_t$ is constructed, the standard linear TS algorithm chooses the arm a that maximizes $x_a^\top \widetilde{\theta}_t$. Algorithm 1 can be naturally applied to solve this MIPS for arm selection. See Algorithm 4 for detail.

Notice that Algorithm 4 assumes that the largest reward expectation is non-negative, as it is more commonly seen (e.g., when the reward corresponds to clicks, purchases, or ratings). When the largest reward expectation is negative, we propose the following extension: we can transform arm's feature x to $\left[\frac{x}{\sqrt{2}}, \frac{\sqrt{2}}{2}\right]$, observed reward r_t to be $\frac{r_t}{2} + \frac{1}{2}$. The corresponding θ^* becomes $\left[\frac{\theta^*}{\sqrt{2}}, \frac{\sqrt{2}}{2}\right]$. In this way, the algorithm sees an environment with the largest reward expectation being positive and properly makes arm selection, while the true environment allows the largest reward expectation to be negative.

Under Assumption 4.1 to 4.3, we can characterize the regret and time complexity of Algorithm 4 as following:

Theorem 6.1 (Regret and time complexity of Algorithm 4, formal version see Theorem B.5)**.** *For any $\delta \in (0,1)$, with probability at least $1-\delta$, the regret is bounded by*

$$R(T) = \widetilde{O}\left(d^{3/2}\sqrt{T} + \eta(T)\cdot T\right),$$

with $\eta(T)$ controlling the approximate MIPS accuracy.

The per-step time complexity is $K^{1-\Theta(\eta(T)^2)+o\left(\log^{-0.45} K\right)}$. The time complexity of the data structure maintenance (line 4) is $K^{1+o(1)}$ which is paid once at initialization.

Algorithm 4 SUBLINEAR TIME THOMPSON SAMPLING

1: **Input:** arm set $\mathcal{A}$, time horizon T, desired failure probability bound δ, desired accuracy $\eta(T)$
2: Set $\beta(\frac{\delta}{4T}) = 1 + \sqrt{2\log\left(\frac{4T}{\delta}\right) + d\log\left(\frac{d+T}{d}\right)}, V_1 = I$
3: Preprocess $x_a, \forall a \in \mathcal{A}$ with Algorithm 1 with $\left\lceil \frac{d}{\eta(T)} \right\rceil$ independent copies. For the i-th copy $\mathcal{M}_i$, use parameter $\left(1 - \frac{1}{i+1}, \frac{i\cdot\eta(T)}{d}, \frac{\eta(T)}{d}, \frac{\delta\cdot\eta(T)}{2d}\right)$
4: Add all arms $a \in \mathcal{A}$ to all $\mathcal{M}_i$
5: **for** $t = 1, 2, \cdots, T$ **do**
6: Add or delete the changing arms a for all $\mathcal{M}_s$ with $s \le \lceil d/\eta(T)\rceil$
7: Sample $\xi_t \sim \mathcal{D}^{TS}$
8: Compute $\widetilde{\theta}_t = \widehat{\theta}_t + \beta(\frac{\delta}{4T})V_t^{-1/2}\xi_t$
9: Query Algorithm 1 with $\widetilde{\theta}_t/\|\widetilde{\theta}_t\|$ and different m, set a_t to be the non-null result with largest m
10: Play arm a_t and observe reward r_t
11: Update $V_{t+1} = V_t + x_t x_t^\top$
12: Update $\widehat{\theta}_{t+1}$ according to Equation (1)
13: **end for**

$\eta(T)$ offers a trade-off between complexity and regret. The following corollaries show examples of choosing $\eta(T)$.

Corollary 6.2. *For any T not scaling with K, one can choose $\eta(T) = \frac{1}{\sqrt{T}}$. The regret bound is $\widetilde{O}(d^{\frac{3}{2}}\sqrt{T})$, and the per-step complexity is $K^{1-\Theta(\frac{1}{T})}$ for sufficiently large K. Note that this retains the regret of the linear TS algorithm and achieves per-step complexity sublinear in K.*

Corollary 6.3. *Consider the regime where K is extremely large and $T = \Theta\left(\log^\gamma K\right)$ for some constant γ. Choosing $\eta(T) = T^{-\frac{0.2}{\gamma}}$, the regret bound is $\widetilde{O}(T^{\frac{1}{2}} + T^{1-\frac{0.2}{\gamma}})$, while the per-step complexity is $o(K)$.*

7. Experiments

In this section, we empirically evaluate the performance of our proposed algorithms in a synthetic environment and a real-world problem on movie recommendation.

We adopt the following algorithms for evaluation:

- **Sublinear Time Elimination (Sub-Elim)**: We implement Algorithm 3 and use HNSW algorithm (Malkov & Yashunin, 2018) as the MIPS solver in Algorithm 2.
- **Sublinear Time Thompson Sampling (Sub-TS)**: We implement Algorithm 4 with HNSW as the MIPS solver.
- **Baselines**: We implement the linear time version of Algorithms 3 and 4, where the MIPS step is solved by the standard linear scan through all the arms. Such baselines allow us to evaluate the performance and accelera-

		Linear Elim	Sub-Elim, shortlist 30	Linear TS	Sub-TS, shortlist 30
$K = 5,000$	Regret	3866 ± 195	3758 ± 190	582 ± 54	605 ± 59
	Time (s)	11.74	2.22 (1.99)	30.08	19.41 (19.29)
	Speedup	$\times 1$	$\times\mathbf{5.28}$ $(\times\mathbf{5.89})$	$\times 1$	$\times\mathbf{1.55}$ $(\times\mathbf{1.56})$
$K = 100,000$	Regret	4804 ± 146	4701 ± 150	721 ± 92	734 ± 89
	Time (s)	221.19	59.40 (3.04)	280.78	32.81 (29.10)
	Speedup	$\times 1$	$\times\mathbf{3.72}$ $(\times\mathbf{72.76})$	$\times 1$	$\times\mathbf{8.56}$ $(\times\mathbf{9.65})$

Table 1: **Synthetic Experiment - Impact of Different K.** "Linear Elim" and "Linear TS" are baselines. "Sub-Elim" and "Sub-TS" are Algorithms 3 and 4, with the shortlist being 30. "Regret" corresponds to the cumulative regret of 20,000 steps, with mean and standard deviation for 10 independent runs. The reported "Time" corresponds to the overall running time of 20,000 steps, averaged over 10 independent runs. The running time excluding preprocessing is reported in the bracket. The "Speedup" is the relative speedup compared with the corresponding baselines. The results demonstrate that Sub-Elim and Sub-TS can deliver significant speedup (e.g., a 72.76 times speedup, excluding preprocessing) especially when the number of arms K is large while obtaining a similar regret as the linear time baselines.

Algorithm	Linear Elim	Sub-Elim, shortlist 10	Sub-Elim, shortlist 100
Regret	4803 ± 146	4691 ± 133	4837 ± 143
Time(s)	221.19	59.07 (2.85)	59.84 (3.91)
Speedup	$\times 1$	$\times\mathbf{3.74}$ $(\times\mathbf{77.61})$	$\times\mathbf{3.69}$ $(\times\mathbf{56.57})$
Algorithm	Linear TS	Sub-TS, shortlist 10	Sub-TS, shortlist 100
Regret	721 ± 92	736 ± 89	721 ± 92
Time(s)	280.78	31.44 (27.75)	36.35 (32.64)
Speedup	$\times 1$	$\times\mathbf{8.93}$ $(\times\mathbf{10.12})$	$\times\mathbf{7.72}$ $(\times\mathbf{8.60})$

Table 2: **Synthetic Experiment - Impact of Approximation Precision.** The algorithms and "Regret", "Time" and "Speedup" are defined the same as in Table 1. Combining with the "shortlist 30" results in Table 1, it shows that a lager shortlist size p (corresponds to a smaller $\eta(T)$ in Algorithms 3 and 4) leads to longer running time. In our evaluated settings, all different shortlist sizes p are large enough to keep regret similar to the linear time baselines.

Algorithm	Linear Elim	Sub-Elim, shortlist 30	Sub-Elim, shortlist 100
Regret	3847 ± 212	3795 ± 206	3806 ± 206
Time(s)	29.55	4.22 (3.47)	4.83 (4.09)
Speedup	$\times 1$	$\times\mathbf{7.00}$ $(\times\mathbf{8.52})$	$\times\mathbf{6.12}$ $(\times\mathbf{7.22})$
Algorithm	Linear TS	Sub-TS, shortlist 30	Sub-TS, shortlist 100
Regret	1193 ± 66	1177 ± 66	1202 ± 68
Time(s)	29.83	19.59 (19.38)	20.63 (20.41)
Speedup	$\times 1$	$\times\mathbf{1.52}$ $(\times\mathbf{1.54})$	$\times\mathbf{1.45}$ $(\times\mathbf{1.46})$

Table 3: **Movie Recommendation - Running time and regret.** "Regret" corresponds to the cumulative regret of 20,000 recommendations, with mean and standard deviation for 300 users. "Time" is the total running time of making 20,000 recommendations, averaged over 300 users. The time excluding preprocessing is reported in the bracket. The results show that "Sub-Elim" is more than 7 times faster; and "Sub-TS" can reduce 30% of the baseline's running time. Both have similar regret as baselines.

tion brought by adopting an approximate MIPS solver.

LSH is not used for our implementation as there is currently no efficient LSH implementation that supports deletions. Note that this is purely an engineering issue - there exist LSH constructions that theoretically support efficient deletions (Andoni et al., 2017).

To control the tradeoff between MIPS accuracy and time complexity, we construct the MIPS solver in the following way: We first use the HNSW algorithm to retrieve a shortlist of p arms, then linearly scan the retrieved p arms for the one with the largest inner product. A larger p gives higher accuracy but slower speed. We take p from $\{10, 30, 100\}$ for our experiments. The choices of different p can be viewed as different $\eta(T)$ for Algorithms 3 and 4.

Synthetic Experiment For the synthetic experiment, we first randomly generated a 16-dimensional vector θ^* from a Gaussian distribution $\mathcal{N}(0, \mathbf{I}_{16})$. The arms $\mathcal{A}$ are generated from the same distribution. The reward noise is unit Gaussian. Further, over the time horizon $T = 20,000$, a batch of $C_{change} = 2$ arms are generated and included into the arm set $\mathcal{A}$ every 20 steps. The final arm set size is K.

Our first result (Table 1) demonstrates the efficiency of Sub-Elim and Sub-TS with different numbers of arms K. In particular, when the number of arms K is large, the Sub-Elim is able to deliver a 72.76 times speedup (excluding the preprocessing time) while retaining the regret of the linear time implementation.

We further evaluate the impact of different choices of shortlist size p (i.e., a larger p corresponding to a more accurate approximate MIPS solver) and the results are presented in Table 2. Moreover, we evaluate our algorithms with $C_{change} \in \{2, 10, 50\}$ and show that all our algorithms can deliver stable speedup in the evaluated settings. We also test Algorithm 4 when there are both additions and

deletions, which demonstrates a speedup and comparable regret as baselines. The results are deferred to Appendix C.

Movie Recommendation The testing environment is derived from a popular recommendation dataset: Movielens-1M (Harper & Konstan, 2015). The dataset contains over 1 million ratings of 3,952 movies by more than 6,000 users.

The environment construction is similar to (Qin et al., 2014). We preserve the ratings of 300 users (each with more than 100 ratings) for testing. With the ratings of more than 5,700 remaining users, we create a 16-dimensional feature for each of the movies by matrix factorization. The movies' features are used as arms' features ($\mathbf{x}_t(i)$).

The algorithm starts with $1,952$ movies, and interacts with the user for $20,000$ times (i.e., time horizon $T = 20,000$). 2 new movies are included for every 20 steps, which in the end leads to all 3,952 movies. In each time step, the regret is 1 if the recommended movie has a rating smaller than 4 or no rating, and otherwise, the regret is 0.

The average regret (and standard deviation) and the running time are reported in Table 3. Our empirical results demonstrate the acceleration and great empirical performance of the proposed sublinear time algorithms.

Acknowledgment

This work was partially supported by NSF grants 1564000, 1934932, 2019844, and 2107037, and the Machine Learning Lab at UT Austin.

References

Abbasi-Yadkori, Y., Pál, D., and Szepesvári, C. Improved algorithms for linear stochastic bandits. In Shawe-Taylor, J., Zemel, R., Bartlett, P., Pereira, F., and Weinberger, K. Q. (eds.), *Advances in Neural Information Processing Systems*, volume 24, pp. 2312–2320. Curran Associates, Inc., 2011.

Abeille, M. and Lazaric, A. Linear Thompson sampling revisited. In *20th International Conference on Artificial Intelligence and Statistics (AISTATS)*, Fort Lauderdale, FL, April 2017.

Abuzaid, F., Sethi, G., Bailis, P., and Zaharia, M. To index or not to index: Optimizing exact maximum inner product search. In *2019 IEEE 35th International Conference on Data Engineering (ICDE)*, pp. 1250–1261. IEEE, 2019.

Agrawal, S. and Goyal, N. Thompson sampling for contextual bandits with linear payoffs. In *International Conference on Machine Learning*, pp. 127–135, 2013.

Andoni, A. and Indyk, P. Near-optimal hashing algorithms for approximate nearest neighbor in high dimensions. In *2006 47th annual IEEE symposium on foundations of computer science (FOCS'06)*, pp. 459–468. IEEE, 2006.

Andoni, A., Laarhoven, T., Razenshteyn, I., and Waingarten, E. Optimal hashing-based time-space trade-offs for approximate near neighbors. In *Proceedings of the Twenty-Eighth Annual ACM-SIAM Symposium on Discrete Algorithms*, pp. 47–66. SIAM, 2017.

Andoni, A., Indyk, P., and Razenshteyn, I. Approximate nearest neighbor search in high dimensions. *arXiv*, pp. arXiv–1806, 2018.

Auer, P. Using confidence bounds for exploitation-exploration trade-offs. *Journal of Machine Learning Research*, 3(Nov):397–422, 2002.

Bachrach, Y., Finkelstein, Y., Gilad-Bachrach, R., Katzir, L., Koenigstein, N., Nice, N., and Paquet, U. Speeding up the Xbox recommender system using a Euclidean transformation for inner-product spaces. In *Proceedings of the 8th ACM Conference on Recommender systems*, pp. 257–264, 2014.

Chen, B., Medini, T., Farwell, J., Gobriel, S., Tai, C., and Shrivastava, A. Slide: In defense of smart algorithms over hardware acceleration for large-scale deep learning systems. *arXiv preprint arXiv:1903.03129*, 2019a.

Chen, B., Xu, Y., and Shrivastava, A. Fast and accurate stochastic gradient estimation. 2019b.

Chen, B., Liu, Z., Peng, B., Xu, Z., Li, J. L., Dao, T., Song, Z., Shrivastava, A., and Re, C. Mongoose: A learnable lsh framework for efficient neural network training. In *OpenReview. net. Retrieved from https://openreview. net/forum*, 2020.

Chu, W., Li, L., Reyzin, L., and Schapire, R. Contextual bandits with linear payoff functions. In *Proceedings of the Fourteenth International Conference on Artificial Intelligence and Statistics*, pp. 208–214, 2011.

Dani, V., Hayes, T. P., and Kakade, S. M. Stochastic linear optimization under bandit feedback. 2008.

Deshpande, Y. and Montanari, A. Linear bandits in high dimension and recommendation systems. In *2012 50th Annual Allerton Conference on Communication, Control, and Computing (Allerton)*, pp. 1750–1754. IEEE, 2012.

Ding, Q., Yu, H.-F., and Hsieh, C.-J. A fast sampling algorithm for maximum inner product search. In *The 22nd International Conference on Artificial Intelligence and Statistics*, pp. 3004–3012, 2019.

Gopalan, A., Mannor, S., and Mansour, Y. Thompson sampling for complex online problems. In Xing, E. P. and Jebara, T. (eds.), *Proceedings of the 31st International Conference on Machine Learning*, volume 32 of *Proceedings of Machine Learning Research*, pp. 100–108, Bejing, China, Jun 2014. PMLR.

Guo, R., Kumar, S., Choromanski, K., and Simcha, D. Quantization based fast inner product search. In *Artificial Intelligence and Statistics*, pp. 482–490, 2016.

Har-Peled, S., Indyk, P., and Motwani, R. Approximate nearest neighbor: Towards removing the curse of dimensionality. *Theory of computing*, 8(1):321–350, 2012.

Harper, F. M. and Konstan, J. A. The movielens datasets: History and context. *Acm transactions on interactive intelligent systems (tiis)*, 5(4):1–19, 2015.

Jun, K.-S., Bhargava, A., Nowak, R., and Willett, R. Scalable generalized linear bandits: Online computation and hashing. In *Advances in Neural Information Processing Systems*, pp. 99–109, 2017.

Kaufmann, E., Korda, N., and Munos, R. Thompson sampling: An asymptotically optimal finite-time analysis. In Bshouty, N. H., Stoltz, G., Vayatis, N., and Zeugmann, T. (eds.), *Algorithmic Learning Theory*, pp. 199–213, Berlin, Heidelberg, 2012. Springer Berlin Heidelberg. ISBN 978-3-642-34106-9.

Keivani, O., Sinha, K., and Ram, P. Improved maximum inner product search with better theoretical guarantee using randomized partition trees. *Machine Learning*, 107 (6):1069–1094, 2018.

Kitaev, N., Kaiser, Ł., and Levskaya, A. Reformer: The efficient transformer. *arXiv preprint arXiv:2001.04451*, 2020.

Lattimore, T., Szepesvari, C., and Weisz, G. Learning with good feature representations in bandits and in rl with a generative model. In *International Conference on Machine Learning*, pp. 5662–5670. PMLR, 2020.

Li, H., Chan, T. N., Yiu, M. L., and Mamoulis, N. Fexipro: fast and exact inner product retrieval in recommender systems. In *Proceedings of the 2017 ACM International Conference on Management of Data*, pp. 835–850, 2017.

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

Malkov, Y. A. and Yashunin, D. A. Efficient and robust approximate nearest neighbor search using hierarchical navigable small world graphs. *IEEE transactions on pattern analysis and machine intelligence*, 42(4):824–836, 2018.

Morozov, S. and Babenko, A. Non-metric similarity graphs for maximum inner product search. *Advances in Neural Information Processing Systems*, 31:4721–4730, 2018.

Neyshabur, B. and Srebro, N. On symmetric and asymmetric LSHs for inner product search. In *International Conference on Machine Learning*, pp. 1926–1934. PMLR, 2015.

Qin, L., Chen, S., and Zhu, X. Contextual combinatorial bandit and its application on diversified online recommendation. In *Proceedings of the 2014 SIAM International Conference on Data Mining*, pp. 461–469. SIAM, 2014.

Russo, D. and Van Roy, B. Learning to optimize via posterior sampling. *Mathematics of Operations Research*, 39 (4):1221–1243, 2014.

Russo, D. and Van Roy, B. An information-theoretic analysis of Thompson Sampling. *The Journal of Machine Learning Research*, 17(1):2442–2471, 2016.

Shen, F., Liu, W., Zhang, S., Yang, Y., and Tao Shen, H. Learning binary codes for maximum inner product search. In *Proceedings of the IEEE International Conference on Computer Vision*, pp. 4148–4156, 2015.

Shrivastava, A. and Li, P. Asymmetric LSH (ALSH) for sublinear time maximum inner product search (MIPS). *Advances in neural information processing systems*, 27: 2321–2329, 2014.

Shrivastava, A., Song, Z., and Xu, Z. Sublinear least-squares value iteration via locality sensitive hashing. *arXiv preprint arXiv:2105.08285*, 2021.

Song, Z., Yang, S., and Zhang, R. Does preprocessing help training over-parameterized neural networks? *Advances in Neural Information Processing Systems*, 34, 2021a.

Song, Z., Zhang, L., and Zhang, R. Training multi-layer over-parametrized neural network in subquadratic time. *arXiv preprint arXiv:2112.07628*, 2021b.

Song, Z., Xu, Z., and Zhang, L. Speeding up sparsification using inner product search data structures. *arXiv preprint arXiv:2204.03209*, 2022.

Spring, R. and Shrivastava, A. Scalable and sustainable deep learning via randomized hashing. In *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pp. 445–454, 2017.

Tan, S., Zhou, Z., Xu, Z., and Li, P. On efficient retrieval of top similarity vectors. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pp. 5239–5249, 2019.

Teflioudi, C., Gemulla, R., and Mykytiuk, O. LEMP: Fast retrieval of large entries in a matrix product. In *Proceedings of the 2015 ACM SIGMOD International Conference on Management of Data*, pp. 107–122, 2015.

Todd, M. J. *Minimum-volume ellipsoids: Theory and algorithms*. SIAM, 2016.

Xu, Z., Song, Z., and Shrivastava, A. Breaking the linear iteration cost barrier for some well-known conditional gradient methods using maxip data-structures. *Advances in Neural Information Processing Systems*, 34, 2021.

Yan, X., Li, J., Dai, X., Chen, H., and Cheng, J. Norm-ranging LSH for maximum inner product search. *Advances in Neural Information Processing Systems*, 31: 2952–2961, 2018.

Yang, S., Shen, Y., and Sanghavi, S. Interaction hard thresholding: Consistent sparse quadratic regression in sub-quadratic time and space. *arXiv preprint arXiv:1911.03034*, 2019.

Yu, H.-F., Hsieh, C.-J., Lei, Q., and Dhillon, I. S. A greedy approach for budgeted maximum inner product search. In *Advances in Neural Information Processing Systems*, pp. 5453–5462, 2017.

Zhou, Z., Tan, S., Xu, Z., and Li, P. Möbius transformation for fast inner product search on graph. In *NeurIPS*, pp. 8216–8227, 2019.

A. Proof for Section 3

A.1. Proof of Proposition 3.2

Andoni et al. (2017) proposed a data structure that solves the approximate nearest neighbor (ANN) search problem. We therefore first present a transformation, which converts a MIPS problem into the nearest neighbor search problem. The transformation is proposed in (Bachrach et al., 2014).

A (c', r')-approximate nearest neighbor search problem aims to find $p' \in P' \subseteq \mathbb{R}^{d+3}$ for a query $q' \in \mathbb{R}^{d+3}$ such that $\|p' - q'\|_2 \leq c'r'$, if there exists $\widetilde{p} \in P'$ such that $\|\widetilde{p} - q'\|_2 \leq r'$. Recall that for MIPS problem, we have $\|p\|_2 \leq 1, \forall p \in P$ and $\|q\|_2 \leq 1$ by Definition 3.1. We take the following transformation

$$p' = \left[\frac{1}{2}; \frac{p}{2}; \sqrt{\frac{3 - \|p\|_2^2}{4}}; 0\right] \in \mathbb{R}^{d+3}, \forall p \in P;$$
$$q' = \left[\frac{1}{2}; \frac{q}{2}; 0; \sqrt{\frac{3 - \|q\|_2^2}{4}}\right] \in \mathbb{R}^{d+3}.$$

Let $P' = \{p' \mid \forall p \in P\}$. Then for any point $p' \in P'$ and any query q', we have

$$\begin{aligned}\|p' - q'\|_2^2 &= \|p'\|_2^2 + \|q'\|_2^2 - 2\langle p', q'\rangle \\ &= \frac{3 - \langle p, q\rangle}{2}.\end{aligned}$$

Therefore the original $(c, r, 0)$-MIPS is equivalent to (c', r')-ANN with $c' = \sqrt{3 - cr}/\sqrt{3 - r}$ and $r' = \sqrt{\frac{3-r}{2}}$. For $c \in [0, 1)$ and $r \in (0, 1]$, we have $r' \in [1, \sqrt{3/2}), c'r' \in (1, \sqrt{3/2}]$ and $c' \in (1, \sqrt{3/2}]$.

Andoni et al. (2017) constructed a data structure that solves (c', r')-ANN with constant success probability. It has $K^{1+o(1)}$ preprocessing time complexity, $K^{\rho_q + o(1)}$ query time complexity and $K^{o(1)}$ time complexity for adding a new point to P. The rest of our proof follows the same procedure as (Andoni et al., 2017), but aims to give a more explicit characterization for the $o(1)$ term in the query time complexity, which turns out to be $o\left(\log^{-0.45} K\right)$. This more explicit form is useful when ρ_q is very close to 1 (i.e., $\rho_q = 1 - o(1)$).

Short description of the data structure construction.

The proposed data structure stores all data points P in a tree, with depth M and branching factor at most B.

During preprocessing, each tree node n draws a unit norm vector u_n uniformly at random. Each point $p \in P$ will traverse down the tree from the root, the point p will descend through a node n if the inner product $\langle p, u_n\rangle \geq \eta_u$, where η_u is a scalar parameter that is shared for the entire tree (i.e., all nodes use the same η_u). The point can descend through multiple nodes at the same level, and will possibly reach multiple leave nodes in the end. The leave nodes will store all the points p that reached it during preprocessing.

During query time, a query q will also descend from the root, and go down through the nodes with $\langle q, u_n\rangle \geq \eta_q$. Similar to a point p in the preprocessing stage, the query q will possibly reach multiple leave nodes in the end. It will then linearly scan through all the points p stored in the corresponding leave nodes. It will stop scanning and return the first point p that solves the (c', r')-ANN problem.

We will omit much detail of the proof but highlight the difference. One can check the full proof in Section 3.3.3 of (Andoni et al., 2017). Define $F(\eta) := \mathbb{P}_{z \sim N(0,1)^d}[\langle z, u\rangle \geq \eta]$, where u is an arbitrary point on the unit sphere. Define $G(s, \eta, \sigma) = \mathbb{P}_{z \sim N(0,1)^d}[\langle z, u\rangle \geq \eta$ and $\langle z, v\rangle \geq \sigma]$, where u, v are two points on the unit sphere with $\|u - v\|_2 = s$.

Preprocessing time complexity

We now prove the preprocessing time complexity. Notice that $F(\eta_u)$ is the probability of one point $p \in P$ descends from a node to a child node.

Lemma A.1. *The data structure construction has the following complexity in expectation:*

$$\textbf{\textit{Time :}}\ K^{1+o(1)} \cdot B \cdot \sum_{i=0}^{M} \cdot (B \cdot F(\eta_u))^i .$$

$$\textbf{\textit{Space :}}\ K^{1+o(1)} \cdot M \cdot (B \cdot F(\eta_u))^M .$$

Proof. The analysis for space complexity is presented in Lemma 3.7 in (Andoni et al., 2017). We show the time complexity analysis in a similar way.

In the preprocessing of P, a point $p \in P$ in expectation descends to $B \cdot F(\eta_u)$ nodes from one node. Thus the expected number of points at depth-i is $K \cdot (B \cdot F(\eta_u))^i$. Each point in at one node incurs a time complexity of $B \cdot K^{o(1)}$. In our regime of interest, K is extremely large and we treat the dimension d as $K^{o(1)}$.

There is an over-estimation at the depth-M node. Since there is no further branching in such nodes, each point at one depth-M node only incurs $K^{o(1)}$ time complexity, instead of $B \cdot K^{o(1)}$. This over-estimation does not hurt further analysis. ■

Next, we show that the preprocessing time complexity is the same as the space complexity.

Lemma A.2. *Both time and space compelxity of data structure construction are* $K^{1+o(1)} \cdot (B \cdot F(\eta_u))^M$.

Proof. As suggested in (Andoni et al., 2017), we set $M = \sqrt{\log K}$, which immediately implies $K^{1+o(1)} \cdot (B \cdot F(\eta_u))^M$ space complexity.

For time complexity, we have

$$\sum_{i=0}^{M} (B \cdot F(\eta_u))^i = O(1)(B \cdot F(\eta_u))^M.$$

This follows from $F(\eta_u) \geq G(r, \eta_u, \eta_q)$, and thus $B \cdot F(\eta_u) \geq B \cdot G(r, \eta_u, \eta_q)$, where in the analysis of (Andoni et al., 2017) it sets $B \cdot G(r, \eta_u, \eta_q) \geq 100$. In the proof of the optimal ρ_u, ρ_q trade-off by Andoni et al. (2017), it showed that when $\rho_u = 0$ (which is the setting we adopted), the specified M, B leads to

$$B^M = K^{c_0}.$$

The c_0 is a constant not depending on K. For $M = \sqrt{\log K}$, we have $B = K^{\frac{c'}{M}} = K^{o(1)}$. Putting these together, we have the time complexity to be $K^{1+o(1)}(B \cdot F(\eta_u))^M$. ■

With the choice of $\eta_u = 0$, the complexity $K^{1+o(1)}(B \cdot F(\eta_u))^M$ is $K^{1+o(1)}$ (see detailed proof in Section 3.3.3 (Andoni et al., 2017)). It therefore achieves the $K^{1+o(1)}$ time complexity.

Query time complexity

Here we show the $K^{\rho_q + o\left(\log^{-0.45} K\right)}$ query complexity extended from (Andoni et al., 2017), where the query complexity was presented as $K^{\rho_q + o(1)}$. Note that this is not an improvement over the original analysis. We are only more explicit about the $o(1)$ term which is necessary for our case.

During query time, the query q recursively descends from the root, with each descending happening with probability $F(\eta_q)$. According to the Lemma 3.8 of (Andoni et al., 2017), the query time complexity is

$$d \cdot B \cdot (B \cdot F(\eta_q))^M + K \cdot d \cdot (B \cdot G(c'r', \eta_u, \eta_q))^M,$$

where $F(\eta_q)$ denotes the probability of the query descending from one node to one of its child node, and $G(c'r', \eta_u, \eta_q)$ denotes the probability of a query and a qualifying point (i.e., distance smaller than $c'r'$) in P both descending to a child node. In the proof of query time complexity, Andoni et al. (2017) take that $F(\eta_u)^M = K^{-\sigma}$ and $F(\eta_q)^M = K^{-\tau}$.

We first present a stronger version of Lemma 3.1 in (Andoni et al., 2017),

$$F(\eta_q) = e^{-(1+o(\eta_q^{-9/5}))\cdot \frac{\eta_q^2}{2}},$$

where the original Lemma 3.1 states $F(\eta_q) = e^{-(1+o(1))\cdot\frac{\eta_q^2}{2}}$. This stronger version of Lemma 3.1 follows immediately from a tight Gaussian tail bound. As $\eta_q \to \infty$, we have

$$\left(\frac{1}{\eta_q} - \frac{1}{\eta_q^3}\right)\frac{e^{-\eta_q^2/2}}{\sqrt{2\pi}} \le F(\eta_q) \le \frac{1}{\eta_q}\cdot\frac{e^{-\eta_q^2/2}}{\sqrt{2\pi}}.$$

Follow the same analysis as in (Andoni et al., 2017), the stronger version of Lemma 3.1 implies a stronger version of Lemma 3.2 of (Andoni et al., 2017),

$$G(c'r', \eta_u, \eta_q) = e^{-(1+o(\eta_q^{-9/5}))\cdot\frac{\eta_u^2+\eta_q^2-2\alpha(c'r')\eta_u\eta_q}{2\beta^2(c'r')}},$$

where $\alpha(s) = 1 - \frac{s^2}{2}$ is the cosine of the angle between two points on a unit Euclidean sphere with distance s between them, and $\beta(s) = \sqrt{1-\alpha^2(s)}$ is the sine of the same angle. Note that the original Lemma 3.2 is $G(s, \eta, \sigma) = e^{-(1+o(1))\cdot\frac{\eta^2+\sigma^2-2\alpha(s)\eta\sigma}{2\beta^2(s)}}$. By requiring that $F(\eta_q)^M = K\cdot G(c'r', \eta_u, \eta_q)^M$, as $\eta_q \to \infty$, we have

$$\frac{\sigma+\tau-2\alpha(c'r')\cdot\sqrt{\sigma\tau}}{\beta^2(c'r')} - 1 = \left(1 + o\left(\eta_q^{-9/5}\right)\right)\tau.$$

As suggested in (Andoni et al., 2017), to have $\eta_u = 0$, we should set $\sqrt{\tau} = \frac{\alpha(r')\beta(c'r')}{1-\alpha(r')\alpha(c'r')}$. With the transformation (from MIPS to ANN) proposed previously, we have $r' \in [1, \sqrt{3/2}), c'r' \in (1, \sqrt{3/2}]$. Therefore τ is bounded by constants as $\tau \in [0.06, 0.34]$, and we have

$$\tau = \frac{\sigma+\tau-2\alpha(c'r')\cdot\sqrt{\sigma\tau}}{\beta^2(c'r')} - 1 + o\left(\eta_q^{-9/5}\right).$$

Notice that with $F(\eta_q)^M = K^{-\tau}$ and τ bounded by constants, we have $\eta_q = \Omega(\log^{1/4} K)$ and therefore,

$$\tau = \frac{\sigma+\tau-2\alpha(c'r')\cdot\sqrt{\sigma\tau}}{\beta^2(c'r')} - 1 + o\left(\log^{-0.45} K\right). \tag{2}$$

In the original analysis by (Andoni et al., 2017), the result was

$$\tau = \frac{\sigma+\tau-2\alpha(c'r')\cdot\sqrt{\sigma\tau}}{\beta^2(c'r')} - 1 + o\left(1\right).$$

Therefore from Equation (2), we have that, up to $o\left(\log^{-0.45} K\right)$ terms,

$$\sqrt{\sigma} = \alpha(c'r')\sqrt{\tau} + \beta(c'r')$$

Further, with $r' \in [1, \sqrt{3/2}), c'r' \in (1, \sqrt{3/2}]$, we have the following term also bounded by constants:

$$\frac{\sigma+\tau-\alpha(r')\sqrt{\sigma\tau}}{\beta(r')^2} \in [0.97, 1.34].$$

The rest of analysis follows the same as Section 3.3.3 in (Andoni et al., 2017), with all $o(1)$ replaced by $o(\log^{-0.45} K)$. As a result, the query time is $K^{\rho_q + o\left(\log^{-0.45} K\right)}$.

Time complexity for adding a new point to P

Adding a point to the data structure takes $K^{o(1)}\cdot B\cdot\sum_{i=0}^{M}\cdot(B\cdot F(\eta_u))^i$ time. We have proven in the **Preprocessing time complexity** that it is $K^{o(1)}$ under the choice of $\eta_u = 0$. Therefore the complexity of adding a new point is $K^{o(1)}$.

A.2. Proof of Theorem 3.3

Proof. Denote Q to be the unit l_2 ball in $\mathbb{R}^d$ centered at 0. We have $q_t \in Q, \forall t \in [T]$. We can discretize Q into lattice $\widehat{Q}$ with precision $\frac{\epsilon}{d}$. Note that every point in $\widehat{Q}$ has all its coordinates being multiples of $\frac{\epsilon}{d}$. We can then bound the size of $\widehat{Q}$ to be $\left|\widehat{Q}\right| \leq \left(\frac{2d}{\epsilon}\right)^d$.

The probability of all κ copies of $\mathcal{S}(c, r, 0)$ fail for any $\widehat{q} \in \widehat{Q}$ and any point $p \in p$ is

$$\mathbb{P}\left(\exists \widehat{q} \in \widehat{Q}, p \in p \text{ s.t. all } \mathcal{S}(c, r, 0, \delta) \text{ fail on } p, \widehat{q}\right) \leq K\left(\frac{2d}{\epsilon}\right)^d 0.1^{\kappa} \leq \delta.$$

the last inequality follows from $\kappa = d \log\left(\frac{Kd}{\epsilon\delta}\right) \geq \log\left(\frac{1}{K}\left(\frac{2d}{\epsilon}\right)^{-d}\delta\right) / \log(0.1)$.

For any query $q \in Q$, rounding it to the nearest point $\widehat{q} \in \widehat{Q}$, it induces ϵ additive error for inner product (recall that $\|p\| \leq 1, \forall p \in P$). Thus, for arbitrary query sequence from Q, running κ copies of $\mathcal{S}(c, r, 0)$ solves (c, r, ϵ)-MIPS problem successfully for all the queries with probability at least $1 - \delta$. This completes the proof. ■

B. Proof for Sections 5 and 6

B.1. Definition of TS distribution

Definition B.1 (Abeille & Lazaric (2017)). $\mathcal{D}^{TS}$ is a multivariate distribution on $\mathbb{R}^d$ absolutely continuous with respect to the Lebesgue measure which satisfies the following properties:

1. (anti-concentration) there exists a strictly positive probability p such that for any $u \in \mathbb{R}^d$ with $\|u\|_2 = 1$,

$$\mathbb{P}_{\xi \sim \mathcal{D}^{TS}}\left(u^\top \xi \geq 1\right) \geq p.$$

2. (concentration) there exists b, b' positive constants such that $\forall \delta \in (0, 1)$

$$\mathbb{P}_{\xi \sim \mathcal{D}^{TS}}\left(\|\xi\| \leq \sqrt{bd \log \frac{b'd}{\delta}}\right) \geq 1 - \delta.$$

B.2. Technical Lemma

We first present 2 previously established supporting lemmas on bounding $\|\widehat{\theta}_t - \theta\|_{V_t}$ and $\sum_{t=1}^T \|x_t\|_{V_t^{-1}}$, which are useful for proving Theorems 5.2 and 6.1.

Lemma B.2 (Thm. 2 of (Abbasi-Yadkori et al., 2011)). *With Assumption 4.1 to 4.3, for the $\widehat{\theta}_t$ estimation according to Equation (1) and for any $\delta > 0$, with probability at least $1 - \delta$ for all $t \geq 0$, we have*

$$\|\widehat{\theta}_t - \theta^*\|_{V_t} \leq 1 + \sqrt{2 \log\left(\frac{1}{\delta}\right) + d \log\left(1 + \frac{T}{d}\right)}.$$

Lemma B.3 (Lemma 4 of (Abbasi-Yadkori et al., 2011)). *Let $\{x_t\}$ be a sequence in $\mathbb{R}^d$. For $V_t = I + \sum_{s=1}^{t-1} x_s x_s^\top$, we have*

$$\sum_{t=1}^T \|x_t\|^2_{V_t^{-1}} \leq 2d \log\left(1 + \frac{T}{d}\right).$$

B.3. Proof of Lemma 5.1

Proof. Let a_t^* be the optimal arm (whose feature is $x_{a_t^*}$) at time t and suppose Ψ_{s^*} is the set that contains a^*. Let t_1 be the time step that a_t^* is placed to Ψ_{s^*}, and t_2 is the time that the played arm a is placed in Ψ_{s_t}, we have

$$\begin{aligned}
x_{a_t^*}^\top \theta^* &\le x_{a_t^*}^\top \widehat{\theta}_{t_1} + \beta(\delta)\|x_{a_t^*}\|_{V_{t_1}^{-1}} \\
&\le x_{a_t^*}^\top \widehat{\theta}_{t_1} + 2^{-s^*} \\
&\overset{(a)}{\le} \underline{r} + 2^{-s^*} + 2^{-s^*} \\
&\overset{(b)}{\le} x_a^\top \widehat{\theta}_{t_2} + 2^{-s_t} + 2\cdot 2^{-s^*} \\
&\le x_a^\top \theta^* + \beta(\delta)\|x_a\|_{V_{t_2}^{-1}} + 2^{-s_t} + 2\cdot 2^{-s^*} \\
&\le x_a^\top \theta^* + 2\cdot 2^{-s_t} + 2\cdot 2^{-s^*} \\
&\overset{(c)}{\le} x_a^\top \theta^* + 4\cdot 2^{-s_t}.
\end{aligned}$$

Inequality (a) holds as $\underline{r} \ge x_{a_t^*}^\top \widehat{\theta}_{t_1} - 2^{-s^*}$ (since $\underline{r}$ is always greater than $x_{a_t^*}^\top \widehat{\theta}_{t_1} - 2^{-s^*}$ after a_t^* advances to a Ψ_{s^*}); inequality (b) holds as arm a is not eliminated from Ψ_{s_t}; inequality (c) holds as $s_t \le s^*$. This completes the proof. ■

B.4. Proof of Theorem 5.2

We first state the formal version of Theorem 5.2.

Theorem B.4 (Formal version of Theorem 5.2)**.** *For any $\delta \in (0,1)$, with probability at least $1-\delta$, the regret of Algorithm 3 is bounded by*

$$R(T) \le 16\beta(\delta/2)\sqrt{Td\log\left(1+\frac{T}{d}\right)} + 64\eta(T)\cdot T,$$

with $\beta(\delta/2) = 1 + \sqrt{2\log\left(\frac{2}{\delta}\right) + d\log\left(1+\frac{T}{d}\right)}$. $\eta(T) \in (0,1)$ controls the approximate MIPS accuracy.

The per-step time complexity is $K^{1-\Theta(\frac{\eta(T)^4}{\log^2 T})+o(\log^{-0.45} K)}$. The overall time complexity overhead (e.g., initialization) is $K^{1+o(1)}$.

Proof of Theorem B.4 - time complexity We break the time complexity into two parts:

Overhead for maintaining Ψ_s: This part contains the overhead induced by maintaining Ψ_s, which includes lines 5, 8-11, 19-24 of Algorithm 3. Line 5 is intializing all the initial K arms, which takes $O(K\log K)$ for the heap related operations, and $O(\kappa\cdot K^{1+o(1)})$ time to add all $a\in\mathcal{A}$ to the adaptive MIPS solver $\mathcal{M}_0$. For lines 8-9 and 19-20, it only happens when an arm a needs to be added (or advanced) to another Ψ_s. Notice that each arm can only be added (or advanced) to a Ψ_s for $\left\lceil \log\frac{1}{8\eta(T)}\right\rceil + 1$ times. Therefore all the arms in total will induce an $\kappa\cdot K^{1+o(1)}\cdot\log\frac{1}{8\eta(T)}$ time complexity in overhead. Further for line 10-11 and 21-24, it only happens when an arm needs to be eliminated. Both the heap $\mathcal{H}_s$ and the adaptive MIPS solver $\mathcal{M}_s$ need to be updated, which in total induces an $\kappa\cdot K^{1+o(1)}+O(K\cdot\log K)$ overhead for all the arms (since all the arms can only be eliminated once). The overall overhead complexity is therefore $\kappa\cdot K^{1+o(1)}\cdot\log\frac{1}{8\eta(T)}+O(K\cdot\log K)$, rearranging the terms gives $K^{1+o(1)}$.

Time complexity for selecting an arm: This includes lines 18 and 26. With the construction of the adaptive MIPS solver $\mathcal{M}$ (Algorithm 1), the query time complexity is given by $\kappa\cdot K^{\rho_q+o(\log^{-0.45} K)}$. Plug in $(c,r,\epsilon) = (1/4, \frac{2^{-2s}(1-\eta(T)^2)}{d^2\beta(\delta/2)^2}, \frac{2^{-2s}\eta(T)^2}{d^2\beta(\delta/2)^2})$ and $s \le \left\lceil \log\frac{1}{8\eta(T)}\right\rceil$, we have $\kappa = K^{O(\frac{1}{\sqrt{\log K}})}$, and $\rho_q = \frac{4c'^2}{(1+c'^2)^2}$, with $c' = 1+\Theta(\frac{\eta(T)^2}{\log T})$ (see Theorem 3.3). Thus we have $\rho_q = 1-\Theta(\frac{\eta(T)^4}{\log^2 T})$, which gives the per-step time complexity $K^{1-\Theta(\frac{\eta(T)^4}{\log^2 T})+o(\log^{-0.45} K)}$.

Therefore the overhead is $K^{1+o(1)}$ and the per-step complexity is $K^{1-\Theta(\frac{\eta(T)^4}{\log^2 T})+o(\log^{-0.45} K)}$.

Proof of Theorem 5.2 - regret bound

Proof. With failure probability being $\frac{\delta}{2s_{max}}$ for Algorithm 1 and $\beta(\delta/2)$ in Algorithm 3, with probability at least $1-\delta$, all queries to Algorithm 1 (line 18 and line 26 of Algorithm 3) are answered correctly and Lemma B.2 holds for all $t \in [T]$. Conditioning on those success events, we proceed to the regret bound.

Suppose at time t, the played arm a_t is chosen from set Ψ_{s_t}. We have that

$$\beta(\delta/2)^2 \left\langle vec(x_{a_t} x_{a_t}^\top), vec(V_t^{-1}) \right\rangle \leq 2^{-2s_t}.$$

By $Lemma$ 5.1, we know that

$$x_{a_t^*}^\top \theta^* - x_{a_t}^\top \theta^* \leq 4 \cdot 2^{-s_t}$$

When $s_t = \left\lceil \log \frac{1}{8\eta(T)} \right\rceil$, we have $2^{-s_t} \leq 8\eta(T)$

$$x_{a_t^*}^\top \theta^* - x_{a_t}^\top \theta^* \leq 32\eta(T) \tag{3}$$

For stage $s_t < \left\lceil \log \frac{1}{8\eta(T)} \right\rceil$, since the action a_t is the result of querying $\mathcal{M}_{s_t}$, we have

$$\beta(\delta/2)^2 \|x_{a_t}\|_{V_t^{-1}}^2 \geq \frac{1}{4} \cdot 2^{-2s_t-2} - \frac{5}{4}\eta(T)^2 \implies \beta(\delta/2)\|x_{a_t}\|_{V_t^{-1}} \geq 2^{-s_t-2} - 2\eta(T),$$

where we used the fact that $2^{-s_t-2} \geq 2\eta(T)$ and $\sqrt{a-b} \geq \sqrt{a} - \sqrt{b}$ for all $a \geq b$. Combining the results, we have

$$x_{a_t^*}^\top \theta^* - x_{a_t}^\top \theta^* \leq 4 \cdot 2^{-s_t} \leq 16\beta(\delta/2)\|x_{a_t}\|_{V_t^{-1}} + 32\eta(T). \tag{4}$$

Combining Equations (3) and (4) and summing over t, we have

$$\begin{aligned} R(T) &\leq 16\beta(\delta/2) \sum_{t=1}^{T} \|x_{a_t}\|_{V_t^{-1}} + 32\eta(T)T + 32\eta(T)T \\ &\leq 16\beta(\delta/2)\sqrt{Td\log\left(1+\frac{T}{d}\right)} + 64\eta(T)T. \end{aligned}$$

The second inequality is by Lemma B.3. This completes the proof.

■

B.5. Proof of Theorem 6.1

We first state the formal version of Theorem 6.1.

Theorem B.5 (Formal version of Theorem 6.1)**.** *For any $\delta \in (0,1)$, with probability at least $1-\delta$, the regret of Algorithm 4 is bounded by*

$$\begin{aligned} R(T) \leq & \frac{4\gamma(\delta/4T)}{p}\left(\sqrt{2Td\log\left(1+\frac{T}{d}\right)} + \sqrt{8T\log\frac{4}{\delta}}\right) \\ & + (\gamma(\delta/4T) + \beta(\delta/4T))\sqrt{2Td\log\left(1+\frac{T}{d}\right)} \\ & + \frac{6(1+\gamma(\delta/4T)+\beta(\delta/4T))}{p} \cdot \frac{\eta(T)}{d} \cdot T, \end{aligned}$$

where $\beta(\delta/4T) = 1 + \sqrt{2\log\frac{4T}{\delta} + d\log\left(1+\frac{T}{d}\right)}$, $\gamma(\delta/4T) = \beta(\delta/4T)\sqrt{bd\log\frac{b'd}{\delta/4T}}$, with b, b', p are constants defined in Definition B.1. $\eta(T) \in (0,1)$ controls the approximate MIPS accuracy.

The per-step time complexity is $K^{1-\Theta(\eta(T)^2)+o(\log^{-0.45} K)}$. The time complexity of the data structure maintenance (line 4) is $K^{1+o(1)}$ which is paid once at initialization.

Proof of Theorem B.5 - time complexity We first show the per-step complexity. The per-step time complexity is $\kappa K^{\rho_q+o(\log^{-0.45} K)} \log \frac{d}{\eta(T)}$, as each query to Algorithm 1 has complexity $\kappa K^{\rho_q+o(\log^{-0.45} K)}$ and line 8 of Algorithm 4 requires a binary search which induces another factor of $\log \frac{d}{\eta(T)}$. By setting $(c, r, \epsilon) = (1 - \frac{1}{i+1}, \frac{i\cdot\eta(T)}{d}, \frac{\eta(T)}{d})$, we have $\kappa = O(\log KT) = K^{O(\frac{1}{\sqrt{\log K}})}$ and $\rho = \frac{4c'^2}{(1+c'^2)^2}$, with $c' = 1 + \Theta(\eta(T))$ (see Theorem 3.3). It then implies $\rho_q = 1 - \Theta(\eta(T)^2)$, which corresponds to the per-step time complexity in Theorem 6.1. Notice that for Line 6 in Algorithm 4, adding new arms to and deleting arms from all $\mathcal{M}_i$ takes at most $C_{change}K^{o(1)} \left\lceil \frac{d}{\eta(t)} \right\rceil$ time, which is negligible comparing with $K^{1-\Theta(\eta(T)^2)+o(\log^{-0.45} K)}$.

Next we prove the preprocessing time complexity. By Theorem 3.3, the preprocessing time complexity is $\kappa K^{1+o(1)}$. With $\kappa = K^{o(1)} \log T$, the complexity becomes $K^{1+o(1)} \cdot \log T$. Note that in line 5, $\left\lceil \frac{d}{\eta(T)} \right\rceil$ copies of Algorithm 1 are constructed. Therefore the preprocessing time complexity is $\frac{dK^{1+o(1)} \log T}{\eta(T)}$ and the per-step complexity is $K^{1-\Theta(\eta(T)^2)+o(\log^{-0.45} K)} \cdot \log T \cdot \log \frac{d}{\eta(T)}$, which can be further simplified as $K^{1+o(1)}$ preprocessing complexity and $K^{1-\Theta(\eta(T)^2)+o(\log^{-0.45} K)}$ per-step complexity.

Proof of Theorem B.5 - regret bound By setting the MIPS solver $\mathcal{M}$'s success probability to be at least $1 - \frac{\delta\cdot\eta(T)}{2d}$, we have the all queries (line 9 of Algorithm 4) are answered correctly with probability at least $1 - \frac{\delta}{2}$.Further, note that with setting of $\gamma(\delta/4T), \beta(\delta/4T)$, with probability at least $1 - \frac{\delta}{2}$, for all $t \le T$, we have

$$\|\widehat{\theta}_t - \theta^*\|_{V_t} \le \beta(\delta/4T), \quad \|\widetilde{\theta}_t - \widehat{\theta}_t\|_{V_t} \le \gamma(\delta/4T),$$

with the first inequality comes from Lemma B.3, and the second one follows from the *concentration* part of Definition B.1. The rest of the proof only considers the case when the events above hold, which happens with probability at least $1 - \delta$.

The regret analysis is similar to the one in (Abeille & Lazaric, 2017). We start with the regret decomposition

$$R(T) = \underbrace{\sum_{t=1}^{T} \left(x_{a_t^*}^\top \theta^* - x_{a_t}^\top \widetilde{\theta}_t \right)}_{R^{TS}(T)} + \underbrace{\sum_{t=1}^{T} \left(x_{a_t}^\top \widetilde{\theta}_t - x_{a_t}^\top \theta^* \right)}_{R^{RLS}(T)},$$

where the $R^{RLS}(T)$ is the regret induced by the "regularized least square" estimation, and $R^{TS}(T)$ measures the regret of making decision based on the $\widetilde{\theta}_t$ drawn by TS.

Bounding $R^{RLS}(T)$.

$$\begin{aligned}
R^{RLS}(T) &\le \sum_{t=1}^{T} \left| x_{a_t}^\top \left(\widetilde{\theta}_t - \widehat{\theta}_t \right) \right| + \sum_{t=1}^{T} \left| x_{a_t}^\top \left(\widehat{\theta}_t - \theta^* \right) \right| \\
&\le \sum_{t=1}^{T} \|x_{a_t}\|_{V_t^{-1}} \left(\|\widetilde{\theta}_t - \widehat{\theta}_t\|_{V_t} + \|\widehat{\theta}_t - \theta^*\|_{V_t} \right) \\
&\le (\gamma(\delta/4T) + \beta(\delta/4T)) \sum_{t=1}^{T} \|x_{a_t}\|_{V_t^{-1}} \\
&\le (\gamma(\delta/4T) + \beta(\delta/4T)) \sqrt{2Td \log \left(1 + \frac{T}{d} \right)}.
\end{aligned}$$

The last inequality follows from Lemma B.3.

Bounding $R^{TS}(T)$. At time t, denote $J_t(\theta) \coloneqq \max_{a\in\mathcal{A}} x_a^\top \theta$. Suppose Algorithm 4 selects a_t. Define $\Delta_t \coloneqq J_t(\widetilde{\theta}_t) - x_{a_t}^\top \widetilde{\theta}_t$, which is the approximation error solving MIPS approximately for $\widetilde{\theta}_t$. Denote $R_t^{TS} = x_{a_t^*}^\top \theta^* - x_{a_t}^\top \widetilde{\theta}_t$, we have

$$R_t^{TS} \le J_t(\theta^*) - J_t(\widetilde{\theta}_t) + \Delta_t.$$

Define $\mathcal{C}_t = \left\{\theta \mid \|\theta - \widehat{\theta}_t\|_{V_t} \leq \gamma(\delta/4T)\right\}$, which implies $\widetilde{\theta}_t \in \mathcal{C}_t$ for all $t \in [T]$. Then

$$R_t^{TS} \leq J_t(\theta^*) - \inf_{\theta \in \mathcal{C}_t} J_t(\theta) + \Delta_t.$$

We denote $\widetilde{\theta}_t$ is *optimistic* if $J_t(\widetilde{\theta}_t) \geq J_t(\theta^*)$. For any step t, $\widetilde{\theta}_t$ is optimistic with probability at least $p/2$, where p is defined in Definition B.1 (see Lemma 3 of (Abeille & Lazaric, 2017)). Condition on $\widetilde{\theta}_t$ being optimistic, we have

$$\begin{aligned}
R_t^{TS} &\leq J_t(\widetilde{\theta}_t) - \inf_{\theta \in \mathcal{C}_t} J_t(\theta) + \Delta_t \\
&\leq x_{a_t}^\top \widetilde{\theta}_t - \inf_{\theta \in \mathcal{C}_t} \max_{a \in \mathcal{A}} x_a^\top \theta + 2\Delta_t \\
&\leq x_{a_t}^\top \widetilde{\theta}_t - \inf_{\theta \in \mathcal{C}_t} x_{a_t}^\top \theta + 2\Delta_t \\
&\leq \sup_{\theta \in \mathcal{C}_t} \|x_{a_t}\|_{V_t^{-1}} \|\widetilde{\theta}_t - \theta\|_{V_t} + 2\Delta_t \\
&\leq 2\gamma(\delta/4T)\|x_{a_t}\|_{V_t^{-1}} + 2\Delta_t.
\end{aligned}$$

Note that the right-hand side is always positive, taking expectation with regard to $\widetilde{\theta}_t$ we have

$$\begin{aligned}
R_t^{TS} &\leq \mathbb{E}_{\widetilde{\theta}_t}\left[2\gamma(\delta/4T)\|x_{a_t}\|_{V_t^{-1}} + 2\Delta_t \;\middle|\; \widetilde{\theta}_t \text{ is optimistic }\right] \\
&\leq \frac{2}{p}\mathbb{E}_{\widetilde{\theta}_t}\left[2\gamma(\delta/4T)\|x_{a_t}\|_{V_t^{-1}} + 2\Delta_t\right].
\end{aligned}$$

Next we proceed to bound Δ_t. Note that as $\|\theta^*\|_2 \leq 1$, $\|\widehat{\theta}_t - \theta^*\|_{V_t} \leq \beta(\delta/4T)$, $\|\widetilde{\theta}_t - \widehat{\theta}_t\|_{V_t} \leq \gamma(\delta/4T)$. For all $t \in [T]$, the multiplicative error (introduced by MIPS according to the parameter in Algorithm 4, line 9) for $J_t(\widetilde{\theta}_t)$ is at most $(1 + \beta(\delta/4T) + \gamma(\delta/4T))\frac{\eta(T)}{d}$ and the additive error ϵ induces another $2(1 + \beta(\delta/4T) + \gamma(\delta/4T))\frac{\eta(T)}{d}$ approximation error.

Therefore for all $t \in [T]$, we have

$$\Delta_t \leq 3\left(1 + \gamma(\delta/4T) + \beta(\delta/4T)\right)\frac{\eta(T)}{d}.$$

It thus implies

$$\begin{aligned}
R^{TS}(T) \leq& \frac{4\gamma(\delta/4T)}{p}\sum_{t=1}^{T}\mathbb{E}_{\widetilde{\theta}_t}\left[\|x_{a_t}\|_{V_t^{-1}}\right] \\
&+ \frac{6\left(1 + \gamma(\delta/4T) + \beta(\delta/4T)\right)}{p}\eta(T)T \\
\leq& \frac{4\gamma(\delta/4T)}{p}\left(\sqrt{2Td\log\left(1 + \frac{T}{d}\right)} + \sqrt{8T\log\frac{4}{\delta}}\right) \\
&+ \frac{6\left(1 + \gamma(\delta/4T) + \beta(\delta/4T)\right)}{p} \cdot \frac{\eta(T)}{d} \cdot T.
\end{aligned}$$

The second inequality follows from Azuma's inequality on bounding the difference between $\sum_{t=1}^{T}\mathbb{E}_{\widetilde{\theta}_t}\left[\|x_{a_t}\|_{V_t^{-1}}\right]$ and $\sum_{t=1}^{T}\|x_{a_t}\|_{V_t^{-1}}$. Combining the bound for $R^{RLS}(T)$ and $R^{TS}(T)$ completes the proof.

C. Deferred Experiment Results

Here we present the deferred experiment results.

Synthetic Experiment - Addition and Deletion We empirically evaluate the performance of Sub-TS when there are both arm additions to and deletions from $\mathcal{A}$. The environment is set as specified in Section 7. Further, we set the number of arms K to be 10,000. For every 20 time steps, there are 2 arms newly generated from the unit spherical Gaussian distribution, and 2 random arms in $\mathcal{A}$ get deleted. The time horizon is set to 20,000 and the results are in Table 4.

Algorithm	Linear TS	Sub-TS, shortlist 10	Sub-TS, shortlist 30	Sub-TS, shortlist 100
Regret	612 ± 43	640 ± 42	612 ± 43	612 ± 43
Time(s)	44.80	25.49 (25.23)	26.37 (26.11)	29.69 (29.43)
Speedup	$\times 1$	$\times\mathbf{1.75}$ ($\times\mathbf{1.77}$)	$\times\mathbf{1.70}$ ($\times\mathbf{1.72}$)	$\times\mathbf{1.51}$ ($\times\mathbf{1.52}$)

Table 4: **Synthetic Experiment - Addition and Deletion.** The algorithms and "Regret", "Time" and "Speedup" are defined the same as in Table 1. We see that the Sub-TS is able to handle arms' changing, including both additions and deletions, and delivers around 1.51 – 1.77 times speedup.

Synthetic Experiment - Impact of C_{change} Here we empirically evaluate the impact of different numbers of arms' changing. The environment is set as specified in Section 7. Further, we set the initial number of arms to be 10,000. For every 20 time steps, there are C_{change} arms newly generated from the Gaussian distribution and included into $\mathcal{A}$. The time horizon is set to 20,000 and the results are in Table 5.

		Linear Elim	Sub-Elim, shortlist 30	Linear TS	Sub-TS, shortlist 30
$C_{change} = 2$	Regret	4433 ± 399	4393 ± 392	566 ± 52	566 ± 52
	Time (s)	35.72	2.85 (2.10)	45.54	21.01 (20.76)
	Speedup	$\times 1$	$\times\mathbf{12.53}$ ($\times\mathbf{17.01}$)	$\times 1$	$\times\mathbf{2.17}$ ($\times\mathbf{2.19}$)
$C_{change} = 10$	Regret	4428 ± 224	4345 ± 244	639 ± 54	638 ± 54
	Time (s)	36.22	3.04 (2.30)	55.99	24.91 (24.65)
	Speed-up	$\times 1$	$\times\mathbf{11.91}$ ($\times\mathbf{15.75}$)	$\times 1$	$\times\mathbf{2.25}$ ($\mathbf{2.27}$)
$C_{change} = 50$	Regret	4106 ± 154	4062 ± 169	581 ± 45	619 ± 62
	Time (s)	36.59	3.94 (3.20)	106.96	48.87 (48.62)
	Speedup	$\times 1$	$\times\mathbf{9.28}$ ($\times\mathbf{11.43}$)	$\times 1$	$\times\mathbf{2.19}$ ($\times\mathbf{2.20}$)

Table 5: **Synthetic Experiment - Impact of Different C_{change}.** The algorithms and "Regret", "Time" and "Speedup" are defined the same as in Table 1. Notice that Linear Elim and Sub-Elim are not much affected by C_{change}, as they will have already removed many arms in the later stages, and therefore the newly added arms do not affect the running time by much. The running time of Linear TS and Sub-TS, however, is significantly affected by C_{change}, as they are running on an increasingly large arm set. The Despite the impact on their individually running time, our algorithms are shown to deliver stable speedup in all evaluated settings.