Hierarchical Learning Algorithms for Multi-scale Expert Problems

LIN YANG, University of Massachusetts Amherst, USA
YU-ZHEN JANICE CHEN, University of Massachusetts Amherst, USA
MOHAMMAD H. HAJIESMAILI, University of Massachusetts Amherst, USA
MARK HERBSTER, University College London, UK
DON TOWSLEY, University of Massachusetts Amherst, USA

In this paper, we study the multi-scale expert problem, where the rewards of different experts vary in different reward ranges. The performance of existing algorithms for the multi-scale expert problem degrades linearly proportional to the maximum reward range of any expert or the best expert and does not capture the non-uniform heterogeneity in the reward ranges among experts. In this work, we propose learning algorithms that construct a hierarchical tree structure based on the heterogeneity of the reward range of experts and then determine differentiated learning rates based on the reward upper bounds and cumulative empirical feedback over time. We then characterize the regret of the proposed algorithms as a function of non-uniform reward ranges and show that their regrets outperform prior algorithms when the rewards of experts exhibit non-uniform heterogeneity in different ranges. Last, our numerical experiments verify our algorithms' efficiency compared to previous algorithms.

CCS Concepts: • **Computing methodologies** → **Online learning settings**; **Sequential decision making**.

Additional Key Words and Phrases: Multi-scale online learning, hedge model, expert problem

ACM Reference Format:
Lin Yang, Yu-zhen Janice Chen, Mohammad H. Hajiesmaili, Mark Herbster, and Don Towsley. 2022. Hierarchical Learning Algorithms for Multi-scale Expert Problems. *Proc. ACM Meas. Anal. Comput. Syst.* 6, 2, Article 34 (June 2022), 29 pages. https://doi.org/10.1145/3530900

1 INTRODUCTION

In the past few decades, a broad range of online learning problems has been studied and applied to various applications, such as online shortest path routing, online advertisement, channel allocation, recommender systems, etc. This paper studies an extension of the expert problem that deals with experts with scaled rewards in different ranges.

Prediction with Expert Advice (PEA) (a.k.a, the expert problem) has been extensively studied since the seminal work [16, 25, 35], and further developed in follow-up works [13, 34]. This paper focuses on the Hedge problem as a classic variant of PEA. The Hedge problem is a repeated game, where an online learner must make sequential decisions over T slots, choosing experts from a given expert set of size K. A series of rewards is associated with each expert, whose values are generated by an oblivious adversary from a given range, e.g., from $[0, 1]$ in the basic setting. At

Authors' addresses: Lin Yang, University of Massachusetts Amherst, USA, linyang@cs.umass.edu; Yu-zhen Janice Chen, University of Massachusetts Amherst, USA, yuzhenchen@cs.umass.edu; Mohammad H. Hajiesmaili, University of Massachusetts Amherst, USA, hajiesmaili@cs.umass.edu; Mark Herbster, University College London, UK, m.herbster@cs.ucl.ac.uk; Don Towsley, University of Massachusetts Amherst, USA, towsley@cs.umass.edu.

2476-1249/2022/6-ART34 $15.00
https://doi.org/10.1145/3530900

each slot, the online player selects an expert and earns the reward associated with the chosen expert. The feedback model is full information, which means that the online player observes the rewards of other experts after selecting an expert. The learner's goal is to minimize its regret with respect to the best expert. The most studied strategy for the above problem is called the Hedge algorithm [16, 17], which achieves the order-optimal regret of $O(\sqrt{T \log K})$ for the Hedge problem.

In this paper, we focus on a practically relevant variant of the Hedge problem, the multi-scale Hedge problem or MSHedge, for short. The basic Hedge problem assumes a homogeneous reward range for each expert, e.g., $[0, 1]$. In practice, however, there is a broad range of applications such as dynamic pricing, portfolio selection, etc. (see §2.3 for motivating examples), where the rewards of different experts are heterogeneous and scaled in different ranges. This motivates the model of our interest, which involves multi-scale experts, those who possess non-uniform reward ranges. In the MSHedge problem, the reward range of expert i is $[L_i, U_i]$, where U_i and L_i serve as the upper and lower bounds of rewards, whose values are known to the online player in advance. For ease of technical presentation, we consider two different models for MSHedge: (1) MSHedge-U, which only considers heterogeneity in upper bounds with lower bounds set to 0; and (2) MSHedge-LU, which allows both upper and lower bounds to be heterogeneous. It is straightforward to show that the naive extension of the Hedge algorithm to the multi-scale setting leads to a regret of $O(M\sqrt{T \log K})$, which linearly scales with M, the maximum reward upper bound among experts. The algorithm in [11] improves the regret to scale linearly proportional to the reward upper bound of the optimal expert instead of the largest reward upper bound among all experts. We review the most relevant literature in §2.4 and more extensively in §7.

Prior algorithms in the mainstream literature for the Hedge problem select experts in a probabilistic manner in each round, i.e., the higher the cumulative empirical reward of an expert, the higher the selection probability. However, this algorithmic idea ignores the impact of multi-scale reward upper bounds on how fast experts with larger cumulative rewards can be changed over time. More specifically, consider a scenario where the cumulative empirical reward of an expert with a large upper bound falls behind some others. Then, the large upper bound of this expert provides room for her to catch up with others quickly, and hence, the leading expert can be changed faster. We propose learning algorithms that explicitly capture this observation in the decision-making process, and we discuss them in the following.

1.1 Contributions

DRate*: A hierarchical learning algorithm with differentiated learning rates.* Our key idea is to explicitly capture the above intuitive observation into the algorithm design. Towards this, we propose to adaptively change the learning rate, and hence the selection probabilities of the expert, based on both the cumulative feedback and upper bound of the leading experts. We develop two learning algorithms based on Differentiated learning RATEs, called DRate-U and DRate-LU for short, which work within the MSHedge-U and MSHedge-LU models, respectively. For simplicity of presentation, we also refer to those two algorithms together as DRate.

To deal with the heterogeneity of multi-scale reward values, DRate partitions the set of experts into smaller subsets, placing experts with similar reward ranges in the same subset. DRate recursively continues to partition the experts into the new subsets and stops partitioning when the upper bounds in the subset are uniform. With a given tree, the decision-making process of DRate is as follows. At each round, DRate traverses the tree by recursively selecting nodes from the root to a leaf node and possibly running the Hedge algorithm in the selected leaf node, and eventually, returning an expert associated with the selected leaf node as its final decision.

The core idea of DRate is to determine *differentiated learning rates* within node algorithms. Generally, the learning rate determines the convergence speed of decisions to the *leading child node*, i.e., the node with the largest cumulative reward. Intuitively, the leading children nodes with different upper bounds have different impacts on the regret of a learning algorithm. Hence, the learning rate should be carefully tuned based on the upper bounds of the leading child node. Take MSHedge-U as an example. When the leading child node only contains experts with small upper bounds, an algorithm has high risk to encounter large regret loss, since the leading child node can be easily catch up with by the other one. In this case, a small or conservative learning rate is preferable. On the other hand, when the leading child node contains experts with large upper bounds, it usually takes longer for the other nodes to catch up. The algorithm may have more confidence in the leading node, and thus a higher or aggressive learning rate is more efficient. In §3, we present the details of DRate and rigorously explain the technical implementation of the intuitive idea of hierarchical partitioning of experts and calculating the differentiated learning rates to traverse the underlying tree.

Regret results for MSHedge-U. We first characterize the regret of DRate-U as a function of the path from root to the node that includes the best expert (Theorem 1). By proper parameter setting and with a balanced binary tree, we show that DRate-U achieves a regret of $O(\sqrt{U_1}\sum_{l=1}^{\log K}\sqrt{U_{2^l-1}T})$, where U_1, w.l.o.g, is the largest reward upper bound assuming a descending order of experts based on their upper bounds. Then in §4.3, we propose an algorithm that constructs an underlying tree that minimizes the regret characterized in Theorem 1. We also demonstrate the significance of the regret of DRate-U as compared to related results in §4.2. When most reward upper bounds are much smaller than the largest one, U_1, DRate-U attains a much smaller regret than the Hedge algorithm, whose regret is $O(U_1\sqrt{T\log K})$. This also improves the result in [11], which is $O(\max_{i\in\mathcal{K}} U_i\sqrt{T\log K})$ in the worst case. We finally derive regret lower bounds for a special case of MSHedge with only two experts and the general case with more than two experts. For the two-experts case, the lower bound is $\Omega(\sqrt{U_1U_2T})$, which matches the regret upper bound of DRate-U.

Regret results for MSHedge-LU. The regret of DRate-LU also depends on how the tree is constructed. Given a tree, DRate-LU provides a provable regret, which is the cumulative regret over the path from root to the node that includes the best expert (Theorem 4). Specifically, by placing experts with similar reward ranges into the same node, DRate-LU can reduce the region that the reward fluctuates in a node, as well as the regret. Note that we also discuss how our results applies to other variations of the expert problem such as the Lipschitz expert setting in §5.3.

Numerical results. Last, we evaluate the performance of DRate through numerical experiments in §6. Our numerical results using different heterogeneity scenarios of the multi-scale expert problem shows that DRate outperforms the Hedge [16, 17], a variant of the Hedge [14], and the algorithm in [11].

2 THE MULTI-SCALE EXPERT PROBLEM

In this section, we formally define the multi-scale expert problem, highlight a few motivating examples, and finally review the most relevant prior results for this problem. A summary of main notations in the system model and algorithms is given in Table 1.

2.1 Prediction with expert advice and the Hedge problem

Prediction with Expert Advice has been studied extensively by the community in past decades. The study of PEA dates back to early work by Littlestone, Warmuth, and Vovk [16, 25, 35], and was further developed in follow-up works [13, 34]. In this paper, we focus on an extended version of

Table 1. Summary of main notations related to MSHedge and DRate

Notation	Description
T	The number of time slots, indexed by t
$\mathcal{K}$	Set of all expert, indexed by i, and $K = \|\mathcal{K}\|$
$x_t(i)$	Reward of expert i at time slot t
U_i	Reward upper bound of expert i
L_i	Reward lower bound of expert i
I_t	The selected expert of the online player at time slot t
v	A node (either internal or leaf) in the decision tree of DRate
o	The root (sink) node of the decision tree in DRate
v_l	The left child of node v
v_r	The right child of node v
v_p^n	The n-th ancestor of node v; specifically, we simplify v's parent node, v_p^1, as v_p
v^*	The leaf node containing the optimal expert
$\texttt{path}(v)$	The set of nodes in the path from the root to node v
U_v	The largest reward upper bound of the experts in node v, i.e., $U_v = \max_{i \in v} U_i$
L_v	The smallest reward lower bound of the experts in node v, i.e., i.e., $L_v = \min_{i \in v} L_i$
$\hat{x}_t(v)$	Feedback of node v at time slot t
$\hat{X}_t(v)$	Cumulative feedback of node v up to time slot t
$\eta_1(v)$ and $\eta_2(v)$	The differentiated learning rates for node v
$\eta'(v)$	The learning rate of the Hedge algorithm in the leaf node v
$\alpha_t(v)$	The child of node v which has larger or equal cumulative reward at time t
$\beta_t(v)$	The child of node v which has smaller cumulative reward at time t
$p_t(v)$	The selection probability of node v at time slot t
$\hat{p}_t(v)$	The selection probability of child node v by node v_p at time slot t
$\hat{p}_t(i)$	The selection probability of expert i by the leaf node v containing expert i
R_T	Regret over T time slots
$R_T(v)$	Individual regret of node v over T time slots

the Hedge problem, as a classic variant of PEA. The Hedge problem is a repeated game, where an online learner is required to make sequential decisions, choosing experts from a given expert set of size K. We denote the expert set as $\mathcal{K} = [K]$. The game lasts T time slots. Associated with each expert i is a series of rewards, $x_i(t)$, $t = 1, 2, \ldots, T$, assigned over time, whose values are generated arbitrarily from a given range. In the standard Hedge setting, the range of the reward is usually $[0, 1]$, and rewards are generated by an oblivious adversary that is unaware of the actions of the online player. At time slot t, the online player selects an expert I_t, and earns the reward associated with the selected expert, i.e., $x_t(I_t)$. The feedback model is full information, which means that the online player can observe the rewards of every expert, $x_t(i)$, $i \in [K]$, after selecting an expert. The goal of the learner is to receive as much cumulative reward as possible, and her performance can be measured by comparing the cumulative rewards of the learner and that of the optimal expert, i.e., the one with the largest cumulative reward. To this end, we define pseudo-regret as

$$R_T := \max_{i \in \mathcal{K}} \sum_{t=1}^{T} x_t(i) - \mathbb{E}\left[\sum_{t=1}^{T} x_t(I_t)\right], \tag{1}$$

where the first term on the right-hand side expresses the cumulative reward of the best expert in hindsight, and the second one corresponds to the cumulative expected reward received by the learning algorithm. In the rest of the paper, we refer to pseudo-regret simply as regret. The most commonly studied strategy for the above problem is called the Hedge algorithm [16, 17], which achieves an order-optimal regret of $O(\sqrt{T \log K})$ for the Hedge problem. The Hedge problem is a

well-established framework for learning under uncertainty, and since the early works in the 1990s there has been substantial literature tackling the extended variants or developing better algorithms. We review the most relevant literature in §2.4 and more extensively in §7.

2.2 MSHedge: Hedge with multi-scale experts

The basic versions of the above framework assume a homogeneous reward range, $[0, 1]$ for each expert. In practice, however, there are a broad range of applications where the rewards of different experts are constrained in different ranges. This motivates the model of our interest, which involves multi-scale experts, those who possess non-uniform reward ranges.

In the Hedge problem with multi-scale experts, or MSHedge for short, the reward range of expert i is defined as $[L_i, U_i]$, where U_i and L_i serve as the upper and lower bounds of rewards, and satisfy $U_i > L_i \geq 0$. For ease of technical presentation of the proposed algorithms, we consider two different models for MSHedge: the first one, referred to as MSHedge-U, only involves heterogeneity in upper bounds, and we set $L_i = 0$, $i \in \mathcal{K}$; and the second one, referred to as MSHedge-LU, allows both upper and lower bounds to be non-uniform. We assume the bounds for those reward ranges are known to the learning algorithm as prior knowledge. In MSHedge, we continue to use the regret defined in Equation (1) as the performance metric for a learning algorithm. Yet, the regrets of learning algorithms in MSHedge are expected to depend on the non-uniform reward ranges, instead of a normalized uniform reward range in the standard Hedge setting, whose influence in regret is usually ignored. To distinguish our multi-scale setting from the basic uniform setting, we refer to the regret in the multi-scale case as the *non-uniform regret*.

2.3 Motivating examples for multi-scale experts

MSHedge can find many practical applications in real world and we highlight a few in the following.

The first application is online auction and dynamic pricing. The dynamic pricing problem has been extensively studied using a broad range of online learning tools including online algorithms, multi-armed bandits, and expert problem [10]. In its basic setting, upon arrival of a buyer, the online decision maker posts a price for an item, and the buyer buys the item only if the posted price is less than their private valuation. In the context of MSHedge, each possible posted price could be considered as an expert with possible rewards of either 0 when the buyers rejects the posted price, or $U > 0$, when the buyer buys the item.

Another example is portfolio selection [9, 11], each type of asset is modeled as an expert, and the learning algorithm decides on the portfolio of assets at each round with the aim to maximize earned profits. In practice, one usually observes heavy-tailed price fluctuations in a financial market, and the prices of different types of assets can vary in much different ranges. Hence, the online learning model must define different reward ranges for each expert.

The last example is the online learning algorithm in recommender systems that are allowed to provide each user with a set of advertisements with different weights, rather than individual one, and the total weights of the recommendations in the set must respect some constraints. For example, a webpage can only contain a finite number of advertisements, since the sum of the sizes of the advertisements have to respect the capacity of the webpage. The reward is proportional to the number of advertisements that the user clicks, and the upper reward limit of a set of recommendations is the sum of those for each individual advertisement in this set. Also, the reward ranges are known to the online learner, since they can observe the advertisements in each set.

2.4 The state-of-the-art results for MSHedge

An initial idea for extending the uniform reward range assumption in the basic Hedge model to multi-scale setting is to normalize the reward ranges based on the largest value and then study

how the normalization of this range impact the regret of existing algorithms. For example, scaling the rewards by $U = \max_i U_i$, yields a new reward range for rewards, $[0, U]$. Without any major modification, the algorithms, such as Hedge [6], developed for the $[0, 1]$ case guarantee the reward bound

$$X_T^* - \sqrt{UX_T^* \log K},$$

where X_T^* is the cumulative reward of the best expert over the time horizon, which could be as large as UT. Hence, the algorithm suffers a regret of $O(U\sqrt{T \log K})$ in the worst case, which degrades linearly with respect to the largest reward upper bound of experts. To improve this linear dependence on reward upper bound, there has been another study in expert problem with non-uniform reward ranges [10], where a model similar to our first model `MSHedge-U` is tackled. In [10] a modified definition of the regret is used to analyze the performance of the proposed learning algorithms. Specifically, with different reward ranges across experts, they define the action-specific regret, $R_{T,i}$, which is

$$R_{T,i} := \sum_{t=1}^{T} x_t(i) - \mathbb{E}\left[\sum_{t=1}^{T} x_t(I_t)\right]. \tag{2}$$

Specifically, we have $R_T = \max_{i \in \mathcal{K}} R_{T,i}$. Then, using the action-specific regret, a multi-scale learning algorithm is devised and its performance is demonstrated by the development of upper bounds of the action-specific regrets for each expert i. The regret of the proposed algorithm for expert i depends linearly on U_i, and thus the worst-case regret defined in Equation (1) depends on the reward upper bound of the best expert. While this is an improvement over dependence on the largest reward upper bound among any expert (either optimal or sub-optimal), in the worst case, where the best expert has the largest upper bound, the regret reduces to the standard result of the Hedge algorithm, i.e., $O(\max_{i \in \mathcal{K}} U_i \sqrt{T \log K})$. In this paper, we will develop learning algorithms that explicitly take into account different reward ranges of experts in their decision making.

Last, we note that in another direction, some works consider additional structured settings such as Lipschitz continuity for the expert problem, which implicitly introduce heterogeneity to the reward ranges. In particular, dependence or correlation can be introduced across the rewards of different experts, resulting in various fluctuations of the rewards. A classic example of the structured setting is the Lipschitz expert problem[27, 37], where the indices of experts lie in a metric space and the rewards of the experts have to satisfy the Lipschitz continuity condition. In §5.3, we compare our model with these structured settings. Specifically, we show that with some modifications of our proposed algorithm for the `MSHedge` model, we can extend our algorithms to capture the Lipschitz expert problem as well.

3 THE DRATE ALGORITHM

In this section, we introduce the `DRate-U` algorithm for the first model of `MSHedge`, where the reward upper bounds of experts scale in different values. The contribution of experts with heterogeneous reward upper bounds to regret is different, so, our high-level idea is to adopt a hierarchical learning policy to effectively capture the multi-scale upper bounds. More specifically, `DRate-U` leverages a tree structure to categorize the experts with different upper bounds, and tackles the original learning problem hierarchically by traversing through the constructed tree.

In Figure 1, we demonstrate the structure of the decision tree where the expert set $\mathcal{K}$ resides at the root and each internal node represents a subset of experts. Each node in the tree associated with more than one experts with different upper bounds can be further partitioned into two smaller subsets as its children. The algorithm may choose not to further partition the node when the upper bounds of the contained experts are the same. Hence, a leaf node may contain more than one expert.

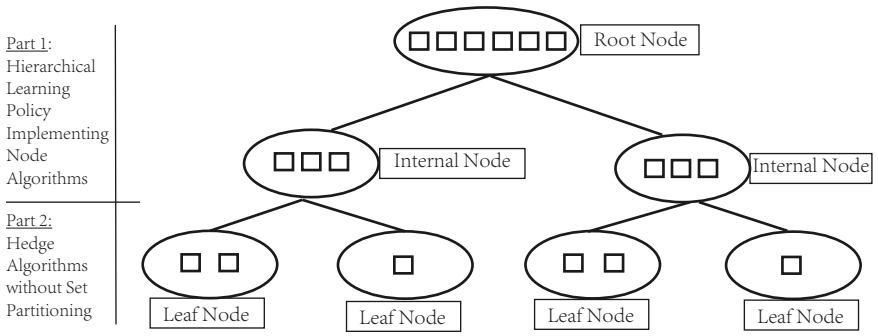


Fig. 1. The hierarchical structure of the DRate-U algorithm

The performance of DRate-U closely depends on how the tree is constructed. However, we first need to characterize the regret performance of the node algorithm in the non-leaf nodes, and then, based on the characterization of the regret, we can optimize the construction of the tree. Hence, we discuss the optimal tree construction given the expert upper bounds in §4.3.

With a given tree, the decision making process of DRate-U is as follows. At each time slot, DRate-U traverses the tree by recursively calling a node algorithm[1] from the root to a leaf node and possibly running the Hedge algorithm in the selected leaf node. Eventually, it returns an expert associated with the selected leaf node as its final decision. In DRate-U, the decision making in each non-leaf node can be considered as an independent *two-expert* problem, where children of the node are taken as super experts. The node algorithm plays a critical role in dealing with multi-scale experts in DRate-U, where a function parameterized by a learning rate is maintained to generate the selection probabilities of the child nodes. The node algorithm in DRate-U adjusts the learning rate based on the upper bounds of the leading child node, i.e., the one with the larger cumulative reward. We present the intuitions and details in §3.1.

By formal definition of the main notations in Table 1, we proceed to explain the the technical details of the algorithm. Let v denote a node (either internal or root) in the tree. The left and right children of node v are denoted as v_l and v_r, respectively.

At time slot t, DRate-U determines the selection probability of choosing each child of node v, which is denoted as $\hat{p}_t(v_l)$ and $\hat{p}_t(v_r)$, respectively. The values of $\hat{p}_t(v_l)$ and $\hat{p}_t(v_r)$ are determined based on past reward feedback, and their sum is one, i.e., $\hat{p}_t(v_l) + \hat{p}_t(v_r) = 1$. Given per-node selection probabilities, we can derive the actual selection probability of each node, denoted as $p_t(v)$, which is

$$p_t(v) = \prod_{v' \in \mathsf{path}(v)} \hat{p}_t(v'),$$

where $\mathsf{path}(v)$ denotes the set of nodes on the path from the root to node v.

When DRate-U commits to an expert, rewards associated with experts are revealed and DRate-U updates the feedback of each node. We denote $\hat{x}_t(v)$ to be the feedback of node v at time t and define it as

$$\hat{x}_t(v) := \sum_{i \in v} \frac{p_t(i)}{p_t(v)} x_t(i).$$

[1]In our terminology, the algorithm executed by a non-leaf node in the tree is called the node algorithm.

Note that $\hat{x}_t(v)$ is the expected reward conditioned on node v is selected. One can calculate $\hat{x}_t(v)$ recursively using feedback from the children, i.e.,

$$\hat{x}_t(v) = \hat{p}_t(v_l)\hat{x}_t(v_l) + \hat{p}_t(v_r)\hat{x}_t(v_r). \tag{3}$$

Last, let $\hat{X}_t(v)$ denote the cumulative feedback of node v up to time slot t, that is

$$\hat{X}_t(v) := \sum_{\tau=1}^{t} \hat{x}_\tau(v).$$

For consistency, we set $\hat{X}_0(v) = 0$.

3.1 Differentiated learning rates for the node algorithm

Now we focus on how to select a child node in the node algorithm of DRate-U, where the goal is to solve a two-expert problem. The idea in the mainstream literature is to select the leading expert, i.e., the expert with the larger cumulative feedback, with higher probability. However, this approach ignores the impact that multi-scale reward upper bounds can have on how fast the leading expert may be changed over time. In contrast, we propose to adaptively change the learning rate, and hence the selection probabilities, depending on both the cumulative feedback and upper bound of the leading expert. We first give an intuition and then formally present the technical idea.

Let $\alpha_t(v)$ be the child of node v with larger cumulative feedback at time t, and $\beta_t(v)$ be the other child, i.e., $\hat{X}_t(\alpha_t(v)) \geq \hat{X}_t(\beta_t(v))$. There are two cases to consider: Case-1 occurs when $\alpha_t(v)$ is the node with the larger upper bound, i.e., $U_{\alpha_t(v)} \geq U_{\beta_t(v)}$; and Case-2, when $\alpha_t(v)$ is the node with smaller upper bound than that of $\beta_t(v)$, i.e., $U_{\alpha_t(v)} < U_{\beta_t(v)}$. The key idea is to assign aggressive (or larger) learning rate to Case-1 and a conservative (or smaller) learning rate when facing Case-2. The reason is intuitive: with multi-scale upper bounds, an important observation is that when Case-2 occurs, i.e., the cumulative feedback of the expert with larger upper bounds falls behind, the risk of large regret is higher, since the larger upper bound provides a room for the expert to quickly catch up with the cumulative feedback of the other node. Thus, in Case-2, a small or conservative learning rate is preferable. On the other hand, in Case-1, i.e., when the expert with lower upper bound falls behind, it takes longer for it to catch up with the other one, so, it is safe to select a more aggressive learning rate.

With the above intuition, we proceed to formally determine the learning rates in the node algorithm of DRate-U. For node v, the node algorithm maintains two learning rates $\eta_1(v)$ and $\eta_2(v)$ associated with Case-1 and Case-2, respectively. During the learning process, the learning rates $\eta_1(v)$ and $\eta_2(v)$ may be alternatively adopted by DRate-U. Further, define $D_t(v) := \hat{X}_t(\alpha_t(v)) - \hat{X}_t(\beta_t(v))$, $t = 0, 1, 2, \ldots, T$, as the gap between the cumulative feedback of the two experts up to time t. Then, we set the selection probabilities for the two children of node v as follows

$$\hat{p}_{t+1}(\alpha_t(v)) = 1 - \left(\frac{1}{2} - D_t(v)\eta_z(v)\right)^+, \ \hat{p}_{t+1}(\beta_t(v)) = \left(\frac{1}{2} - D_t(v)\eta_z(v)\right)^+, \tag{4}$$

where $\eta_z(v)$, $z \in \{1, 2\}$ represents the learning rates used in the two cases and plays a critical role in determining the speed at which the node algorithm converges to the better child. The larger $\eta_z(v)$ is, the faster the algorithm converges to the leading child. In our analysis, we derive values $\eta_1(v)$ and $\eta_2(v)$ that optimize the regret.

3.2 The Hedge algorithm in leaf nodes with multiple experts

Once a leaf node with more than one experts is selected, DRate-U calls the Hedge algorithm to select an expert in the leaf node as the final decision. For any leaf node v with more than one expert,

Algorithm 1 Node algorithm for node v at time t (subscript t is dropped)

Initialization: $\alpha(v) = v_l$, $\beta(v) = v_r$, $\eta_1(v) > 0$, $\eta_2(v) > 0$, $\hat{X}(v_l) = 0$, $\hat{X}(v_r) = 0$
if node v is selected by v_p **then** ▷ *Select a child node*
  if $U_{\alpha(v)} \geq U_{\beta(v)}$ **then** ▷ *represents* Case-1
    $\eta_z(v) \leftarrow \eta_1(v)$
  end if
  if $U_{\alpha(v)} < U_{\beta(v)}$ **then** ▷ *represents* Case-2
    $\eta_z(v) \rightarrow \eta_2(v)$
  end if
  Select a child node with probability $\hat{p}_t(\alpha(v))$ and $\hat{p}_t(\beta(v))$, whose values are given in Equation (4)
end if
Receive feedback from node v's children, $\hat{x}_t(v_l)$ and $\hat{x}_t(v_r)$ ▷ *Update feedback of nodes*
Update $\hat{X}(v_l)$ and $\hat{X}(v_r)$: $\hat{X}(v_l) \leftarrow \hat{X}(v_l) + \hat{x}_t(v_l)$, $\hat{X}(v_r) \leftarrow \hat{X}(v_r) + \hat{x}_t(v_r)$
Calculate the feedback of node v, $\hat{x}_t(v)$, based on Equation (3), which will be used to calculate the feedback of node v_p
if $\hat{X}(\alpha(v)) < \hat{X}(\beta(v))$ **then**
  Swap the values of $\alpha(v)$ and $\beta(v)$
end if

DRate-U maintains a series of weights $w_t(i)$, $i \in v$. For convenience, we set $w_0(i) = 1$. At each time slot t, DRate-U normalizes the rewards observed on each expert and update the weights as follows.

$$w_t(i) = w_{t-1}(i) \cdot \exp(\eta'(v) x_t(i)/U_i),$$

where $\eta'(v) > 0$ is the learning rate in the Hedge algorithm. Then, the selection probability of expert i in node v at time slot $t+1$ is

$$\hat{p}_{t+1}(i) = \frac{w_t(i)}{\sum_{i' \in v} w_t(i')}, \ i \in v.$$

For consistency, we set $\hat{p}_t(i) = 1$ if there is only one expert in a leaf node. Let v be the leaf node that contains expert i. Then, the actual selection probability of expert i at time slot t is

$$p_t(i) = p_t(v)\hat{p}_t(i) = \left(\prod_{v' \in \mathsf{path}(v)} \hat{p}_t(v') \right) \hat{p}_t(i).$$

Last, we recall that our discussion on how to design a decision tree for the DRate-U algorithm is given in §4.3. Our regret analysis in the next section characterizes the regret of DRate-U as a function of the regret introduced by the nodes in $\mathsf{path}(v^*)$ (see Lemma 1). Thus, the optimal construction of the underlying tree should minimize the aggregate regret over different paths.

4 REGRET ANALYSIS FOR DRATE-U

We first state our main results and remarks for the regret of DRate-U in §4.1. In §4.2, we provide a few examples to clarify the significance of the regret of DRate-U. The regret of DRate-U depends on how the underlying tree structure is constructed. In §4.3, we present an algorithm for the optimal construction of the tree.

4.1 Main results and remarks

We define the notion of *node regret* of DRate-U. In the following, non-leaf nodes refer to either the root or internal nodes. All proofs are given in the appendix of the paper.

Definition 1. *(Node Regret) Given a tree for* DRate-U*, define node regret $R_T(v)$ for node v over horizon T as*

(1) If v is a non-leaf node,

$$R_T(v) := \max\left\{\sum_{t=1}^{T}\hat{x}_t(v_l), \sum_{t=1}^{T}\hat{x}_t(v_r)\right\} - \sum_{t=1}^{T}\left(\hat{p}_t(v_l)\hat{x}_t(v_l) + \hat{p}_t(v_r)\hat{x}_t(v_r)\right). \tag{5}$$

(2) If v is a leaf node,

$$R_T(v) := \begin{cases} \max_{i\in v}\sum_{t=1}^{T} x_t(i) - \sum_{t=1}^{T}\sum_{i\in v}\hat{p}_t(i)x_t(i), & |v| > 1; \\ 0, & |v| = 1 \end{cases} \tag{6}$$

The first term on the right-hand side of Equation (5) corresponds to the maximum cumulative feedback of the children of node v, and the second term corresponds to the cumulative expected feedback of node v. Generally, the node regret of node v characterizes the speed that its decisions converge to the child generating the larger cumulative feedback.

The following lemma implies that the regret of DRate-U is upper bounded by the sum of the node regrets on the path from the root node to the leaf node which contains the optimal expert.

Lemma 1. *The regret of* DRate-U *satisfies*

$$R_T \le \sum_{v\in\mathsf{path}(v^*)} R_T(v).$$

With Lemma 1, the remaining analysis is to upper bound the node regret for each non-leaf node and the leaf node containing the optimal expert, respectively.

Let U_v be the maximum reward upper bound of the experts in node v, i.e., $U_v := \max_{i\in v} U_i$. In the following proposition, we provide an upper bound for the node regret of any non-leaf node v, which depends on the largest reward upper bounds of v's children, i.e., U_{v_l} and U_{v_r}.

Proposition 1. *(Node Regret of a Non-leaf Node) Let v be a non-leaf node of the tree of* DRate-U*, assume $U_{v_l} \ge U_{v_r}$. With learning rates $\eta_1(v) > 0$ and $\eta_2(v) > 0$, we have*

$$R_T(v) \le \frac{1}{\eta_1(v)} + TU_{v_r}^2\eta_1(v) + \frac{1}{\eta_2(v)} + 2TU_{v_l}U_{v_r}\eta_2(v) + U_{v_l} + U_{v_r}.$$

Proposition 2. *(Node Regret of a Leaf Node, **Theorem 1.5** in [18]) Let v be a leaf node in the tree of* DRate-U *and $|v| > 1$. With learning rate $\eta'(v) > 0$, we have*

$$R_T(v) \le \eta'(v)U_v^2T + \frac{\log|v|}{\eta'(v)}.$$

By assuming, without loss of generality, $U_{v_l} \ge U_{v_r}$ holds for every non-leaf node v in the constructed tree, and combining propositions 1 and 2, and Lemma 1, we present the main result.

Theorem 1. *(Regret of* DRate-U*) With learning rates $\eta_1(v) > 0$ and $\eta_2(v) > 0$ for non-leaf node v and $\eta'(v') > 0$ for leaf node v', the regret of* DRate-U *satisfies the following upper bound.*

$$R_T \le \sum_{v\in\mathsf{path}(v^*)\setminus\{v^*\}}\left(\frac{1}{\eta_1(v)} + TU_{v_r}^2\eta_1(v) + \frac{1}{\eta_2(v)} + 2TU_{v_l}U_{v_r}\eta_2(v) + U_{v_l} + U_{v_r}\right) + Q_1,$$

where $Q_1 = \eta'(v^*)U_{v^*}^2T + \frac{\log|v^*|}{\eta'(v^*)}$, if $|v^*| > 1$; 0, otherwise.

The following corollary gives the result of DRate-U under the learning rates minimizing the upper bound in Theorem 1.

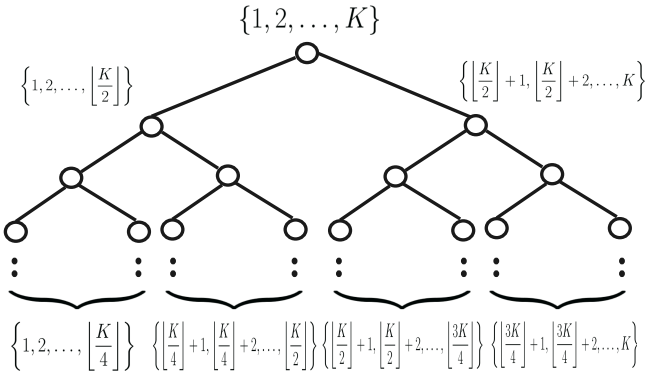


Fig. 2. Balanced binary tree for DRate-U.

Corollary 1. *With* $\eta_1(v) = 1/(U_{v_r}\sqrt{T})$, $\eta_2(v) = 1/\sqrt{2U_{v_l}U_{v_r}T}$, *and* $\eta'(v') = \sqrt{\log|v'|/T}/U_{v'}$, *the regret of* DRate-U *satisfies the following upper bound.*

$$R_T \leq \sum_{v\in \mathsf{path}(v^*)\setminus\{v^*\}} \left(\sqrt{2U_{v_l}U_{v_r}T} + U_{v_r}\sqrt{T}\right) + U_{v^*}\sqrt{T\log|v^*|}.$$

The above results imply that the regret of DRate-U depends on the way that the tree is constructed. In the following corollary, we further specify the regret of DRate-U when the expert set is fully partitioned into a balanced binary tree as shown in Figure 2.

Corollary 2. *(Regret of* DRate-U *with a Balanced Binary Tree) Given a balanced binary tree in* DRate-U, *assuming* $U_1 \geq U_2 \geq \cdots \geq U_K$, *and with* $\eta_1(v) = 1/(U_{v_r}\sqrt{T})$, $\eta_2(v) = 1/\sqrt{2U_{v_l}U_{v_r}T}$, *the regret of* DRate-U *satisfies*

$$R_T = O\left(\sqrt{U_1}\sum_{l=1}^{\log K}\sqrt{U_{2^l-1}T}\right).$$

Remark 1. *(Comparison with the Hedge Algorithm [18]) We first clarify that the regret in Corollary 2 is not always better than the optimal result by Hedge. For example, when all upper bounds are all equal to* U, DRate-U *yields a regret of* $O(U\sqrt{T}\log K)$, *while Hedge achieves a better regret of* $O(U\sqrt{T\log K})$. *However,* DRate-U *achieves much better performance than Hedge in heterogeneous settings with non-uniform upper bounds across experts. We further scrutinize this claim by providing several examples of non-uniform upper bound structures in §4.2. A promising future direction is to develop an algorithm that can simultaneously achieve the optimal regret bound under both uniform and non-uniform settings.*

Remark 2. *(Comparison with Multi-scale Learning Algorithm in [11]) The proposed algorithm in [11], analyzes the results for an action-specific regret defined in Equation (2) through a multi-scale learning algorithm. Specifically, the action-specific regret for expert* i *is* $O(U_i\sqrt{T\log(KT)})$. *Hence, the real regret of the multi-scale learning algorithm proposed in [11] is* $O(U_{i^*}\sqrt{T\log(KT)})$, *which can be as large as* $O(\max_{i\in[K]} U_i\sqrt{T\log(KT)})$ *in the worst case, similar to the result by Hedge. Hence, the similar arguments for non-uniform upper bounds in Remark 1 could be applied to the algorithm proposed in [11]. Note that in addition to more analytical examples in §4.2, we numerically compare the performance of these algorithms with* DRate-U *in §6.*

In the following, we present the lower bounds for MSHedge. We first present the lower bound for a special case with two experts.

Theorem 2. *Consider a two-expert problem, where the upper bounds is U_1 and U_2, respectively. The regret of any algorithm is $\Omega\left(\sqrt{U_1U_2T}\right)$.*

Combined with Corollary 1, we observe that the above bound is tight for the two expert case. It is much more complicated to analyze the lower bound in the general case, due to the difficulty in summarizing the lower bounds with different values of the upper bounds. In the following, we provide a (loose) lower bound for the general MSHedge problem with more than two experts.

Theorem 3. *Consider the* MSHedge *problem with K experts. For any $1 \le i \le K$, the regret of any algorithm is $\Omega\left(\frac{1}{\mathtt{num}(U_i)}\sum_{i':U_{i'}\ge U_i}\sqrt{U_iU_{i'}T\log \mathtt{num}(U_i)}\right)$, where $\mathtt{num}(U_i)$ denotes the number of experts whose upper bounds are larger than or equal to U_i.*

Remark 3. *Assume $U_1 \ge U_2 \ge \cdots \ge U_{K-1} \ge U_K$. A straightforward lower bound for the* MSHedge *algorithm is $O(U_k\sqrt{T\log K})$. Theorem 3 improves the naive lower bound, since we can easily derive a tighter lower bound from it, which is $\Omega(\frac{1}{K}\sum_{i=2}^{K}\sqrt{U_iU_KT\log K})$. Also, in the uniform setting with reward ranges in $[0, U]$ for each expert, the above lower bound becomes tight.*

4.2 Additional examples on the impact of tree structure on the regret

In this section, we provide two special instances of MSHedge with particular structures on non-uniform reward upper bounds of experts, and show how they require different tree structures to achieve the optimal value of regret for DRate-U as characterized in Theorem 1.

Example 1. *Consider a special instance of* MSHedge-U *with K experts, with the first expert with upper bound $U \gg 1$, and the remaining $K-1$ experts with upper bound of one.*

To construct the optimal tree, we place the first expert into one set as the left child of the root node and put the other $K-1$ experts as the right child of the root without further partitioning it into smaller nodes, since the contained experts have identical reward ranges. Then, a node algorithm is implemented in the root node and a Hedge algorithm in its right child, respectively. The node algorithm in the root node uses differentiated learning rates, with an aim to select a child node; and the Hedge algorithm selects an expert from the right child.

With $\eta_1(o) = 1/\sqrt{T}$ and $\eta_2(o) = 1/\sqrt{2UT}$, the node regret of the root node o is $O(\sqrt{UT})$. Further, the node regret of o_r is $O(\sqrt{T\log(K-1)})$, as the direct result of the regret of Hedge. By Lemma 1, the regret of DRate-U with a two-layer tree is the sum of the node regrets, and thus the regret of DRate-U is $O(\sqrt{UT}+\sqrt{T\log(K-1)})$. With sufficiently large U, the regret is dominated by the node regret of the root, i.e., $O(\sqrt{UT})$. For comparison, the regret of Hedge in this case is $O(U\sqrt{T\log K})$ for this special case; hence, DRate-U outperforms Hedge. We also note that combining Theorem 2 and the lower bound result for the basic Hedge problem, yields a regret of any algorithm for the above setting to be $\Omega(\sqrt{UT}+\sqrt{T\log(K-1)})$, which implies that the regret of DRate-U is order optimal.

In the above example, there is only one expert whose upper bound is different from others and we can achieve the best regret of DRate-U by a hierarchical learning structure with only two layers. When there is greater heterogeneity in the reward upper bounds, more layers have to be added to the tree. In the following, we show another example where the upper bounds of experts differ from any other's and DRate-U achieves the best result with many more layers.

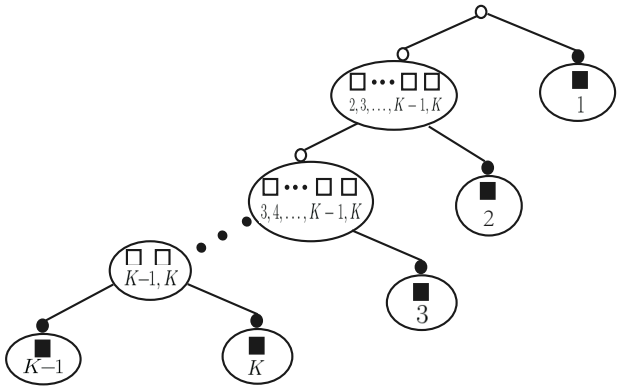


Fig. 3. Unbalanced tree constructed for Example 2.

Example 2. *Consider a special case of* MSHedge-U *with K experts with different reward upper bounds in a descending order. With* $U > 1$ *and* $a < 1$*, the upper bounds are set as* $U_1 = a^0U, U_2 = a^1U, \ldots, U_{K-1} = a^{K-1}U, U_K = a^KU$.

For the above special instance of MSHedge, we construct an unbalanced tree as shown in Figure 3. In this tree, the right child all non-leaf nodes is always one expert. At the l-th layer, the upper bound of the rewards of the experts associated with the node on the right hand side is $a^{l-1}U$. Given this tree construction, by applying the results in Corollary 2, DRate-U guarantees the following regret upper bound.

$$R_T = O\left(\sqrt{U}\sum_{l=1}^{K+1}\sqrt{U}a^{\frac{l-1}{2}}\sqrt{T}\right) = O\left(U\cdot\frac{1-\sqrt{a}^{K+1}}{1-\sqrt{a}}\sqrt{T}\right) \le O\left(\frac{U}{1-\sqrt{a}}\sqrt{T}\right).$$

We first compare the above regret with that of the Hedge algorithm and the result in [11]. The above regret holds for any $K > 1$, and thus it implies that DRate-U attains a regret of $O(U\sqrt{T})$ for an arbitrary number of experts. When $1-\sqrt{a}$ is not too small, DRate-U enjoys substantial improvement over the Hedge algorithm and the result in [11], whose regrets are $O(U\sqrt{\log(K)T})$ in the worst case. For a two-expert problem with upper bounds being U and aU, the lower bound of the regret for any algorithm is $\Omega(aU\sqrt{T})$ [11]. This also serves as a lower bound of the regret for our example, since our example involves an extended expert set. Ignoring the constant factor a, the above regret for DRate-U is order optimal.

4.3 Optimal tree construction

In the previous examples, we outlined two special cases of MSHedge-U and showed how the non-uniform structure of expert upper bounds can lead to different underlying trees that minimize the regret of DRate-U based on the main result in Theorem 1. In this section, we present an algorithm to generate the optimal tree for any instance of MSHedge-U given the reward upper bounds.

Let $R_T(\phi)$ be the regret of DRate-U with a tree ϕ. From Lemma 1, we have

$$R_T(\phi) \le \max_{v\in\texttt{leafnodes}}\sum_{v'\in\texttt{path}(v)}R_T(v') \le \max_{v\in\texttt{leafnodes}}\sum_{v'\in\texttt{path}(v)\setminus\{v\}}\left(\sqrt{2U_{v'_l}U_{v'_r}T}+U_{v'_r}\sqrt{T}\right)+U_v\sqrt{T\log|v|}.$$

The above equation suggests that a good way to guarantee a small regret for DRate-U is to balance the cumulative node regrets across the paths from the root node to leaf nodes. With this intuition, we present a simple algorithm to construct the optimal tree.

Algorithm 2 optimal_regret(v)

$r = U_v\sqrt{\log|v|}$
if $v = \emptyset$ **then**
  return 0, $\emptyset$, $\emptyset$
end if
for any $v' \subset v$ and $v' \neq \emptyset$ **do**
  temp $\leftarrow \max\{\text{optimal\_regret}_1(v'), \text{optimal\_regret}_1(v\backslash v')\} + \sqrt{2U_{v'}U_{v\backslash v'}} + \min\{U_{v'}, U_{v\backslash v'}\}$
  if temp $< r$ **then**
    $r \leftarrow$ temp
    $v_l \leftarrow v'$
  end if
end for
if $r = U_v\sqrt{\log|v|}$ **then** ▷ *Compare with the case of no splitting*
  return $U_v\sqrt{\log|v|}$, v, $\emptyset$
end if
return r, v_l, $v\backslash v_l$

Define the function optimal_regret(v), implemented in a recursive manner. Given a node v, optimal_regret(v) returns the optimal regret for the tree with v being the root. Specifically, optimal_regret(v) returns three output arguments in order: the smallest cumulative regret from v to v^* (optimal_regret(o) corresponds to the regret of DRate-U), the optimal left child and the optimal right child. When optimal_regret(v) returns empty set, it implies DRate-U cannot gain from adding additional levels by splitting v and thus we will add v to the tree as the leaf node. For simplicity, we use optimal_regret_i(v) ($i = 1, 2, 3$) to denote the i-th returned argument. The recursive procedure of optimal_regret(v) is formally defined in Algorithm 2. Since for each node v, optimal_regret(v) exhaustively compares the optimal regrets of all possible partitions on the experts in v (include no partition), it returns the optimal partition as children of node v. Note that the maximum cumulative regret on the path from node v to a leaf node is equal to the the node regret of v plus the larger one of the maximum cumulative regret on the path from v_l and v_r. Hence, by recursively calling function optimal_regret(v), one gets the optimal children of each node, and eventually the optimal tree.

5 MSHedge WITH NON-UNIFORM UPPER AND LOWER BOUNDS

In this section, we study the MSHedge-LU, where both lower and upper bounds of the rewards are in different scales, i.e, the reward range of the i-th expert is within $[L_i, U_i]$. We propose DRate-LU as an extended version of DRate-U to deal with the new challenge in the model.

5.1 The DRate-LU algorithm

Similar to DRate-U, DRate-LU also uses a binary decision tree, which is constructed exactly in the same way as that of DRate-U. Hence, we skip the details on how to construct the decision tree for DRate-LU, and one can refer to §4.3 for more details.

Associated with each node, DRate-LU also assigns a randomized node algorithm to each non-leaf node, and a Hedge algorithm to each leaf node with more than one expert. In non-leaf nodes, DRate-LU uses the function as defined in Equation (4) to determine the selection probabilities, but it redefines the per-time-slot expected feedback. Let v be any node except the root node. The expected feedback of node v at time slot t, $\hat{x}_t(v)$, is redefined as

$$\hat{x}_t(v) := \sum_{i\in v} \frac{p_t(i)}{p_t(v)} \left(x_t(i) - L_{v_p}\right) = \sum_{i\in v} \frac{p_t(i)}{p_t(v)} x_t(i) - L_{v_p},$$

where L_v is the smallest reward lower bound of the experts in node v, i.e., $L_v = \min_{i \in v} L_i$. By the above definition, we can derive a range for $\hat{x}_t(v)$, which is $\left[0, \hat{U}_v\right]$, where $\hat{U}_v := U_v - L_{v_p}$. Note that, $\hat{U}_v$ represents the gap between the largest and smallest rewards of the experts in node v, and thus reflects the fluctuation of the reward associated with the experts in node v. Similarly, we define $\hat{X}_t(v) := \sum_{\tau=1}^{t} \hat{x}_\tau(v)$ as the cumulative reward of node v up to time t (set $\hat{X}_0(v) = 0$) and $D_t(v) := \hat{X}_t(\alpha_t(v)) - \hat{X}_t(\beta_t(v))$ as the gap between the cumulative rewards of the children nodes of v up to time t. With this redefinition of notation and similar to DRate-U, DRate-LU calculates the selection probabilities for the children of node v as follows.

$$\hat{p}_t(\alpha_t(v)) = 1 - \left(\frac{1}{2} - D_t(v)\eta_z(v)\right)^+, \ \hat{p}_t(\beta_t(v)) = \left(\frac{1}{2} - D_t(v)\eta_z(v)\right)^+, \tag{7}$$

where $\eta_z(\cdot)$, $z = \{1, 2\}$ are the differentiated learning rates in node v for Case-1 and Case-2, respectively, as defined in §3.1.

5.2 Regret results for DRate-LU

DRate-LU redefines the feedback for each node v except the root node, whose range is $[0, \hat{U}_v]$. The regret of DRate-LU can be analyzed by similar techniques as those for DRate-U. In order to analyze the regret of DRate-LU, we first give the following propositions demonstrating the node regret for DRate-LU, whose definition is the same as that in (5). Without loss of generality, we assume $\hat{U}_{v_l} \geq \hat{U}_{v_r}$ holds for each non-leaf node v.

Proposition 3. *(Node Regret of* DRate-LU*) Let v be a non-leaf node of the tree constructed by* DRate-LU*. With learning rates $\eta_1(v)$ and $\eta_2(v)$,* DRate-LU *guarantees the following per-node regret for node v.*

$$R_T(v) \leq \frac{1}{\eta_1(v)} + T\hat{U}_{v_r}^2\eta_1(v) + \frac{1}{\eta_2(v)} + 2T\hat{U}_{v_l}\hat{U}_{v_r}\eta_2(v) + \hat{U}_{v_l} + \hat{U}_{v_r}.$$

Proposition 4. *(Node Regret of Leaf Node) Let v be the leaf node of the tree constructed by* DRate-LU *and $|v| > 1$. With learning rate $\eta'(v) > 0$,* DRate-U *guarantees the following node regret for leaf node v.*

$$R_T(v) \leq \eta'(v)\hat{U}_v^2 T + \frac{\log|v|}{\eta'(v)}.$$

The above propositions can be proved by slightly modifying the proofs of Proposition 1 and 2, respectively, and thus we skip the proofs. The result in Lemma 1 is applicable to DRate-LU. Hence, combining the results in Lemma 1 and the above propositions yields the following theorem.

Theorem 4. *(Regret of* DRate-LU*) The regret of* DRate-LU *satisfies the following upper bound.*

$$R_T \leq \sum_{v \in \mathsf{path}(v^*) \setminus \{v^*\}} \left(\frac{1}{\eta_1(v)} + T\hat{U}_{v_r}^2\eta_1(v) + \frac{1}{\eta_2(v)} + 2T\hat{U}_{v_l}\hat{U}_{v_r}\eta_2(v) + \hat{U}_{v_l} + \hat{U}_{v_r}\right) + Q_2,$$

where $Q_2 = \eta'(v^*)\hat{U}_{v^*}^2 T + \frac{\log|v^*|}{\eta'(v^*)}$, if $|v^*| > 1$; 0, otherwise.

The following corollary demonstrates the regret with the optimal learning rates.

Corollary 3. *By setting learning rates $\eta_1(v) = 1/(\hat{U}_{v_r}\sqrt{T})$ and $\eta_2(v) = 1/\sqrt{2\hat{U}_{v_l}\hat{U}_{v_r}T}$ for non-leaf nodes, and $\eta'(v') = \sqrt{\log|v'|/T}/\hat{U}_{v'}$ for leaf nodes with more than one experts,* DRate-LU *achieves the following regret upper bound.*

$$R_T \leq \sum_{v \in \mathsf{path}(v^*) \setminus \{v^*\}} \left(\sqrt{2\hat{U}_{v_l}\hat{U}_{v_r}T} + \hat{U}_{v_r}\sqrt{T}\right) + \hat{U}_{v^*}\sqrt{T\log|v^*|}.$$

We skip the proofs for DRate-LU, since once can prove the above results by slightly modifying the proofs for DRate-U. With similar arguments as in Remark 1, we can show that with non-uniform reward ranges, the regret of DRate-LU is much smaller than that of Hedge. The regret of DRate-LU also depends how the tree is constructed. It is much more complicated to construct a tree for DRate-LU than for DRate-U, since DRate-LU contains two sources of heterogeneity in the reward ranges. Intuitively, placing the experts with similar reward ranges into the same subset can reduce the values of $\hat{U}_v$, as well as the entire regret of DRate-LU. In order to show the efficiency of DRate-LU, we construct a simple example as follows.

Example 3. *Consider a special case of* MSHedge-LU *with two sets of experts: one expert with reward range in* $[L_1, U_1]$*; and the second group of the remaining* $K-1$ *experts, whose reward range is* $[L_2, U_2]$*. We assume* $[L_2, U_2] \subset [L_1, U_1]$.

To construct a tree of the problem instance in Example 3 for DRate-LU, we place the experts with the same reward range in the same set and construct a two-layer tree as follows.

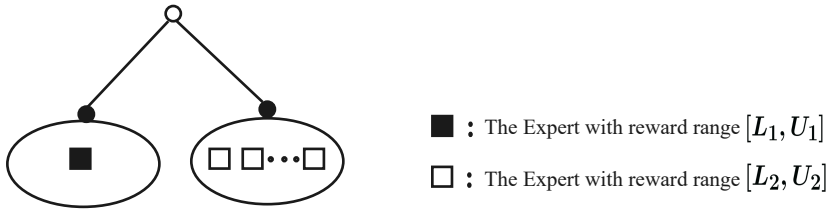


Fig. 4. A two-layer tree structure for DRate-LU.

For the root note, DRate-LU executes the node algorithm with differentiated learning rate; for the right child of the root node, the Hedge algorithm is used to select among the contained $K-1$ experts. With the above construction, the node regret of the root node is $O(\sqrt{(U_1 - L_1)(U_2 - L_1)T})$, and the node regret of the right child of the root node is $O((U_2 - L_2)\sqrt{T \log K})$. By Lemma 1, the total regret of DRate-LU for the above problem is $O(\sqrt{(U_1 - L_1)(U_2 - L_1)T} + (U_2 - L_2)\sqrt{T})$.

5.3 MSHedge and the Lipschitz expert problem

In this section, we discuss the relationship between MSHedge-LU and another structured settings of PEA, known as the Lipschitz expert problem [22, 27]. We call this problem LipHedge and define it formally in the following.

Definition 2. *(The* LipHedge *Problem) In* LipHedge*, non-uniform reward ranges of each expert depends on neighbouring experts. Assumes there are K experts. Specifically, the time-varying reward range for the i-th expert in* LipHedge *can be denoted as* $[L_{i,t}, L_{i,t} + 1]$, $i \in [K]$*. In addition, the lower bounds of any two adjacent experts satisfy the "Lipschitz Condition", that is*

$$|L_{i,t} - L_{i-1,t}| \leq c, c > 0, \text{ for all } i = 2, 3, \ldots, K, \text{ and } t = 1, 2, \ldots, T. \tag{8}$$

Compared to MSHedge, LipHedge involves an additional condition on the rewards of adjacent experts, that is similar to the Lipschitz continuous condition in the Lipschitz Expert problem [22, 27]. We can extend DRate-LU for the LipHedge problem. Toward this, we first partition the expert set recursively and construct a tree with subsets being nodes and construct the same balanced tree as Figure 2.

We can use the continuous condition for adjacent experts to bound the rewards of the experts in a node. Consider a node v, which lies in the l-th layer (take the layer containing the root

node as the first layer). Then, there are $K/2^{l-1}$ adjacent experts associated with node v, and the reward gap between any two experts will be at most $(K/2^{l-1}+1)c$. Thus, $\hat{U}_v$, satisfies $\hat{U}_v \leq (K/2^{l-2}+1)c$. Substituting it to Theorem 4 and Corollary 3 yields the regret result of DRate-LU for LipHedge. By a balanced tree with leaf nodes containing only one expert, DRate-LU attains the following regret. With setting the learning rates as Corollary 3, DRate-LU guarantees the regret of $R_T = O\left(\sum_{l=0}^{\log K-1} c\frac{K}{2^l}\sqrt{T}\right) = O\left(cK\sqrt{T}\right)$ for LipHedge. Note that the gap between the maximum and minimum rewards of experts can be as large as Kc. Thus, Hedge can attain a regret of $O(cK\sqrt{T\log K})$. The above theorem shows that DRate-LU is much more efficient to address LipHedge than Hedge. Also, compared to standard techniques used in Lipschitz Expert problems [24, 27, 37], our method attains the same or better performance, but our model can be applied to more general models of structured PEA.

6 EXPERIMENTAL RESULTS

In this section, we evaluate the performance of our proposed algorithms for MSHedge through numerical experiments and compare them to the basic and extended versions the Hedge algorithm and prior algorithms for multi-scale experts [11].

6.1 Overview of setup and baseline algorithms

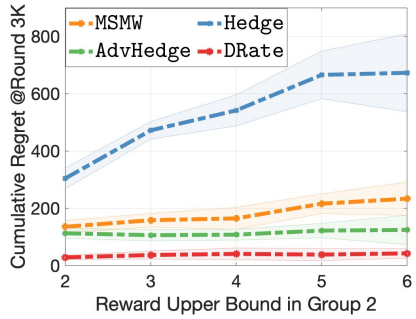
We consider a scenario with $K = 8$ experts with Bernoulli rewards. We scale the Bernoulli processes by different factors, and take them as rewards for the experts. The lower bound of the reward range is zero, and the upper bound is different based on the scaling factor. This captures the first model of MSHedge. In all experiments, we report the cumulative regret after 3,000 rounds. All reported values are averaged over 10 independent trials and standard deviations are plotted as shaded areas.

Comparison algorithms. We compare DRate with three prior algorithms: (1) the Hedge algorithm with learning rate $\eta = \sqrt{\log(K)/T}/(\max_{i\in[K]} 2U_i)$; (2) AdvHedge, which is an extended variant of Hedge [14] that adjusts the learning rate based on empirical effective reward ranges of experts, which can be non-uniform over time. We set the learning rate of AdvHedge to $\eta = \sqrt{\log(K)/T}/(2 * \texttt{EmpRR})$, where EmpRR stands for the empirical effective reward ranges of experts. The third baseline is the Multi-Scale Multiplicative-Weight (MSMW) algorithm in [11]. MSMW first normalizes the reward of each expert for updating weights and then projects the weights to a simplex by a smooth multi-scale projection for making the sampling probability distribution over the experts. In our experiment, we set the learning rate of MSMW to $\eta = \sqrt{\log(KT)/T}/3$. We note that all learning rates for the baseline algorithms are chosen according to the suggested values in the original work to optimize the regret. Last, since the reward lower bound is set to zero in our experiments, in our experiments we only implement DRate-U with $\eta_1(v) = 1/(U_{v_l}\sqrt{T})$ and $\eta_2(v) = 1/(\sqrt{2U_{v_l}U_{v_r}T})$, where $U_{v_l} \geq U_{v_r}$. We expect similar results for the case of multi-scale lower bounds.

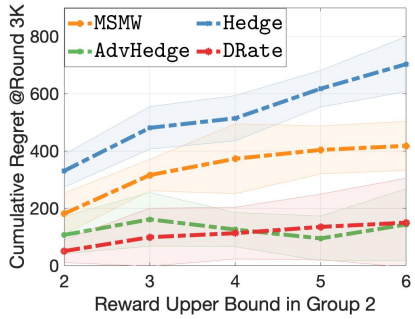
6.2 Experimental results

In the following, we evaluate the performance of DRate in two different scenarios.

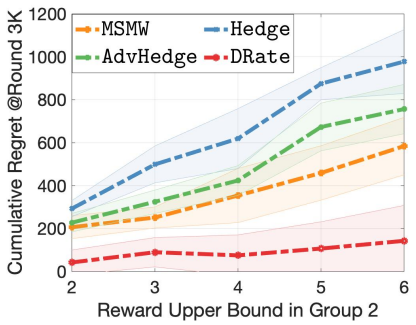
Non-uniform reward ranges with uniformly selected reward means. In the first experiment, eight experts are categorized into two groups of four experts each. The experts in the same group have a common reward range of $[0, 1]$, while that for the second group is $[0, U]$, $U > 1$. Mean rewards of all experts are randomly selected from $[0, 1]$. The results in Figures 5(a) and 5(b) demonstrate the regrets of the investigated algorithms with different reward ranges for the experts in the second group, i.e., $[0, 2]$, $[0, 3]$, $[0, 4]$, $[0, 5]$, $[0, 6]$. Specifically, Figure 5(a) corresponds to the case where the optimal expert lies in the first group, and Figure 5(b) corresponds to the case where the optimal



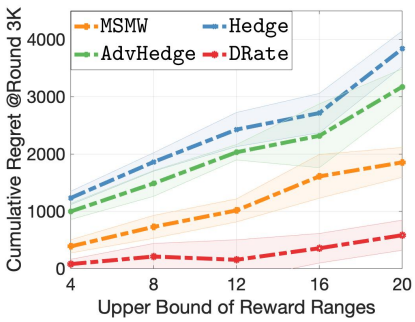
(a) Best expert lies in the first group



(b) Best expert lies in the second group



(c) Fixed non-uniform reward ranges and means



(d) Random reward ranges and means

Fig. 5. Performance comparison with non-uniform reward ranges and means.

expert lies in the second group. Figure 5(a) shows that DRate, MSMW and AdvHedge significantly outperform the Hedge algorithm. This observation matches our expectation that the performance of Hedge degrades quickly with large reward upper bounds. In contrast to Hedge, DRate efficiently deals with non-uniform reward ranges, with regret slightly increasing as heterogeneity in reward ranges increases. In Figure 5(b), DRate and AdvHedge outperform others, and the performance of MSMW is largely degraded. The performance degradation of MSMW matches its theoretical results that the performance of MSMW linearly degrades as a function of the upper bound of the best expert. Compared to the first set of results in Figure 5(a), one observes that the performance of AdvHedge remains unchanged and works as well as DRate. That is because, AdvHedge tunes its learning rate based on empirical reward gaps, and thus mitigates the impact of large reward upper bounds when empirical rewards of all experts are much smaller than the upper bound. However, as we show in the next experiment, AdvHedge fails to be competitive in some other experimental scenarios, where mean rewards of experts also becomes non-uniform.

Non-uniform reward ranges with non-uniform mean rewards. In the second experiment, we assign non-uniform reward ranges to experts, and then randomly select mean rewards from the corresponding reward ranges. Results are shown in Figures 5(c) and 5(d). In the first experiment, experts are evenly divided into two groups. The reward ranges of experts in the first group are all set to $[0, 1]$, and those for the second group are set to $[0, U]$, $U > 1$. Figure 5(c) demonstrates the regrets of different algorithms for $U = 2, 3, 4, 5, 6$. In the second experiment in Figure 5(d), no

fixed upper bounds are assigned to experts. Instead, the upper bound of every expert is randomly selected from a given range. The larger the range, the greater the heterogeneity in reward ranges. Figure 5(d) gives the regret results when reward upper bounds of experts are randomly selected from the ranges $[0, 4]$, $[0, 8]$, $[0, 12]$, $[0, 16]$, $[0, 20]$.

Both Figures 5(c) and 5(d) verify the advantages of `DRate` over the three baseline algorithms. Notable observations are summarized as follows. (1) When the heterogeneity of reward ranges increases, the regrets of all algorithms increase, yet, the regret of `DRate` is much smaller and increases at a slower rate than the others. (2) `AdvHedge` and `MSMW` perform slightly better than Hedge, but their regrets increase quickly with non-uniform reward ranges when mean rewards are randomly selected from the corresponding reward ranges. Specifically, for the `AdvHedge` algorithm, when mean rewards are randomly selected, it cannot benefit from mitigating the influence of small empirical rewards as in the first set of experiments where mean rewards for all experts lie in a very small range. The performance of `MSMW` algorithm also degrades quickly, since the best expert is likely to have a larger upper bound, resulting in a worse regret.

7 RELATED WORK

7.1 PEA with generalized reward ranges

Non-uniform reward ranges. In §2.4, we have reviewed this kind of work and compared the results in those works with ours. Thus, we skip the details here.

Structured rewards. In some works, the reward ranges are characterized by defining dependence or correlation among the rewards of the experts, which introduces heterogeneity to reward ranges implicitly. For example, the works in [22, 23] assume a structured class of payoff functions over experts (or decision points called in their work), whose structure is induced by a metric on the decision space, where correlation or dependence is defined among decision points. That is, there are no fixed reward ranges for decision points, but the reward of each decision point has to satisfy some dependence or correlation conditions among the decision points. By leveraging the structure as prior knowledge of algorithms, one is able to design efficient learning algorithms [31, 33]. For example, the authors in [24, 27, 37] consider Lipschitz experts in a Euclidean space of constant dimension, with proven regret upper bound. The above model can be generalized by considering discontinuity points in the continuous condition, and one can refer to [8, 32]. For the structured settings of PEA, we have shown the efficiency of our frameworks by theoretical results in §5.3. Last, for bandit version of this kind of work, where only partial information on the decision points can be acquired at each time slot, one can refer to [1, 2, 7, 20, 21] etc.

Time-varying reward ranges. Some other work considers time-varying reward upper bounds. For example, the works in [3, 19, 36] considered the model where the reward upper bounds can be time-varying and even unbounded. For example, the work in [36] considered rewards to satisfy time-varying upper bounds over time, and proposed an algorithm which is optimal as the scaled fluctuations of one-step losses of experts of the pool tend to zero. In addition to [36], Poland and Hutter in [19] have studied the games where one-step losses of all experts at each step t are bounded from above, and the upper bounds consist of an increasing sequence U_t, $t = 1, 2, \ldots, T$, which are given in advance. They presented a learning algorithm that is asymptotically consistent for $U_t = t^{1/16}$. However, all of the above works did not consider internal heterogeneity among experts in a single time slot.

7.2 PEA with specific reward realizations

Another direction of work focuses on devising learning algorithms whose performance depends on the realization of rewards. Those works assume uniform reward ranges, but the resulting regret upper bounds can vary with different realizations. Even though the setting of their model differs from ours, we are still interested in comparing their works with ours from the angle of algorithm design and theoretical results.

In [28], the authors investigated learning algorithms, that can achieve the best performance of both worlds, that is, attaining optimal regret both for adversarial rewards and for stochastic rewards. Similarly, the authors in [5] considered a setting where the environment is benign and generates losses stochastically, but the feedback observed by the learner is subject to moderate adversarial corruption. They proved that a variant of the classical Multiplicative Weights algorithm with decreasing step sizes achieves constant regret in this setting and performs optimally in a wide range of environments, regardless of the magnitude of the injected corruption. Specifically, regret can be expressed as a function of the amount of corruption. Also, one finds similar results in the bandit setting [12, 29, 30].

In addition, there are many works on designing parameter-free algorithms, whose regret is independent of the number of experts to some degree. To do this, we can order the cumulative payoffs of all actions from highest to lowest and define the *regret of the learner to the top ϵ-quantile,* $\epsilon \in [0, 1]$ to be the difference between the cumulative reward of the $\lceil N\epsilon \rceil$-th element in the sorted list and that of the learner. In [15], the authors proposed a novel algorithm, NormalHedge, that achieves a regret of $O(\sqrt{T \ln \frac{1}{\epsilon}} + \ln^2 K)$ with respect to the above new definition, which means the algorithm suffers at most this amount of regret for all but an ϵ fraction of the experts. Note that this bound does not depend on K at all and is at most $O(\sqrt{T \ln K} + \ln^2 K)$, since $\epsilon \geq 1/K$. This is close to the result for the Hedge algorithm differing by a small additive term $\ln^2 K$. Then, [26] obtains a regret of $O(\sqrt{T \ln(\ln T)/\epsilon}))$ for a new algorithm, NormalHedge.DT, which drops the dependence on K. Note that, in the cases where there are many "good" experts whose rewards are close to each other, the above result is substantially better than that by the Hedge algorithm.

Last, there is another kind of work which attains better regret performance with small expectation and variance of realized rewards. For example, the authors in [14] reported a regret of $O(\sqrt{\sum_{t=1}^{T} G_t^2})$, where G_t is the effective range of the rewards at round t, i.e., $G_t = \max_{i \in [K]} x_t(i) - \min_{i' \in [K]} x_t(i')$. Apparently, when the realization of payoffs is much smaller than the upper bound, the algorithm significantly outperforms the Hedge algorithm.

Generally, all the above works focus on the performance of the algorithm with specific realizations of inputs, and the resulting regrets can be based on the heterogeneous payoff realizations over different experts. However, those work can not improve on the performance of the algorithm under the worst case. Moreover, those works only leverage the effective reward range of all experts, but none of them take into account the non-uniform reward ranges across experts in a single time slot.

8 CONCLUDING REMARKS

Motivated by the problems of dynamic pricing and portfolio selection, in this paper, we studied an extended version of the classic Hedge problem where there are multiple experts with non-uniform reward ranges, and the goal of an online learner is to find the best expert. We developed hierarchical learning algorithms that explicitly consider the heterogeneity of the reward range of experts into their sequential decision-making. Our theoretical regret analysis shows that the proposed algorithms outperform the existing algorithms when the heterogeneity of the expert rewards is

high. We also verified our theoretical observations by numerical experiments and showed the our algorithms outperforms multiple existing algorithms in the literature.

A limitation of this work is that in the current result, the tight lower bound and optimal result for our setting in the general case is still missing. We leave developing an algorithm with the optimal regret for MSHedge as an open problem. Developing a unified algorithm that performs well under both uniform and non-uniform reward structures is another promising future direction. Another interesting future work is to extend MSHedge to the bandit setting where the algorithm can only observe the reward of the selected expert (or arm in the bandit context). Specifically, an interesting problem for the bandit setting with multi-scaled reward ranges is whether we can achieve a better regret by using the hierarchical learning policy.

ACKNOWLEDGMENTS

This research is supported by NSF CPS 2136199, CAREER 2045641, CNS 2102963, CNS 1908298, and CNS 2106299, U.S. Army Research Laboratory Army Research Laboratory under Cooperative Agreement W911NF-17-2-0196 (IoBT CRA), and U.S. Army Research Laboratory and the U.K. Ministry of Defence under Agreement W911NF-16-3-0001.

REFERENCES

[1] Yasin Abbasi-Yadkori, Dávid Pál, and Csaba Szepesvári. 2011. Improved algorithms for linear stochastic bandits. *Advances in neural information processing systems* 24 (2011), 2312–2320.

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

[3] Chamy Allenberg, Peter Auer, László Györfi, and György Ottucsák. 2006. Hannan consistency in on-line learning in case of unbounded losses under partial monitoring. In *International Conference on Algorithmic Learning Theory*. Springer, 229–243.

[4] Noga Alon and Joel H Spencer. 2004. *The probabilistic method*. John Wiley & Sons.

[5] Idan Amir, Idan Attias, Tomer Koren, Roi Livni, and Yishay Mansour. 2020. Prediction with corrupted expert advice. *arXiv preprint arXiv:2002.10286* (2020).

[6] Sanjeev Arora, Elad Hazan, and Satyen Kale. 2012. The multiplicative weights update method: a meta-algorithm and applications. *Theory of Computing* 8, 1 (2012), 121–164.

[7] Peter Auer, Ronald Ortner, and Csaba Szepesvári. 2007. Improved rates for the stochastic continuum-armed bandit problem. In *International Conference on Computational Learning Theory*. Springer, 454–468.

[8] Maria-Florina Balcan, Travis Dick, and Ellen Vitercik. 2018. Dispersion for data-driven algorithm design, online learning, and private optimization. In *2018 IEEE 59th Annual Symposium on Foundations of Computer Science (FOCS)*. IEEE, 603–614.

[9] Avrim Blum, Vijay Kumar, Atri Rudra, and Felix Wu. 2004. Online learning in online auctions. *Theoretical Computer Science* 324, 2-3 (2004), 137–146.

[10] Sébastien Bubeck, Nikhil R Devanur, Zhiyi Huang, and Rad Niazadeh. 2017. Multi-scale online learning and its applications to online auctions. *arXiv preprint arXiv:1705.09700* (2017).

[11] Sebastien Bubeck, Nikhil R Devanur, Zhiyi Huang, and Rad Niazadeh. 2017. Online auctions and multi-scale online learning. In *Proceedings of the 2017 ACM Conference on Economics and Computation*. 497–514.

[12] Sébastien Bubeck and Aleksandrs Slivkins. 2012. The best of both worlds: Stochastic and adversarial bandits. In *Conference on Learning Theory*. JMLR Workshop and Conference Proceedings, 42–1.

[13] Nicolo Cesa-Bianchi, Yoav Freund, David Haussler, David P Helmbold, Robert E Schapire, and Manfred K Warmuth. 1997. How to use expert advice. *Journal of the ACM (JACM)* 44, 3 (1997), 427–485.

[14] Nicolo Cesa-Bianchi, Yishay Mansour, and Gilles Stoltz. 2007. Improved second-order bounds for prediction with expert advice. *Machine Learning* 66, 2 (2007), 321–352.

[15] Kamalika Chaudhuri, Yoav Freund, and Daniel J Hsu. 2009. A parameter-free hedging algorithm. In *Advances in neural information processing systems*. 297–305.

[16] Yoav Freund and Robert E Schapire. 1997. A decision-theoretic generalization of on-line learning and an application to boosting. *Journal of computer and system sciences* 55, 1 (1997), 119–139.

[17] Yoav Freund and Robert E Schapire. 1999. Adaptive game playing using multiplicative weights. *Games and Economic Behavior* 29, 1-2 (1999), 79–103.

[18] Elad Hazan. 2019. Introduction to online convex optimization. *arXiv preprint arXiv:1909.05207* (2019).

[19] Marcus Hutter and Jan Poland. 2004. Prediction with expert advice by following the perturbed leader for general weights. In *International Conference on Algorithmic Learning Theory*. Springer, 279–293.
[20] Robert Kleinberg. 2004. Nearly tight bounds for the continuum-armed bandit problem. *Advances in Neural Information Processing Systems* 17 (2004), 697–704.
[21] Robert Kleinberg, Aleksandrs Slivkins, and Eli Upfal. 2008. Multi-armed bandits in metric spaces. In *Proceedings of the fortieth annual ACM symposium on Theory of computing*. 681–690.
[22] Robert Kleinberg, Aleksandrs Slivkins, and Eli Upfal. 2019. Bandits and experts in metric spaces. *Journal of the ACM (JACM)* 66, 4 (2019), 1–77.
[23] Robert David Kleinberg. 2005. *Online decision problems with large strategy sets*. Ph.D. Dissertation. Massachusetts Institute of Technology.
[24] Walid Krichene, Maximilian Balandat, Claire Tomlin, and Alexandre Bayen. 2015. The hedge algorithm on a continuum. In *International Conference on Machine Learning*. PMLR, 824–832.
[25] Nick Littlestone and Manfred K Warmuth. 1994. The weighted majority algorithm. *Information and computation* 108, 2 (1994), 212–261.
[26] Haipeng Luo and Robert E Schapire. 2014. A drifting-games analysis for online learning and applications to boosting. *Advances in Neural Information Processing Systems* 27 (2014), 1368–1376.
[27] Odalric-Ambrym Maillard and Rémi Munos. 2010. Online learning in adversarial lipschitz environments. In *Joint european conference on machine learning and knowledge discovery in databases*. Springer, 305–320.
[28] Jaouad Mourtada and Stéphane Gaïffas. 2019. On the optimality of the Hedge algorithm in the stochastic regime. *Journal of Machine Learning Research* 20 (2019), 1–28.
[29] Chloé Rouyer and Yevgeny Seldin. 2020. Tsallis-INF for Decoupled Exploration and Exploitation in Multi-armed Bandits. In *Conference on Learning Theory*. PMLR, 3227–3249.
[30] Yevgeny Seldin and Aleksandrs Slivkins. 2014. One practical algorithm for both stochastic and adversarial bandits. In *International Conference on Machine Learning*. PMLR, 1287–1295.
[31] Pier Giuseppe Sessa, Ilija Bogunovic, Maryam Kamgarpour, and Andreas Krause. 2019. No-regret learning in unknown games with correlated payoffs. *Advances in Neural Information Processing Systems* 32 (2019), 13624–13633.
[32] Dravyansh Sharma, Maria-Florina Balcan, and Travis Dick. 2020. Learning piecewise Lipschitz functions in changing environments. In *International Conference on Artificial Intelligence and Statistics*. PMLR, 3567–3577.
[33] Aleksandrs Slivkins. 2011. Contextual bandits with similarity information. In *Proceedings of the 24th annual Conference On Learning Theory*. JMLR Workshop and Conference Proceedings, 679–702.
[34] Vladimir Vovk. 1998. A game of prediction with expert advice. *J. Comput. System Sci.* 56, 2 (1998), 153–173.
[35] Volodimir G Vovk. 1990. Aggregating strategies. *Proc. of Computational Learning Theory, 1990* (1990).
[36] Vladimir V V'yugin. 2011. Online Learning in Case of Unbounded Losses Using Follow the Perturbed Leader Algorithm. *Journal of Machine Learning Research* 12, 1 (2011).
[37] Lin Yang, Lei Deng, Mohammad H Hajiesmaili, Cheng Tan, and Wing Shing Wong. 2018. An optimal algorithm for online non-convex learning. *Proceedings of the ACM on Measurement and Analysis of Computing Systems* 2, 2 (2018), 1–25.

A SUPPLEMENTARY PROOFS

In this section, we give the proofs of the upper and lower bounds. Note that, given the upper bounds, the learning rate selected by node v is actually based on the index of the leading child, which is $\alpha_t(v)$. Hence, we can use $\eta(\alpha_t(v))$ to denote the learning rate adopted by node v at time slot t. For example, if $U_{v_l} \geq U_{v_r}$, we have

$$\eta(\alpha_t(v)) = \begin{cases} \eta_1(v), & \alpha_t(v) = v_l; \\ \eta_2(v), & \alpha_t(v) = v_r. \end{cases}$$

A.1 A proof of Lemma 1

Lemma 1 can be proved by recursively applying the definition of feedback of nodes. Specifically, we have

$$\begin{aligned}
R_T =& \max_{i\in\mathcal{K}} \sum_{t=1}^{T} x_t(i) - \mathbb{E}\left[\sum_{t=1}^{T} x_t(I_t)\right] = \max_{i\in\mathcal{K}} \sum_{t=1}^{T} x_t(i) - \sum_{t=1}^{T}\sum_{i\in\mathcal{K}} p_t(i)x_t(i) \\
=& \sum_{t=1}^{T} \hat{x}_t(v^*) - \sum_{t=1}^{T} \hat{x}_t(o) = \sum_{t=1}^{T} \hat{x}_t(v^*) - \sum_{t=1}^{T} \hat{x}_t((v^*)_p) + \sum_{t=1}^{T} \hat{x}_t((v^*)_p) - \sum_{t=1}^{T} \hat{x}_t(o) \\
\leq& R_T((v^*)_p) + \sum_{t=1}^{T} \hat{x}_t((v^*)_p) - \sum_{t=1}^{T} \hat{x}_t(o) \\
\leq& R_T(v^*)_p) + \sum_{t=1}^{T} \hat{x}_t((v^*)_p) - \sum_{t=1}^{T} \hat{x}_t((v^*)_p^2) + \sum_{t=1}^{T} \hat{x}_t((v^*)_p^2) - \sum_{t=1}^{T} \hat{x}_t(o) \\
=& R_T((v^*)_p) + R_T((v^*)_p^2) + \sum_{t=1}^{T} \hat{x}_t((v^*)_p^2) - \sum_{t=1}^{T} \hat{x}_t(o).
\end{aligned}$$

By recursively applying the definition of $\hat{x}_t(\cdot)$, the above equation can be rewritten as

$$R_T \leq \sum_{v\in\mathsf{path}(v^*)} R_T(v).$$

This completes the proof.

A.2 Analysis of the Node Regret of DRate-U (Proof of Proposition 1)

In this subsection, we provide a detailed proof of Proposition 1, which characterizes the node regret of any non-leaf node v when executing DRate-U. Generally, the proof consists of the following three parts.

Part I: Define Per-step Regret.
To ease our analysis, we first define the "per-step regret" for node v, which is

$$r_t(v) := \hat{X}_t(\alpha_{v,t}) - \hat{X}_{t-1}(\alpha_{t-1}(v)) - (\hat{p}_t(\alpha_t(v))\hat{x}_t(\alpha_t(v)) + \hat{p}_t(\beta_t(v))\hat{x}_t(\beta_t(v))).$$

Intuitively, the per-step regret, $r_t(v)$, which can be negative, characterizes the changing amount of the node regret at any time slot t. It follows from the definition of $r_t(v)$ that

$$\begin{aligned}
R_T(v) &= \max\left\{\sum_{t=1}^{T} \hat{x}_t(v_l), \sum_{t=1}^{T} \hat{x}_t(v_r)\right\} - \sum_{t=1}^{T} (\hat{p}_t(v_l)\hat{x}_t(v_l) + \hat{p}_t(v_l)\hat{x}_t(v_r)) \\
&= \hat{X}_T(\alpha_T(v)) - \sum_{t=1}^{T} (\hat{p}_t(v_l)\hat{x}_t(v_l) + \hat{p}_t(v_l)\hat{x}_t(v_r)) \\
&= \sum_{t=1}^{T} \left(\hat{X}_t(\alpha_t(v)) - \hat{X}_{t-1}(\alpha_{t-1}(v))\right) + \hat{X}_0(\alpha_0(v)) - \sum_{t=1}^{T} (\hat{p}_t(\alpha_t(v))\hat{x}_t(\alpha_t(v)) + \hat{p}_t(\beta_t(v))\hat{x}_t(\beta_t(v))) = \sum_{t=1}^{T} r_t(v).
\end{aligned}$$

The above equation implies that the node regret of v can be rewritten as the summation of its per-step regrets over all time slots.

Part II: Define stack S.

To analyze the per-step regret, we define stack S which only exists in our analysis. Our later analysis will show that the above defined per-step regret can be characterized through the elements in S. Specifically, S is empty at the beginning, and its elements are updated at each time slot based on the feedback gap between the two children of node v. Specifically, we denote the feedback gap at time slot t as g_t, i.e., $g_t := \hat{x}_t(\alpha_t(v)) - \hat{x}_t(\beta_t(v))$. With g_t, the elements in S are updated according to the following three rules.

(1) When $g_t = 0$: do nothing to the stack.

(2) When $g_t > 0$: push g_t to the top of the stack.

(3) When $g_t < 0$: if $|g_t|$ is larger than the sum of all the elements in S, denoted as $\texttt{sum}(S)$, empty the stack and add a new element with value being the $|g_t| - \texttt{sum}(S)$ as a new top; otherwise, pop items until the sum of the values of popped items is larger than or equal to $|g_t|$. If their sum is larger than $|g_t|$, a new item with value being the sum minus $|g_t|$ will be pushed into the stack as a new top. We use the following figure to demonstrate the updates of the stack with different values of g_t.

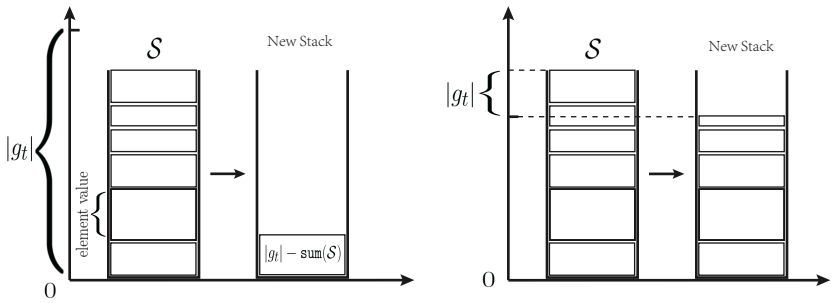


Fig. 6. Stack evolution according to different values of $|g_t|$.

Considering that the elements of S can be time-varying, we use S_t to denote the state of the stack at the end of time slot t and $s_{t,k}$ to denote the value of the k-th element from the bottom at time slot t.

Part III: Analyze Per-step Regret.

In this part, we proceed to analyze the per-step regret case by case.

Case 1: $\alpha_t(v) = \alpha_{t-1}(v)$. In this case, the index of the leading child is the same as the previous time slot. By the definition of the per-step regret, there is

$$\begin{aligned}
r_t(v) &= \hat{X}_t(\alpha_t(v)) - \hat{X}_{t-1}(\alpha_{t-1}(v)) - (\hat{p}_t(\alpha_t(v))\hat{x}_t(\alpha_t(v)) + \hat{p}_t(\beta_t(v))\hat{x}_t(\beta_t(v))) \\
&= \hat{x}_t(\alpha_t(v)) - (\hat{p}_t(\alpha_t(v))\hat{x}_t(\alpha_t(v)) + \hat{p}_t(\beta_t(v))\hat{x}_t(\beta_t(v))) \\
&= (1 - \hat{p}_t(\alpha_t(v)))\,\hat{x}_t(\alpha_t(v)) - \hat{p}_t(\beta_t(v))\hat{x}_t(\beta_t(v)) \\
&= \hat{p}_t(\beta_t(v))\,(\hat{x}_t(\alpha_t(v)) - \hat{x}_t(\beta_t(v))) \\
&= \left(\frac{1}{2} - D_{t-1}(v)\eta(\alpha_{t-1}(v))\right)^+ (\hat{x}_t(\alpha_t(v)) - \hat{x}_t(\beta_t(v))) \\
&= \left(\frac{1}{2} - D_{t-1}(v)\eta(\alpha_t(v))\right)^+ (\hat{x}_t(\alpha_t(v)) - \hat{x}_t(\beta_t(v)))\,,
\end{aligned} \tag{9}$$

Note that, the sum of the values of all elements in the stack is always equal to the gap of cumulative feedback between the leading child and the other one, i.e., $\texttt{sum}(S_t) = D_t(v)$, for any t.

Hence,

$$\begin{aligned} r_t(v) &= \left(\frac{1}{2} - D_{t-1}(v)\eta(\alpha_t(v))\right)^+ (\hat{x}_t(\alpha_t(v)) - \hat{x}_t(\beta_t(v))) \\ &= \left(\frac{1}{2} - \mathsf{sum}(S_{t-1})\eta(\alpha_t(v))\right)^+ (\hat{x}_t(\alpha_t(v)) - \hat{x}_t(\beta_t(v))) \\ &= \left(\frac{1}{2} - \sum_{k=1}^{\mathsf{len}(S_{t-1})} s_{t,k}\eta(\alpha_t(v))\right)^+ g_t, \end{aligned} \tag{10}$$

where $\mathsf{len}(\cdot)$ returns the length of the stack.

We continue to consider the following two cases.

Case 1(a): When $g_t \geq 0$, we have

$$r_t(v) = \left(\frac{1}{2} - \sum_{k=1}^{\mathsf{len}(S_{t-1})} s_{t,k}\eta(\alpha_t(v))\right)^+ g_t = \Phi(S_t) - \Phi(S_{t-1}),$$

where $\Phi(S_t)$ is a potential function with respect to the elements in stack S at time slot t, and defined as

$$\Phi(S_t) := \sum_{k=1}^{\mathsf{len}(S_t)} \left(\frac{1}{2} - \sum_{k'=1}^{k} s_{t,k'}\eta(\alpha_t(v))\right)^+ s_{t,k}.$$

Case 1(b): When $g_t < 0$, we have

$$\begin{aligned} r_t(v) &= \left(\frac{1}{2} - \sum_{k=1}^{\mathsf{len}(S_{t-1})} s_{t-1,k}\eta(\alpha_t(v))\right)^+ g_t \\ &= \left(\frac{1}{2} - \sum_{k=1}^{\mathsf{len}(S_t)} s_{t,k}\eta(\alpha_t(v))\right)^+ g_t + \left(\left(\frac{1}{2} - \sum_{k=1}^{\mathsf{len}(S_{t-1})} s_{t-1,k}\eta(\alpha_t(v))\right)^+ - \left(\frac{1}{2} - \sum_{k=1}^{\mathsf{len}(S_t)} s_{t,k}\eta(\alpha_t(v))\right)^+\right) g_t. \end{aligned} \tag{11}$$

By applying the analysis in A.3, we have

$$r_t(v) \leq \left(\frac{1}{2} - \sum_{k=1}^{\mathsf{len}(S_t)} s_{t,k}\eta(\alpha_t(v))\right)^+ g_t + \eta(\alpha_t(v))g_t^2. \tag{12}$$

If $\alpha_t(v) = v_l$, we have $g_t = \hat{x}_t(v_l) - \hat{x}_t(v_r) < 0$, and thus $|g_t| \leq |\hat{x}_t(v_r)| \leq U_{v_r}$. The second term in Equation (12) satisfies

$$\eta(\alpha_t(v))g_t^2 \leq \eta(v_l)U_{v_r}^2. \tag{13}$$

If $\alpha_t(v) = v_r$, we have

$$\begin{aligned} &|g_t| \leq U_{v_l}, \\ &|g_t| = \sum_{k=1}^{\mathsf{len}(S_{t-1})} s_{t-1,k} - \sum_{k=1}^{\mathsf{len}(S_t)} s_{t,k} \leq (n_t + 1)U_{v_r}. \end{aligned}$$

In the above equation, n_t is the number of elements popped from the stack at time slot t. The last inequality of the second equation uses the fact that $s_{t,k} \leq U_{\alpha_t(v)}$. Then, we have

$$\eta(\alpha_t(v))g_t^2 \leq \eta(v_r)(n_t + 1)U_{v_r}U_{v_l}. \tag{14}$$

Combining Equation (12), (13) and (14) yields

$$\begin{aligned} r_t(v) \leq & \left(\frac{1}{2} - \sum_{k=1}^{\texttt{len}(S_t)} s_{t,k}\eta(\alpha_t(v))\right)^+ g_t + (n_t+1)U_{v_l}U_{v_r}\eta(v_r) + U_{v_r}^2\eta(v_l) \\ \leq & \Phi(S_t) - \Phi(S_{t-1}) + (n_t+1)U_{v_l}U_{v_r}\eta(v_r) + U_{v_r}^2\eta(v_l), \end{aligned}$$

where the second inequality is based on the definition of the the potential function $\Phi(S_t)$.

Concluding cases (1.1) and (1.2), we have

$$r_t(v) \leq \Phi(S_t) - \Phi(S_{t-1}) + (n_t+1)U_{v_l}U_{v_r}\eta(v_r) + U_{v_r}^2\eta(v_l).$$

Case 2: $\alpha_t(v) \neq \alpha_{t-1}(v)$**.**

In this case, we also upper bound the per-step regret by using the potential function with respect to the defined stack. Based on the definitions, we rewrite the per-step regret as follows.

$$\begin{aligned} & r_t(v) = \hat{X}_t(\alpha_t(v)) - \hat{X}_{t-1}(\alpha_{t-1}(v)) - (\hat{p}_t(\alpha_t(v))\hat{x}_t(\alpha_t(v)) + \hat{p}_t(\beta_t(v))\hat{x}_t(\beta_t(v))) \\ & = \hat{X}_t(\alpha_t(v)) - \hat{X}_{t-1}(\alpha_{t-1}(v)) \\ & - \left(\hat{p}_t(\alpha_t(v))\hat{X}_t(\alpha_t(v)) - \hat{p}_t(\alpha_t(v))\hat{X}_{t-1}(\alpha_t(v)) + \hat{p}_t(\beta_t(v))\hat{X}_t(\beta_t(v)) - \hat{p}_t(\beta_t(v))\hat{X}_{t-1}(\beta_t(v))\right) \\ & = \hat{X}_t(\alpha_t(v)) - \hat{X}_{t-1}(\alpha_{t-1}(v)) \\ & - \left(\hat{p}_t(\alpha_t(v))\hat{X}_t(\alpha_t(v)) - \hat{p}_t(\alpha_t(v))\hat{X}_{t-1}(\beta_{t-1}(v)) + \hat{p}_t(\beta_t(v))\hat{X}_t(\beta_t(v)) - \hat{p}_t(\beta_t(v))\hat{X}_{t-1}(\alpha_{t-1}(v))\right) \\ & = (1-\hat{p}_t(\alpha_t(v)))\,\hat{X}_t(\alpha_t(v)) - (1-\hat{p}_t(\beta_t(v)))\hat{X}_{t-1}(\alpha_{t-1}(v)) + \hat{p}_t(\alpha_t(v))\hat{X}_{t-1}(\beta_{t-1}(v)) - \hat{p}_t(\beta_t(v))\hat{X}_t(\beta_t(v)) \\ & = \hat{p}_t(\beta_t(v))\left(\hat{X}_t(\alpha_t(v)) - \hat{X}_t(\beta_t(v))\right) - \hat{p}_t(\alpha_t(v))\left(\hat{X}_{t-1}(\alpha_{t-1}(v)) - \hat{X}_{t-1}(\beta_{t-1}(v))\right) \\ & = \hat{p}_t(\beta_t(v))D_t(v) - \hat{p}_t(\alpha_t(v))D_{t-1}(v). \end{aligned}$$

It follows from the definition of the potential function that

$$\hat{p}_t(\beta_t(v))D_t(v) \leq \Phi(S_t),$$

and

$$-\hat{p}_t(\alpha_t(v))D_{t-1}(v) \leq -\frac{1}{2}D_{t-1}(v) \leq -\Phi(S_{t-1}).$$

Combining the above three equations yields

$$r_t(v) \leq \Phi(S_t) - \Phi(S_{t-1}).$$

Concluding cases (1) and (2), we have

$$r_t(v) \leq \Phi(S_t) - \Phi(S_{t-1}) + (n_t+1)U_{v_l}U_{v_r}\eta(v_r) + U_{v_r}^2\eta(v_l), \text{ for any } t \in [T].$$

Summing $r_t(v)$ up yields the node regret of node v:

$$\begin{aligned} R_T(v) = \sum_{t=1}^{T} r_t \leq & \Phi(S_T) - \Phi(S_0) + \sum_{t=1}^{T}\left((n_t+1)U_{v_l}U_{v_r}\eta(v_r) + U_{v_r}^2\eta(v_l)\right) \\ \leq & \frac{1}{\eta(v_l)} + TU_{v_r}^2\eta(v_l) + \frac{1}{\eta(v_r)} + 2TU_{v_l}U_{v_r}\eta(v_r), \end{aligned}$$

where the inequality uses the facts that $\Phi(S_0) = 0$, $\Phi(S_T) \leq \max\{\eta^{-1}(v_l), \eta^{-1}(v_r)\}$, and $\sum_{t=1}^{T}(n_t + 1) \leq 2T$. Substituting $\eta_1(v)$ and $\eta_2(v)$ into $\eta(v_l)$ and $\eta(v_r)$ yields the final result. This completes the proof.

A.3 A proof of Equation (12)

We prove the equation case by case.

Case 1: $\frac{1}{2} - \sum_{k=1}^{\text{len}(S_{t-1})} s_{t-1,k}\eta(\alpha_t(v)) \leq 0$.

It is straightforward that

$$r_t(v) = 0 \leq \left(\frac{1}{2} - \sum_{k=1}^{\text{len}(S_t)} s_{t,k}\eta(\alpha_t(v))\right)^+ g_t. \tag{15}$$

Case 2: $\frac{1}{2} - \sum_{k=1}^{\text{len}(S_{t-1})} s_{t-1,k}\eta(\alpha_t(v)) > 0$.

In this case, we have $\frac{1}{2} - \sum_{k=1}^{\text{len}(S_t)} s_{t,k}\eta(\alpha_t(v)) > 0$, since $\sum_{k=1}^{\text{len}(S_{t-1})} s_{t-1,k} \geq \sum_{k=1}^{\text{len}(S_t)} s_{t,k}$ when $g_t < 0$. It follows that

$$\begin{aligned}
&\left(\frac{1}{2} - \sum_{k=1}^{\text{len}(S_{t-1})} s_{t-1,k}\eta(\alpha_t(v))\right)^+ - \left(\frac{1}{2} - \sum_{k=1}^{\text{len}(S_t)} s_{t,k}\eta(\alpha_t(v))\right)^+ \\
=&\frac{1}{2} - \sum_{k=1}^{\text{len}(S_{t-1})} s_{t-1,k}\eta(\alpha_t(v)) - \frac{1}{2} + \sum_{k=1}^{\text{len}(S_t)} s_{t,k}\eta(\alpha_t(v)) = g_t\eta(\alpha_t(v)).
\end{aligned}$$

Applying the above results to Equation (11), we have

$$r_t(v) \leq \left(\frac{1}{2} - \sum_{k=1}^{\text{len}(S_t)} s_{t,k}\eta(\alpha_t(v))\right)^+ g_t + \eta(\alpha_t(v))g_t^2.$$

One refers to Figure 7 for a visualized characterization of the terms in the above equation. Specifically, the absolute value of the first term on the right hand side of the above equation corresponds to the sum of the sizes of Area 1 and 2 in Figure 7, and the second term corresponds to the size of Area 1.

Concluding the above two cases, we complete the proof.

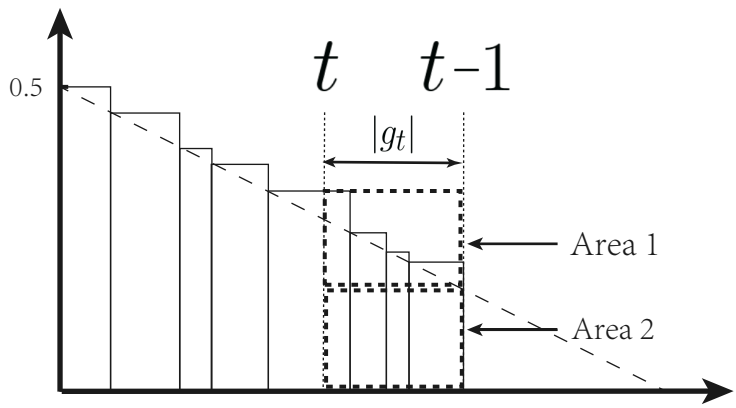


Fig. 7. Visualized proof for Case (2).

A.4 A proof of Corollary 2

The sum of the node regrets except the leaf node is

$$\sum_{v\in\mathsf{path}(v^*)\setminus\{v^*\}}\left(\frac{1}{\eta_1(v)}+TU_{v_r}^2\eta_1(v)+\frac{1}{\eta_2(v)}+2TU_{v_l}U_{v_r}\eta_2(v)+U_{v_l}+U_{v_r}\right)$$
$$=O\left(\sum_{v\in\mathsf{path}(v^*)\setminus\{v^*\}}\sqrt{2U_{v_l}U_{v_r}T}+U_{v_r}\sqrt{T}\right)=O\left(\sqrt{U_1}\sum_{l=1}^{\log K}\sqrt{U_{2^l-1}T}\right).$$

The leaf node only contains one expert, and thus its regret is 0. This completes the proof.

Adding the node regret of the leaf node v^* yields the final result.

A.5 Proof of the Lower Bounds

We first prove the lower bound for the two-expert case in Theorem 2. To prove the lower bound, we construct the rewards for the two experts as follows. For the first expert, we set rewards to be $U_1/2$ over all the T time slots. For the second expert, we first randomly select TU_1/U_2 time slots from the entire time horizon with the same probability. The set of selected time slots is denoted by $\mathcal{T}_s$. Then, we assign 0 or U_2 rewards to those time slots with equal probability. In this way, the reward means of both experts are $U_1/2$.

For $h\le T/4U_1U_2$, the probability that the cumulative empirical rewards of the second expert is larger than or equal to $TU_1/2+h$ is

$$\begin{aligned}\Pr\left[\sum_{t\in\mathcal{T}_s}x_t(2)\ge\frac{U_1T}{2}+h\right]&=\Pr\left[\sum_{t\in\mathcal{T}_s}x_t(2)\ge\frac{TU_1}{U_2}\cdot\frac{U_2}{2}+h\right]\\&=\Pr\left[\sum_{t\in\mathcal{T}_s}\frac{x_t(2)}{U_2}\ge\frac{TU_1}{U_2}\cdot\frac{1}{2}+\frac{h}{U_2}\right]\\&\ge\frac{1}{15}\exp\left(-16\frac{TU_1}{U_2}\left(\frac{h}{TU_1}\right)^2\right)=\frac{1}{15}\exp\left(-16\frac{h^2}{U_1U_2T}\right),\end{aligned}$$

The last inequality is based on the standard concentration results for binomial distribution with mean $1/2$ for TU_1/U_2 trails (see [4]). Let $h=\sqrt{U_1U_2T}/8$, we have

$$\Pr\left[\sum_{t\in\mathcal{T}_s}x_t(2)\ge\frac{TU_1}{2}+\frac{1}{8}\sqrt{U_1U_2T}\right]\ge\frac{1}{15}\exp\left(-\frac{1}{4}\right)>\frac{1}{20}.$$

It follows that

$$\mathbb{E}\left[\max_{i=1,2}\sum_{t=1}^{T}x_t(i)\right]\ge\frac{1}{20}\left(\frac{TU_1}{2}+\frac{\sqrt{U_1U_2T}}{8}\right)+\frac{19}{20}\cdot\frac{TU_1}{2}=\frac{TU_1}{2}+\frac{1}{160}\sqrt{U_1U_2T}.$$

Any algorithm can only attain $U_1T/2$ reward in expectation. Thus, the regret of any algorithm is $\Omega(\sqrt{U_1U_2T})$. This completes the proof.

In the following, we proceed to analyze the general cases with more than two experts. Specifically, we use $\mathsf{num}(U_i)$ to denote the number of experts whose upper bound is larger than or equal to U_i.

Let i be any expert in $\mathcal{K}$. We set rewards of the i-th expert to be $U_i/2$ over all the T time slots and those in the experts whose upper bound less than U_i as 0. For the i'-th expert whose upper bound is large than or equal to U_i, we first randomly select $TU_i/U_{i'}$ time slots from the entire time horizon with the same probability. The set of selected time slots is denoted by $\mathcal{T}_{i'}$. Then, we assign

0 or $U_{i'}/U_i$ rewards to those time slots with equal probability. In this way, the reward means of all experts are $U_i/2$.

Similar to the above proof, for i' satisfying $U_{i'} \geq U_i$, we have

$$\Pr\left[\sum_{t\in\mathcal{T}_{i'}} x_t(i') \geq \frac{U_i T}{2} + h\right] = \Pr\left[\sum_{t\in\mathcal{T}_{i'}} \frac{x_t(i')}{U_{i'}} \geq \frac{U_i T}{2U_{i'}} + \frac{h}{U_{i'}}\right]$$
$$\geq \frac{1}{15}\exp\left(-16\frac{TU_i}{U_{i'}}\left(\frac{h}{TU_i}\right)^2\right) = \frac{1}{15}\exp\left(-16\frac{h^2}{U_iU_{i'}T}\right),$$

where $h \leq T/4U_iU_{i'}$. Define $A_{i'}$ as the event $\sum_{t\in\mathcal{T}_{i'}} x_t(i') \geq \frac{U_iT}{2}+h_{i'}$, where $h_{i'} = \sqrt{U_iU_{i'}T\ln \mathsf{num}(U_i)}/4$.

It follows that

$$P\left(\bigcup_{i':U_{i'}\geq U_i} \bar{A}_{i'}\right) \leq \prod_{i':U_{i'}\geq U_i}\left(1-\frac{1}{15\mathsf{num}(U_i)}\right) \leq \left(1-\frac{1}{15\mathsf{num}(U_i)}\right)^{\mathsf{num}(U_i)} \leq \exp(-1/15) < 0.95.$$

$A_{i'}$ happens with equal probability. Hence,

$$\mathbb{E}\left[\max_{i\in\mathcal{K}}\sum_{t=1}^{T} x_t(i)\right] \geq \sum_{i':U_{i'}\geq U_i}\frac{1}{20}\frac{1}{\mathsf{num}(U_i)}\left(\frac{TU_{i'}}{2}+\frac{\sqrt{U_iU_{i'}T\ln\mathsf{num}(U_i)}}{8}\right)+\frac{19}{20}\cdot\frac{TU_i}{2}$$
$$=\frac{TU_i}{2}+\frac{1}{160}\frac{1}{\mathsf{num}(U_i)}\sum_{i':U_{i'}\geq U_i}\sqrt{U_iU_{i'}T\ln\mathsf{num}(U_i)}.$$

Any algorithm can only attain $U_iT/2$ reward in expectation. Thus, the regret of any algorithm is $\Omega\left(\frac{1}{\mathsf{num}(U_i)}\sum_{i':U_{i'}\geq U_i}\sqrt{U_iU_{i'}T\ln\mathsf{num}(U_i)}\right)$. We complete the proof of Theorem 3.

Received February 2022; revised March 2022; accepted April 2022