An Online Minimax Optimal Algorithm for Adversarial Multi-Armed Bandit Problem

Kaan Gokcesu and Suleyman S. Kozat, *Senior Member, IEEE*

***Abstract*— We investigate the adversarial multi-armed bandit problem and introduce an online algorithm that asymptotically achieves the performance of the best switching bandit arm selection strategy. Our algorithms are truly online such that we do not use the game length or the number of switches of the best arm selection strategy in their constructions. Our results are guaranteed to hold in an individual sequence manner since we have no statistical assumptions on the bandit arm losses. Our regret bounds, i.e., our performance bounds with respect to the best bandit arm selection strategy, are minimax optimal up to logarithmic terms. We achieve the minimax optimal regret with computational complexity only log-linear in the game length. Thus, our algorithms can be efficiently used in applications involving big data. Through extensive set of experiments involving synthetic and real data, we demonstrate significant performance gains achieved by the proposed algorithm with respect to the state-of-the-art switching bandit algorithms. We also introduce a general efficiently implementable bandit arm selection framework, which can be adapted to various applications.**

***Index Terms*— Adversarial multi-armed bandit, switching bandit, minimax optimal, individual sequence manner, big data.**

I. Introduction

A. Preliminaries

In contemporary online learning literature, one of the main subjects of interest is reinforcement learning, where an agent (or algorithm) takes actions to maximize a certain reward in a given environment [1]. This area of online learning is heavily investigated in different fields from decision theory [2], game theory [3], [4] and signal processing [5] to control theory [6] and multi-agent systems [7]. In these applications of reinforcement learning, we encounter the fundamental dilemma of exploration-exploitation trade-off, which is most throughly studied in the multi-armed bandit problem [8]. The multi-armed bandit problem is generally considered to be the limited feedback version of the well studied prediction with expert advice [9]–[13]. It has attracted significant attention since the bandit setting can be successfully applied to a wide range of learning applications from recommender systems [14] and dimensionality reduction [15] to probability matching [16]. In applications regarding multi-armed bandit problems, we have a set of M algorithms (or action generating mechanisms) that run in parallel on a task (such as different advertisements or campaigns) and each of them is considered as an arm of a multi-armed bandit [17]. At each round of the decision process, we select one of the arms. Due to the nature of these applications only the loss (or the gain) of this selected arm is observed (where the performance of all the other arms remain hidden). An example application is the ad placement on a website. Suppose that we have M number of advertisements (M bandit arms) at our disposal that we can show to the visitors of a website and because of the space considerations, we can only show one of these ads (i.e, we need to choose one of these bandit arms). If there is an interaction by the visitor and the ad is clicked, the ad placement is successful. Otherwise, we incur a loss because the ad is ignored. We can only know the outcome of the ad we selected and cannot know whether or not the other ads would have been clicked.

We study the multi-armed bandit problem in an online setting, where we operate continuously on a stream of observations from a possibly nonstationary, chaotic or even adversarial environment. We assume no statistical assumptions on the loss sequence (the environment) so that our results are guaranteed to hold in an individual sequence manner. To this end, we investigate the multi-armed bandit problem from a competitive algorithm perspective. Since we have no statistical assumptions on the losses of the bandit arms, we define our performance with respect to a competing class of strategies. As the competition class, we use the class of switching bandit arm selection strategies and define our performance with respect to the best strategy (minimum loss) in this class. We point out that each such strategy constitutes a predetermined arm selection sequence (e.g., in a game of length T with M bandit arms, we have a total of M^T strategies). We emphasize that competing against the class of switching experts (or bandit arms in our setting) is extensively studied in control theory [18], [19], computational learning theory [20], [21], neural networks [22], [23], graph theory [9], signal processing [10], universal source coding [24]–[26] and multi-agent systems [27] due to their ability to construct algorithms that work in real life conditions. We also emphasize that in the competitive algorithm perspective, we do not need to explicitly know the actions (the bandit arms) we are presented with. Each bandit arm can even be separately running algorithms that learn throughout time. The only prior knowledge we need about the bandit arms is that there are M options (whatever they may be) that we can select from. At each time t, the action is chosen solely based on the sequential performance.

In this class, the optimal strategy is the one whose arm selection at each round of the game is optimum and has minimum loss. If the optimal strategy changes its arm selection a total of $S-1$ times, hence, has $S-1$ switches, we say the optimal strategy has S segments. Each such segment constitutes a part of the game (with possibly different lengths) where the optimum strategy's arm selection stays the same.

In this context, we introduce a truly online algorithm that asymptotically achieves the performance of the optimal strategy by observing only the loss of the selected arm at each

This work was supported in part by the Turkish Academy of Sciences Outstanding Researcher Programme and the Scientific and Technological Research Council of Turkey under Contract 113E517.

The authors are with the Department of Electrical and Electronics Engineering, Bilkent University, Bilkent, Ankara 06800, Turkey, Tel: +90 (312) 290-2336, Fax: +90 (312) 290-1223, (e-mail: gokcesu@ee.bilkent.edu.tr, kozat@ee.bilkent.edu.tr).

round t. Our algorithm does not require any knowledge of the game length T, the number of segments S, the lengths of these segments and the location of these segments. We emphasize that due to the limited information access of the bandit setting compared to the widely studied predicting with expert advice setting of the online learning literature, directly using the mixing techniques such as [11] are highly problematic. Since we do not have direct access to the loss of the non-selected arms in contrast to the expert setting, we represent their performance with unbiased estimates. However, since the optimal number of switches and the true loss values are not completely known, we cannot directly use merging (or derandomizing) strategies of [28]. Instead, we first design an online probability sharing network that sequentially combines the selections of all possible bandit arm selection strategies with carefully constructed weights, where the number of possible strategies grow with M^T. We efficiently implement this network by creating equivalence classes, which group certain strategies together, to store and update their weights collectively. After that, we construct multiple carefully designed probability sharing networks and tune each network's parameters (transition weights and learning rates) to different number of switches. By combining the beliefs of these probability sharing networks, we achieve the minimax optimal regret up to logarithmic factors with computational complexity only log-linear in the game length T without any knowledge of the number of switches the optimal strategy has.

There exists a significant amount of prior art to construct algorithms that can compete with the best bandit arm selection sequence chosen in hindsight (the optimal strategy). However, we, as the first time in the literature, introduce a truly online, low complexity (log-linear) algorithm whose performance with respect to the optimal strategy is minimax optimal (up to logarithmic factors). Through extensive set of experiments involving synthetic and real data, we also demonstrate significant performance gains achieved by the proposed algorithm with respect to the state-of-the-art algorithms [29]–[31].

B. Prior Art and Comparisons

The adversarial multi-armed bandit problem where the player competes against the best fixed arm has a regret lower bound of $O(\sqrt{MT})$[1] for M bandits in a T round game [32]. When competing against the best switching bandit arm strategy (as opposed to the best fixed arm strategy), we can apply $O(\sqrt{MT})$ bound separately to each one of S segment (if we know the switching instants). Hence, maximization of the total regret bound yields a minimax bound of $O(\sqrt{MTS})$ since the square-root function is concave and the bound is maximum when each segment is of equal length T/S. The state of the art algorithms [29]–[32] achieve an expected regret upper bound $\tilde{O}(\sqrt{MTS})$ when both the number of switches S and the game length T are known a priori. To achieve this regret in the absence of the knowledge of the game length T, [29]–[32] employ the doubling trick [33]. However, in that setting, while the total number of switches during the entire game S is known, the number of switches that occur in each epoch of the algorithms are not known. Therefore, the authors also construct a variant of their algorithms for the version of the game where the game length T is known and instead of the number of switches S, an upper bound $S_{\max}$ is known such that $S \leq S_{\max}$. Employing the doubling trick for these variants of the algorithms in [29]–[32] produce an expected regret upper bound of $\tilde{O}(\sqrt{MTS_{\max}})$, which in turn makes it possible for these algorithms to achieve an expected regret of $\tilde{O}(\sqrt{MTS})$ by knowing S but not T. However, their approach is not enough to achieve the minimax optimal regret in the absence of the knowledge of S as well.

In [30], the authors provide an algorithm that is able to attain an expected regret upper bound of $\tilde{O}(S\sqrt{MT})$ if the number of switches S is not known a priori. The analysis done in [30] is also applicable to [29], [31], [32] and the regret bound is achievable in the setting where game length T is also not known a priori as well. However, this bound differs from the optimal regret bound by a factor of $\sqrt{S}$. Therefore, if the number of switches S is not known a priori, the optimal regret bound on the expected regret is not achievable with the algorithms [29]–[32]. To this end, we introduce an online randomized algorithm that has an expected regret bound of $\tilde{O}(\sqrt{MTS})$ whether or not the number of switches S and the game length T are known. The computational complexity of our final algorithm is $O(M \log T)$ per round while the complexity of the algorithms [29]–[32] is $O(M)$ per round. Thus, our algorithm uniformly achieves the minimax optimal regret with only a logarithmic increase in computational complexity.

C. Contributions

Our main contributions are as follows.

- We introduce an online algorithm, which achieves the performance of the best bandit arm selection strategy, where the regret of our algorithm is minimax optimal up to logarithmic factors. Our results are uniformly guaranteed to hold in an individual sequence manner for all possible arm loss sequences since we refrain from making any statistical assumptions on the bandit arms.
- Our algorithm is truly online such that neither the length of the game T nor the number of switches S of the optimal strategy are used to achieve the performance of the best bandit arm selection strategy whose loss is minimum at each round individually.
- We achieve this performance with a computational complexity and storage demand only log-linear in the game length T. Thus, our algorithm can be efficiently used in applications involving big data.
- Through extensive set of experiments involving synthetic and real data, we demonstrate significant performance gains achieved by the proposed algorithm with respect to the state-of-the-art adversarial multi-armed bandit algorithms in the reinforcement learning and computational learning theory literature [29]–[32].
- We introduce a general and efficient arm selection framework that can be adapted to various applications.

D. Organization of the Paper

The organization of this paper is as follows. We first define the adversarial multi-armed bandit problem in Section II. In Section III, we introduce a general bandit arm selection framework. We provide the brute force approach in Section

[1]We use big-O notation, i.e., $O(f(x))$ to ignore constant factors and use soft-O notation, i.e., $\tilde{O}(f(x))$ to ignore the logarithmic factors as well.

III-A and introduce the efficient and elegant implementation of the brute force approach with computational complexity only polynomial in the game length in Section III-B. In Section IV, we construct an algorithm that achieves the minimax optimal regret with prior information of S and T. In Section V, we construct a truly online algorithm that achieves the minimax optimal regret with no prior information. We demonstrate the performance of our algorithms via extensive set of experiments in Section VI and conclude with final remarks in Section VII.

II. Problem Description

In this paper[2], we study the adversarial multi-armed bandit problem where we have M bandit arms and randomly select one of the arms at each round t. Based on our online selection $\{u_t\}_{t\geq 1}$, $u_t \in \{1, 2, \ldots, M\}$, we receive only the loss of the selected arm $\{l_{t,u_t}\}_{t\geq 1}$, $l_{t,u_t} \in [0, 1]$ and we do not know the losses of the arms we did not chose. We assume $l_{t,u_t} \in [0, 1]$ for notational simplicity, however, our derivations hold for any bounded loss after shifting and scaling in magnitude. In a T round game, we define $\mathbf{u}_T$ as the column vector containing the user selections up to time T as $\mathbf{u}_T = [u_1, \ldots, u_T]^T$. We define the variable $\mathbf{s}_T$ as the column vector representing a deterministic bandit arm selection sequence of length T as $\mathbf{s}_T = [s_1, \ldots, s_T]^T$ such that $s_t \in \{1, 2, \ldots, M\}$ for all t. In the rest of the paper, we refer to each such deterministic bandit arm selection sequence, $\mathbf{s}_T$, as a strategy. We define $\boldsymbol{l}_{\mathbf{s}_T}$ as the loss sequence of the strategy $\mathbf{s}_T$, $\boldsymbol{l}_{\mathbf{s}_T} = [l_{1,s_1}, \ldots, l_{T,s_T}]^T$. Thus, the loss sequence of $\mathbf{u}_T$ becomes $\boldsymbol{l}_{\mathbf{u}_T} = [l_{1,u_1}, \ldots, l_{T,u_T}]^T$.

We work in the adversarial bandit setting such that we do not assume any statistical model on the behavior of the bandit arms [32] and our algorithms are guaranteed to work in an individual sequence manner. The output u_t of our algorithm at each round t is strictly online and randomized. It is a function of only the past selections and observed losses as

$$u_t \triangleq u_t(\boldsymbol{l}_{\mathbf{u}_{t-1}}; \mathbf{u}_{t-1}), \;\; u_t \in \{1, ..., M\}. \tag{1}$$

We denote the accumulated loss at time T of any strategy $\mathbf{s}_T$ by $L_{\mathbf{s}_T} = \sum_{t=1}^{T} l_{t,s_t}$. Since we assume no statistical assumptions on the loss sequence, we define our performance with respect to the optimum strategy $\mathbf{s}_T^* = [s_1^*, \ldots, s_T^*]$, which is given as

$$\mathbf{s}_T^* = \arg\min_{\mathbf{s}_T} L_{\mathbf{s}_T} \quad \text{or} \quad s_t^* = \arg\min_{s_t} l_{t,s_t}, 1 \leq t \leq T. \tag{2}$$

We use the notion of regret to define our performance as

$$R_T \triangleq \sum_{t=1}^{T} l_{t,u_t} - \sum_{t=1}^{T} l_{t,s_t^*} = L_{\mathbf{u}_T} - L_{\mathbf{s}_T^*}, \tag{3}$$

where we denote the regret accumulated in T rounds as R_T. The regret R_T depends on how hard it is to learn the optimum strategy $\mathbf{s}_T^*$. We quantify the hardness of learning the optimum strategy by the number of switches it has since at every switch we need to learn again the optimal arm from scratch. For $\mathbf{s}_T^*$, we define a switch as the event when its arm selection changes between consecutive rounds. We denote S as the total number of such switches where we also count the beginning as a switch such that $S = 1 + \sum_{t=2}^{T} \mathbb{1}_{s_t^* \neq s_{t-1}^*}$, and $\mathbb{1}_x$ is the indicator function that outputs 1 if the statement x is true, and 0 otherwise. Each part of $\mathbf{s}_T^*$ starting with a switch constitute a segment. Thus, the selected arm throughout a segment is the same, however, the selected arms at successive segments are strictly different. As an example, consider the strategy

$$\mathbf{s}_{10} = \{3, 3, 1, 1, 1, 1, 7, 3, 3, 3\}. \tag{4}$$

The strategy $\mathbf{s}_{10}$ in (4) has $S = 4$ switches and it selects the arms 3,1,7 and 3 in segments 1,2,3 and 4 respectively.

Our goal is to introduce an algorithm that achieves the minimax optimal expected regret up to logarithmic terms, i.e., $\mathbf{E}[R_T] \leq \tilde{O}(\sqrt{MTS})$, without any information on the bandit arms, the game length T and the number of switches S.

III. General Bandit Arm Selection Framework

In this section, to achieve the performance of the best a priori selected strategy, we consider all possible arm selection strategies and combine them with exponential weights similar to [32]. To this end, we need to combine M^T different arm selection strategies in a T round game since one can select one of M bandit arms in each round yielding M^T strategies. Naturally, one of these M^T strategies has the optimal selection with the minimum loss, at every round t.

To achieve the performance of the optimum strategy over any loss sequence, we can consider each one of M^T strategies as a deterministic expert and combine them with performance weights just like the algorithm "Exponential-weight algorithm for Exploration and Exploitation using Expert advice" (Exp4) [30], since one of the M^T strategies is by definition optimal (but it is not known a priori). However, mixture algorithms, when used with uniform weights, produce $O(\sqrt{T \log N})$ regret and have $O(N)$ computational complexity where N is the number of algorithms combined. Hence, a straightforward combination of these exponential number of algorithms produces a non-vanishing regret bound $O(T)$ and has an exponential in time computational complexity. Therefore, we need to intelligently combine these strategies with carefully selected and efficiently computable combination weights in an online manner to produce vanishing regret in polynomial time. To achieve this, we assign different weights for each strategy based on its complexity cost (the number of switches a strategy has [24]). This weight selection is in line with the complexity penalty of AIC and MDL [34], [35]. Strategies with larger number of switches are assigned lower weights.

We first consider the brute force approach to explain the framework in detail and to derive the regret bounds, which will also hold for the efficient implementation.

A. Brute Force Approach

We hypothetically assume that we run all possible M^T strategies in parallel, i.e., all the possible bandit arm selection sequences for a T length game, in an online manner. However, note that, at time t, we have M^t parallel running strategies. Each of these strategies suggest a different bandit arm (in correspondence to their selection sequence) to use at time t. We assign each of these strategies, $\mathbf{s}_t$, a weight $w_{\mathbf{s}_t}$ (as detailed later in this section), which shows our trust on this particular strategy. Based on these weights, we create a probability simplex and assign each parallel running strategies a probability by normalizing the weights $w_{\mathbf{s}_t}$ as

$$P_{\mathbf{s}_t} = \frac{w_{\mathbf{s}_t}}{\sum_{\mathbf{s}_t' \in \mathbb{M}_t} w_{\mathbf{s}_t'}}, \tag{5}$$

[2]All vectors are column vectors and denoted by boldface lower case letters. We work with real data for notational simplicity. We use $\log(x)$ to denote logarithm with base 2 and $\ln(x)$ to denote the natural logarithm.

where $\mathbb{M}_t$ is the class of all strategies up to time t, and its size is M^t, i.e., $|\mathbb{M}_t| = M^t$. To make our selection at time t, for each arm m, we find the strategies among all M^t strategies that suggest m at round t and sum their assigned probabilities

$$p_{t,m} = \sum_{\mathbf{s}_t(t:t)=m} P_{\mathbf{s}_t}, \tag{6}$$

where $\mathbf{s}_t(i:j)$ is the vector consisting of i^{th} through j^{th} elements of $\mathbf{s}_t$, e.g., $\mathbf{s}_t(t:t) = s_t$. By summing the probabilities of strategies that suggests the same bandit arm, we construct the probabilities of each bandit arm at time t.

The strategies to be used are not specifically selected a priori. Instead, at each time t, all of the strategies $\mathbf{s}_t$ that compromise the class $\mathbb{M}_t$ are treated as experts in our online learning problem [11], [13], [21]. These strategies (or experts) are combined according to their weights $w_{\mathbf{s}_t}$, which indicates our trust in different strategies, to achieve the performance of the optimal expert. Hence, our algorithm intrinsically achieves the performance of the optimal strategy without knowing which strategy specifically has the best performance because of its universal prediction perspective [36].

The construction of (5) and (6) directly depends on $w_{\mathbf{s}_t}$, the weight we assign to each parallel running strategy. The weight assignment $w_{\mathbf{s}_t}$ has two components. As the first component, we assign an a priori weight $\mathcal{T}(\mathbf{s}_t)$ to each $\mathbf{s}_t$, which only depends on the complexity of $\mathbf{s}_t$, e.g., how many switches it has, (more detail will be given for $\mathcal{T}(\mathbf{s}_t)$ later on). The second part directly depends on the past performance of $\mathbf{s}_t$, which is the exponential weight, $\exp(-\eta \tilde{L}_{\mathbf{s}_t(1:t-1)})$. Hence, the combined weight is given by

$$w_{\mathbf{s}_t} = \mathcal{T}(\mathbf{s}_t) e^{-\eta \tilde{L}_{\mathbf{s}_t(1:t-1)}}, \tag{7}$$

where η is the learning rate and $\tilde{L}_{\mathbf{s}_t(1:t-1)}$ is the unbiased estimator of $L_{\mathbf{s}_t(1:t-1)}$.

The combination weights, $\mathcal{T}(\mathbf{s}_t)$, are designed by us and are predetermined. To get a truly online algorithm, we choose sequentially calculable $\mathcal{T}(\mathbf{s}_t)$ such that we have a telescoping rule as: $\mathcal{T}(\mathbf{s}_t) = \mathcal{T}(\mathbf{s}_t|\mathbf{s}_t(1:t-1))\mathcal{T}(\mathbf{s}_t(1:t-1))$, where we denote the relative weight update from the strategy $\mathbf{s}_t(1:t-1)$ to the strategy $\mathbf{s}_t$ by $\mathcal{T}(\mathbf{s}_t|\mathbf{s}_t(1:t-1))$. To get a probability score, we design the relative weight updates such that

$$\sum_{m=1}^{M} \mathcal{T}([\mathbf{s}_t; m]|\mathbf{s}_t) = 1, \ \ \forall \mathbf{s}_t, \ \ t \in \{0, \ldots, T-1\}, \tag{8}$$

where $[\mathbf{s}_t; m]$ denotes concatenation of the vector $\mathbf{s}_t$ and m (creating a new strategy of length $t+1$), $\mathbf{s}_0 = [\emptyset]^T$ and $\mathcal{T}(\mathbf{s}_0) = 1$. The exponential weights are the exponential losses of each strategy up to time $t-1$. Thus, the joint weight, $w_{\mathbf{s}_t}$, assigned to each strategy is sequentially constructible such that

$$w_{\mathbf{s}_t} = w_{\mathbf{s}_t(1:t-1)} \mathcal{T}(\mathbf{s}_t|\mathbf{s}_t(1:t-1)) e^{-\eta \tilde{l}_{t,\mathbf{s}_t(t-1:t-1)}}. \tag{9}$$

Note that we only observe the loss of the selected arm at time t, l_{t,u_t}. Thus, we construct estimates $\tilde{l}_{t,m}$ for the other losses. We use the well known unbiased estimator

$$\tilde{l}_{t,m} = \begin{cases} l_{t,m}/p_{t,m} & m = u_t \\ 0 & m \neq u_t \end{cases}, \tag{10}$$

which gives $\mathbf{E}[\tilde{l}_{t,m}] = l_{t,m}$. Hence, expected value of each estimate is its true value [32]. We also define an expectation operator $\mathbf{E}_m$ over bandit arms $m \in \{1, \ldots, M\}$ such that $\mathbf{E}_m[f(m)] = \sum_{m=1}^{M} p_{t,m} f(m)$, hence, $\mathbf{E}_m[\tilde{l}_{t,m}] = \sum_{m=1}^{M} p_{t,m} \tilde{l}_{t,m} = l_{t,u_t}$.

Using the bandit arm selection probability assignment given in (5), (6), (7), we have the following regret result.

Theorem 1. *Let $m \in \{1, ..., M\}$ be the arms of an adversarial multi-armed bandit and $l_{t,m}$ be the loss incurred from selecting the arm m at round t such that $l_{t,m} \in [0,1]$. Using any sequentially constructible combination weight assignment $\mathcal{T}(\cdot)$ satisfying* (8) *and exponential losses as in* (7) *to determine the selection probabilities of each arm yields an expected regret*

$$\mathbf{E}[R_T] \leq \min_{\boldsymbol{s}_T} \left(\frac{\eta M T}{2} + \frac{1}{\eta} \ln W(\boldsymbol{s}_T) + L_{\boldsymbol{s}_T} - L_{\boldsymbol{s}_T^*} \right), \tag{11}$$

accumulated in a T round game where $\eta \geq 0$ is the learning rate in the exponential weights and $W(\boldsymbol{s}_T) \triangleq 1/\mathcal{T}(\boldsymbol{s}_T)$, i.e., the reciprocal of the combination weight of the strategy $\boldsymbol{s}_T$ (the complexity cost). $L_{\boldsymbol{s}_T}$ is the cumulative loss of the strategy $\boldsymbol{s}_T$ and $L_{\boldsymbol{s}_T^*}$ is the cumulative loss of the optimum arm selection strategy, $\boldsymbol{s}_T^*$, chosen a priori with full information of the losses of every arm $m \in \{1, ..., M\}$ in every round $t \in \{1, ..., T\}$.

The result in Theorem 1 indicates that by careful design of $\mathcal{T}(\mathbf{s}_t)$ and η, one can achieve sublinear and even optimal regret. However, the weight assignment $\mathcal{T}(\mathbf{s}_t)$ needs to be sequentially constructible and also satisfy (8).

If we naively assign equal probability to each strategy, the predetermined weight updates $\mathcal{T}(\mathbf{s}_t|\mathbf{s}_t(1:t-1))$ become $1/M$ for all $\mathbf{s}_t$. However, such a selection makes the combination weight $\mathcal{T}(\mathbf{s}_T^*)$ of the optimal strategy $\mathbf{s}_T^*$ at the end of T rounds, M^{-T}. Since by Theorem 1, the regret is dependent on the negative logarithm of this combination weight, we end up with a linear regret bound, which is undesirable (the average regret does not diminish). Hence, we need to penalize the strategies according to their complexity much like the complexity penalty of AIC and MDL [34], [35]. In Section IV, we construct an algorithm that can achieve the minimax optimal regret bound and demonstrate how the combination weight update $\mathcal{T}(\mathbf{s}_t|\mathbf{s}_t(1:t-1))$ should be determined for that particular situation.

Moreover, the result in Theorem 1 implies that the performance of our framework is dependent on not only the complexity cost of the optimum strategy ($W(\mathbf{s}_T^*) = 1/\mathcal{T}(\mathbf{s}_T^*)$) but also the complexity cost of the strategies whose loss is relatively close to the loss of the optimum strategy (the optimum loss). Therefore, even if the optimal strategy were to have a high complexity cost (high number of switches), our algorithm can attain a relatively low regret if there exist a strategy with low complexity cost (small number of switches) that has a loss sufficiently close to the optimal loss. We emphasize that the expectation in (11) is due to the randomization in our algorithm such that the result uniformly holds for any sequence of bandit losses without any statistical assumption.

Proof of Theorem 1. The regret against the optimum selection strategy $\mathbf{s}_T^*$, where $\mathbf{s}_T^* = [s_1^*, s_2^*, \ldots, s_{T-1}^*, s_T^*]$, at time t is given by $r_t = l_{t,u_t} - l_{t,s_t^*}$. We transform r_t into a more manageable form and construct two distinct terms, which we will bound separately as

$$r_t = \left(l_{t,u_t} + \frac{\ln \mathbf{E}_m[e^{-\eta \tilde{l}_{t,m}}]}{\eta} \right) - \left(\frac{\ln \mathbf{E}_m[e^{-\eta \tilde{l}_{t,m}}]}{\eta} + l_{t,s_t^*} \right). \tag{12}$$

We bound the first term in (12) by using $\ln x \leq x-1$ for $x>0$

$$\ln \mathbf{E}_m[e^{-\eta \tilde{l}_{t,m}}] \leq \mathbf{E}_m\left[e^{-\eta \tilde{l}_{t,m}}-1\right]. \quad (13)$$

Using $e^{-x}-1+x \leq x^2/2$ for $x>0$ bounds (13) as

$$\ln \mathbf{E}_m[e^{-\eta \tilde{l}_{t,m}}] \leq \mathbf{E}_m\left[\frac{\eta^2 \tilde{l}_{t,m}^2}{2}\right]-\eta \mathbf{E}_m[\tilde{l}_{t,m}] \leq \frac{\eta^2 l_{t,u_t}^2}{2p_{t,u_t}}-\eta l_{t,u_t}. \quad (14)$$

Putting (14) into the first term of (12) yields

$$r_t \leq \frac{\eta}{2p_{t,u_t}}+\left[-\frac{1}{\eta}\ln \mathbf{E}_m[e^{-\eta \tilde{l}_{t,m}}]-l_{t,s_t^*}\right], \quad (15)$$

since $l_{t,u_t} \leq 1$. To upper bound the second term in (15), we calculate the expectation using (6) and (5) as

$$\mathbf{E}_m[e^{-\eta \tilde{l}_{t,m}}] = \sum_{m=1}^{M} p_{t,m} e^{-\eta \tilde{l}_{t,m}} = \sum_{\mathbf{s}'_t \in \mathbb{M}^t} P_{\mathbf{s}'_t} e^{-\eta \tilde{l}_{t,\mathbf{s}'_t(t:t)}},$$

$$= \sum_{\mathbf{s}'_t \in \mathbb{M}^t} \frac{w_{\mathbf{s}'_t}}{\sum_{\mathbf{s}''_t \in \mathbb{M}^t} w_{\mathbf{s}''_t}} e^{-\eta \tilde{l}_{t,\mathbf{s}'_t(t:t)}}. \quad (16)$$

Using (7) in (16) provides

$$\mathbf{E}_m[e^{-\eta \tilde{l}_{t,m}}] = \frac{\sum_{\mathbf{s}'_t \in \mathbb{M}_t} \mathcal{T}(\mathbf{s}'_t) e^{-\eta \tilde{L}_{\mathbf{s}'_t}}}{\sum_{\mathbf{s}''_t \in \mathbb{M}_t} \mathcal{T}(\mathbf{s}''_t) e^{-\eta \tilde{L}_{\mathbf{s}''_t(1:t-1)}}} = \frac{\sum_{\mathbf{s}'_t \in \mathbb{M}_t} \mathcal{T}(\mathbf{s}'_t) e^{-\eta \tilde{L}_{\mathbf{s}'_t}}}{\sum_{\mathbf{s}''_{t-1} \in \mathbb{M}_{t-1}} \mathcal{T}(\mathbf{s}''_{t-1}) e^{-\eta \tilde{L}_{\mathbf{s}''_{t-1}}}}, \quad (17)$$

where we used (8). Henceforth, summing the logarithm of (17) for all T rounds yields

$$\sum_{t=1}^{T} \ln \mathbf{E}_m[e^{-\eta \tilde{l}_{t,m}}] = \ln \sum_{\mathbf{s}'_T \in \mathbb{M}^T} \mathcal{T}(\mathbf{s}'_T) e^{-\eta \tilde{L}_{\mathbf{s}'_T}}. \quad (18)$$

Then, we multiply (18) with $-1/\eta$ and upper-bound as

$$\sum_{t=1}^{T} -\frac{1}{\eta}\ln \mathbf{E}_m[e^{-\eta \tilde{l}_{t,m}}] = -\frac{1}{\eta}\ln \sum_{\mathbf{s}'_T \in \mathbb{M}^T} \mathcal{T}(\mathbf{s}'_T) e^{-\eta \tilde{L}_{\mathbf{s}'_T}},$$

$$\leq -\frac{1}{\eta}\ln\left[\mathcal{T}(\mathbf{s}_T) e^{-\eta \tilde{L}_{\mathbf{s}_T}}\right] \leq -\frac{1}{\eta}\ln \mathcal{T}(\mathbf{s}_T) + \tilde{L}_{\mathbf{s}_T}, \quad (19)$$

for any $\mathbf{s}_T \in \mathbb{M}_T$. Summing (15) for all T rounds yields the regret accumulated in a T round game as

$$R_T = \sum_{t=1}^{T} \frac{\eta}{2p_{t,u_t}} - \frac{1}{\eta}\sum_{t=1}^{T} \ln \mathbf{E}[e^{-\eta \tilde{l}_{t,m}}] - \sum_{t=1}^{T} l_{t,s_t^*}. \quad (20)$$

Putting (19) into (20) gives the total regret R_T as

$$R_T \leq \sum_{t=1}^{T} \frac{\eta}{2p_{t,u_t}} - \frac{1}{\eta}\ln \mathcal{T}(\mathbf{s}_T) + \tilde{L}_{\mathbf{s}_T} - L_{\mathbf{s}_T^*}. \quad (21)$$

Our selections u_t, hence p_{t,u_t}, for every t are random variables in R_T. We take the expectation of (21) with respect to the arm selection probabilities (over $\{u_t\}_{t\geq 1}^T$), which gives

$$\mathbf{E}[R_T] \leq \frac{\eta MT}{2} - \frac{1}{\eta}\ln \mathcal{T}(\mathbf{s}_T) + L_{\mathbf{s}_T} - L_{\mathbf{s}_T^*}.$$

For notational simplicity, we change the notation from combination weight to the complexity cost of the strategy such that $W(\mathbf{s}_T) \triangleq 1/\mathcal{T}(\mathbf{s}_T)$. Hence,

$$\mathbf{E}[R_T] \leq \frac{\eta MT}{2} + \frac{1}{\eta}\ln W(\mathbf{s}_T) + L_{\mathbf{s}_T} - L_{\mathbf{s}_T^*}. \quad (22)$$

Since (22) is satisfied for any strategy $\mathbf{s}_T$, a tighter bound can be found by minimizing (22) over $\mathbf{s}_T$, which gives (11). □

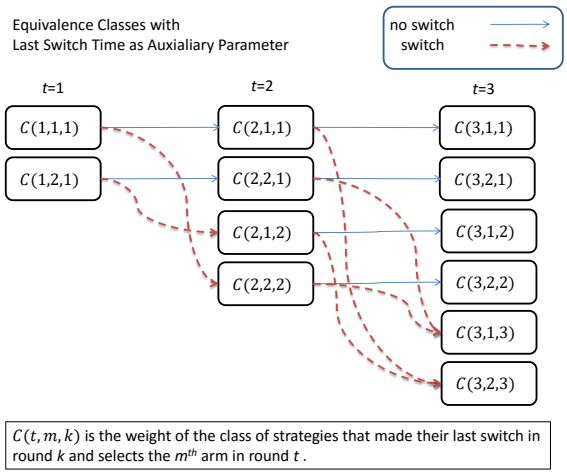


Fig. 1: Efficient combination example for a 2-armed bandit case using last switch time as the auxiliary parameter for the first three rounds. Equivalence classes are denoted as $C(t,m,k)$ where t is the present round, m is the bandit arm associated with that class and k is the time index of the last switch, i.e time index of the start of the present segment. In this case, the auxiliary parameter vector $\boldsymbol{\sigma}_t$ includes just k and the possible values it can take increase linearly with time. Thus, this formulates an algorithm with linear complexity since k can only have t different values.

Corollary 1. *To get an upper bound independent from the loss sequence, we can set $\boldsymbol{s}_T = \boldsymbol{s}_T^*$ in Theorem 1, which will upper bound* (11) *with*

$$\mathbf{E}[R_T] \leq \frac{\eta MT}{2} + \frac{1}{\eta}\ln W(\boldsymbol{s}_T^*), \quad (23)$$

where $W(\boldsymbol{s}_T^*)$ is the reciprocal of the combination weight of the optimum arm selection strategy $\boldsymbol{s}_T^*$.

The computational complexity of the brute force approach is $O(M^t)$ in each round t since both the combination weights and exponential weights of each strategy are updated in every time instant. However, both the combination and exponential weights can be designed to be sequentially constructible and efficiently computable. In Section III.B, we introduce an efficient implementation and reduce the computational complexity to polynomial in time.

B. Efficient Implementation

The complexity of the brute force algorithm increases exponentially since we combine an exponentially increasing number of strategies. Every strategy, $\mathbf{s}_t \in \mathbb{M}_t$, has a weight $w_{\mathbf{s}_t}$, which has to be stored and updated at every round t. To circumvent this, we create equivalence classes that group together certain strategies to reduce computational complexity.

We define $C(t,m,\boldsymbol{\sigma}_t)$ as the weight of the equivalence class of arm m and auxiliary parameters $\boldsymbol{\sigma}_t$ at time t. The equivalence class $C(t,m,\boldsymbol{\sigma}_t)$ includes all the strategies $\mathbf{s}_t$ that selects m^{th} arm at time t and whose behavior match with the parameter vector $\boldsymbol{\sigma}_t$. As an example, consider the case in Fig. 1, where $\boldsymbol{\sigma}_t$ only includes the time index of the last switch the strategies have made. Grouping the strategies together by their last switch time results in linearly increasing number of equivalence classes, e.g., strategies whose last switch was at time $t-1$, whose last switch was at time $t-2$ and so on. We emphasize that the auxiliary parameter vector $\boldsymbol{\sigma}_t$ can include different groupings such as the number of switches the strategies have made, e.g., the strategy $\mathbf{s}_{10}$ in (4) selects the

arm $m = 3$ at time $t = 10$ and has $S = 4$ number of switches, hence, it is in the equivalence class $C(10, 3, 4)$ since $\boldsymbol{\sigma}_{10} = 4$. If the auxiliary parameter were to include both the last switch time ($t' = 8$) and the number of switches ($S = 4$), $\mathbf{s}_{10}$ in (4) would be in the equivalence class with the auxiliary variable $\boldsymbol{\sigma}_{10} = [8, 4]^T$, i.e., $C(10, 3, [8, 4]^T)$.

The parameters included in $\boldsymbol{\sigma}_t$ determine its extend and how many different strategies it will represent, which in turn determines how many equivalence classes we will have at the end. The reason for using auxiliary parameters $\boldsymbol{\sigma}_t$ is to group together certain strategies whose weight update in (9), i.e., $\mathcal{T}(\mathbf{s}_t|\mathbf{s}_t(1:t-1))\exp(-\eta\tilde{l}_{t,\mathbf{s}_t(t-1:t-1)})$, is the same. Therefore, we need to include in $\boldsymbol{\sigma}_t$ all the possible parameters that are related to the combination weight update $\mathcal{T}(\mathbf{s}_t|\mathbf{s}_t(1:t-1))$. Hence, the design of the combination weight assignment $\mathcal{T}(\cdot)$ influences the parameters required to be included in $\boldsymbol{\sigma}_t$. We define Λ_t as the vector space including all possible $\boldsymbol{\sigma}_t$ vectors.

The weight of an equivalence class is simply the summation of the weights of the strategies whose behavior conforms with its class parameters $t, m, \boldsymbol{\sigma}_t$, such that

$$C(t, m, \boldsymbol{\sigma}_t) = \sum_{\substack{\mathbf{s}_t(t:t)=m \\ \boldsymbol{\sigma}(\mathbf{s}_t)=\boldsymbol{\sigma}_t}} w_{\mathbf{s}_t}, \tag{24}$$

where $\boldsymbol{\sigma}(\cdot)$ is the mapping from strategies $\mathbf{s}_t$ to the auxiliary parameters $\boldsymbol{\sigma}_t$, $\boldsymbol{\sigma} : \mathbb{M}_t \rightarrow \Lambda_t$, and $w_{\mathbf{s}_t}$ is defined in (7). In Fig. 1, example equivalence classes for a 2-armed bandit game for the first 3 rounds are given, where $C(t, m, k)$ denotes the weight of the class of strategies that made their last switch at the k^{th} round and selects m^{th} arm at round t. As an example, $C(3, 1, 3)$ is the weight of the class (set) of strategies $\mathbf{s}_3 \in \{[2, 2, 1]^T, [1, 2, 1]^T\}$. Since our goal is to do the multiplicative update in (9) at the same time for a number of strategies, the update $\mathcal{T}(\mathbf{s}_t|\mathbf{s}_t(1:t-1))\exp(-\eta\tilde{l}_{t,\mathbf{s}_t(t-1:t-1)})$ needs to be the same for every strategy in the mix. The exponential loss update is the same if the arms to be chosen by the strategies are the same, which is satisfied if the strategies belong to the same equivalence class. We design the combination weight update to be dependent on the class parameters, which are made up of the present round t, the choice of bandit arm in the present round m and the auxiliary parameters $\boldsymbol{\sigma}_t$ such that each strategy in the same class have the same combination weight update. We denote the common combination weight update from the equivalence class $C(t,m',\boldsymbol{\sigma}'_t)$ to $C(t+1,m,\boldsymbol{\sigma}_{t+1})$ by $\mathcal{T}(t+1,m,\boldsymbol{\sigma}_{t+1}|t,m',\boldsymbol{\sigma}'_t)$, where we use m' and $\boldsymbol{\sigma}'_t$ to differentiate between subsequent time instances. Thus,

$$C(t+1,m,\boldsymbol{\sigma}_{t+1}) = \sum_{m',\boldsymbol{\sigma}'_t} C(t,m',\boldsymbol{\sigma}'_t)\mathcal{T}(t+1,m,\boldsymbol{\sigma}_{t+1}|t,m',\boldsymbol{\sigma}'_t)e^{-\eta\tilde{l}_{t,m'}}, \tag{25}$$

since each equivalence class weight is the summation of the joint weights of all strategies that conform to its parameters.

Naturally, the computational complexity per round is dependent on the number of equivalence classes. Therefore, if we design the equivalence classes in such a way that their number increases polynomially, the computational complexity will reduce to polynomial in time from exponential in time.

Remark 1. *In the brute force approach, $\boldsymbol{\sigma}_t$ includes all the past selections of a strategy. Therefore, each equivalence class included only one strategy hence the number of classes increased exponentially with time (M^t).*

Algorithm 1 Efficient General Framework

Initialize constant $\eta \in \mathbb{R}^+$
Select combination weight assignment
Set Λ_t for $t \in 1, ..., T$ accordingly
Initialize $\boldsymbol{\sigma}_1$, which is the extend of Λ_1
Set $C(1, m, \boldsymbol{\sigma}_1) = 1/M$ for $m \in 1, ..., M$
Initialize $p_{1,m} = C(1, m, \boldsymbol{\sigma}_1)$
for $t = 1$ **to** T **do**
  Select one of the M arms with probability $p_{t,m}$
  Receive loss l_{t,u_t}
  Set $\tilde{l}_{t,m} = \frac{l_{t,m}\mathbb{1}_{m=u_t}}{p_{t,m}}$ for $m \in 1, ..., M$
  for $\boldsymbol{\sigma}_{t+1} \in \Lambda_{t+1}$ **do**
    for $m = 1$ **to** M **do**
      Do the update in (25)
    end for
  end for
  for $m = 1$ **to** M **do**
    Set $p_{t+1,m} = \frac{\sum_{\boldsymbol{\sigma}_{t+1}\in\Lambda_{t+1}} C(t+1,m,\boldsymbol{\sigma}_{t+1})}{\sum_{m=1}^{M}\sum_{\boldsymbol{\sigma}_{t+1}\in\Lambda_{t+1}} C(t+1,m,\boldsymbol{\sigma}_{t+1})}$
  end for
end for

We need to emphasize that the equivalence classes have to group together the strategies by the last selected arm since the exponential loss update is dependent on the last selected arm. The auxiliary parameters in $\boldsymbol{\sigma}_t$ are used for updating the combination weights. In Algorithm 1, we provide the complete efficient implementation of the general framework.

The efficient implementation in Algorithm 1 directly implements the weight assignment of the brute force approach in Section III-A since (25) is the direct implementation of (9) by using (24). Therefore, all the regret analysis done for the brute force approach holds for the efficient implementation as well, i.e., Theorem 1 and Corollary 1 holds for Algorithm 1. We have shown that by using equivalence classes we can reduce the computational complexity from exponential in time to polynomial in time since the computational complexity is related to the number of equivalence classes, which is $M|\Lambda_t|$ at each round t (where $|\Lambda_t|$ is the total number of distinct auxiliary parameter vectors $\boldsymbol{\sigma}_t$).

Using more auxiliary parameters in $\boldsymbol{\sigma}_t$ provides more flexibility in the general weight assignments. The generality of this framework and the weight assignments provide a wide range of possibilities for various applications. Different weighting assignments can be designed for different environments. The weighting assignment can be tailored for different complexity cost functions (instead of the number of switches, which is the focus of our paper). For example, instead of treating all switches equally, one can design a weighting scheme that places more emphasis on switches made after segments with longer lengths. If segments lower than a certain length are considered anomalous and are not regarded as a switch, we can design an appropriate weighting scheme by using the last switch time as an auxiliary variable. Moreover, this general framework can be used for combining only a feasible subset of strategies instead of the whole set $\mathbb{M}_T$. For example, if the optimum arm does not change for at least K rounds, one can use the length of the last segment as an auxiliary variable and

combine only the strategies with at least K length segments by preventing switches between arms when a segment does not yet reach length K. If a certain arm m cannot be the optimum arm immediately after the optimum arm is m', the weighting scheme can be designed to prevent switches from m' to m to combine only the feasible strategies.

However, achieving the minimax regret bound for the number of switches complexity cost in the adversarial bandit setting without any constraints requires no auxiliary variables, only the last selected arm m and the current round t.

In Section IV, we construct an algorithm that can achieve the minimax optimal regret bound with prior knowledge of the number of switches S and the game length T. In Section V, we remove both of these prior information requirements and introduce a truly online minimax optimal algorithm.

IV. Achieving the Minimax Optimal Regret for Known Number of Switches and Game Length

In this section, we first introduce an algorithm that achieves the minimax optimal regret ($\tilde{O}\sqrt{MTS}$) with a priori knowledge of the game length T and the number of switches S (we will remove these requirements in Section V). Since the regret we are trying to achieve is dependent only on the number of switches (our complexity cost) irrespective of the length of the segments, we design the combination weights, $\mathcal{T}(\cdot)$, in a way that its update at each round t does not depend on anything except whether a switch is made or not. Therefore, keeping track of only the last arm selection of a strategy and the current time t will suffice. Hence, this algorithm will have a computational complexity of $O(M)$ per round. We optimize the combination weight assignment according to S and T since we assume that the number of switches of the optimum strategy and the game length T are known beforehand.

We start our design by making the update $\mathcal{T}(\mathbf{s}_t|\mathbf{s}_t(1:t-1))$ only depend on the last arm selection such that

$$\mathcal{T}(\mathbf{s}_t|\mathbf{s}_t(1:t-1))=\begin{cases}\frac{\alpha}{M-1} & s_t\neq s_{t-1}\ \text{(switch)}\\ 1-\alpha & s_t=s_{t-1}\ \text{(no switch)}\end{cases}, \quad (26)$$

where α is the switching update parameter (or switch probability). This weight assignment that uses no auxiliary variables is sufficient for our problem since the complexity cost (the number of switches) is defined only by the successive arm selections. When the weight update in (26) is used, the combination weight of the optimum strategy with S segments becomes

$$\mathcal{T}(\mathbf{s}_T^*)=\frac{1}{M}\frac{1}{(M-1)^{S-1}}\alpha^{S-1}(1-\alpha)^{T-S},$$

where the initial $1/M$ comes from the uniform start between M bandit arms. Since $W(\mathbf{s}_T^*)=1/\mathcal{T}(\mathbf{s}_T^*)$,

$$\ln W(\mathbf{s}_T^*)=\ln M+(S-1)\ln\left(\frac{M-1}{\alpha}\right)-(T-S)\ln(1-\alpha). \quad (27)$$

Minimizing (27) yields the optimal value for α, which is $\alpha^*=(S-1)/(T-1)$. Next, we provide a general result which will be useful in our derivations in Section V.

Lemma 1. *Using the fixed switching update parameter, $\alpha=(S_\alpha-1)/(T-1)$ produces the following complexity cost, $\ln W(\boldsymbol{s}_T^*)$, for the optimum strategy, if $S_\alpha\leq S$,*

$$\ln W(\boldsymbol{s}_T^*)\leq \ln M+(S-1)\ln\left(\frac{T-1}{S_\alpha-1}(M-1)e\right). \quad (28)$$

Algorithm 2 Switching Network Forecaster for Known S,T (SNF.F)

Initialize constant $\alpha,\eta\in\mathbb{R}^+$
Set $C(1,m)=1/M$ for $m\in 1,...,M$
Initialize $p_{1,m}=\frac{C(1,m)}{\sum_{m=1}^{M}C(1,m)}$ for $m\in 1,...,M$
for $t=1$ **to** T **do**
  Select one of the M arms with probability $p_{t,m}$
  Receive loss l_{t,u_t}
  Set $\tilde{l}_{t,m}=\frac{l_{t,m}\mathbb{1}_{m=u_t}}{p_{t,m}}$ for $m\in 1,...,M$
  for $m=1$ **to** M **do**
    $C(t+1,m)=(1-\alpha)C(t,m)e^{-\eta\tilde{l}_{t,m}}+\frac{\alpha}{M-1}\sum_{m'\neq m}C(t,m')e^{-\eta\tilde{l}_{t,m'}}$
  end for
  for $m=1$ **to** M **do**
    $p_{t+1,m}=\frac{C(t+1,m)}{\sum_{m=1}^{M}C(t+1,m)}$
  end for
end for

where S_α is the switch parameter used in selecting α and S is the number of switches of the optimum strategy.

Proof of Lemma 1. Substituting the switching update parameter α with $\alpha=(S_\alpha-1)/(T-1)$ in (27) yields the following complexity cost for $\mathbf{s}_T^*$,

$$\ln W(\mathbf{s}_T^*)=\ln M+(S-1)\ln\left(\frac{T-1}{S_\alpha-1}(M-1)\right)+(T-S)\ln\left(\frac{T-1}{T-S_\alpha}\right).$$

We can bound the last term as

$$(T-S)\ln\left(\frac{T-1}{T-S_\alpha}\right)=(T-S)\ln\left(1+\frac{S_\alpha-1}{T-S_\alpha}\right),$$
$$=\ln\left(\left[1+\frac{S_\alpha-1}{T-S_\alpha}\right]^{T-S}\right)\leq\ln\left(\left[1+\frac{S-1}{T-S}\right]^{T-S}\right)\leq S-1,$$

when $S_\alpha\leq S$. Thus, the upper bound of $\ln W(\mathbf{s}_T^*)$ is (28). □

In Algorithm 2, we provide a version of Algorithm 1 that has no auxiliary variables in its equivalence classes and uses the combination weight update in (26). This algorithm has the following performance result.

Theorem 2. *Running Algorithm 2 with parameters $\alpha=(S_\alpha-1)/(T-1)$ and*

$$\eta=\sqrt{\frac{2}{MT}\left[\ln M+(S_\eta-1)\ln\left(\frac{T-1}{S_\alpha-1}(M-1)e\right)\right]}$$

yields the regret bound

$$\mathbf{E}[R_T]\leq\frac{2\ln M+(S+S_\eta-2)\ln\left(e(M-1)\frac{(T-1)}{(S_\alpha-1)}\right)}{\sqrt{2\ln M+(2S_\eta-2)\ln\left(e(M-1)\frac{(T-1)}{(S_\alpha-1)}\right)}}\sqrt{MT}$$

where $S_\alpha\leq S$ is the switch parameter of α and S_η is the switch parameter of the learning rate η.

Proof of Theorem 2. The result directly follows from putting (28) into (23) and substituting in η. □

Corollary 2. *If S is known beforehand (as was our assumption is this section), setting $S_\eta=S_\alpha=S$ in the parameters α and η yields the compact regret bound*

$$\mathbf{E}[R_T]\leq\sqrt{2MTS\ln(eMT/S)}.$$

We have successfully achieved $\tilde{O}(\sqrt{MTS})$ upper bound on the expected regret when S is known beforehand. Up to now, we have achieved the optimal regret performance when the number of switches of the optimum strategy, S, is known. In the next section, we introduce an algorithm that achieves the optimal regret bound without knowing S and T beforehand.

V. Achieving the Minimax Optimal Regret for Unknown Number of Switches and Game Length

Algorithm 2 in Section IV achieves the optimal regret bound $\tilde{O}(\sqrt{MTS})$ when it is given the game length T and number of switches S a priori as inputs. To remove the requirement of knowing S, we can run T copies of this algorithm where each copy is optimized for different number of switches, $S_0 = \{1, ..., T\}$.

In batch setting, it is possible to set parameters by trying different values and using the optimal one obtained from cross-validation. However, in the online setting, we do not have the opportunity to try different values and select the optimal one. Instead, we combine them in a mixture of experts framework [11]. The combination structure achieves the performance of each expert with some redundancy. Hence, we achieve the performance of the optimally set value, i.e., the performance achievable by knowing the number of switches S a priori, without knowing S beforehand. The candidate values cover all possible values of S, hence, there is no need to have any information beforehand.

We use S_0 as a common switch parameter and set both S_α and S_η to S_0 such that $S_\alpha = S_\eta = S_0$. We can then combine the outputs of these T parallel running algorithms to achieve the performance of the optimal algorithm with the optimal number of switches since one of these algorithms naturally has the best switching parameter ($S_0 = S$).

However, naively combining all these algorithms (for all possible number of switches) increases the computational complexity to linear in the game length, which would prove to be highly problematic for games of long duration. Therefore, instead of combining Algorithm 2's optimized for each possible number of switches, we combine Algorithm 2's optimized for only the number of switches that are powers of 2. As an example, for a game length of 100 rounds, instead of combining switches $S_0 \in \{1, 2, 3, 4, ..., 99, 100\}$, we combine switches $S_0 \in \{1, 2, 4, 8, 16, 32, 64\}$ so as to ensure that one of the elements in the set of S_0 is such that $S_0 \leq S \leq 2S_0 - 1$. This combination will increase the computational complexity by only $O(\log(T))$ instead of $O(T)$.

We consider each Algorithm 2 with a different S_0 as an expert, $\mathbf{e}_{S_0}$. Note that each algorithm $\mathbf{e}_{S_0}$ is itself a randomized algorithm, which provides a probability distribution at each round t. In Algorithm 3, we provide the description of the algorithm that achieves the optimal regret when S is not known, which has the following performance result.

Theorem 3. *If S is not known beforehand but T is known, then using Algorithm 3 (SNF.U) produces the regret bound*

$$\mathbf{E}[R_T] \leq O\left(\frac{3}{2}\sqrt{2MTS\ln\left(\frac{eMT}{S}\right)} + \sqrt{2MT\ln(\log T + 1)}\right). \tag{29}$$

The first part of the expected regret is similar to the expected regret we can obtain with the knowledge of S and T beforehand, there is only a multiplicative increase. The second part of the regret is caused by not knowing the number of switches S beforehand. Hence, Algorithm 3 achieves an expected regret bound of $\mathbf{E}[R_T] \leq \tilde{O}(\sqrt{MTS})$.

Algorithm 3 Switching Network Forecaster for Unknown S (SNF.U)

Initialize experts $\mathbf{e}_{S_0}$ according to SNF.F where $S_0 = 2^i$ for $i \in \{0, 1, 2, \ldots, \lfloor \log T \rfloor\}$ such that $S_0 \in \{1, 2, 4, ...\}$
Initialize expert mixture weights $\mathbf{q}$ to be uniform.
Let $\mathbf{p}_m$ be the vector of probabilities of selecting m^{th} arm by the experts and initialize all elements of $\mathbf{p}_m$ to $1/M$ for all m
Initialize $p_{1,m} = 1/M$ for all m
for $t = 1$ **to** T **do**
  Select one of the M arms with probability $p_{t,m}$
  Receive loss l_{t,u_t}
  Update $\mathbf{q}$ according to Exp4
  Feed the selection u_t and the loss l_{t,u_t} to the experts
  for $m = 1$ **to** M **do**
    Update each element of $\mathbf{p}_m$ according to Algorithm 2 (SNF.F)
  end for
  for $m = 1$ **to** M **do**
    $p_{t+1,m} = \mathbf{q}^T \mathbf{p}_m$
  end for
end for

Algorithm 4 Switching Network Forecaster for Unknown S and T (SNF.U.1)

Initialize $T_0 = 1$, $T_r = 2^{r-1}$ for $r > 0$
for $r = 0, 1, \ldots$ **do**
  Run Algorithm 3 (SNF.U) for $T = T_r$
end for

We run $\lfloor \log T \rfloor$ copies of the algorithm (where $\lfloor \cdot \rfloor$ is the floor function) each with a different switch parameter that is a power of 2 and combine them with Exp4. The total regret now has two terms. The first one is due to the regret of one of the $\lfloor \log T \rfloor$ parallel running algorithms whose common switch parameter S_0 is such that $S_0 \leq S \leq 2S_0 - 1$. The second term is the regret (redundancy) incurred from combining the parallel running algorithms. We denote the first regret term by $\mathbf{E}[R_T^S]$. Since the performance of the best algorithm will dominate, we assume that the optimal algorithm will behave similar to its individual run. We find this regret by putting $S_\alpha = S_\eta = S_0$ where $S_0 \leq S \leq 2S_0 - 1$ in Theorem 2, which yields

$$\mathbf{E}[R_T^S] \leq \frac{2\ln M + (S + S_0 - 2)\ln\left(e(M-1)\frac{(T-1)}{(S_0-1)}\right)}{\sqrt{2\ln M + (2S_0 - 2)\ln\left(e(M-1)\frac{(T-1)}{(S_0-1)}\right)}}\sqrt{MT}, \tag{30}$$

$$\leq \frac{3}{\sqrt{2}}\sqrt{MT(\ln M + (S_0 - 1)\ln(e(M-1)\frac{(T-1)}{(S_0-1)}))}, \tag{31}$$

$$\leq \frac{3}{\sqrt{2}}\sqrt{MT\left(S_0\ln\left(\frac{eMT}{S_0}\right)\right)} \leq \frac{3}{2}\sqrt{2MTS\ln\left(\frac{eMT}{S}\right)}, \tag{32}$$

where we used $S \leq 2S_0 - 1$ to get (31) and $S_0 \leq S$ to get (32). We denote the second term, i.e., the regret (redundancy) we incur for not knowing S beforehand, by R_T^R, which is

$$\mathbf{E}[R_T^R] \leq \sqrt{2MT\ln(\lfloor \log T \rfloor + 1)} \leq \sqrt{2MT\ln(\log T + 1)}. \tag{33}$$

Hence, by combining (33) and (32), we have the total regret $\mathbf{E}[R_T]$, which is in the order of their summation. The total regret is $\mathbf{E}[R_T] \leq O\left(\mathbf{E}[R_T^S] + \mathbf{E}[R_T^R]\right)$, which is given in (29), without the knowledge of S beforehand. If we also have no knowledge of the game length T, we can use the doubling-trick and run Algorithm 3 in lengths of powers of 2 and reset the algorithm after each run. We call each such run an epoch and denote this algorithm as Algorithm 4, which has the following result showing its optimality.

Theorem 4. *If S and T are not known beforehand, then Algorithm 4 (SNF.U.1) has the regret bound*

$$\mathbf{E}[R_T] \leq O\Bigg(3\sqrt{2MT(S+\log 2T)\ln(eMT/S)} + (2\sqrt{2}+2)\sqrt{MT\ln(\log 2T)}\Bigg). \tag{34}$$

In comparison to Theorem 3, not knowing the game length causes a multiplicative increase in the redundancy regret and a seemingly additive increase in the number of switches since Algorithm 4 resets each time an epoch ends, thus creating a phantom switch. Nevertheless, the redundant terms decay significantly faster. Our algorithm attains an expected regret of $\mathbf{E}[R_T] \leq \tilde{O}(\sqrt{MTS})$ and achieves the minimax regret.

We remove the requirement of knowing T beforehand by using the doubling-trick. Taking intervals of $T_0 = 1$, $T_r = 2^{r-1}$ for $r > 0$ and resetting the final algorithm after each interval's end gives an expected regret of

$$\mathbf{E}[R_T] \leq \sum_{r=0}^{n} \mathbf{E}[R_{T_r}],$$

where $n = \lceil \log T \rceil$ ($\lceil \cdot \rceil$ is the ceil function), and $T_n < T \leq 2T_n$. Let S_r denote the number of switches made in epoch 'r'. Then, applying Theorem 3 in each epoch yields the expected regret

$$\mathbf{E}[R_T] \leq \sum_{r=0}^{n} O\left(\frac{3}{2}\sqrt{2MT_rS_r\ln\left(\frac{eMT_r}{S_r}\right)} + \sqrt{2MT_r\ln(\log 2T_r)}\right).$$

The summation of the first term of each epoch's regret is maximum when S_r and T_r are proportional, therefore we replace S_r with T_rS'/T, where $S' = \sum_{r=0}^{n} S_r$, which gives

$$\mathbf{E}[R_T] \leq \sum_{r=0}^{n} O\left(\frac{3}{2}\sqrt{\frac{2MT_r^2S'}{T}\ln\left(\frac{eMT}{S'}\right)} + \sqrt{2MT_r\ln(\log 2T_r)}\right).$$

Since $T_r < T$ for all r, we substitute T_r with T inside the logarithm in the second part. Thus the bound becomes

$$\mathbf{E}[R_T] \leq \sum_{r=0}^{n} O\left(\frac{3T_r}{2}\sqrt{\frac{2MS'}{T}\ln\left(\frac{eMT}{S'}\right)} + \sqrt{2MT_r\ln(\log 2T)}\right).$$

Sum of T_r is equal to $2T_n$, which is upper bounded by $2T$, and summation of their square-roots are upper bounded by $\sqrt{2T}/(\sqrt{2}-1)$. Thus, the bound can be written as

$$\mathbf{E}[R_T] \leq O\Bigg(3\sqrt{2MTS'\ln\left(\frac{eMT}{S'}\right)} + (2\sqrt{2}+2)\sqrt{MT\ln(\log 2T)}\Bigg).$$

By definition, $S \leq S' \leq S + n \leq S + \log 2T$, and we get (34).

Since we run $O(\log T)$ instances of Algorithm 2 as subroutines and combine them, the computational complexities of Algorithms 3 and 4 are $O(M \log T)$.

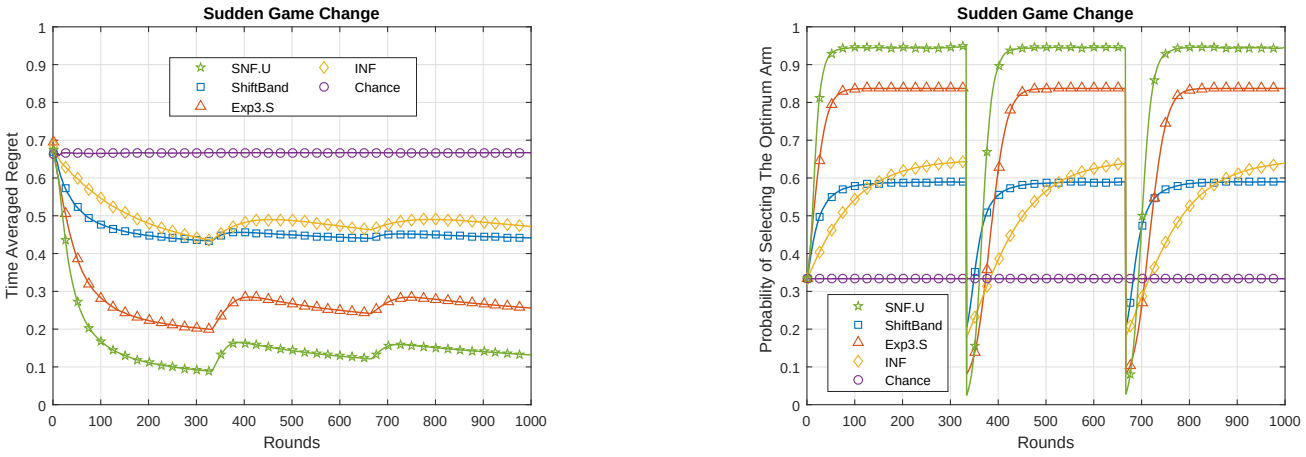
VI. Experiments

In this section, we demonstrate the performance of our algorithm both on real and synthetic data. We use two synthesized datasets and three real datasets to show how our algorithm performs individually and in comparison to state of the art techniques: ShiftBand [29], Implicitly Normalized Forecaster (INF) [31] and Exp3.S [30]. All the simulated algorithms are constructed as instructed in their original publications. We run Algorithm 3 (SNF.U) when game length T is provided and run Algorithm 4 (SNF.U.1) otherwise. We also compare each algorithm against the trivial algorithm, Chance, (i.e., random guess) for a baseline comparison.

A. Sudden Game Change

We first construct a game with 3-armed bandit where in all rounds the optimum arm has zero loss while the other arms have loss of 1. This synthesized dataset can be thought as representing a classification problem where the loss is binary, e.g zero loss is a correct classification and a loss of one is a misclassification. The parameters of the individual algorithms are set as instructed by their respective publications [29]–[31]. The information of both the game length T and the number of switches S have been given a priori to all of the algorithms since the compared algorithms are unable to compete otherwise, as we shall see in the subsequent sections.

A game of length 1000 has been created with switches at rounds 333 and 666. The optimum arms at consecutive segments are different. The loss of the arms changes abruptly at the switching time instances, e.g., the optimum arm of the first segment has a loss of 1 throughout the second segment. The same game has been presented to the algorithms and we calculate their time averaged accumulated regrets and track their probabilities of selecting the optimum arm. This process is repeated for 100 times and the ensemble averages are plotted in Fig. 2a, 2b respectively. Our algorithm significantly outperforms the other algorithms as illustrated in Fig. 2a. The switches at rounds 333 and 666 are apparent with a slight increase in the regret. In Fig. 2b, the probability of selecting the optimum arm in each round is shown. We emphasize that this figure plots the selection probability of the optimum arm for each round individually. For example, if the optimum arm is 1 up to round 333 and 2 up to round 666 and 1 again after that, then the figure plots the probability of selecting arm 1 up to round 333 and arm 2 after that and so on. This is the reason for the sudden dips in probability at each switch. Fig. 2b clearly demonstrates why our algorithm outperforms the others. Since SNF.U saturates at a higher probability, the dip in the probability is much greater. However, since SNF.U has a faster convergence, it is able to select more frequently the optimum arm than the other algorithms. Fig. 2a and 2b show that while the performances of ShiftBand and INF are similar for this dataset, their behavior is slightly different. While ShiftBand has a faster convergence, its probability saturates at a lesser value than INF. The behavior of SNF.U and Exp3.S are similar to each other. However, in terms of convergence speed and saturation level, SNF.U is significantly better than Exp3.S. SNF.U is able to achieve a better performance than the other algorithms because it reacts to the change better.



(a) (b)

Fig. 2: (a) Regret performances of the algorithms in a 3-armed bandit game with sudden game (concept) change at every 333 rounds. (b) Probabilities of selecting the optimum arm for all of the algorithms in sudden game change setting.

B. Random Game and Benchmark

In this part, we construct a game whose behavior is completely random with the only regularization condition being a single arm should be optimum throughout a segment. We start to synthesize the dataset by randomly selecting losses in [0,1] for all arms for all rounds. We predetermine the optimum arms in each segment and then switch the minimum losses with the loss of the optimum arm at each round. This synthesized dataset creates a game with randomly determined losses while maintaining that one arm is uniformly optimum throughout each segment. We synthesize multiple datasets to analyze the effects of the parameters of the game individually, where we compare the algorithms' performances for varying game length (T), number of switches (S), number of arms (M) and prior information given. We start with the control group of $T = 1000$, $M = 3$, $S = 3$ and both T and S is known a priori. Then, for each case, we vary one of the above four parameters. Differently from before, the time instances of switches are not fixed to 333 and 666 but instead selected randomly to be in anywhere in the game. Thus, we create random games with 3 arms and 3 segments. We provided the algorithms with the prior information of both game length and number of switches. We selected the game lengths to be Fibonacci numbers between 100 and 10000. In Fig. 3a, we have plotted the average regret incurred at the end of the game by all of the algorithms at different values of game length while fixing the other parameters. For any set of parameters we have simulated the setting for 10 times with recreating the game each time to get a value closer to the true mean. The algorithms ShiftBand and INF perform close to random guess up to approximately game length of 400 rounds. After that, they start to perform better than chance. SNF.U and Exp3.S however perform better than chance for all values of game length. There is a significant performance difference between SNF.U and Exp3.S, especially in games of shorter lengths.

To observe the effect of the number of switches on the performances, we created random change games with 3 arms and game length of 1000. We provided the algorithms with the prior information of both the game length and the number of switches. We selected the number of switches to be Fibonacci numbers between 1 and 1000. In Fig. 3b, we have plotted the average regret incurred at the end of the game by all of the algorithms at different values of number of switches while fixing the other parameters. For any set of parameters, we have simulated the setting for 10 times with recreating the game each time. The algorithm ShiftBand perform similar to random guess after approximately 10 switches, i.e., $S = 10$. Both INF and Exp3.S behave no better than random guess after $S = 50$, however, Exp3.S increases from a much lower value of regret. SNF.U on the other hand catches random guess at $S = 400$, even for number of switches comparable to game length, SNF.U manages to provide better performance than random guess unlike the other algorithms.

To observe the effect of the number of bandit arms on the performances, we created random change games with 3 segments and game length of 1000. We provided the algorithms with the prior information of both the game length and the number of switches. We selected the number of bandit arms to be Fibonacci numbers between 2 and 100. In Fig. 3c, we have plotted the average regret incurred at the end of the game by all of the algorithms at different values of number of bandit arms while fixing the other parameters. For any set of parameters, we have simulated the setting for 10 times with recreating the game each time. The algorithms ShiftBand and INF perform similar to random guess after approximately 10 bandit arms. Exp3.S reduces to random guess after 50 bandit arms. However, SNF.U always outperforms random guess. Even for 89 bandit arms SNF.U provides slight performance gain while the others are indistinguishable from chance. SNF.U outperforms all algorithms for all values of bandit arms uniformly.

To observe the effect of prior information on the performances, we created random change games with 3 bandit arms, 3 segments and game length of 1000. Case 1,2,3,4 represents providing both S and T, only S, only T, none of them a priori respectively. In Fig. 3d, we have plotted the average regret incurred at the end of the game by all of the algorithms at different settings of prior information while fixing the other parameters. For any set of parameters, we have simulated the setting for 10 times with recreating the game each time. ShiftBand and INF perform similar to each other in all cases. Exp3.S has better performance than them when S is known a priori. All competition performs close to

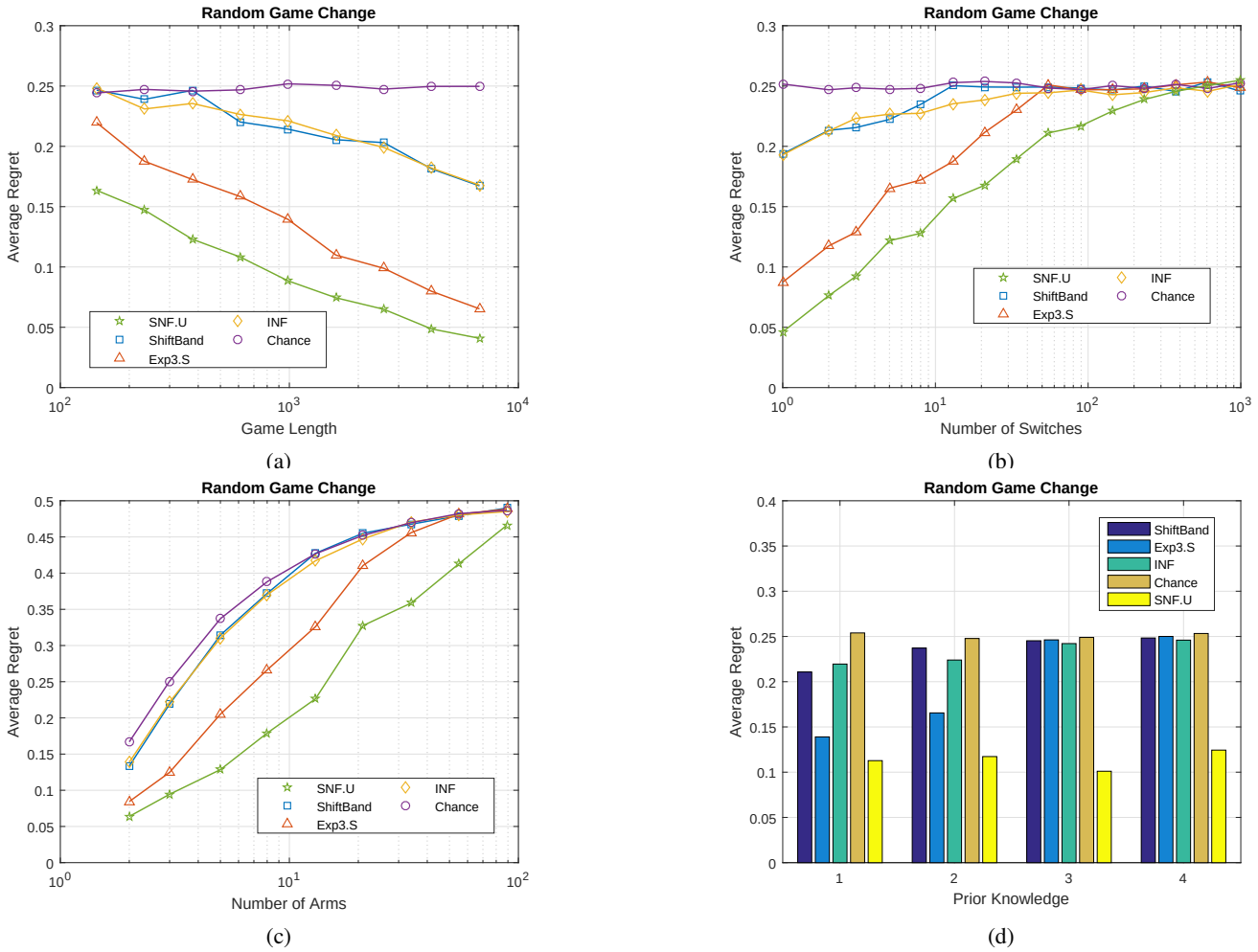


Fig. 3: (a) Per round regret performances of the algorithms with increasing game length. (b) Per round regret performances of the algorithms with increasing number of switches. (c) Per round regret performances of the algorithms with increasing number of bandit arms. (d) Per round regret performances of the algorithms according to the prior information given.

random guess when S is not known, since $S_{\max}$ is set to be T as per their rules. What we observe here tells us that the other algorithms are simply not cut out for settings when number of switches are not known. In [29], the author provides a version for Exp3.S that uses minimum possible number of switches, $S = 1$, to initialize the parameters, which in turn provides guaranteed regret linear with the number of switches instead of square-root dependence, as was in the variant of Exp3.S when parameters are set according to $S_{\max}$. However, such an initialization gives guaranteed lower regret only if we can be sure that $S \leq \sqrt{T}$ (which is not the case). Therefore, that kind of initialization is meaningless when no information regarding S is known since setting the parameters according to $S_{\max} = T$ has lower guaranteed regret. The prior information given does not effect SNF.U drastically since SNF.U is a universal algorithm. The knowledge of T has a bit of an effect on its performance because then it does not have to reset the algorithm in every epoch. SNF.U outperforms all algorithms, especially when the number of switches are not known.

C. Real Data Benchmark

We use a real-world networking dataset that corresponds to the retrieval latencies of more than 700 universities homepages. The pages have been probed every 10mins for more than one week in May 2004 from an internet connection located in New York, NY, USA. The data includes 760 URLs and 1361 latencies per URL. The initial purpose of this dataset was to be used as a benchmarking dataset for the multi-armed bandit problem [37]. An agent must retrieve data through a network with several redundant sources available. For each retrieval, the agent selects one source and waits until the data is retrieved. The objective of the agent is to minimize the sum of the delays for the successive retrievals. Vermorel and Mohri have used the homepages of 760 universities. The home pages have been retrieved roughly every 10 min for about 10 days (1361 rounds), where the retrieval latencies are in milliseconds. Intuitively, each page is associated to a bandit arm, and each latency to a loss [37].

Using the universities as the bandit arms and 1361 latencies as the losses at each round, we have extracted games with 2 bandit arms and at most 3 segments 100 times. For each game, we have normalized the losses into $[0, 1]$ and plotted the time-averaged regret averaged over 100 trials in Fig. 4a, where we have given only the knowledge of S a priori. Similar to all of the tests before, ShiftBand and INF perform the worst. Exp3.S performs better than the other two but SNF.U outperforms all of them. However, even chance has a time averaged regret of 0.03, which implies that the dataset is too diverge and

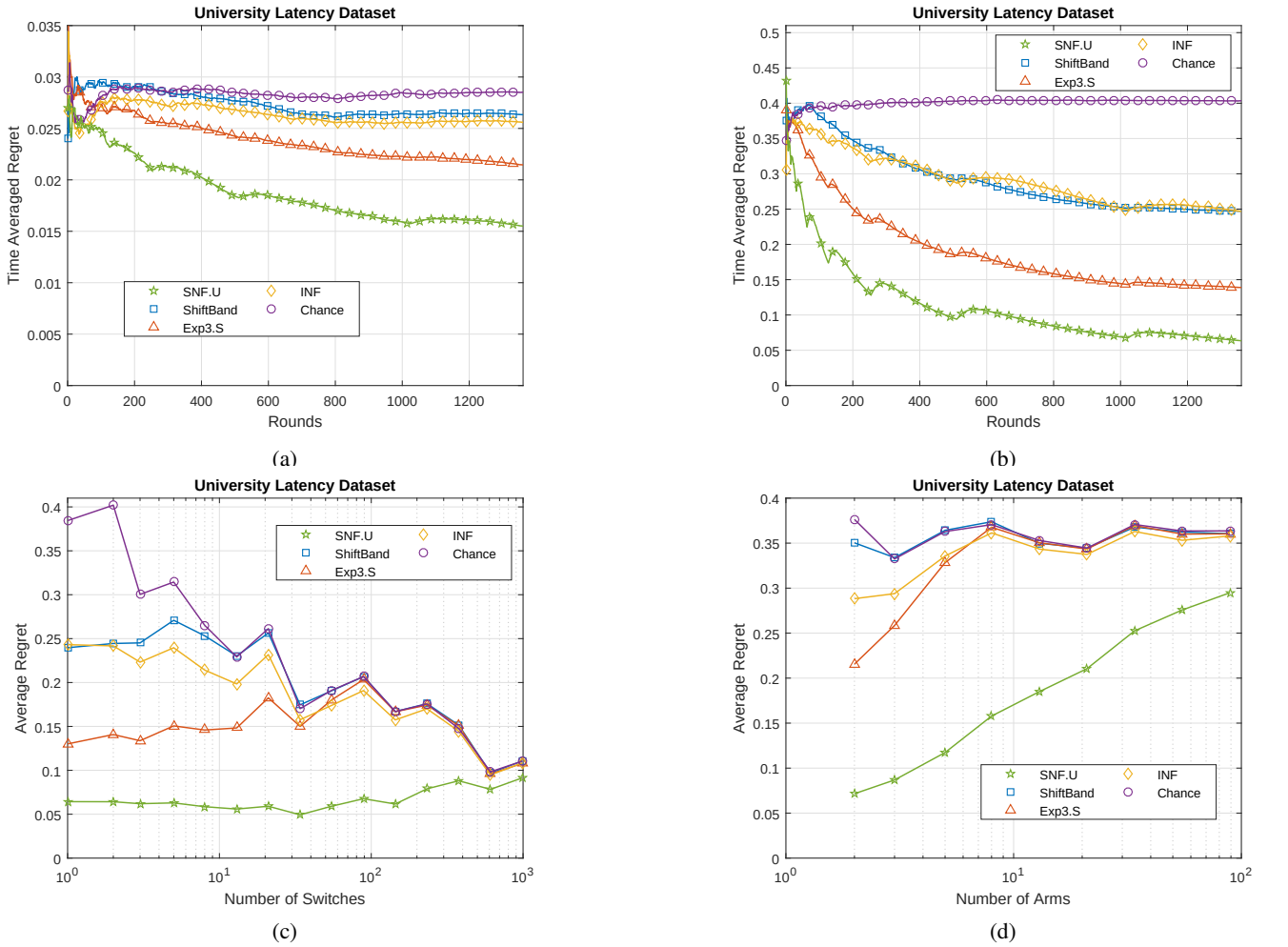


Fig. 4: (a) Time averaged regret performances in "univ-latencies" dataset when $M = 2$ and $S_{\max} = 3$ (b) Time averaged regret performances in "univ-latencies" dataset with time-out when $M = 2$ and $S_{\max} = 3$ (c) Per round regret performances of the algorithms with increasing number of switches. (d) Per round regret performances of the algorithms with increasing number of bandit arms.

sparse. Because of some very high latencies, normalization considerably diminishes most values in the dataset. Therefore, we have truncated the latencies at 1000 ms, which can be thought of as timeout for a more realistic real-world setting. Re-performing the same experiments after truncation gives the results in Fig. 4b, which provides a clearer comparison.

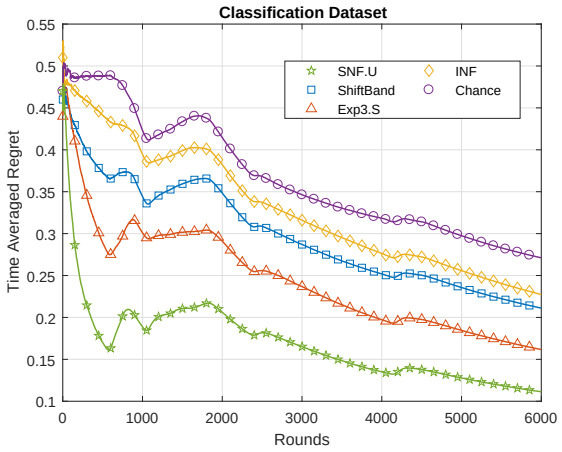
Using the truncated data, we do benchmarks on the number of switches and the number of arms. For the number of switches benchmark, we set the number of bandit arms to 2 and set $S_{\max}$ to fibonacci numbers between 1 and 1000. For each value of $S_{\max}$, we extract games with S between the current $S_{\max}$ and previous $S_{\max}$ so as to not use the same game for different values of $S_{\max}$. Interestingly, as $S_{\max}$ increases the regret of random guess decreases, which can only mean that high number of switches occur in the dataset when the latency values are closer together. All of the other algorithms perform similar to chance after $\approx$ 30 switches, while our algorithm performs similar after ≈ 600 switches.

For the number of bandit arms, we set $S_{\max}$ to 20, since this is a real-world dataset it is not always possible to get high-armed games with low number of switches. We set M to fibonacci numbers between 2 and 100. Interestingly, as M increases the regret of random guess approximately stays the same. All of the other algorithms perform similar to chance after $M = 10$, while our algorithm always outperforms chance when $M \leq 100$. Its apparent in all cases that INF performs not really well because by design only an additive bias term is applied to each arm's potential function. ShiftBand and Exp3.S are similar in spirit with their pooling of the arm selection probabilities. However, Exp3.S performs significantly better than ShiftBand. SNF.U considerably outperforms INF, ShiftBand, Exp3.S since it creates a probability sharing network between all of the arms. The power of SNF.U is especially clear in high number of bandit arms and switches.

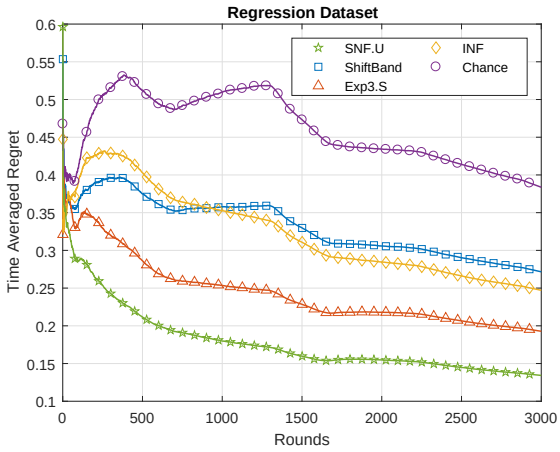
The behaviors of the algorithms in synthesized and real dataset benchmarks may seem quite different at first glance. However, considering the performance of the Chance algorithm as a baseline and observing the performance increases of the algorithms in each individual case shows us that the synthetic and real experiments significantly agree with each other.

D. Classification Task

For the classification experiment, we have used the "Occupancy Detection Dataset" [38], which contains a binary classification task for detecting the room occupancy based on the room temperature, relative humidity, light, CO2 and humidity ratio. The dataset includes 20560 data samples. In



(a) Classification



(b) Regression

Fig. 5: (a) Regret performances of the algorithms in the "Occupancy Detection Dataset" (b) Regret performances of the algorithms in the "Combined Cycle Power Plant Dataset" .

each trial of the experiment, we have randomly split the dataset into 6000 test set and 14560 training set. The feature vector in the dataset is 5-dimensional. We have split the training data according to the orthant each sample belongs to. Hence, we have $2^5 = 32$ disjoint training sets. Using each training set, we train a perceptron with the learning rate $\sqrt{1/n}$ for the n^{th} sample. We feed the training set 10 times to the perceptrons. Each such perceptron has been trained on a different part of the training set. We treat the perceptrons as the arms of a multi-armed bandit and run the bandit algorithms to select the correct model. For the test phase, we split the test dataset according to their orthants. First, we sequentially feed the samples belonging to the first orthant, then we feed the second orthant and so on until we have fed all of the test data (i.e., the orthants are fed in order). Thus, if the dimension of the feature vector is D (in this case 5), then the number of bandit arms is $M = 2^D$ and the number of switches $S = M = 2^D$. The experiment has been done for 100 trials. In Figure 5a, we have illustrated the performances of the competing bandit algorithm. Similar to the rest of the experiments, our algorithm SNF.U has outperformed all of the other algorithms. As can be seen in Figure 5a, the convergence rate of SNF.U is much higher in comparison to the other algorithms, which is most clear up to the approximately 600^{th} round. Our algorithm is much more robust to orthant switches. The algorithms, ShiftBand and INF again perform the worst and has very close performance to chance. Exp3.S performs the best except our algorithm but even that falls behind considerably. Nonetheless, none of the algorithms' performance come close to our algorithm SNF.U.

E. Regression Task

For the regression experiment, we have used the "Combined Cycle Power Plant Dataset" [39], which contains 9568 data points collected from a Combined Cycle Power Plant over 6 years (2006-2011), when the plant was set to work with full load. Features of the dataset are the hourly average ambient variables: temperature, ambient pressure, relative humidity and exhaust vacuum. The goal is to predict the net hourly electrical energy output of the plant. In each trial of the experiment, we have randomly split the dataset into 3000 test set and 6568 training set. The feature vector in the dataset is 4-dimensional. Similarly to the classification task, we have split the training data according to the orthant each sample belongs to. Hence, we have $2^4 = 16$ disjoint training sets. Using each training set, we train a linear predictor with the learning rate $0.1\sqrt{1/n}$ for the n^{th} sample. We feed the training set 10 times to the linear predictors. Each such predictor has been trained on a different part of the training set. We again treat the predictors as the arms of a multi-armed bandit and run the bandit algorithms to select the correct model. For the test phase, we split the test dataset according to their orthants. First, we sequentially feed the samples belonging to the first orthant, then we feed the second orthant and so on until we have fed all of the test data (i.e., the orthants are fed in order). Thus, if the dimension of the feature vector is D (in this case 4), then the number of bandit arms is $M = 2^D$ and the number of switches $S = M = 2^D$. In Figure 5b, we have illustrated the performances of the competing bandit algorithms. Similar to the rest of the experiments, our algorithm SNF.U has outperformed all of the other algorithms. Our algorithm uniformly outperforms the others and is nearly nonresponsive to the changes and holds down its regret. The algorithms, ShiftBand and INF again perform the worst and have the closest performance to chance. They switch orders at the 900^{th} round and perform quite similar. Exp3.S performs the best except our algorithm but even that falls behind considerably. All in all, SNF.U. has performed better than the other bandit algorithms for varying real-world datasets (both classification and regression).

VII. Concluding Remarks

We studied an important problem in the field of reinforcement learning, which is the adversarial multi-armed bandit problem, and introduced, as the first time in the literature, a truly online, low complexity (log-linear) algorithm whose performance with respect to the best bandit arm selection strategy is minimax optimal (up to logarithmic factors). We emphasize that we achieve this minimax optimal regret without any knowledge about the best arm selection strategy (e.g., its number of segments S, the lengths of these segments and the location of these segments) and the game length T. The results we provide are uniformly guaranteed to hold in an individual sequence manner for all possible arm loss sequences since we

refrain from making any statistical assumptions on the bandit arms. We achieved these results by first introducing a general efficiently implementable bandit arm selection framework that works with any kind of weighting scheme for various applications. Then, based on this framework, we designed an online probability sharing network that sequentially combines the selections of all possible bandit arm selection strategies with carefully constructed weights, where the number of possible strategies grow with M^T. We efficiently implement this network by creating equivalence classes, which group certain strategies together to store and update their weights collectively. Then, by combining the beliefs of different carefully designed probability sharing networks with its parameters (transition weights and learning rates) carefully tuned to different number of switches, we achieve the minimax optimal regret up to logarithmic factors with computational complexity only log-linear in game length T. Thus, our algorithm can be efficiently used in applications involving big data. Through extensive set of experiments involving synthetic and real data, we demonstrated significant performance gains achieved by our algorithm with respect to the state-of-the-art adversarial multi-armed bandit algorithms in the reinforcement learning and computational learning theory literature [29]–[31].

REFERENCES

[1] R. S. Sutton and A. G. Barto, *Reinforcement learning: An introduction*. MIT press Cambridge, 1998, vol. 1, no. 1.

[2] J. Moody and M. Saffell, "Learning to trade via direct reinforcement," *IEEE Transactions on Neural Networks*, vol. 12, no. 4, pp. 875–889, Jul 2001.

[3] R. Song, F. L. Lewis, and Q. Wei, "Off-policy integral reinforcement learning method to solve nonlinear continuous-time multiplayer nonzero-sum games," *IEEE Transactions on Neural Networks and Learning Systems*, vol. PP, no. 99, pp. 1–10, 2016.

[4] H. S. Chang, J. Hu, M. C. Fu, and S. I. Marcus, "Adaptive adversarial multi-armed bandit approach to two-person zero-sum markov games," *IEEE Transactions on Automatic Control*, vol. 55, no. 2, pp. 463–468, Feb 2010.

[5] H. Ozkan, M. A. Donmez, S. Tunc, and S. S. Kozat, "A deterministic analysis of an online convex mixture of experts algorithm," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 26, no. 7, pp. 1575–1580, July 2015.

[6] H. R. Berenji and P. Khedkar, "Learning and tuning fuzzy logic controllers through reinforcements," *IEEE Transactions on Neural Networks*, vol. 3, no. 5, pp. 724–740, Sep 1992.

[7] N. D. Vanli, M. O. Sayin, I. Delibalta, and S. S. Kozat, "Sequential nonlinear learning for distributed multiagent systems via extreme learning machines," *IEEE Transactions on Neural Networks and Learning Systems*, vol. PP, no. 99, pp. 1–13, 2016.

[8] P. Auer, N. Cesa-Bianchi, Y. Freund, and R. E. Schapire, "Gambling in a rigged casino: The adversarial multi-armed bandit problem," in *Foundations of Computer Science, 1995. Proceedings., 36th Annual Symposium on*, Oct 1995, pp. 322–331.

[9] A. J. Bean and A. C. Singer, "Universal switching and side information portfolios under transaction costs using factor graphs," *IEEE Journal of Selected Topics in Signal Processing*, vol. 6, no. 4, pp. 351–365, Aug 2012.

[10] T. Moon, "Universal switching fir filtering," *IEEE Transactions on Signal Processing*, vol. 60, no. 3, pp. 1460–1464, March 2012.

[11] A. C. Singer and M. Feder, "Universal linear prediction by model order weighting," *IEEE Transactions on Signal Processing*, vol. 47, no. 10, pp. 2685–2699, Oct 1999.

[12] T. Moon and T. Weissman, "Universal fir mmse filtering," *IEEE Transactions on Signal Processing*, vol. 57, no. 3, pp. 1068–1083, March 2009.

[13] A. C. Singer and M. Feder, "Universal linear least-squares prediction," in *Information Theory, 2000. Proceedings. IEEE International Symposium on*, 2000, pp. 81–81.

[14] C. Tekin, S. Zhang, and M. van der Schaar, "Distributed online learning in social recommender systems," *IEEE Journal of Selected Topics in Signal Processing*, vol. 8, no. 4, pp. 638–652, 2014.

[15] C. Tekin and M. van der Schaar, "Releaf: An algorithm for learning and exploiting relevance," *IEEE Journal of Selected Topics in Signal Processing*, vol. 9, no. 4, pp. 716–727, 2015.

[16] M. R. W. Dawson, B. Dupuis, M. L. Spetch, and D. M. Kelly, "Simple artificial neural networks that match probability and exploit and explore when confronting a multiarmed bandit," *IEEE Transactions on Neural Networks*, vol. 20, no. 8, pp. 1368–1371, Aug 2009.

[17] R. Zheng and C. Hua, *Adversarial Multi-armed Bandit*. Cham: Springer International Publishing, 2016, pp. 41–57. [Online]. Available: https://doi.org/10.1007/978-3-319-50502-2˙4

[18] A. Heydari and S. N. Balakrishnan, "Optimal switching and control of nonlinear switching systems using approximate dynamic programming," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 25, no. 6, pp. 1106–1117, June 2014.

[19] X. Liu, H. Su, and M. Z. Q. Chen, "A switching approach to designing finite-time synchronization controllers of coupled neural networks," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 27, no. 2, pp. 471–482, Feb 2016.

[20] P. Auer and M. K. Warmuth, "Tracking the best disjunction," *Machine Learning*, vol. 32, no. 2, pp. 127–150, 1998.

[21] M. Herbster and M. K. Warmuth, "Tracking the best expert," *Machine Learning*, vol. 32, no. 2, pp. 151–178, 1998.

[22] P. Lim, C. K. Goh, K. C. Tan, and P. Dutta, "Multimodal degradation prognostics based on switching kalman filter ensemble," *IEEE Transactions on Neural Networks and Learning Systems*, vol. PP, no. 99, pp. 1–13, 2016.

[23] A. Heydari, "Feedback solution to optimal switching problems with switching cost," *IEEE Transactions on Neural Networks and Learning Systems*, vol. PP, no. 99, pp. 1–1, 2015.

[24] F. M. J. Willems, "Coding for a binary independent piecewise-identically-distributed source," *IEEE Transactions on Information Theory*, vol. 42, no. 6, pp. 2210–2217, 1996.

[25] N. Merhav, "On the minimum description length principle for sources with piecewise constant parameters," *IEEE Transactions on Information Theory*, vol. 39, no. 6, pp. 1962–1967, Nov 1993.

[26] G. I. Shamir and N. Merhav, "Low-complexity sequential lossless coding for piecewise-stationary memoryless sources," *IEEE Transactions on Information Theory*, vol. 45, no. 5, pp. 1498–1519, Jul 1999.

[27] X. Liu, J. Lam, W. Yu, and G. Chen, "Finite-time consensus of multiagent systems with a switching protocol," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 27, no. 4, pp. 853–862, April 2016.

[28] V. Vovk, "Derandomizing stochastic prediction strategies," *Machine Learning*, vol. 35, no. 3, pp. 247–282, 1999.

[29] P. Auer, "Using confidence bounds for exploitation-exploration trade-offs," *J. Mach. Learn. Res.*, vol. 3, pp. 397–422, Mar. 2003.

[30] P. Auer, N. Cesa-Bianchi, Y. Freund, and R. E. Schapire, "The non-stochastic multiarmed bandit problem," *SIAM J. Comput.*, vol. 32, no. 1, pp. 48–77, Jan. 2003.

[31] J.-Y. Audibert and S. Bubeck, "Regret bounds and minimax policies under partial monitoring," *J. Mach. Learn. Res.*, vol. 11, pp. 2785–2836, Dec. 2010.

[32] S. Bubeck and N. Cesa-Bianchi, "Regret analysis of stochastic and nonstochastic multi-armed bandit problems," *Foundations and Trends in Machine Learning*, vol. 5, no. 1, pp. 1–122, 2012.

[33] N. Cesa-Bianchi, Y. Freund, D. Haussler, D. P. Helmbold, R. E. Schapire, and M. K. Warmuth, "How to use expert advice," *J. ACM*, vol. 44, no. 3, pp. 427–485, May 1997.

[34] H. Akaike, "A new look at the statistical model identification," *IEEE Transactions on Automatic Control*, vol. 19, no. 6, pp. 716–723, Dec 1974.

[35] J. Rissanen, "Modeling by shortest data description," *Automatica*, vol. 14, no. 5, pp. 465–471, 1978.

[36] N. Merhav and M. Feder, "Universal prediction," *IEEE Transactions on Information Theory*, vol. 44, no. 6, pp. 2124–2147, 1998.

[37] J. Vermorel and M. Mohri, "Multi-armed bandit algorithms and empirical evaluation," in *In European Conference on Machine Learning*. Springer, 2005, pp. 437–448.

[38] L. M. Candanedo and V. Feldheim, "Accurate occupancy detection of an office room from light, temperature, humidity and co 2 measurements using statistical learning models," *Energy and Buildings*, vol. 112, pp. 28–39, 2016.

[39] P. Tüfekci, "Prediction of full load electrical power output of a base load operated combined cycle power plant using machine learning methods," *International Journal of Electrical Power & Energy Systems*, vol. 60, pp. 126–140, 2014.