Ranking in Contextual Multi-Armed Bandits

Amitis Shidani
Department of Statistics
University of Oxford
amitis.shidani@stats.ox.ac.uk

George Deligiannidis
Department of Statistics
University of Oxford
deligian@stats.ox.ac.uk

Arnaud Doucet
Department of Statistics
University of Oxford
doucet@stats.ox.ac.uk

Abstract

We study a ranking problem in the contextual multi-armed bandit setting. A learning agent selects an ordered list of items at each time step and observes stochastic outcomes for each position. In online recommendation systems, showing an ordered list of the most attractive items would not be the best choice since both position and item dependencies result in a complicated reward function. A very naive example is the lack of diversity when all the most attractive items are from the same category. We model position and item dependencies in the ordered list and design UCB and Thompson Sampling type algorithms for this problem. We prove that the regret bound over T rounds and L positions is $\tilde{O}(L\sqrt{dT})$, which has the same order as the previous works with respect to T and only increases linearly with L. Our work generalizes existing studies in several directions, including position dependencies where position discount is a particular case, and proposes a more general contextual bandit model.

1 Introduction

1.1 Background and motivation

The *multi-armed bandit* (MAB) problem is a sequential decision-making problem in which there are K possible choices called arms, each with an unknown reward distribution. At each time step t, the decision-maker can choose one arm and see a reward sample drawn from its distribution. The goal is to minimize the regret, which in the simplest case is defined as the difference between the total expected reward when playing the optimal action over time horizon T and the total expected reward collected by the decision-maker [13, 20]. Since the reward distributions are unknown, the decision-maker should keep a balance between *exploration* and *exploitation*; i.e., pulling each arm long enough to estimate its reward and short enough to dedicate the most of time pulling the optimal arm. In the classic MAB problem, the arms are assumed independent and we are interested in finding the arm with the highest expected reward [13].
Under these assumptions, various popular policies have been developed such as UCB, Thompson Sampling, and KL-UCB [12, 3, 18]. However, in the real world, arms are often dependent and pulling one arm gives information about other arms. Also, in some cases, like recommendation systems, the goal is to show an ordered list of items that best engages the users with the system and provides more rewards (i.e., clicks, watch time). There have been some recent papers addressing these issues in different ways.
On the one hand, [19] introduced the concept of *Contextual Bandits*, which considers the information

Preprint. Under review.

of arms and the state of the environment in terms of feature vectors and context. Contextual Bandits were then generalized in many works [23, 33, 37, 10]. In order to incorporate arms dependencies, [25], [31] and [7] take another approach and use the graph-based feedback setting based on the work of [26]. In this work, when the learner selects arm a they also observe the rewards of all adjacent arms. A generalization of the feedback graph approach can be found in [32], where the reward distributions of arms are dependent. More specifically, arms form clusters, and the reward distributions of arms belonging to the same cluster are known functions of an unknown parameter vector. [14] introduced another approach where rewards obtained by pulling different arms are correlated.
On the other hand, regarding the ranking problem, [28] proposes algorithms that learn a marginal utility for each document at each rank separately by either exploring and then committing to the best arms or running separate bandit algorithms for each position. [15] introduced an algorithm relying on a Lipschitz assumption on the reward functions, which on some levels provides useful information about the similarity between arms. [34] combined these two previous works with the contextual bandit and introduced a more practical algorithm. Here, the context for each position is defined as the event that previous items have not been clicked. This work was first generalized in [16], and then in [22]. [11] leverages contextual bandits for position based models. [20] introduces a more general approach for click models [8] where the objective is to identify the most attractive list of L items. Even though these works do not assume that the items are independent, they do not use any information on items' similarities either. In [24], the authors proposed a more general cascading bandit model using the position discount and contextual information between arms.

1.2 Our contribution

In these previous works, the expected reward function is non-decreasing with respect to items' attractiveness, i.e., if the user finds item a more attractive than item a', any ordered list with item a' replaced by item a provides a higher expected reward. This assumption can be very restrictive since the expected reward function may not always be monotonic in realistic scenarios; a realistic example is when the most attractive items are from the same category, and the user prefers to see a diverse list of items. In other words, an item's attractiveness may depend on the neighboring items.
In this paper, we generalize the previous works on the ranking problem in the MAB setting in two ways. First, the reward function we propose can be non-monotonic, addressing the issue mentioned above. Second, it is reasonable to assume that items can receive different levels of attention from users in different positions, i.e. having different attractiveness in each position. [24] addressed this challenge as a discount factor over positions. However, the real difference between positions may be more complicated. Therefore, we let here items share different contextual information at each position. To the best of our knowledge, this is the first paper addressing these issues.

2 Notation and setting

2.1 Problem formulation

Let $A = \{1, \ldots, K\}$ be the finite set of arms, where for each arm i there exists a vector $v_i \in \mathbb{R}^d$. We have L slots available, for which we want to find the best ordered list of L items, where $L \ll K$. At each round $t \in [T]$ the learner chooses an ordered list of L arms called an *action* $a_t = \{a_t^1, \ldots, a_t^L\} \in \mathcal{A}$, where for each i, $a_t^i \in A$ and $\mathcal{A}$ denotes the set of all the possible actions. At the end of each round, the learner observes the sample reward $r_{a_t}^l$ for each position $1 \leq l \leq L$. The goal is to minimize the expected regret $\mathcal{R}_T$ over the time horizon T, where the regret is defined as the gap between the cumulative reward received by the learner compared to the case where the optimal action was taken at all steps; i.e. we have:

$$\mathcal{R}_T = \mathbb{E}\left[\sum_{t=1}^{T} \max_{a \in \mathcal{A}} \sum_{l=1}^{L} \mathbb{E}\left[r_a^l\right] - \sum_{t=1}^{T}\sum_{l=1}^{L} r_{a_t}^l\right].$$

In the most general case, i.e. when there is no assumption on the reward function, the reward at each position may depend on all the items in the ordered list. In this case, choosing the optimal L-tuple of items is NP-hard since it is equivalent to the maximum coverage problem [27]. The standard greedy algorithm for the maximum coverage problem translates to iteratively choosing the items with the highest reward, which provides the same result as [28]. Since the problem is intractable in the general case, we impose some assumptions on the reward function. As mentioned in Section

1, previous works relied on a reward function monotonic with respect to items' attractiveness and assumed the attractiveness is independent of other items. We propose here a simple linear contextual reward function that is tractable and addresses these restrictions to some extent.
For each action a_t at round t and position l, let the reward function $r^l_{a_t}$ be as follows:

$$r^l_{a_t} = \langle \theta^l \ , \ v_{a^l_t} + w_l v_{a_t{}^{l-1}} \rangle + \eta^l_t, \tag{1}$$

where $\langle \cdot \ , \ \cdot \rangle : \mathbb{R}^d \times \mathbb{R}^d \to \mathbb{R}$ is the Euclidean inner product, θ^l is an unknown d-dimensional vector for position l, $w_l \in \mathbb{R}$ is a known parameter measuring the dependency of the reward function at position l to the item in the previous position, and $\{\eta^l_t\}_{t,l}$ is a family of centred, independent 1-subgaussian random variables. Here, we assume that the reward at position l depends on the attractiveness of both the items at positions l and $l-1$. The result can be generalized to a window of neighboring items instead, resulting however in more complicated calculations. Moreover, the parameter w_l can attain negative values allowing the reward function to be non-monotonic. In a more general case, w_l may be replaced by weights depending on both items and positions. Finally, θ^l allows the arms to share contextual information at each position. Having different parameters at each position allows us to model a more general case of discount factors, i.e., model different users' behavior for each position. The discount factor model is a special case of $\theta^{l+1} = d_l \theta^l$, where d_l denotes the discount parameter.
For the first position, as there is no previous item, the simplest approach is to assume that $v_{a^0_t} = \vec{0}$ or $w_1 = 0$, for any action a_t. However, more interesting approaches could also be considered. For instance, we could recommend a list based on the user's last action in a movie recommendation system. In this case, $v_{a^0_t}$ would be a vector embedding the user's last action, and w_1 would indicate its *importance*, i.e. the degree to which it affects the list. To address the most general case, we denote $v_{a^0_t}$ by v_0 in the rest of this paper. Now, let $\mathcal{H}_t$ denote the history *before* the learner chooses action at time t. Essentially, $\mathcal{H}_t$ contains the information at all times $s \leq t$, i.e. $\mathcal{H}_t = \{a_1, (r^1_{a_1}, \dots, r^L_{a_1}), \dots, a_{t-1}, (r^1_{a_{t-1}}, \dots, r^L_{a_{t-1}})\}$. The learner may use the history to choose the action a_t. Moreover, $\mathbb{E}[\eta^l_t | \mathcal{H}_t] = \mathbb{E}[\eta^l_t] = 0$ for any time t and position l.
Letting $a_\star$ be the optimal action, i.e. $a_\star = \operatorname{argmax}_{a \in \mathcal{A}} \sum_{l=1}^L \mathbb{E}\left[r^l_a\right]$, we can rewrite the regret function as follows

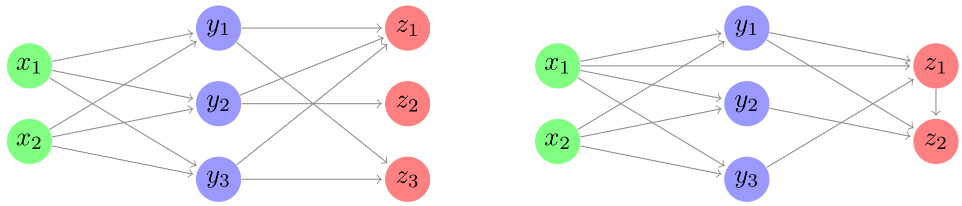
$$\mathcal{R}_T = \mathbb{E}\left[\textstyle\sum_{t=1}^T \sum_{l=1}^L \langle \theta^l \ , \ v_{a^l_\star} + w_l v_{a_\star^{l-1}} \rangle - \langle \theta^l \ , \ v_{a^l_t} + w_l v_{a_t^{l-1}} \rangle - \eta^l_t \right].$$

The expectation is over η_t and the randomness of the algorithm selecting a_t. In the rest of the paper, we first explain the challenges of this model, present our approach, and extend the UCB and Thompson Sampling algorithms to this ranking problem and provide upper bounds for their regrets.

2.2 Challenges

We can reformulate the problem by defining a new set of arms, called "super-arms of set A", as pairs of arms, i.e. (i, j) where $i, j \in A$, denoting the items in the previous and present position respectively. As there is no previous item for the first position, we denote the corresponding super-arms by $(0, i)$. In this case, the number of super-arms is $O(K^2)$. It may seem that the problem is finding the best super-arm for each position simultaneously. However this approach may result in an invalid action list for the main problem. Suppose e.g. that the best super-arms for positions $l, l+1$ are (i, j), (p, q) respectively; this sequence will only be valid for the original problem if $j = p$ as otherwise the super-arms disagree on the item proposed for position l.

One solution to the above problem is to run the algorithm sequentially conditioned to the previous item; a similar approach can be found in [28]. However, we argue that this may lead to linear regret. Assume for simplicity that $L = 2$. The algorithm starts by choosing a super-arm for the first position, and then continues to the next position. When the algorithm converges, the first item of the list will be fixed and would be the super-arm with the maximum estimated reward for the first position. However, w_2 and θ^2 may be chosen so that for any remaining acceptable arms, the reward for the second position is very low, such that the sum of the rewards of both positions is always less than choosing an arm with a lower expected reward for the first position. One way to address this issue is to define the super-arms as the whole list $(i_1, \dots, i_L)$, where $i_j \in A$. This approach leads to an action set with cardinality $O(K^L)$, resulting to vacuous regret bounds. We now propose a graph-based approach, which keeps the cardinality of the super-arms as $O(K^2)$ and tackles the aforementioned challenges.



(a) A valid 3-layered graph

(b) An invalid 3-layered graph

Figure 1: An illustration of a valid (1a) and an invalid (1b) 3-layered graph. The graph presented in 1b is invalid as the conditions (a) and (b) of Definition 1 are not established.

2.3 General approach

2.3.1 Further comments and definitions

We first give some preliminary definitions. All graphs we consider are weighted directed graphs.

Definition 1. *The directed graph $G = (V, E)$ is "L-layered" if and only if (a) $V = \bigcup_{j=1}^{L} V_j$, where $V_i \cap V_j = \emptyset$ for $i \neq j$, (b) all edges $e \in E$ have the form $e = (v, w)$ where $v \in V_l, w \in V_{l+1}$ for some $0 \leq l \leq L-1$, (c) there are no edges $e = (v, w)$ with $v \in V_0$, and (d) l-th layer, V_l, with $l \geq 2$, consists of the nodes with a depth of exactly $l-1$ from the nodes of the first layer V_1.*

With this definition, if G is L-layered and there are two vertices v and w in the same layer, from which there exist paths to vertex z, then z should be in the same depth from v and w. An illustration of a valid and an invalid 3-layered graph is presented in Figure 1.
Now, we want to build a L-layered graph G using the super-arms defined in the previous section. We add K vertices to the first layer, and K^2 vertices for each layer $2 \leq l \leq L$. We denote the nodes at layer one as u^1_{0i} for $i \in [K]$, correspoding to the super-arm $(0, i)$; at layer l we write u^l_{ij}, where $1 \leq i, j \leq K$ for the vertex assigned to super-arm (i, j) at position l. We connect the vertex u^l_{ij}, $l \in [L-1]$, to all the vertices u^{l+1}_{jq}, where $q \in [K]$. It is not hard to see that G is L-layered. Also, note that $G = \bigcup_{i=1}^{K} G_i$, where G_i is the induced subgraph of G that includes all the paths of G containing u^1_{0i}.
Now we need to define the weights of the edges for the weighted graph G. For the vector $\theta = (\theta^1, \ldots, \theta^L)$, if e is an edge between vertices u^l_{ij} and u^{l+1}_{jq}, then the weight of e denoted by c_e is defined as follows:

$$c_e = \begin{cases} \frac{1}{2}(2\langle \theta^1 \ , \ v_j + w_1 v_0 \rangle + \langle \theta^2 \ , \ v_q + w_2 v_j \rangle) & \text{if } l = 1; \\ \frac{1}{2}(\langle \theta^{L-1} \ , \ v_j + w_{L-1} v_i \rangle + 2\langle \theta^L \ , \ v_q + w_L v_j \rangle) & \text{if } l = L-1; \\ \frac{1}{2}(\langle \theta^l \ , \ v_j + w_l v_i \rangle + \langle \theta^{l+1} \ , \ v_q + w_{l+1} v_j \rangle) & \text{otherwise.} \end{cases}$$

Note that c_e can also be written as follows:

$$c_e = \begin{cases} \frac{1}{2}(2\mathbb{E}\left[r^1_a | a^1 = j, a^0 = 0\right] + \mathbb{E}\left[r^2_a | a^2 = q, a^1 = j\right]) & \text{if } l = 1; \\ \frac{1}{2}(\mathbb{E}\left[r^{L-1}_a | a^{L-1} = j, a^{L-2} = i\right] + 2\mathbb{E}\left[r^L_a | a^L = q, a^{L-1} = j\right]) & \text{if } l = L-1; \\ \frac{1}{2}(\mathbb{E}\left[r^l_a | a^l = j, a^{l-1} = i\right] + \mathbb{E}\left[r^{l+1}_a | a^{l+1} = q, a^l = j\right]) & \text{otherwise.} \end{cases} \tag{2}$$

We call this process of building G and G_is as "L-layering" over super-arms of set A and vector θ or the reward functions r^l. In the next part, we will explain how this L-layering simplifies the main problem of finding the best-ordered list.

2.3.2 General approach

Assume that the L-layering process over the super-arms of set A and the vector $\theta = (\theta^1, \ldots, \theta^L)$ is complete. Consider a path p of length $L-1$ or path p with L vertices. In a L-layered graph, this path starts with one of the first layer vertices and ends at a vertex from the L-th layer resulting in a sequence of the form $\{u^1_{0i_1}, u^2_{i_1 i_2}, \ldots, u^L_{i_{L-1} i_L}\}$. The sum of the weights of this path is equal to the expected reward of playing the action $(i_1, \ldots, i_L)$. The interesting thing about this graph is that every path of length $L-1$ provides a valid ordered list for the main ranking problem. Moreover the problem of finding the best-ordered list corresponds to finding the longest weighted path. Two problems arise, for which we propose solutions in the rest of this section:

- The time complexity of finding the longest weighted path may be high.
- The vector θ (i.e. the reward functions r^l) are unknown; therefore, we cannot find the path with the largest sum of weights.

Longest weighted path. Finding the longest weighted path of an arbitrary graph G is NP-hard. However, the longest path in a weighted graph G is the same as the shortest path in the graph $-G$ derived from G by replacing every weight by its opposite. Therefore, if shortest paths can be found in $-G$, then longest paths can also be found in G [30]. In most cases, this transformation is not useful because it creates negative length cycles in $-G$. However, if G is a directed acyclic graph, then no negative cycles can be created, and the longest path in G can be found in linear time by applying the linear time algorithm for shortest paths in $-G$ given in [9]. By definition, if G is L-layered, it is a directed acyclic graph, and we can easily find the shortest path in the graph $-G$, giving us the best ordered list of items. Moreover, the L-layered property already gives a topological ordering for the graph G, which might be a requirement for some algorithms to find the shortest path [9]. Also, note that we can reduce the time complexity of finding the shortest path of G by running the algorithm for each G_i separately and simultaneously and then comparing the shortest paths of sub-graphs. For instance, if we use Dijkstra's algorithm [35], the worst-case running time complexity would be $O\left(|E_G| + |V_G|\log(|V_G|)\right)$, where $|E_G|$ and $|V_G|$ represent the number of edges and the number of vertices of graph G. For the L-layered graph G over super-arms of set A, it would be $O(K^3)$.

Unknown vector θ. In order to find the longest path, we need to know the weights of the graph, which is not possible if the vector θ is unknown. This situation is similar to the MAB problem where we cannot play the optimal action from the beginning. In the MAB setting, we estimate the expected reward for each action at each round and then play the action with the highest expected reward. This approach would lead to an expected regret for the algorithm that estimates the rewards. Now, we will take a similar approach to address the problem of finding the longest path. For any algorithm that can estimate the expected reward of each super-arm, we will be able to find the longest path in the graph with these estimated weights. At each round, we update the weights based on the reward history of the super-arms. When the algorithm converges to the actual values of the expected reward, the longest path would also converge to the best-ordered list.

In the following sections, we will mostly focus on two famous algorithms, UCB [6] and Thompson Sampling [29], and will show how to adapt them to find the best ordered list using the L-layering technique.

3 Ranking UCB algorithm

3.1 Algorithm

Before presenting the algorithm, we explain the main ideas behind it. By Equation 1, we have:

$$\mathbb{E}\left[r_a^l|\mathcal{H}_t\right] = {\theta^l}^{\mathrm{T}}(v_{a^l} + w_l v_{a^{l-1}}).$$

To estimate the expected reward for position l, we need to first estimate θ^l, which can be viewed as a regression problem with samples $x_t^l = v_{a_t^l} + w_l v_{a_t^{l-1}}$ and labels $r_{a_t}^l$. Therefore, we can estimate θ^l using a regularized least square estimator. Thus, we have:

$$\hat{\theta}_t^l = \mathrm{argmin}_{\theta^l} \sum_{s=1}^t (r_{a_s}^l - \langle \theta^l\ ,\ x_s^l \rangle)^2 + \lambda \|\theta^l\|_2^2 = V_t^l(\lambda)^{-1}\left[\sum_{s=1}^t r_{a_s}^l x_s^l\right],$$

where $V_0^l(\lambda) = \lambda I$, and $V_t^l(\lambda) = V_0^l(\lambda) + \sum_{s=1}^t x_s^l {x_s^l}^{\mathrm{T}}$. Note that $V_t^l(\lambda) \in \mathbb{R}^{d\times d}$ is a symmetric strictly positive definite matrix, and for any strictly positive definite matrix V we can define a norm on $\mathbb{R}^d$ given by $\|x\|_V = (x^{\mathrm{T}} V x)^{\frac{1}{2}}$.

Now, we would like to construct a confidence set $\mathcal{C}_t^l \subset \mathbb{R}^d$ based on $\mathcal{H}_t$ and centered at $\hat{\theta}_{t-1}^l$ that contains the unknown parameter θ^l with high probability. To this end, we use the approach proposed in Theorem 2 in [1], restated as follows:

Lemma 1. *[1] Let $\delta \in (0,1)$. Then, with probability at least $1-\delta$, it holds that for any time $t \in \mathcal{N}$,*

$$\|\hat{\theta}_t^l - \theta^l\|_{V_t^l(\lambda)} \leq \sqrt{\lambda}\|\theta^l\|_2 + \sqrt{2\log\left(\tfrac{1}{\delta}\right) + \log\left(\frac{\det\left(V_t^l(\lambda)\right)}{\lambda^d}\right)}. \tag{3}$$

In other words, Lemma 1 shows that if we define $\mathcal{C}_t^l$ as follows, then $\mathbb{P}(\exists t \in \mathcal{N} : \theta^l \notin \mathcal{C}_t^l) \leq \delta$ for

$$\mathcal{C}_t^l = \left\{ \theta \in \mathbb{R}^d : \|\hat{\theta}_{t-1}^l - \theta\|_{V_{t-1}^l(\lambda)} \leq \sqrt{\lambda}\|\theta^l\|_2 + \sqrt{2\log\left(\tfrac{1}{\delta}\right) + \log\left(\frac{\det\left(V_{t-1}^l(\lambda)\right)}{\lambda^d}\right)} \right\}. \tag{4}$$

Thus, we can define the optimistic estimated reward for any super-arm (i, j) and position l in the UCB algorithm as the following:

$$\mathrm{UCB}_t^l(i,j) = \max_{\theta \in \mathcal{C}_t^l} \langle \theta \ , \ v_j + w_l v_i \rangle. \tag{5}$$

Now that we have an algorithm to estimate the rewards, we can use Equation 2 and the discussion in Section 2.3.2 to build the L-layering graph G over the super-arms and the estimated rewards. Namely, at each round t, Equation 5 allows us to replace the weight c_e for the edge $e = (u_{ij}^l, u_{jq}^{l+1})$ by the estimate $\hat{c}_e$:

$$\hat{c}_e = \begin{cases} \frac{1}{2}(2\mathrm{UCB}_t^1(0,j) + \mathrm{UCB}_t^2(j,q) & \text{if } l = 1; \\ \frac{1}{2}(\mathrm{UCB}_t^{L-1}(i,j) + 2\mathrm{UCB}_t^L(j,q)) & \text{if } l = L-1; \\ \frac{1}{2}(\mathrm{UCB}_t^l(i,j) + \mathrm{UCB}_t^{l+1}(j,q)) & \text{otherwise.} \end{cases} \tag{6}$$

Finding the longest path of G leads us to the best ordered list for each round t using the UCB algorithm. The complete algorithm, RankUCB, is described in Algorithm 1. In the next section, we will provide a regret bound for this algorithm.

Algorithm 1 RankUCB

Input: $\lambda > 0$, $\delta \in (0, 1)$, L, $\{w_l\}_{l \leq L}$, T, arm set $A = \{1, \ldots, K\}$, and vector v_0
Create L-layered graph $G = \bigcup_{i=1}^K G_i$ over super-arms of set A
Initialization: $\hat{\theta}_0^l = 0$, $V_0^l = \lambda I$ for $l \in [L]$, and for any edge e of G, set $\hat{c}_e = 0$
for $t = 1, 2, \ldots, T$ **do**
  Obtain $p_i \leftarrow \mathrm{ShortestPathAlgorithm}(-G_i)$ for all $i \in [K]$ simultaneously
  $p_\star \leftarrow \mathrm{argmin}_{p_i} \sum_{e \in p_i} \hat{c}_e$
  Choose action a_t as the ordered vertices of path $p_\star$
  Play a_t and observe $r_{a_t}^l$ for $l \in [L]$
  for $l = 1, \ldots L$ **do**
    $V_t^l(\lambda) \leftarrow V_{t-1}^l + (v_{a_t^l} + w_l v_{a_t^{l-1}})(v_{a_t^l} + w_l v_{a_t^{l-1}})^{\mathrm{T}}$
    $\hat{\theta}_t^l \leftarrow V_t^l(\lambda)^{-1} \left[\sum_{s=1}^t r_{a_s}^l (v_{a_s^l} + w_l v_{a_s^{l-1}}) \right]$
    Create $\mathcal{C}_{t+1}^l$ based on Equation 4
    $\mathrm{UCB}_{t+1}^l(i,j) \leftarrow \max_{\theta \in \mathcal{C}_{t+1}^l} \langle \theta \ , \ v_j + w_l v_i \rangle$ for all super-arms (i, j)
    Update $\hat{c}_e$, for any edge e, based on Equation 6
  end for
end for

3.2 Regret bound

In this section, we provide a regret bound for the RankUCB algorithm under the assumptions that the confidence intervals for each position indeed contain the true parameters with high probability and that the action set is bounded.

Assumption 1. *For some $m_1, m_2 > 0$, the following hold: (a) for any arm $i \in A$, $\|v_i\|_2 \leq m_1$, (b) for all $l \in L$, $\|\theta^l\|_2 \leq m_2$, (c) $\sup_{l \in [L]} \sup_{i \in A} |\langle \theta^l \ , \ v_i \rangle| \leq 1$, (d) There exist $\delta \in (0,1)$ such that with probability at least $1 - \delta$, for all $t \in [T]$ and $l \in [L]$, $\theta^l \in \mathcal{C}_t^l$ where $\mathcal{C}_t^l$ satisfies the Equation 4.*

Now, we prove a bound on the regret achieved by RankUCB. The proof is provided in Appendix A.

Theorem 1. *Under Assumption 1, with probability at least $1 - \delta$, the expected regret of the RankUCB algorithm satisfies:*

$$\mathcal{R}_T \leq 2\sqrt{2}(1 + \max_{l \in [L]} |w_l|) L \sqrt{dT\beta_T \log\left(1 + \frac{T\left((1 + \max_{l \in [L]} |w_l|) m_1\right)^2}{d\lambda}\right)} \tag{7}$$

where $\sqrt{\beta_T} = \max_{l \in [L]} \sqrt{\lambda} m_2 + \sqrt{2 \log\left(\frac{1}{\delta}\right) + \log\left(\frac{\det\left(V_T^l(\lambda)\right)}{\lambda^d}\right)}$.

The conditions of Assumption 1 are not restrictive since usually the arms and contexts are coming from compact sets of $\mathbb{R}^d$ and can be normalized to satisfy the conditions. Theorem 1 provides an upper bound of $\tilde{O}(L\sqrt{dT})$ for the ranking MAB problem. This upper bound increases only linearly on L while capturing both item and position dependencies compared to the previous works, where these dependencies were either ignored or simplified. Additionally, the bound has a similar order of complexity in terms of T and d compared to previous works [1, 24].

4 Ranking Thompson Sampling algorithm

4.1 Algorithm

Thompson Sampling (TS) [36] assumes there exists a prior distribution $\mathcal{Q}$ on the parameter $\theta \in \Theta$ of the conditional reward distribution $\mathcal{P}(\cdot|\theta)$, which in our case is a subgaussian distribution as η is subgaussian. At each round t, the algorithm draws a sample from the posterior distribution $\hat{\theta}_t \sim \mathcal{Q}(\cdot|\mathcal{H}_t)$, selects the best action according to the sample, and updates the distribution based on the observed reward. We are assuming here that each θ^l is sampled independently from a prior distribution $\mathcal{Q}^l$, and we will update their posterior distributions separately. The prior distribution $\mathcal{Q}^l$ for different l can be different, i.e. the samples are not necessarily identically distributed.
Also, note that for finding the best action according to the samples $\hat{\theta}_t^l$ for $l \in [L]$, we use the L-layering graph technique. In other words, we use the samples of the vector $\hat{\theta} = (\hat{\theta}^1, \ldots, \hat{\theta}^L)$ to estimate the weights of each edge e in the L-layered graph G over super-arms of set A, and find the longest path in the graph as the best action for round t. Thus, the estimated weight of $\hat{c}_e$, where e is the edge from u_{ij}^l to u_{jq}^{l+1} would be defined as follows:

$$\hat{c}_e = \begin{cases} \frac{1}{2}(2\langle\hat{\theta}^1\ ,\ v_j + w_1 v_0\rangle + \langle\hat{\theta}^2\ ,\ v_q + w_2 v_j\rangle) & \text{if } l = 1; \\ \frac{1}{2}(\langle\hat{\theta}^{L-1}\ ,\ v_j + w_{L-1} v_i\rangle + 2\langle\hat{\theta}^L\ ,\ v_q + w_L v_j\rangle) & \text{if } l = L-1; \\ \frac{1}{2}(\langle\hat{\theta}^l\ ,\ v_j + w_l v_i\rangle + \langle\hat{\theta}^{l+1}\ ,\ v_q + w_{l+1} v_j\rangle) & \text{otherwise.} \end{cases} \tag{8}$$

The final adaptation of TS algorithm, RankTS, is described in Algorithm 2. In the next section, we obtain an upper bound for the regret of this algorithm.

Algorithm 2 RankTS

Input: L, prior distributions $\{\mathcal{Q}^l\}_{l=1}^L$, $\{w_l\}_{l \le L}$, T, arm set $A = \{1, \ldots, K\}$, and vector v_0
Create L-layered graph $G = \bigcup_{i=1}^K G_i$ over super-arms of set A
Initialization: For any edge e of G, set $\hat{c}_e = 0$
for $t = 1, 2, \ldots, T$ **do**
  $(\hat{\theta}^1, \ldots, \hat{\theta}^L) \sim \mathcal{Q}^1(\cdot|\mathcal{H}_t) \otimes \ldots \otimes \mathcal{Q}^L(\cdot|\mathcal{H}_t)$
  Update $\hat{c}_e$, for any edge e, based on Equation 8
  Obtain $p_i \leftarrow \mathrm{ShortestPathAlgorithm}(-G_i)$ for all $i \in [K]$ simultaneously
  $p_\star \leftarrow \mathrm{argmin}_{p_i} \sum_{e \in p_i} \hat{c}_e$
  Choose action a_t as the ordered vertices of path $p_\star$
  Play a_t and observe $r_{a_t}^l$ for $l \in [L]$
  $\mathcal{H}_{t+1} \leftarrow \mathcal{H}_t \cup \{a_t, (r_{a_t}^1, \ldots, r_{a_t}^L)\}$
  Update $\mathcal{Q}^l(\cdot|\mathcal{H}_{t+1})$ for $l \in L$
end for

4.2 Regret bound

The first result providing an upper bound for TS with linear reward functions was obtained in [4]. Then, [2] presented a new proof, which can also be applied to generalized or regularized linear models. Our upper bound for RankTS borrows the techniques from these two papers. We first need the following assumption to state the main theorem:

Assumption 2. *For some $m_1, m_2 > 0$, the following hold: (a) for any arm $i \in A$, $\|v_i\|_2 \leq m_1$, (b) for all $l \in L$, $\|\theta^l\|_2 \leq m_2$ with $\mathcal{Q}^l$-probability one, (c) $\sup_{l\in[L]} \sup_{i\in A} |\langle\theta^l\ ,\ v_i\rangle| \leq 1$.*

Now, we have the following theorem and the proof is available in Appendix B:

Theorem 2. *Under Assumption 2, the expected regret of the RankTS algorithm is bounded by:*

$$\mathcal{R}_T \leq 2L(1 + \max_{l\in[l]} |w_l|)\left(1 + \sqrt{2Td\beta^2 \log\left(1 + \frac{T\left((1+\max_{l\in[L]}|w_l|)m_1\right)^2}{d\lambda}\right)}\right) \tag{9}$$

where $\beta = 1 + \sqrt{4\log(T) + d\log\left(1 + T\left((1+\max_{l\in[L]}|w_l|)m_1\right)^2/d\lambda\right)}$.

The upper bound obtained for RankTS matches the upper bound obtained by RankUCB, which is consistent with previous results on TS and UCB. An implementation of RankTS needs to sample from the posterior, which is not straightforward for some priors and might need numerical methods such as Markov chain Monte Carlo [5] or variational inference [38]. Having sampled θ^l, finding the best action requires solving a linear optimization problem. By comparison, RankUCB needs to solve 5, which can be intractable for large or continuous action sets.

5 Generalization: estimating position dependencies

In the previous sections, we have assumed that the dependency parameters $w_l, l \in [L]$ are known. However, in realistic scenarios, we need to estimate them. We now reformulate the problem in a way that allows us to jointly estimate $\theta = (\theta^1, \dots, \theta^l)$ and $w = (w_1, \dots, w_l)$. Then, we modify the RankUCB algorithm for this general case and provide the corresponding regret bound.
Let us rewrite the expected reward of action $a = (a^1, \dots, a^L)$ at position l as follows:

$$\mathbb{E}\left[r^l\right] = \langle\theta^l\ ,\ v_{a^l} + w_l v_{a^{l-1}}\rangle = \langle\phi^l\ ,\ \tilde{x}^l_a\rangle,$$

where $\phi^l = \left(\theta^l\ \ w_l\theta^l\right)^{\mathrm{T}} \in \mathbb{R}^{2d}$ and $\tilde{x}^l_a = \left(v_{a^l}\ \ v_{a^{l-1}}\right)^{\mathrm{T}} \in \mathbb{R}^{2d}$. We can follow a similar procedure to that in Section 3. We estimate

$$\hat{\phi}^l_t = \mathrm{argmin}_{\phi^l} \textstyle\sum_{s=1}^t (r^l_{a_s} - \langle\phi^l\ ,\ \tilde{x}^l_{a_s}\rangle)^2 + \lambda\|\phi^l\|_2^2 = \tilde{V}^l_t(\lambda)^{-1}\left[\sum_{s=1}^t r^l_{a_s}\tilde{x}^l_{a_s}\right]$$

where $\tilde{V}^l_0(\lambda) = \lambda I \in \mathbb{R}^{2d\times 2d}$, and $\tilde{V}^l_t(\lambda) = \tilde{V}^l_0(\lambda) + \sum_{s=1}^t \tilde{x}^l_{a_s}\tilde{x}^{l\mathrm{T}}_{a_s}$. Now, we can use Lemma 1 to create a confidence interval for ϕ^l denoted by $\tilde{\mathcal{C}}^l_t$. Thus, the estimated reward for super-arm (i, j) at position l, which is denoted by vector $\tilde{x}_{ji} = \left(v_j\ \ v_i\right)^{\mathrm{T}}$, would be

$$\mathrm{UCB}^l_t(i,j) = \max_{\phi\in\tilde{\mathcal{C}}^l_t} \langle\phi\ ,\ \tilde{x}_{ji}\rangle. \tag{10}$$

Finally, to find the best ordered list at each round, we can build the L-layered graph G as before and use the Equations 10 and 6 to update the weights of the edges. The generalized algorithm, genRankUCB, is provided in Appendix C.1. The next theorem upper bounds the regret of this algorithm. The proof is provided in the Appendix C.2. First, we need the following assumption:

Assumption 3. *For some $m_1, m_2, m_3 > 0$, the following hold: (a) for any arm $i \in A$, $\|v_i\|_2 \leq m_1$, (b) for all $l \in L$, $\|\theta^l\|_2 \leq m_2$, (c) for all $l \in L$, $|w_l| \leq m_3$, (d) $\sup_{l\in[L]}\sup_{i\in A} |\langle\theta^l\ ,\ v_i\rangle| \leq 1$, (e) There exist $\delta \in (0, 1)$ such that with probability at least $1 - \delta$, for all $t \in [T]$ and $l \in [L]$, $\phi^l \in \tilde{\mathcal{C}}^l_t$ where $\tilde{\mathcal{C}}^l_t$ satisfies the Equation 4 for ϕ^l.*

Theorem 3. *Under the conditions of Assumption 3, with probability at least $1 - \delta$, the expected regret of the genRankUCB algorithm satisfies:*

$$\mathcal{R}_T \leq 4(1 + m_3)L\sqrt{dT\beta_T \log\left(1 + \frac{2Tm_2^2}{d\lambda}\right)} \tag{11}$$

where $\sqrt{\beta_T} = \max_{l\in[L]} \sqrt{\lambda}\left(m_2\sqrt{1+m_3^2}\right) + \sqrt{2\log\left(\frac{1}{\delta}\right) + \log\left(\frac{\det\left(\tilde{V}^l_T(\lambda)\right)}{\lambda^{2d}}\right)}$.

The upper bound provided in Theorem 3 has a larger coefficient factor and is looser than the bound reported in Theorem 1, which was predictable since there are more unknown parameters.

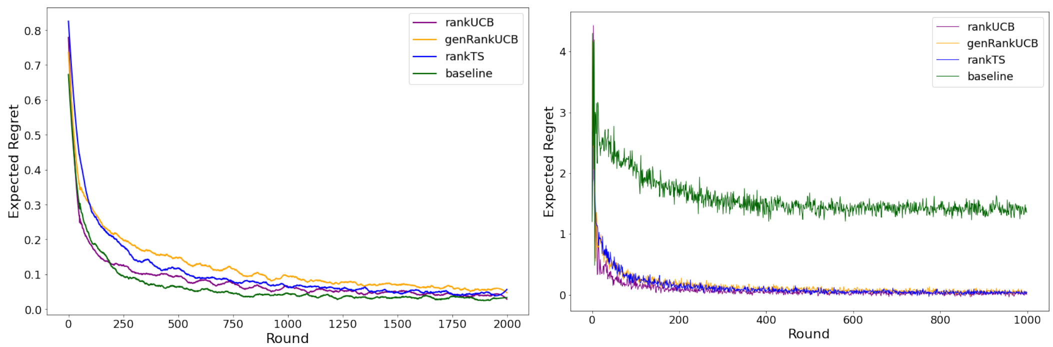


Figure 2: Expected regret for $K = 100$. **Left:** $w_l = 0 \ \forall l \in [L]$, **Right:** $\max_{l\in[L]} |w_l| = 10$.

6 Experiments

In this section, we compare RankUCB, genRankUCB and RankTS to the baseline algorithm [28], where there is no assumption on the dependency between positions. The experiments are contextual bandits with $d = 10$, $L = 4$ and $K \in \{10, 100\}$, and various values for weights w_l to be small, large and zero. The case where $w_l = 0$ for all $l \in [L]$ would be similar to the setting discussed in [28]. Regarding the parameters, we randomly choose $\theta'_l \in \mathbb{R}^{d-1}$ with $\|\theta'_l\|_2 = 1$ and let $\theta^l = (\frac{\theta'_l}{2}\ ,\ \frac{1}{2})$. We let the vector associated to arm i be $v_i = (v'_i\ ,\ 1)$, where $v'_i \in \mathbb{R}^{d-1}$ with $\|v'_i\|_2 = 1$. This process will guarantee that $\sup_{l\in[L]} \sup_{i\in A} |\langle\theta^l\ ,\ v_i\rangle| \leq 1$, which is required for Assumptions 1, 2 and 3. Next, we generate the weight w_l by a random sample from the uniform distribution.
The results are reported in Figure 2 and Appendix D. When $\max_{l\in[L]} |w_l|$ is very small or zero, left figure of Figure 2, all the algorithms perform well and display similar performances. As $\max_{l\in[L]} |w_l|$ increases, the baseline algorithm, which does not capture the position dependencies, does not converge to the optimal action. This leads to a non-zero regret over time T, i.e. leads to a linear regret. In contrast, our algorithms perform well with all ranges of w_l. One remarkable result is that genRankUCB performs essentially as well as RankUCB and RankTS while estimating the dependencies. However, one practical challenge is efficiently solving Equation 5 for UCB-based algorithms. In this regard, since $\mathcal{C}_t$ from Equation 4 is ellipsoid, we can estimate:

$$\hat{\mathrm{UCB}}^l_t(i,j) = \langle\hat{\theta}^l_{t-1}\ ,\ v_j + w_l v_i\rangle + \sqrt{\beta_T}\|v_j + w_l v_i\|_{V^{l-1}_{t-1}}.$$

Moreover, the prior can have a significant effect on the performance of RankTS; see Appendix D for further experimental details. This sensitivity is due to the fact that the selected prior may result in not playing an arm over a horizon of length T. In Figure 2, we are using multivariate normal distribution as prior for vector θ in RankTS, which reduces the computation cost as the noise η_t is Gaussian as well. In this case, we are sampling $\hat{\theta}^l_t \sim \mathcal{N}(\mu^l_{t-1}, \Sigma^l_{t-1})$ where μ^l and Σ^l are updated based on the history at each round; see Appendix D for details. Finally, all algorithms' run times were very similar. As K increases, the main bottleneck becomes finding the longest path in graph G. We addressed this issue to some extent by multiprocessing in our implementation.

7 Conclusion

We have studied here the ranking problem in contextual multi-armed bandits with position dependency. We have proposed two algorithms, RankUCB and RankTS, and proved a T-round expected regret of order $\tilde{O}(L\sqrt{dT})$ for the linear reward function. The key idea in both algorithms is to formulate the optimal ordered list as the longest path in a graph corresponding to possible actions. The experiments conducted demonstrate the advantage of involving position dependency.
A future direction of study would be to extend the ranking problem with non-linear reward functions like Generalized Linear Models [17] and Neural Contextual Bandits [39]. We believe the idea of finding the longest path and position dependency could be easily generalized. Moreover, the L-layered graph proposed in this paper is $(K, 0)$-sparse [21] and has a specific structure. Optimizing the shortest path algorithm for this graph could also be an interesting future work, which might significantly reduce the time complexity of the algorithms.

Acknowledgement

Arnaud Doucet is partly supported by the EPSRC grant EP/R034710/1. He also acknowledges the support of the UK Defence Science and Technology Laboratory (DSTL) and EPSRC under grant EP/R013616/1. This is part of the collaboration between US DOD, UK MOD, and UK EPSRC, under the Multidisciplinary University Research Initiative.

References

[1] Y. Abbasi-Yadkori, D. Pál, and C. Szepesvári. Improved algorithms for linear stochastic bandits. *Advances in Neural Information Processing Systems*, 24, 2011.

[2] M. Abeille and A. Lazaric. Linear Thompson sampling revisited. In *Artificial Intelligence and Statistics*, pages 176–184. PMLR, 2017.

[3] S. Agrawal and N. Goyal. Analysis of Thompson sampling for the multi-armed bandit problem. In *Conference on learning theory*, pages 39–1. JMLR Workshop and Conference Proceedings, 2012.

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

[5] C. Andrieu, N. De Freitas, A. Doucet, and M. I. Jordan. An introduction to MCMC for machine learning. *Machine Learning*, 50(1):5–43, 2003.

[6] P. Auer, N. Cesa-Bianchi, and P. Fischer. Finite-time analysis of the multiarmed bandit problem. *Machine Learning*, 47(2):235–256, 2002.

[7] S. Buccapatnam, A. Eryilmaz, and N. B. Shroff. Stochastic bandits with side observations on networks. In *The 2014 ACM international conference on Measurement and modeling of computer systems*, pages 289–300, 2014.

[8] A. Chuklin, I. Markov, and M. de Rijke. Click models for web search. *Synthesis lectures on information concepts, retrieval, and services*, 7(3), 2015.

[9] T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein. *Introduction to Algorithms*. MIT press, 2022.

[10] M. Dimakopoulou, Z. Zhou, S. Athey, and G. Imbens. Balanced linear contextual bandits. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, 2019.

[11] B. Ermis, P. Ernst, Y. Stein, and G. Zappella. Learning to rank in the position based model with bandit feedback. In *Proceedings of the 29th ACM International Conference on Information & Knowledge Management*, pages 2405–2412, 2020.

[12] A. Garivier and O. Cappé. The KL-UCB algorithm for bounded stochastic bandits and beyond. In *Proceedings of the 24th annual Conference on Learning Theory*, pages 359–376. JMLR Workshop and Conference Proceedings, 2011.

[13] J. C. Gittins. Bandit processes and dynamic allocation indices. *Journal of the Royal Statistical Society: Series B (Methodological)*, 41(2):148–164, 1979.

[14] S. Gupta, S. Chaudhari, G. Joshi, and O. Yagan. Multi-armed bandits with correlated arms. *IEEE Transactions on Information Theory*, 67(10), 2021.

[15] R. Kleinberg, A. Slivkins, and E. Upfal. Multi-armed bandits in metric spaces. In *Proceedings of the fortieth annual ACM symposium on Theory of Computing*, pages 681–690, 2008.

[16] B. Kveton, C. Szepesvari, Z. Wen, and A. Ashkan. Cascading bandits: Learning to rank in the cascade model. In *International Conference on Machine Learning*, pages 767–776. PMLR, 2015.

[17] B. Kveton, M. Zaheer, C. Szepesvari, L. Li, M. Ghavamzadeh, and C. Boutilier. Randomized exploration in generalized linear bandits. In *Proceedings of the Twenty Third International Conference on Artificial Intelligence and Statistics*, volume 108, 2020.

[18] T. L. Lai and H. Robbins. Asymptotically efficient adaptive allocation rules. *Advances in Applied Mathematics*, 6(1):4–22, 1985.

[19] J. Langford and T. Zhang. The epoch-greedy algorithm for multi-armed bandits with side information. In *Advances in Neural Information Processing Systems*, volume 20. Curran Associates, Inc., 2008.

[20] T. Lattimore and C. Szepesvári. *Bandit Algorithms*. Cambridge University Press, 2020.

[21] A. Lee and I. Streinu. Pebble game algorithms and sparse graphs. *Discrete Mathematics*, 308 (8):1425–1437, 2008.

[22] C. Li, H. Feng, and M. de Rijke. Cascading hybrid bandits: Online learning to rank for relevance and diversity. In *Fourteenth ACM Conference on Recommender Systems*, 2020.

[23] L. Li, W. Chu, J. Langford, and R. E. Schapire. A contextual-bandit approach to personalized news article recommendation. In *Proceedings of the 19th International Conference on World Wide Web*, pages 661–670, 2010.

[24] S. Li, B. Wang, S. Zhang, and W. Chen. Contextual combinatorial cascading bandits. In *Proceedings of The 33rd International Conference on Machine Learning*, volume 48 of *Proceedings of Machine Learning Research*. PMLR, 2016.

[25] T. Lykouris, E. Tardos, and D. Wali. Feedback graph regret bounds for Thompson sampling and UCB. In *Algorithmic Learning Theory*, pages 592–614. PMLR, 2020.

[26] S. Mannor and O. Shamir. From bandits to experts: On the value of side-observations. *Advances in Neural Information Processing Systems*, 24, 2011.

[27] G. L. Nemhauser, L. A. Wolsey, and M. L. Fisher. An analysis of approximations for maximizing submodular set functions–i. *Mathematical programming*, 14(1), 1978.

[28] F. Radlinski, R. Kleinberg, and T. Joachims. Learning diverse rankings with multi-armed bandits. In *International Conference on Machine Learning*, pages 784–791, 2008.

[29] D. J. Russo, B. Van Roy, A. Kazerouni, I. Osband, and Z. Wen. A tutorial on Thompson sampling. *Foundations and Trends in Machine Learning*, 11(1), 2018.

[30] R. Sedgewick and K. Wayne. *Algorithms, 4th Edition.* Addison-Wesley, 2011.

[31] R. Singh, F. Liu, X. Liu, and N. Shroff. Contextual bandits with side-observations. *arXiv preprint arXiv:2006.03951*, 2020.

[32] R. Singh, F. Liu, Y. Sun, and N. Shroff. Multi-armed bandits with dependent arms. *arXiv preprint arXiv:2010.09478*, 2020.

[33] A. Slivkins. Contextual bandits with similarity information. In *Proceedings of the 24th Annual Conference on Learning Theory*, volume 19 of *Proceedings of Machine Learning Research*. PMLR, 2011.

[34] A. Slivkins, F. Radlinski, and S. Gollapudi. Ranked bandits in metric spaces: learning diverse rankings over large document collections. *Journal of Machine Learning Research*, 14(Feb): 399–436, 2013.

[35] M. Sniedovich. Dijkstra's algorithm revisited: the dynamic programming connexion. *Control and Cybernetics*, 35(3):599–620, 2006.

[36] W. R. Thompson. On the likelihood that one unknown probability exceeds another in view of the evidence of two samples. *Biometrika*, 25(3-4):285–294, 1933.

[37] M. Valko, N. Korda, R. Munos, I. Flaounas, and N. Cristianini. Finite-time analysis of kernelised contextual bandits. In *Uncertainty in Artificial Intelligence*, page 654, 2013.

[38] M. J. Wainwright and M. I. Jordan. Graphical models, exponential families, and variational inference. *Foundations and Trends® in Machine Learning*, 1(1–2):1–305, 2008.

[39] D. Zhou, L. Li, and Q. Gu. Neural contextual bandits with UCB-based exploration. In *International Conference on Machine Learning*, pages 11492–11502. PMLR, 2020.

A Proof of Theorem 1

In order to provide the proof for Theorem 1, we first need the following lemma, which often is called the elliptical potential lemma:

Lemma 2. *Let $V_0 \in \mathbb{R}^{d\times d}$ be a positive definite matrix and $b_1, \dots, b_T \in \mathbb{R}^d$ be a sequence of vectors with $\|b_t\|_2 \leq M < \infty$. For all $t \in [T]$, define $V_t = V_0 + \sum_{s\leq t} b_s b_s^{\mathrm{T}}$. Then,*

$$\sum_{t=1}^{T} \min\{1\ ,\ \|b_t\|^2_{V_t^{-1}}\} \leq 2\log\left(\frac{\det(V_T)}{\det(V_0)}\right) \leq 2d\log\left(\frac{\mathrm{tr}(V_0)+TM^2}{d\det(V_0)^{\frac{1}{d}}}\right).$$

Proof. If V is a symmetric positive definite matrix, then $V+U = V^{1/2}(I + V^{-1/2}UV^{-1/2})V^{1/2}$. Moreover, for each $t \in [T]$, we have that V_t is a symmetric positive definite matrix. Thus, for any $t \geq 1$, we can write:

$$V_t = V_{t-1} + b_t b_t^{\mathrm{T}} = V_{t-1}^{1/2}\left(I + V_{t-1}^{-1/2} b_t b_t^{\mathrm{T}} V_{t-1}^{-1/2}\right)V_{t-1}^{-1/2}$$

By noting that $\det(VU) = \det(V)\det(U)$, we have that:

$$\det(V_t) = \det(V_{t-1})\det\left(I + V_{t-1}^{-1/2} b_t b_t^{\mathrm{T}} V_{t-1}^{-1/2}\right) = \det(V_{t-1})\left(1 + \|b_t\|^2_{V_{t-1}^{-1}}\right).$$

The last equality is due to the fact that the determinant of a matrix is the product of its eigenvalues, and matrix $I + xx^{\mathrm{T}}$ has eigenvalues $1 + \|x\|_2^2$ and 1. By repeatedly applying this equality, we have that:

$$\det(V_t) = \det(V_0)\prod_{s=1}^{t}\left(1 + \|b_s\|^2_{V_{s-1}^{-1}}\right).$$

Therefore, we obtain

$$\frac{\det(V_t)}{\det(V_0)} = \prod_{s=1}^{t}\left(1 + \|b_s\|^2_{V_{s-1}^{-1}}\right). \tag{12}$$

Now, using Equation 12 and the fact that for any $x \geq 0$, $\min\{1, x\} \leq 2\log(1+x)$, we get the following:

$$\sum_{t=1}^{T}\min\left\{1\ ,\ \|b_t\|^2_{V_t^{-1}}\right\} \leq 2\sum_{t=1}^{T}\log\left(1 + \|b_t\|^2_{V_t^{-1}}\right) = 2\log\left(\frac{\det(V_t)}{\det(V_0)}\right).$$

This proves the first inequality in the lemma. For the second inequality, we use the inequality of arithmetic and geometric means. So, we have that:

$$\det(V_T) = \prod_{i=1}^{d}\lambda_i \leq \left(\frac{1}{d}\sum_{i=1}^{d}\lambda_i\right)^d = \left(\frac{1}{d}\mathrm{tr}(V_T)\right)^d \leq \left(\frac{\mathrm{tr}(V_0)+TM^2}{d}\right)^d,$$

where $\lambda_1, \dots, \lambda_d$ denote the eigenvalues of V_T, and the proof is complete. □

Now, we give the proof of Theorem 1.

Theorem 1. *Under the conditions of Assumption 1, with probability at least $1-\delta$, the expected regret of the RankUCB algorithm satisfies*

$$\mathcal{R}_T \leq 2\sqrt{2}(1 + \max_{l\in[L]}|w_l|)L\sqrt{dT\beta_T\log\left(1 + \frac{T\left((1+\max_{l\in[L]}|w_l|)m_1\right)^2}{d\lambda}\right)}, \tag{13}$$

where $\sqrt{\beta_T} = \max_{l\in[L]}\sqrt{\lambda}m_2 + \sqrt{2\log\left(\frac{1}{\delta}\right) + \log\left(\frac{\det\left(V_T^l(\lambda)\right)}{\lambda^d}\right)}$.

Proof. By Assumption 1, it is suffices to prove the bound on the event that for all $l \in [L]$, $\theta^l \in \mathcal{C}_t^l$. Let $a_\star = \text{argmax}_{a \in \mathcal{A}} \sum_{l=1}^{L} \langle \theta^l \ , \ v_{a^l} + w_l v_{a^{l-1}} \rangle$, and R_t be the instantaneous total regret in round t. Then,

$$R_t = \sum_{l=1}^{L} \langle \theta^l \ , \ v_{a_\star^l} + w_l v_{a_\star^{l-1}} \rangle - \sum_{l=1}^{L} \langle \theta^l \ , \ v_{a_t^l} + w_l v_{a_t^{l-1}} \rangle.$$

For each $l \in [L]$, let $\tilde{\theta}_t^l \in \mathcal{C}_t^l$ be the parameter for which $\langle \tilde{\theta}^l \ , \ v_{a_t^l} + w_l v_{a_t^{l-1}} \rangle = \text{UCB}_t^l(a_t^{l-1}, a_t^l)$. Now, the fact that $\theta^l \in \mathcal{C}_t^l$ and Equation 5 lead us to the following:

$$\begin{aligned} &\langle \theta^l \ , \ v_{a_\star^l} + w_l v_{a_\star^{l-1}} \rangle \leq \text{UCB}_t^l(a_\star^{l-1}, a_\star^l) \\ \implies &\sum_{l=1}^{L} \langle \theta^l \ , \ v_{a_\star^l} + w_l v_{a_\star^{l-1}} \rangle \leq \sum_{l=1}^{L} \text{UCB}_t^l(a_\star^{l-1}, a_\star^l). \end{aligned}$$

Note that $a_\star$ corresponds to a path in graph G of Algorithm 1, and since the longest path of graph G at round t has been a_t, we can write:

$$\sum_{l=1}^{L} \text{UCB}_t^l(a_\star^{l-1}, a_\star^l) \leq \sum_{l=1}^{L} \text{UCB}_t^l(a_t^{l-1}, a_t^l) = \sum_{l=1}^{L} \langle \tilde{\theta}^l \ , \ v_{a_t^l} + w_l v_{a_t^{l-1}} \rangle.$$

Therefore,

$$\begin{aligned} R_t &= \sum_{l=1}^{L} \langle \theta^l \ , \ v_{a_\star^l} + w_l v_{a_\star^{l-1}} \rangle - \sum_{l=1}^{L} \langle \theta^l \ , \ v_{a_t^l} + w_l v_{a_t^{l-1}} \rangle \\ &\leq \sum_{l=1}^{L} \langle \tilde{\theta}_t^l \ , \ v_{a_t^l} + w_l v_{a_t^{l-1}} \rangle - \sum_{l=1}^{L} \langle \theta^l \ , \ v_{a_t^l} + w_l v_{a_t^{l-1}} \rangle \\ &= \sum_{l=1}^{L} \langle \tilde{\theta}_t^l - \theta^l \ , \ v_{a_t^l} + w_l v_{a_t^{l-1}} \rangle \\ &\leq \sum_{l=1}^{L} \| v_{a_t^l} + w_l v_{a_t^{l-1}} \|_{V_{t-1}^{l-1}} \| \tilde{\theta}_t^l - \theta^l \|_{V_{t-1}^l}. \end{aligned}$$

The last line follows from the Cauchy-Schwartz inequality. By Assumption 1, we can write that $R_t \leq 2(1 + \max_{l \in [L]} |w_l|)L$. Hence,

$$R_t \leq \min \left\{ 2(1 + \max_{l \in [L]} |w_l|)L \ , \ \sum_{l=1}^{L} \| v_{a_t^l} + w_l v_{a_t^{l-1}} \|_{V_{t-1}^{l-1}} \| \tilde{\theta}_t^l - \theta^l \|_{V_{t-1}^l} \right\}$$

Using Equation 3 and Assumption 1, we have that:

$$\| \tilde{\theta}_t^l - \theta^l \|_{V_{t-1}^l} \leq 2 \left(\sqrt{\lambda} m_2 + \sqrt{2 \log \left(\frac{1}{\delta} \right) + \log \left(\frac{\det \left(V_{t-1}^l(\lambda) \right)}{\lambda^d} \right)} \right)$$

Thus,

$$\begin{aligned} R_t \leq 2(1 + \max_{l \in [L]} |w_l|) \sum_{l=1}^{L} &\left(\sqrt{\lambda} m_2 + \sqrt{2 \log \left(\frac{1}{\delta} \right) + \log \left(\frac{\det \left(V_{t-1}^l(\lambda) \right)}{\lambda^d} \right)} \right) \times \\ \min &\left\{ 1 \ , \ \| v_{a_t^l} + w_l v_{a_t^{l-1}} \|_{V_{t-1}^{l-1}} \right\} \end{aligned} \tag{14}$$

Moreover, the expected regret can be written as $\mathcal{R}_T = \mathbb{E} \left[\sum_{t=1}^{T} R_t \right]$, which can be upper bounded by Equation 14. Also, note that $\left\{ \det \left(V_t^l \right) \right\}_{t=0}^{T}$ is an increasing sequence.[1] Therefore, we can upper

[1]For more details, see the proof of Lemma 2.

bound the regret as follows:

$$\begin{aligned}
\mathcal{R}_T &= \mathbb{E}\left[\sum_{t=1}^{T} R_t\right] \\
&\leq 2(1+\max_{l\in[L]}|w_l|)\sum_{t=1}^{T}\sum_{l=1}^{L}\sqrt{\beta_T}\min\left\{1\ ,\ \|v_{a_t^l}+w_l v_{a_t^{l-1}}\|_{V_{t-1}^{l-1}}\right\} \\
&\leq 2(1+\max_{l\in[L]}|w_l|)\sqrt{LT\sum_{l=1}^{L}\sum_{t=1}^{T}\beta_T\min\left\{1\ ,\ \|v_{a_t^l}+w_l v_{a_t^{l-1}}\|^2_{V_{t-1}^{l-1}}\right\}}.
\end{aligned}$$

The last inequality follows from Cauchy–Schwartz inequality. Now, we can use Lemma 2 to upper bound $\sum_{t=1}^{T}\min\left\{1\ ,\ \|v_{a_t^l}+w_l v_{a_t^{l-1}}\|^2_{V_{t-1}^{l-1}}\right\}$. One can check that if we define $b_t = v_{a_t^l}+w_l v_{a_t^{l-1}}$, and $M=(1+\max_{l\in[L]}|w_l|)m_1$, then we can write:

$$\mathcal{R}_T \leq 2(1+\max_{l\in[L]}|w_l|)\sqrt{LT\beta_T\sum_{l=1}^{L}2d\log\left(\frac{\text{tr}(V_0^l)+T\left((1+\max_{l\in[L]}|w_l|)m_1\right)^2}{d\det\left(V_0^l\right)^{\frac{1}{d}}}\right)}.$$

By replacing $\text{tr}(V_0^l)=d\lambda$ and $\det(V_0^l)=\lambda^d$, we get the following bound:

$$\mathcal{R}_T \leq 2\sqrt{2}(1+\max_{l\in[L]}|w_l|)L\sqrt{dT\beta_T\log\left(1+\frac{T\left((1+\max_{l\in[L]}|w_l|)m_1\right)^2}{d\lambda}\right)}.$$

This completes the proof. □

B Proof of Theorem 2

We need the following corollary of Lemma 2to prove Theorem 2.

Corollary 1. *Let $V_0=\lambda I\in\mathbb{R}^{d\times d}$, and $b_1,\ldots,b_T\in\mathbb{R}^d$ be a sequence of vectors with $\|b_t\|_2\leq M<\infty$. For all $t\in[T]$, define $V_t=V_0+\sum_{s\leq t}b_s b_s^{\mathrm{T}}$. Then,*

$$\frac{\det\left(V_t(\lambda)\right)}{\lambda^d}\leq\left(\text{tr}\left(\frac{V_t(\lambda)}{\lambda d}\right)^d\right)\leq\left(1+\frac{nM^2}{\lambda d}\right)^2.$$

We can now give the proof of Theorem 2.

Theorem 2. *Under Assumption 2, the expected regret of the RankTS algorithm is bounded by:*

$$\mathcal{R}_T \leq 2L(1+\max_{l\in[l]}|w_l|)\left(1+\sqrt{2Td\beta^2\log\left(1+\frac{T\left((1+\max_{l\in[L]}|w_l|)m_1\right)^2}{d\lambda}\right)}\right) \tag{15}$$

where $\beta = 1+\sqrt{4\log(T)+d\log\left(1+\frac{T\left((1+\max_{l\in[L]}|w_l|)m_1\right)^2}{d\lambda}\right)}$.

Proof. Let us denote the set of the super-arms of set A by $\mathcal{S}(A)$. We start by defining upper confidence bound functions $U_t^l:\mathcal{S}(A)\mapsto\mathbb{R}$ for all $l\in[L]$ as follows:

$$U_t^l(i,j)=\langle\hat{\theta}_{t-1}^l\ ,\ v_j+w_l v_i\rangle+\beta\|v_j+w_l v_i\|_{V_{t-1}^{l-1}}$$

where $V_t^l=\frac{1}{m_2^2}I+\sum_{s=1}^{t}(v_{a_s^l}+w_l v_{a_s^{l-1}})(v_{a_s^l}+w_l v_{a_s^{l-1}})^{\mathrm{T}}$. By Lemma 1 and Lemma 2, and setting $\lambda=\frac{1}{m_2^2}$ and $\delta=\frac{1}{T^2}$, we have that $\mathbb{P}(\exists t\in[T]:\|\hat{\theta}_{t-1}^l-\theta^l\|_{V_{t-1}^l}>\beta)\leq\frac{1}{T^2}$. Let E_t^l be the event that $\|\hat{\theta}_{t-1}^l-\theta^l\|_{V_{t-1}^l}\leq\beta$, and define $E^l=\bigcap_{t=1}^{T}E_t$, $E=\bigcap_{l=1}^{L}E^l$, and $a_\star=$

$\text{argmax}_{a\in\mathcal{A}}\sum_{l=1}^{L}\langle\theta^l\ ,\ a^l+w_la^{l-1}\rangle$. Since $\{\theta^l\}_{l=1}^L$ are random, $a_\star$ is a random variable. Now, we can write the regret as follows:

$$\begin{aligned}\mathcal{R}_T &= \mathbb{E}\left[\sum_{t=1}^{T}\sum_{l=1}^{L}\langle\theta^l\ ,\ a_\star^l-a_t^l+w_l(a_\star^{l-1}-a_t^{l-1})\rangle\right]\\ &= \mathbb{E}\left[\mathbf{1}_E\sum_{t=1}^{T}\sum_{l=1}^{L}\langle\theta^l\ ,\ a_\star^l-a_t^l+w_l(a_\star^{l-1}-a_t^{l-1})\rangle\right]\\ &\quad+\mathbb{E}\left[\mathbf{1}_{E^c}\sum_{t=1}^{T}\sum_{l=1}^{L}\langle\theta^l\ ,\ a_\star^l-a_t^l+w_l(a_\star^{l-1}-a_t^{l-1})\rangle\right].\end{aligned}\tag{16}$$

Here $\mathbf{1}_E$ is the indicator function of event E. Now, for the second term which is on the event E^c, we can bound the term inside the expectation based on Assumption 2:

$$\begin{aligned}&\mathbb{E}\left[\mathbf{1}_{E^c}\sum_{t=1}^{T}\sum_{l=1}^{L}\langle\theta^l\ ,\ a_\star^l-a_t^l+w_l(a_\star^{l-1}-a_t^{l-1})\rangle\right]\\ &=\sum_{l=1}^{L}\mathbb{E}\left[\mathbf{1}_{E^{l^c}}\sum_{t=1}^{T}\langle\theta^l\ ,\ a_\star^l-a_t^l+w_l(a_\star^{l-1}-a_t^{l-1})\rangle\right]\\ &\le 2T(1+\max_{l\in[l]}|w_l|)\sum_{l=1}^{L}\mathbb{P}(E^{l^c}).\end{aligned}$$

The first line is due to the fact that events E^l for any $l\in[L]$ are independent because in Algorithm 2 we have that $(\hat\theta^1,\ldots,\hat\theta^L)\sim\mathcal{Q}^1(\cdot|\mathcal{H}_t)\otimes\ldots\otimes\mathcal{Q}^L(\cdot|\mathcal{H}_t)$. Now, for $\mathbb{P}(E^{l^c})$ we have that:

$$\mathbb{P}(E^{l^c})=\mathbb{P}(\bigcup_{t=1}^{T}E_t^{l^c})\le\sum_{t=1}^{T}\mathbb{P}(E_t^{l^c})\le T\frac{1}{T^2}=\frac{1}{T}.$$

Therefore, the second term of Equation 16 is bounded by $2L(1+\max_{l\in[l]}|w_l|)$. Now, for the first term we can write:

$$\begin{aligned}&\mathbb{E}\left[\mathbf{1}_E\sum_{t=1}^{T}\sum_{l=1}^{L}\langle\theta^l\ ,\ a_\star^l-a_t^l+w_l(a_\star^{l-1}-a_t^{l-1})\rangle\right]\\ &\le\mathbb{E}\left[\sum_{t=1}^{T}\sum_{l=1}^{L}\mathbf{1}_{E_t^l}\langle\theta^l\ ,\ a_\star^l-a_t^l+w_l(a_\star^{l-1}-a_t^{l-1})\rangle\right]\\ &=\mathbb{E}\left[\sum_{t=1}^{T}\sum_{l=1}^{L}\mathbb{E}\left[\mathbf{1}_{E_t^l}\langle\theta^l\ ,\ a_\star^l-a_t^l+w_l(a_\star^{l-1}-a_t^{l-1})\rangle|\mathcal{H}_t\right]\right].\end{aligned}\tag{17}$$

To bound this, note that for any $l\in[L]$ both θ^l and $\hat\theta_t^l$ are drawn from the same prior, which basically means that $\mathbb{P}(\theta^l\in\cdot|\mathcal{H}_t)=\mathbb{P}(\hat\theta_t^l\in\cdot|\mathcal{H}_t)$. Hence, we can conclude that $\mathbb{P}(a_\star=\cdot|\mathcal{H}_t)=\mathbb{P}(a_t=\cdot|\mathcal{H}_t)$ and $\mathbb{E}\left[U_t^l(a_\star^{l-1},a_\star^l)|\mathcal{H}_t\right]=\mathbb{E}\left[U_t^l(a_t^{l-1},a_t^l)|\mathcal{H}_t\right]$. Thus,

$$\begin{aligned}\mathbb{E}\left[\mathbf{1}_{E_t^l}\langle\theta^l\ ,\ a_\star^l-a_t^l+w_l(a_\star^{l-1}-a_t^{l-1})\rangle|\mathcal{H}_t\right]&=\mathbf{1}_{E_t^l}\mathbb{E}\left[\langle\theta^l\ ,\ v_{a_\star^l}+w_lv_{a_\star^{l-1}}\rangle-U_t^l(a_\star^{l-1},a_\star^l)\right]\\ &\quad+\mathbf{1}_{E_t^l}\mathbb{E}\left[U_t^l(a_t^{l-1},a_t^l)-\langle\theta^l\ ,\ v_{a_t^l}+w_lv_{a_t^{l-1}}\rangle\right]\\ &\le\mathbf{1}_{E_t^l}\mathbb{E}\left[U_t^l(a_t^{l-1},a_t^l)-\langle\theta^l\ ,\ v_{a_t^l}+w_lv_{a_t^{l-1}}\rangle\right]\\ &\le\mathbf{1}_{E_t^l}\mathbb{E}\left[\langle\hat\theta_{t-1}^l-\theta^l\ ,\ v_{a_t^l}+w_lv_{a_t^{l-1}}\rangle\right]\\ &\quad+\beta\|v_{a_t^l}+w_lv_{a_t^{l-1}}\|_{V_{t-1}^{l-1}}\\ &\le\mathbf{1}_{E_t^l}\mathbb{E}\left[\|\hat\theta_{t-1}^l-\theta^l\|_{V_{t-1}^l}\|v_{a_t^l}+w_lv_{a_t^{l-1}}\|_{V_{t-1}^{l-1}}\right]\\ &\quad+\beta\|v_{a_t^l}+w_lv_{a_t^{l-1}}\|_{V_{t-1}^{l-1}}\\ &\le2\beta\|v_{a_t^l}+w_lv_{a_t^{l-1}}\|_{V_{t-1}^{l-1}}.\end{aligned}$$

The second line is due to the fact that, by the definition of U_t^l functions, the first term of the first line is negative or zero. Now, we can bound the Equation 17 by noting that according to Assumption 2, $\mathbf{1}_{E_t^l}\langle\theta^l\ ,\ a_\star^l - a_t^l + w_l(a_\star^{l-1} - a_t^{l-1})\rangle \le 2(1 + \max l \in [L]|w_l|)$. Therefore, we have:

$$\mathbb{E}\left[\mathbf{1}_E \sum_{t=1}^T \sum_{l=1}^L \langle\theta^l\ ,\ a_\star^l - a_t^l + w_l(a_\star^{l-1} - a_t^{l-1})\rangle\right] \le 2\beta(1 + \max_{l\in[L]}|w_l|)\ \times$$
$$\mathbb{E}\left[\sum_{t=1}^T \sum_{l=1}^L \min\left\{1, \|v_{a_t^l} + w_l v_{a_t^{l-1}}\|_{V_{t-1}^{l-1}}\right\}\right].$$

Using Cauchy–Schwartz inequality and Lemma 2, we will have:

$$\begin{aligned}
\mathbb{E}\left[\sum_{t=1}^T \sum_{l=1}^L \min\left\{1, \|v_{a_t^l} + w_l v_{a_t^{l-1}}\|_{V_{t-1}^{l-1}}\right\}\right] &\le \sqrt{LT\mathbb{E}\left[\sum_{t=1}^T \sum_{l=1}^L \min\left\{1, \|v_{a_t^l} + w_l v_{a_t^{l-1}}\|^2_{V_{t-1}^{l-1}}\right\}\right]} \\
&\le \sqrt{LT\sum_{l=1}^L 2d\log\left(1 + \frac{T\left((1+\max_{l\in[L]}|w_l|)m_1\right)^2}{d\lambda}\right)} \\
&= L\sqrt{2Td\log\left(1 + \frac{T\left((1+\max_{l\in[L]}|w_l|)m_1\right)^2}{d\lambda}\right)}.
\end{aligned}$$

By substituting all the above bounds to Equation 16, we get the following bound and the proof is complete.

$$\mathcal{R}_T \le 2L(1 + \max_{l\in[l]}|w_l|)\left(1 + \beta\sqrt{2Td\log\left(1 + \frac{T\left((1+\max_{l\in[L]}|w_l|)m_1\right)^2}{d\lambda}\right)}\right).$$

□

C Generalized RankUCB algorithm

C.1 Algorithm

The generalized version of RankUCB algorithm is Algorithm 3, where we estimate $\phi^l = (\theta^l\ \ w_l\theta^l)$.

Algorithm 3 genRankUCB

```
Input: λ > 0, δ ∈ (0,1), L, T, arm set A = {1, ..., K}, and vector v_0
Create L-layered graph G = ⋃_{i=1}^K G_i over super-arms of set A
Initialization: φ̂_0^l = 0, Ṽ_0^l = λI for l ∈ [L], and for any edge e of G, set ĉ_e = 0
for t = 1, 2, ..., T do
    Obtain p_i ← ShortestPathAlgorithm(−G_i) for all i ∈ [K] simultaneously
    p_⋆ ← argmin_{p_i} Σ_{e∈p_i} ĉ_e
    Choose action a_t as the ordered vertices of path p_⋆
    Play a_t and observe r_{a_t}^l for l ∈ [L]
    for l = 1, ... L do
        Ṽ_t^l(λ) ← Ṽ_{t−1}^l + x̃_{a_s}^l x̃_{a_s}^{l T}
        φ̂_t^l ← Ṽ_t^l(λ)^{−1} [Σ_{s=1}^t r_{a_s}^l x̃_{a_s}^l]
        Create C̃_{t+1}^l based on Lemma 1
        UCB_{t+1}^l(i, j) ← max_{φ∈C̃_{t+1}^l} ⟨φ , x̃_{ji}⟩ for all super-arms (i, j)
        Update ĉ_e, for any edge e, based on Equation 6
    end for
end for
```

C.2 Proof of Theorem 3

Theorem 3. *Under the conditions of Assumption 3, with probability at least* $1-\delta$*, the expected regret of the genRankUCB algorithm satisfies:*

$$\mathcal{R}_T \leq 4(1+m_3)L\sqrt{dT\beta_T \log\left(1+\frac{2Tm_2^2}{d\lambda}\right)} \tag{18}$$

where $\sqrt{\beta_T} = \max_{l\in[L]} \sqrt{\lambda}\left(m_2\sqrt{1+m_3^2}\right) + \sqrt{2\log\left(\frac{1}{\delta}\right) + \log\left(\frac{\det\left(\tilde{V}_T^l(\lambda)\right)}{\lambda^{2d}}\right)}$.

Proof. The proof is similar to the proof of Theorem 1 in Appendix A. By Assumption 3, it is suffices to prove the bound on the event that for all $l \in [L]$, $\phi^l \in \tilde{\mathcal{C}}_t^l$. Let $a_\star = \operatorname{argmax}_{a\in\mathcal{A}} \sum_{l=1}^{L} \langle \phi^l\ ,\ x_a^l \rangle$, where $x_a^l = (v_{a^l}\ \ v_{a^{l-1}})^\mathsf{T}$, and R_t be the instantaneous total regret in round t. Then,

$$R_t = \sum_{l=1}^{L} \langle \phi^l\ ,\ x_{a_\star}^l \rangle - \sum_{l=1}^{L} \langle \phi^l\ ,\ x_{a_t}^l \rangle.$$

For each $l \in [L]$, let $\tilde{\phi}_t^l \in \tilde{\mathcal{C}}_t^l$ be the parameter for which $\langle \tilde{\phi}^l\ ,\ x_{a_t}^l \rangle = \mathrm{UCB}_t^l(a_t^{l-1}, a_t^l)$. Now, the fact that $\phi^l \in \tilde{\mathcal{C}}_t^l$ and Equation 10 lead us to the following:

$$\langle \phi^l\ ,\ x_{a_\star}^l \rangle \leq \mathrm{UCB}_t^l(a_\star^{l-1}, a_\star^l). \tag{19}$$

Using Equation 19 and the facts that $a_\star$ corresponds to a path in graph G of Algorithm 3, and the longest path of graph G at round t has been a_t, we can write:

$$\sum_{l=1}^{L} \langle \phi^l\ ,\ x_{a_\star}^l \rangle \leq \sum_{l=1}^{L} \mathrm{UCB}_t^l(a_\star^{l-1}, a_\star^l) \leq \sum_{l=1}^{L} \mathrm{UCB}_t^l(a_t^{l-1}, a_t^l) = \sum_{l=1}^{L} \langle \tilde{\phi}^l\ ,\ x_{a_t}^l \rangle.$$

Therefore,

$$\begin{aligned}
R_t &= \sum_{l=1}^{L} \langle \phi^l\ ,\ x_{a_\star}^l \rangle - \sum_{l=1}^{L} \langle \phi^l\ ,\ x_{a_t}^l \rangle \\
&\leq \sum_{l=1}^{L} \langle \tilde{\phi}_t^l\ ,\ v_{a_t}^l \rangle - \sum_{l=1}^{L} \langle \phi^l\ ,\ x_{a_t}^l \rangle = \sum_{l=1}^{L} \langle \tilde{\phi}_t^l - \phi^l\ ,\ x_{a_t}^l \rangle \\
&\leq \sum_{l=1}^{L} \|x_{a_t}^l\|_{\tilde{V}_{t-1}^{l-1}} \|\tilde{\phi}_t^l - \phi^l\|_{\tilde{V}_{t-1}^l} \\
&\leq \min\left\{2(1+m_3)L\ ,\ \sum_{l=1}^{L} \|x_{a_t}^l\|_{\tilde{V}_{t-1}^{l-1}} \|\tilde{\phi}_t^l - \phi^l\|_{\tilde{V}_{t-1}^l}\right\}.
\end{aligned}$$

The third line is followed by the Cauchy-Schwartz inequality, and the last line is due to Assumption 3, which bounds $R_t \leq 2(1+m_3)L$. Now, using Lemma 1 and Assumption 3, we have that:

$$\|\tilde{\phi}_t^l - \phi^l\|_{\tilde{V}_{t-1}^l} \leq 2\left(\sqrt{\lambda}\left(m_2\sqrt{1+m_3^2}\right) + \sqrt{2\log\left(\frac{1}{\delta}\right) + \log\left(\frac{\det\left(\tilde{V}_{t-1}^l(\lambda)\right)}{\lambda^{2d}}\right)}\right).$$

Thus,

$$\begin{aligned}
R_t \leq 2(1+m_3)\sum_{l=1}^{L}&\left(\sqrt{\lambda}\left(m_2\sqrt{1+m_3^2}\right) + \sqrt{2\log\left(\frac{1}{\delta}\right) + \log\left(\frac{\det\left(\tilde{V}_{t-1}^l(\lambda)\right)}{\lambda^{2d}}\right)}\right) \times \\
&\min\left\{1\ ,\ \|x_{a_t}^l\|_{\tilde{V}_{t-1}^{l-1}}\right\}.
\end{aligned} \tag{20}$$

Table 1: Average response-time (ART) for $d = 10$, $L = 4$, $T = 1e4$ and 100 runs

Algorithm	K	ART (ms)
baseline	10	8.43
baseline	100	730.88
RankUCB	10	8.85
RankUCB	100	780.02
RankTS	10	8.02
RankTS	100	729.57
genRankUCB	10	9.30
genRankUCB	100	801.89

Now, we can upper bound the expected regret $\mathcal{R}_T = \mathbb{E}\left[\sum_{t=1}^{T} R_t\right]$ by Equation 20. Noting that $\left\{\det\left(\tilde{V}_t^l\right)\right\}_{t=1}^{T}$ is an increasing sequence, we can write:

$$\begin{aligned}\mathcal{R}_T &= \mathbb{E}\left[\sum_{t=1}^{T} R_t\right] \\ &\leq 2(1+m_3)\sum_{t=1}^{T}\sum_{l=1}^{L}\sqrt{\beta_T}\min\left\{1\ ,\ \|x_{a_t}^l\|_{\tilde{V}_{t-1}^{l^{-1}}}\right\} \\ &\leq 2(1+m_3)\sqrt{LT\sum_{l=1}^{L}\sum_{t=1}^{T}\beta_T\min\left\{1\ ,\ \|x_{a_t}^l\|^2_{\tilde{V}_{t-1}^{l^{-1}}}\right\}}.\end{aligned}$$

The last claim follows from Cauchy–Schwartz inequality. Using Lemma 2 and defining $b_t = x_{a_t}^l = (v_{a_t^l}\ \ v_{a_t^{l-1}})^{\mathrm{T}}$, and $M = 2m_2$, we can write:

$$\mathcal{R}_T \leq 2(1+m_3)\sqrt{LT\beta_T\sum_{l=1}^{L}4d\log\left(\frac{\mathrm{tr}(\tilde{V}_0^l) + 4Tm_2^2}{2d\det\left(\tilde{V}_0^l\right)^{\frac{1}{2d}}}\right)}.$$

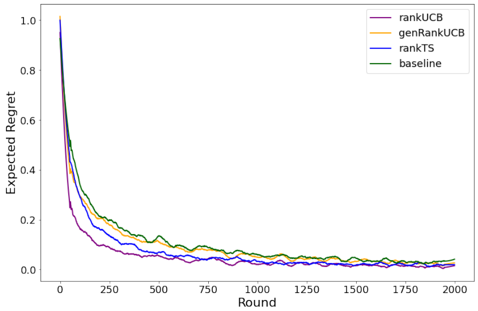
By replacing $\mathrm{tr}(\tilde{V}_0^l) = 2d\lambda$ and $\det(\tilde{V}_0^l) = \lambda^{2d}$, we get the following bound:

$$\mathcal{R}_T \leq 4(1+m_3)L\sqrt{dT\beta_T\log\left(1 + \frac{2Tm_2^2}{d\lambda}\right)}.$$

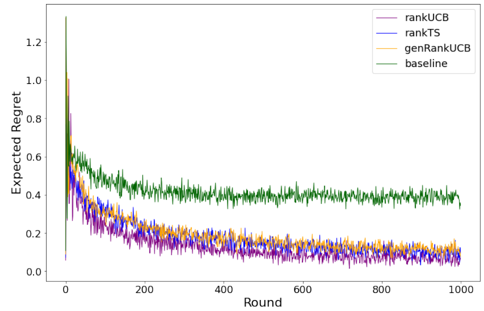
This completes the proof. $\square$

D More details on experiments

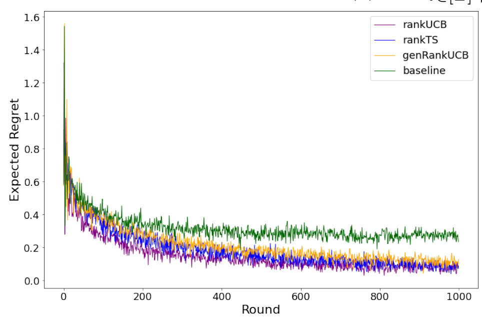
We conduct additional experiments to compare the algorithms. In Figure 3, the expected regret for all the four algorithms under different initial parameters is shown. As it was discussed in Section 6, all algorithms perform well when $\max_{l\in[L]}|w_l|$ is close to zero. This can be seen in Figures 2 (Left figure) and 3a. However, the baseline algorithm cannot capture the true behavior of the optimal action when $|w_l|$ becomes larger. Even in relatively small $\max_{l\in[L]}|w_l|$ like Figure 3c, the baseline algorithm converges to a non-optimal action. In contrast, the other three algorithms proposed in this paper follow the optimal regret. Note that the regret at each time step t is averaged over 100 runs.
As mentioned in Section 6, the Figures 2 and 3 are using the multivariate normal distribution as prior in RankTS, i.e. sampling $\hat{\theta}_t^l \sim \mathcal{N}(\mu_{t-1}^l, \Sigma_{t-1}^l)$ for each $\hat{\theta}^l$ separately. We assume the noise is Gaussian as well; therefore, the parameters of the normal distribution for the posterior can easily be



(a) $\max_{l\in[L]}|w_l| = 0.1$, $K = 10$



(b) $\max_{l\in[L]}|w_l| = 10$, $K = 10$



(c) $\max_{l\in[L]}|w_l| = 0.5$, $K = 100$

Figure 3: Expected regret for $d = 10$, and $L = 4$.

updated by the following equations:

$$\mu^l_{t-1} = \Sigma^{l-1}_{t-1}\left[\sum_{s=1}^{t} r^l_{a_s}(v^l_{a_s} + w_l v^{l-1}_{a_s})\right],$$
$$\Sigma^l_t = \Sigma^l_{t-1} + (v_{a^l_t} + w_l v_{a^{l-1}_t})(v_{a^l_t} + w_l v_{a^{l-1}_t})^{\mathrm{T}}.$$

It is noteworthy to mention that Thompson Sampling heavily relies on the prior distribution; a poor prior may prevent an arm from being played enough times, leading to linear regret. A detailed study of prior sensitivity is out of scope of this work. In addition, it can be challenging to find a practical example of this sensitivity.

The algorithms are straightforward to implement. Since the majority of computations in UCB and TS are matrix multiplications, they are very fast. However, when K is high, the shortest path algorithm becomes very slow. This is because shortest path algorithms are not optimized for a specific graph structure, which in our case is the L-layered graph. Changing to shortest path algorithms for sparse graphs, however, may speed up the process. An analysis of the run times of algorithms can be found in Table 1. Moreover, multiprocessing can be used to improve the run time of finding the shortest path to each induced subgraph of G_i, as defined in Section 2.3. The specifications of the system that generated the data for Table 1 are AMD Ryzen 5 5600x @ 3.7GHz. The importance of contextual bandit algorithms for practical applications such as recommendation systems and online advertising services makes the theoretical and practical investigation of the shortest path optimization problem essential. By applying algorithms with the shortest possible run time, societal negative impacts in the systems mentioned earlier can be mitigated.

Moreover, in Figure 4, we show that the bounds achieved by Theorems 1, 2, and 3 are quite tight, i.e., the cumulative expected regret would converge to the upper bound given in the corresponding theorems within a small coefficient factor.

Furthermore, it would be interesting to explore the robustness of these algorithms. Considering a small deviation from the main assumptions, such as non-linearity in the reward function, how well the algorithms would perform in finding the optimal action. For this case, we assume that the reward

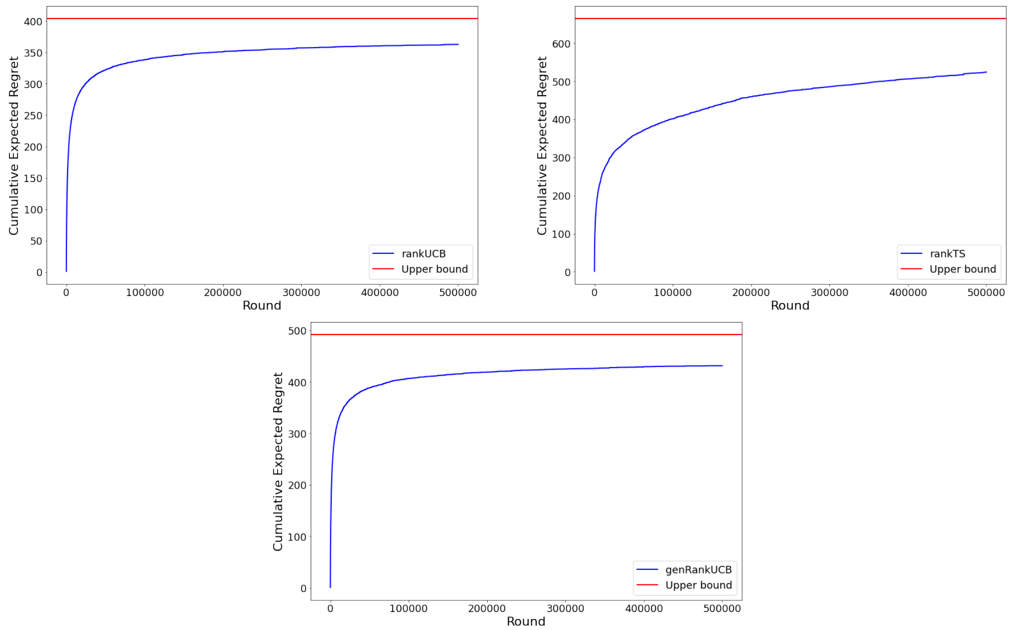


Figure 4: Tightness of the regret bounds. Cumulative expected regrets for $d = 10$, $L = 4$, $K = 10$ and $\max_{l \in [L]} |w_l| = 2$. The real environment's parameters is the same for all the subfigures.

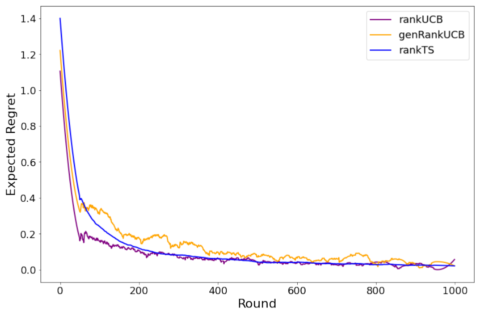
function for position l and action a is as follows:

$$r_a^l = f\left(\langle \theta^l \ , \ v_{a^l} + w_l v_{a^{l-1}} \rangle\right) + \eta^l$$
$$\text{where } f(x) = x + \exp(\epsilon x)$$

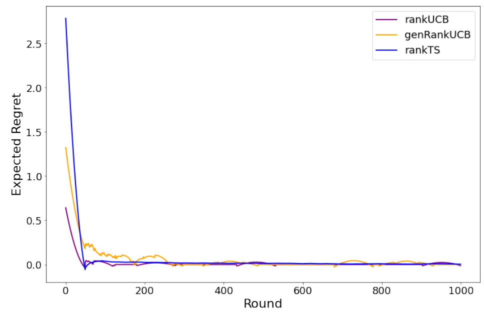
Another form of perturbation is when the noise is not subgaussian, for instance, when it is sampled from a Laplace distribution. In this case, we have the following changes:

$$r_a^l = \langle \theta^l \ , \ v_{a^l} + w_l v_{a^{l-1}} \rangle + \eta^l + \epsilon \tilde{\eta}^l$$
$$\text{where } \tilde{\eta}^l \sim \text{Laplace}(0, 1)$$

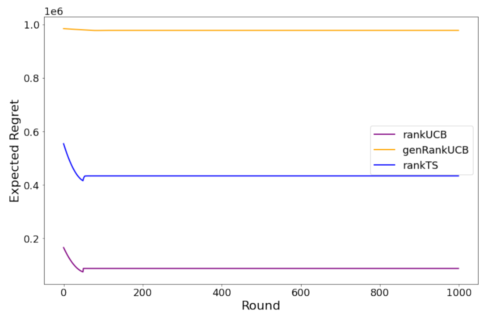
Here, η^l is a subgaussian noise that matches the main assumptions, and $\tilde{\eta}^l$ is a sample from a zero-mean Laplace distribution with scale 1. Figure 5 and 6 show the results. In both perturbation cases, algorithms can come close to the optimal action when the scale of perturbation is relatively small. However, a large perturbation scale might result in a potentially non-optimal action, resulting in linear regrets. The interesting point is that adding some perturbations seems to help the algorithms to converge faster, like Figures 5b and 6b.
The codes to reproduce the result are available at `https://anonymous.4open.science/r/rankingcontextualbandits-B313`.



(a) $\epsilon = 1\mathrm{e}{-4}$

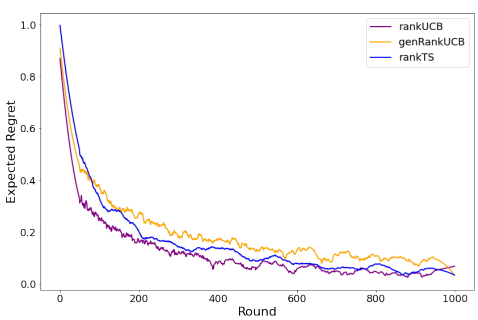


(b) $\epsilon = 0.1$

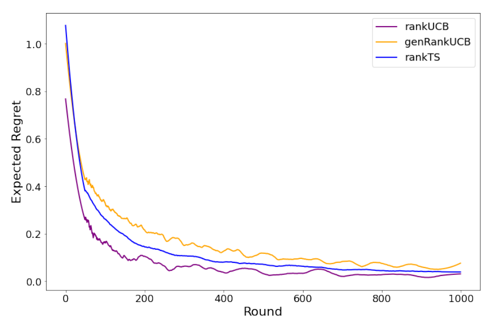


(c) $\epsilon = -2$

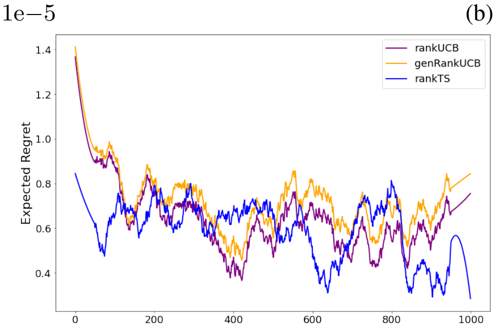
Figure 5: Robustness of algorithms in presence of nonlinearity in the reward function. $\max_{l\in[L]} |w_l| = 2$, and $K = 10$



(a) $\epsilon = 1\mathrm{e}{-5}$



(b) $\epsilon = 0.1$



(c) $\epsilon = 3$

Figure 6: Robustness of algorithms in presence non-subgaussian noise. $\max_{l\in[L]} |w_l| = 1$, and $K = 10$