Quantifying the Merits of Network-Assist Online Learning in Optimizing Network Protocols

Xiangxiang Dai[†1], Zhiyong Wang[†1], Jiancheng Ye[*2], John C.S. Lui[1]
[1]The Chinese University of Hong Kong, [2]The University of Hong Kong
Email: {xxdai23, zywang21, cslui}@cse.cuhk.edu.hk, jcye@connect.hku.hk

Abstract—**Optimizing network protocols is crucial for improving application performance. Recent research works use multi-armed bandit (MAB) online learning methods to address network optimization problems, aiming to improve cumulative payoffs such as network throughput. However, existing MAB frameworks are ineffective since they inherently assume the network environment is static, or they have high complexity in detecting environmental changes. In this work, we advocate using lightweight "network-assist" techniques together with online learning to optimize network protocols, and show it can effectively detect environmental changes and maximize network performance. Furthermore, optimizing network protocols often face two types of decision (or arm) spaces: discrete and continuous choices, while most prior MAB models only handle discrete settings. This paper proposes a framework capable of managing both spaces. To our best knowledge, we are the first to develop an MAB framework that incorporates network-assist signals in handling dynamic environments, while considering the distinct characteristics of discrete and continuous arm spaces. Our framework can achieve optimality by showing its sub-linear regret bound, matching the state-of-the-art results in several degenerate cases. We also illustrate how to apply our framework to two network applications: (1) wireless network channel selection, and (2) rate-based TCP congestion control. We demonstrate the merits of our algorithms via both numerical simulations and packet-level experiments.**

I. Introduction

As network applications continue to proliferate, optimizing them to deliver the best possible performance is essential. Achieving optimal performance for network applications requires developing a policy that can make astute decisions, like resource allocation, and crowdsourcing [1], [2]. However, it is difficult to know the proper operating parameters of these applications/applications beforehand, e.g., in mobile edge computing, the optimal lightweight services providing good services for users are not fixed, but vary over time [3].

To explore the right operating parameters of these network applications without any *prior* knowledge, researchers consider online learning approaches where the learning agent estimates the appropriate operating parameters based on the perceived network performance. Under such context, the multi-armed bandit (MAB) is a dominant online learning method that has been adopted for many network optimization problems [1]. The basic MAB setting is as follows. A set of arms (or decisions) represent different choices for operating parameter values. If an arm is selected, it returns a payoff from an unknown distribution to the agent, who then attempts to make a decision based on historically observed payoffs. Despite the lack of prior knowledge on the operating parameters, the MAB framework can explore the appropriate operating parameters of an application and use such information to make better decisions. In recent years, researchers have used this online learning framework to optimize different network applications, i.e., authors in [4] propose an MAB approach with guarantee and apply it in network applications. The authors of [5] develop a general discounted UCB framework to determine the initial window size in TCP congestion control. Authors in [6] design a linear MAB-based algorithm to guide content caching of mobile edges. In [2], the authors develop a general combinatorial multi-armed bandit that addresses channel selection among other related problems.

However, the basic MAB model assumes a *static environment*, e.g., rewards come from a stationary distribution, thus ignoring the constantly changing network conditions. For instance, in a cognitive radio network, the channel throughput is unavoidably influenced by the arrival and departure of users. Therefore, policies designed for stationary environments may not be suitable as condition changes, leading to degraded application performance. Only few MAB works consider state changes in a dynamic environment. Authors in [7] rely on observations of specific metrics to detect state changes, such as arm payoff and its deviation. [8] performs an offline mapping from states to decisions. However, in a time-varying network environment, these self-detection or offline mechanisms can be inefficient without considering the specific characteristics of the network scenarios. Moreover, these methods explore the entire arm space from scratch when detecting a state change. This slows down the convergence of the MAB algorithms.

In contrast, we aim to develop a general learning method that incorporates *"network-assist"* signals for promptly detecting state changes and expediting the learning process. Network-assist implies *transmitting low-overhead signals in the network "control-plane", without requiring heavy computation or architectural support from network elements*. Another requirement of our work is that these network-assist signals can be easily implemented on existing network protocols. Note that network-assist signals alone exist in various network applications, e.g., viewer-assisted transcoding in crowdsourced transcoding, interference alignment in wireless networks, and explicit congestion notification (ECN) in network congestion control [9], [10]. A more specific example is the active queue

[†]Equal contribution. [*]Jiancheng Ye is the corresponding author.

management (AQM) policies such as Random Early Detection (RED), which drops packets when the queue length exceeds a predefined threshold. TCP clients respond to these signals by adjusting their transmission rates or window sizes, based on the absence of acknowledgments for the dropped packets. Despite the benefits of network-assist signals in various network applications, *their potential has not been explored within the context of online learning and protocol optimizations*.

Furthermore, many network protocols, such as choosing the sending rate of TCP, involve a continuous arm space rather than a discrete arm set. This poses new challenges for previous MAB works which only deal with discrete arm sets. Facing infinite arm set and how to make the online learning protocols operate within memory constraints is very challenging [11]. These memory constraints hinder the agent's capability to gain knowledge about the infinite arm space since oversampling non-optimal arms can lead to subpar performance, while infinite arms in the continuous arm space necessitate more samples to guarantee sufficient exploration [12]. These challenges make it difficult for previous MAB works to be applicable for general network applications.

Facing the challenges above, we propose a new *non-stationary* online learning framework under a time-varying environment, and formulate an MAB model for both discrete and continuous arm spaces. Specifically, the system environment is dynamic during the learning process, i.e., the state may abruptly change. The information on available arm subspace will be revealed at the beginning of each new state utilizing network-assist signals, which the agent can receive after some *signal delay*. Our contributions are as follows:

1) We introduce a new mathematical model that elucidates the advantages of network-assist online learning within the context of dynamic, non-stationary MAB environments, incorporating network-assist signals to better capture the evolving nature of these scenarios.
2) To address both discrete and continuous arm spaces, we develop two online learning algorithms for the dynamic state change model: DNS-UCB for handling the discrete arm space, and CNS-UCB for optimizing granularity in the continuous arm space.
3) We provide theoretical analysis to validate the efficiency of both algorithms, which match the state-of-the-art results in some specialized or simpler scenarios, and highlight their advantages over other leading algorithms in both discrete and continuous arm spaces.
4) We apply our proposed algorithms to two network applications covering two types of arm spaces: channel selection in cognitive radio networks and rate-based TCP congestion control. In addition to our numerical simulations, we also conduct packet-level ns-3 simulations to validate the advantages of our approach.

II. SYSTEM MODEL AND PROBLEM FORMULATION

In this section, we introduce our online learning framework under a "*non-stationary*" MAB setting with time-varying state changes, taking into account two distinct types of arm spaces.

Firstly, in the MAB literature, arm spaces are categorized into two types: discrete or continuous arm space. A countable number of distinct arms are available for selection in the discrete arm space. In contrast, the continuous arm space offers infinite arms that can take on any value within a continuous range. One example of network applications in the discrete arm space is mobile crowdsensing, where the task organizer has to decide which participants to select to enhance the quality of crowd-sensed data [4]. An instance of network applications with continuous arm space is network traffic engineering, where the agent needs to decide the resource allocation, such as bandwidth and link capacity. We respectively denote the discrete arm set as $\mathcal{A}$, with cardinality $|\mathcal{A}|$, and the continuous arm space as $\mathcal{X} \triangleq [m, n]$, $m, n \in \mathbb{R}$, where m (n) is the lower (upper) bound on the feasible decision range.

We focus on dynamic environment scenarios with time-varying states. For instance, in an opportunistic multichannel access network, the system's state depends on the availability of channels, which is influenced by the movement of primary users (PUs) [13]. Based on coordination routines to judge the usage of current channels, the available channels from all existing channels can be sensed by the secondary users (SUs), but without any knowledge of the channel throughput [14]. Another example is the explicit signal like ECN, which can reveal the current network congestion level. Accordingly, ECN values can represent different states and guide the sender to adjust its sending rate [15]. The specific assistance information from the network for the above examples is referred to as "*network-assist*" signals, which enables an agent to find pertinent arms in an appropriate subset. Formally, let $\mathcal{S}$ represents the set of all states throughout the learning process, with cardinality $|\mathcal{S}|$. The set $\mathcal{S}$ is not known in advance but expanded based on observed states. A network-assist signal is a function $\mathcal{N} : \mathcal{S} \times \mathcal{T} \to 2^{\mathcal{A}} \cup \mathcal{X}$ that maps each state $s \in \mathcal{S}$ at time $t \in \mathcal{T}$ to a subset of arms $\mathcal{A}_s \subseteq \mathcal{A}$ in the discrete arm space or a corresponding subspace $\mathcal{X}_s \subseteq \mathcal{X}$ ($\mathcal{X}_s \triangleq [m_s, n_s]$, with m_s and n_s satisfying $m_s, n_s \in [m, n]$) in the continuous arm space. This mapping is informed by real-time network conditions and is designed to direct the agent's attention to a more pertinent set of choices that are likely to yield higher payoffs under the current network state. The network-assist signal serves two primary roles: (1) **State Reflection**: Each state $s \in \mathcal{S}$, obtained through network assist, reflects the current condition of the environment; (2) **Arm Guidance**: The refined arm subset $\mathcal{A}_s$ or subspace $\mathcal{X}_s$ are customized for different states, typically including arms that are expected to offer higher payoffs. To enhance readability, we use $\mathcal{A}$ to represent both arm spaces in the rest of this section, with continuous space discussed later in Section III.

Next, we use an example in Fig. 1 to illustrate a channel selection problem and describe the time-varying state evolution process. The entire learning process is modeled as a slotted system, where time is divided into a sequence of slots $\mathcal{T} = \{1, 2, 3, \ldots, T\}$. At a certain time slot, the system may make a transition from a state s to another state s'

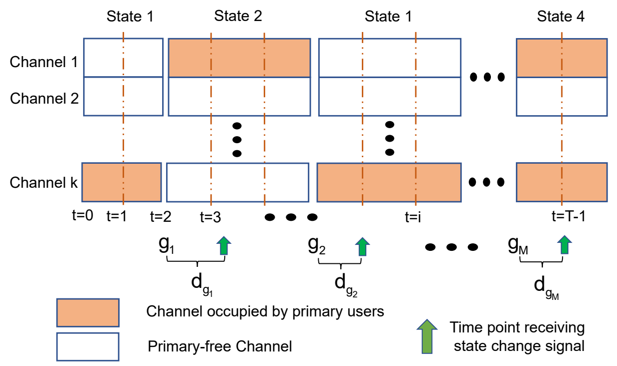


Fig. 1: State evolution process in a wireless channel selection problem, which involves multiple wireless channels with varying unknown channel throughput under different system states at change point g_i. An agent, responsible for selecting the channel with the highest throughput, experiences delays d_{g_i} in knowing state changes.

($s, s' \in \mathcal{S}$). The state at each time slot $t \in \mathcal{T}$ is denoted as s_t, $\forall s_t \in \mathcal{S}$. The total number of state changes during the learning process, denoted as M, is unknown beforehand and defined as $M = \sum_{t=1}^{T-1} \mathbb{I}\{s_t \neq s_{t+1}, \forall s_t, s_{t+1} \in \mathcal{S}\}$, where $\mathbb{I}\{\cdot\}$ is the indicator function of the event argument. Let $[z] \triangleq \{1, \ldots, z\}$ for any $z \in \mathbb{Z}^+$. We denote g_i as the change-point for each state change, where $i \in [M]$ and $g_i \in \mathcal{T}$. We also consider the potential delay for receiving the signal of a state change, where the delay of the state change-point g_i is denoted as d_{g_i}, and $(g_i + d_{g_i}) \in \mathcal{T}$ for all $i \in [M]$. Given that the delay of network-assist signals is significantly less than the state change periods [13], [16], we assume that the maximum delay C in measurements is smaller than the time interval until the next change-point, expressed formally as:

$$C < \min_{i \in [M-1]} \{g_{i+1} - g_i\}. \tag{1}$$

Under a specific state $s_t \in \mathcal{S}$ at time slot t, each arm $a \in \mathcal{A}_{s_t}$ is associated with a prior unknown payoff distribution, which is denoted as $\mathcal{D}_a^{s_t}$ and assumed to be stationary. This implies that the payoff distributions for the same arm can vary across different states. For instance, a higher transmission rate can lead to a higher utility value during non-congested states, but a lower utility value during congested states [17]. After an arm $a_t \in \mathcal{A}_{s_t}$ is selected, an immediate payoff $u_{s_t,t}(a_t)$ is observed, which is an independently and identically distributed sample from the distribution $\mathcal{D}_a^{s_t}$. Without loss of generality, the payoff is normalized to the range $[0, 1]$ for all $t \in \mathcal{T}$. In the scenario where the arm space is continuous, we assume that the payoff function $u_{s,t}(\cdot)$ satisfies the *uniformly local Lipschitz* condition [11] for all $a, a' \in \mathcal{A}$ with $\|a - a'\|_2 \leq \delta$:

$$|u_{s,t}(a) - u_{s,t}(a')| \leq L \|a - a'\|_2^{\beta}, \tag{2}$$

where $0 < \beta \leq 1$ and $L > 0$ is *Lipschitz constant*. Eq. (2) implies that similar arms yield similar payoffs, applicable to diverse network applications like congestion control [18].

During the whole learning process, we aim to maximize the expected total payoffs $\mathbb{E}\left[\sum_{s \in \mathcal{S}} \sum_{t \in \mathcal{T}: s_t = s} u_{s_t}(a_t)\right]$ under all states. However, for all $s \in \mathcal{S}$, the expected payoff function $\bar{u}_s(a)$ from the distribution $\mathcal{D}_a^s$ is not known to the agent beforehand. Consequently, the agent must balance the exploitation of their current estimate of mean payoffs with the exploration of unknown information to obtain more accurate estimates of these mean payoffs. To accomplish this goal, our objective is to devise algorithms for two types of arm spaces that can learn a policy denoted as $\boldsymbol{p} = \{p_1, p_2, \ldots, p_{|\mathcal{S}|}\}$ based on observed states and payoffs. The policy undergoes continuous refinement and adaptation by minimizing the *regret* between the expected cumulative payoff obtained from an optimal policy and the performance of the proposed algorithms. The regret for the policy $\boldsymbol{p}$ is defined as follows:

$$R_{\boldsymbol{p}}(T) = \mathbb{E}\left[\sum_{s \in \mathcal{S}} \sum_{\substack{t \in \mathcal{T}: \\ s_t = s}} \left(u_{s_t,t}(a_{s_t}^{\star}) - u_{s_t,t}(a_t^{p_{s_t}})\right)\right], \tag{3}$$

where $a_{s_t}^{\star} \in \arg\max \mathcal{A}_{s_t}$ denotes the optimal arm with the highest expected payoff in the corresponding subset of arm $\mathcal{A}_{s_t} \subseteq \mathcal{A}$ under the given state s_t.

III. ALGORITHM DESIGN

We propose two algorithms, one for discrete and another for continuous arm spaces, respectively. For both algorithms, the agent can freely choose any subroutine of the finite-armed bandit algorithms, such as Upper Confidence Bound (UCB) [19], Thompson Sampling (TS) [20] (or their variants) and other methods. Here, a UCB-based method is used as an illustration only. Note that the two proposed algorithms are specific designs that leverage signals from network assistance. The insight behind *"network-assist"* is crucial as it reduces the scope of the search and accelerates the learning speed for finding the optimal arm under the dynamic environment.

A. Discrete Arm Space Algorithm (DNS-UCB)

Algorithm 1 DNS-UCB

```
1: Initialize $\mathcal{S}_0 = \emptyset$, $\mathcal{A}_s = \mathcal{A}$, $j = 1$;
2: for all $t = 1, 2, \ldots, T$ do
3:     if Receive current network-assist state $s_t$ then
4:         Update the arm subset $\mathcal{A}_{s_t}$;
5:         Remove information, check if $s_t$ appears and perform corresponding operations using Subroutine 1;
6:     end if
7:     Select an arm, update the statistical information, and maintain the sliding window using Subroutine 2;
8: end for
```

In the scenario where the arm space is discrete, we design an online learning algorithm, called Discrete Network-assist State-based UCB (DNS-UCB) for the non-stationary MAB problem described in Section II.

The two primary challenges we face are: 1) how to maximize expected payoffs ((or minimize cumulative regret) without knowledge of the payoff distributions for all arms, and 2) how to adapt to the dynamic environment with different time-varying states. Note that these challenges cannot be tackled independently, as state change occurrences are sequential over time, and the uncertain sequence of states is only revealed after the process is completed. To overcome these challenges, we first utilize network-assist signals to adapt to state changes and then decompose the problem by devising a more progressive policy where a distinct policy p_s is learned for each individual state $s \in \mathcal{S}$. In what follows, we will describe the proposed DNS-UCB algorithm, which minimizes the separate regret instances for all states. To reduce code redundancy, we separate two portions of the algorithm by dividing them into two subroutines, which will be reused for the continuous arm space setting: (a) remove-check-perform (RCP, in Subroutine 1), and (b) select-update-maintain (SUM, in Subroutine 2).

Subroutine 1 RCP

```
1: Remove the wrong-state information stored in sliding window $\mathcal{W}$ according to Eq. (6);
2: if $s_t \in \mathcal{S}_t$ then
3:     Read the stored information at round $j$:
           $\left(n_a^{s_t}(j), \hat{u}_{s_t,j}(a)\right), \forall a \in \mathcal{A}_{s_t}$;
4: else
5:     $\mathcal{S}_t = \mathcal{S}_{t-1} \bigcup \{s_t\}$;
6:     Set round $j = 1$ for new state $s_t$;
7: end if
```

Subroutine 2 SUM

```
1: Select arm $a_t$ according to Eq. (5);
2: Observe the corresponding payoff $u_{s_t,j}(a_t)$;
3: $n_{a_t}^{s_t}(j) \leftarrow n_{a_t}^{s_t}(j) + 1$, update the mean payoff $\hat{u}_{s_t,j}(a_t)$ according to $u_{s_t,j}(a_t)$, $j \leftarrow j + 1$;
4: Add $u_{s_t,j}(a_t)$ of selected arm $a_t$ into $\mathcal{W}$;
5: if $|\mathcal{W}| = \lceil C \rceil$ then
6:     Remove the earliest one stored in $\mathcal{W}$;
7: end if
```

As outlined in Algorithm 1, we maintain a set of all detected states $\mathcal{S}_t$ at each time slot $t \in \mathcal{T}$, which is initialized as an empty set (Line 1). At each time slot t, the subset of arms $\mathcal{A}_{s_t} \subseteq A$ is obtained from network-assist signal if a specific state s_t is detected (Line 3, 4). Considering that each state $s \in \mathcal{S}$ is independent, the entire time slots $\mathcal{T}$ can be partitioned into $|\mathcal{S}|$ previously unknown sub-horizons, which is denoted as T_s, with $\sum_{s \in \mathcal{S}} T_s = T$. To keep track of the unknown sub-horizon for each state s, we maintain a round counter j within each sub-horizon T_s, $j \in [T]_s$. Next, we introduce how to select arms and update estimation under each state $s \in \mathcal{S}$. For all $a \in \mathcal{A}_s$, let $n_a^s(j)$ be the number of times arm a has been selected up to round $j \in [T]_s$. We also define $\hat{u}_{s,j}(a)$ as the empirical mean of the payoffs for arm a up to round j, i.e., $\hat{u}_{s,j}(a) \triangleq \frac{1}{n_a^s(j)} \sum_{i=1}^{n_a^s(j)} u_{s,i}(a)$. Based on the historically received payoffs generated from arm subset $\mathcal{A}_s$ under state s, the UCB index of arm a at round j is defined as follows:

$$\mathrm{UCB}_a^s(j) = \hat{u}_{s,j}(a) + \sqrt{\frac{2\log(j)}{n_a^s(j)}}, \tag{4}$$

where the first term $\hat{u}_{s,j}(a)$ in Eq. (4) represents exploitation of current information of empirically mean payoffs, and the second term $\sqrt{2\log(j)/n_a^s(j)}$ represents exploration for potential better arm not yet found, thereby mitigating the risk of excessive exploitation. Let $\mathcal{K}^s(j)$ be the set of arms that have not been selected until round j under state s, i.e., $\mathcal{K}^s(j) = \{a \in \mathcal{A}_s \mid n_a^s(j) = 0, \forall a \in \mathcal{A}_s\}$. The arm a_j to be selected is then determined (Line 1 in Subroutine 2):

$$a_j = \begin{cases} \text{random choice of } \mathcal{K}^s(j-1), & \mathcal{K}^s(j-1) \neq \varnothing \\ \arg\max_{a \in \mathcal{A}_s} \mathrm{UCB}_a^s(j-1), & \text{otherwise} \end{cases} \tag{5}$$

Subsequently, we update the statistics information based on the payoff of the selected arm (Line 2, 3 in Subroutine 2). Note that it is possible to revisit the same states, so we leverage previous learning efforts rather than restarting an entirely new learning process for the revisited state. Hence, we continue the learning process based on the previous record if the newly detected state s_t at time slot t has been visited before, i.e., $\exists h \in [t-1]$, s.t. $s_t = s_h$; otherwise, we start from scratch (Line 2-7 in Subroutine 1). This approach offers cost and latency reductions in the learning process, which is particularly beneficial when time and resources are limited.

It is important to note that, in general, state changes may not be detected instantaneously and could be delayed, namely, a state change is received at time slot t after an unknown delay d_{g_i} at the change-point g_i ($i \in [M]$). This can result in erroneous samples of mixed states during time interval d_{g_i}. To tackle this issue, we propose a sliding-window technique. The sliding window, denoted as $\mathcal{W}$, aims to identify a learning process that occurs within the time interval d_{g_i} when mixed states are present. Considering the unknown delay of network-assist signals, the size of the sliding window is set to the maximum delay, i.e., $|\mathcal{W}| \triangleq \lceil C \rceil$ (Line 5-7 in Subroutine 2). In practice, this can be achieved via history measurements and periodic updates [21]. If a state change is detected, indicated by $s_t \neq s_{t-1}$, we remove the recorded payoffs and selection counts within the time period $[t - |\mathcal{W}| + 1, t]$ for all $a \in \mathcal{W}$ (Line 1 in Subroutine 1). The specific operations for removing the wrong information are defined as follows:

$$\begin{cases} j = j - |\mathcal{W}|, \\ n_a^s(j) = n_a^s(j) - N_a(k), \\ \hat{u}_{s,j}(a) = \hat{u}_{s,j}(a) - U_a(k). \end{cases} \tag{6}$$

Here, for each arm a in the sliding window $\mathcal{W}$, $N_a(k)$ is the number of occurrences of arm a and $U_a(k)$ is the sum of empirical payoffs of arm a. Induced from Eq. (1), the learning round is greater than the size of the sliding window, i.e., $j - |\mathcal{W}| > 0$. This sliding-window approach allows us to eliminate incorrect samples within the window while preserving the rest of the learned statistic information, albeit at the sacrifice of some learning efforts within the time interval of $C - d_{g_i}$.

B. *Continuous Arm Space Algorithm (CNS-UCB)*

Let us introduce the algorithm proposed for the continuous arm space, called Continuous Network-assist State-based UCB (CNS-UCB). In addition to the challenges in the discrete arm space, a new technical challenge in continuous scenarios involves effectively managing the infinite arm space to avoid endless search for the optimal arm. To address this challenge, we technically discretize the continuous arm space to constrain the number of arms to choose based on the *uniformly local Lipschitz* condition, defined as Eq. (2).

Algorithm 2 CNS-UCB

```
1: Initialize $\mathcal{S}_0 = \emptyset$, $\mathcal{X}s = \mathcal{X}$, $j = 1$, $H = 2$;
2: while $H \leq T$ do
3:     Set discretization parameter $D$ according to Eq. (7);
4:     Determine the arm subset $\mathcal{A}_s$ of $\mathcal{X}_s$ using Eq. (8);
5:     for all $t = H, H+1, \ldots, \min(2H-1, T)$ do
6:         if Receive current network-assist state $s_t$ then
7:             Update the arm subspace $\mathcal{X}_{s_t}$;
8:             Obtain the subset $\mathcal{A}_{s_t}$ according to Eq. (8);
9:             Remove information, check if $s_t$ appears and
               perform corresponding operations using Subroutine 1;
10:        end if
11:        Select an arm, update the statistical information,
           and maintain the sliding window using Subroutine 2;
12:    end for
13:    $H = 2H$;
14: end while
```

The details of CNS-UCB are shown in Algorithm 2. CNS-UCB consists of the outer and inner loop, respectively responsible for transforming the continuous space into discrete space and learning a policy $\boldsymbol{p}$ to minimize cumulative regret. For the outer loop, the discretization parameter D is carefully designed as follows (Line 3):

$$D = \left\lceil \left(\frac{H}{\ln H} \right)^{\frac{1}{2\beta+1}} \right\rceil, \tag{7}$$

where H represents the inner loop horizon. To transform a non-uniform algorithm into a uniform one, the outer loop applies a standard doubling strategy to double the scale of H in an iterative manner (Line 5, 13). In the inner loop, to handle state changes, we obtain a subspace $\mathcal{X}_s$ based on the network-assist signal instead of exploring the entire continuous arm space $\mathcal{X}$ (Line 7). For instance, some network devices may provide available bandwidth estimation, which can be used to construct the subspace. The continuous arm space $\mathcal{X}_s = [m_s, n_s] \subseteq \mathcal{X}$,$\forall s \in \mathcal{S}$ is discretized into a finite set $\mathcal{A}_s$, whose size is determined by the discretization parameter D, with $D = |\mathcal{A}_s|$ (Line 8):

$$\mathcal{A}_s = \left\{ m_s + \frac{d}{D}(n_s - m_s) \;\middle|\; d = \{0, 1, 2, \ldots, D\} \right\}. \tag{8}$$

IV. THEORETICAL ANALYSIS AND SIMULATIONS

In this section, we present a comprehensive theoretical analysis of the performance of our proposed algorithms. We show that both the DNS-UCB and CNS-UCB algorithms achieve *sub-linear* regret upper bounds, i.e., $\lim_{T\to\infty} \frac{R_{\boldsymbol{p}}(T)}{T} = 0$. Our algorithms' policy $\boldsymbol{p}$ achieves near-optimal performance, showcasing its superiority over the linear regret of the uniform arm selection policy and providing insight into the advantages of our approach. Then, we compare our proposed algorithms with state-of-the-art algorithms in both arm spaces.

A. Performance Guarantee of DNS-UCB

We firstly present Theorem 1, which provides the upper bound of the time-average regret of Algorithm 1.

Theorem 1. *The regret upper bound of DNS-UCB is*

$$O\left(\sqrt{|\mathcal{S}||\mathcal{A}|_{max} T \log(T)}\right) + O\left(CM\right), \tag{9}$$

where $|\mathcal{A}|_{max} = max_{s\in\mathcal{S}}|\mathcal{A}_s|$ *and* $|\cdot|$ *represents the number of arms in the arm set.*

Remark 1. *The regret in Theorem 1 can be decomposed into two terms: the first term* $O\left(\sqrt{|\mathcal{S}||\mathcal{A}|_{max} T \log(T)}\right)$ *arises from the difference between the arm chosen under each state and the optimal arm, while the second term* $O\left(CM\right)$ *represents the regret incurred by delayed feedback of state changes. The magnitude of the regret upper bound depends on the number of states* $|\mathcal{S}|$*, and we then analyze two degenerate cases. In the first case, assuming there is only one observed state, the regret upper bound without signal delay is* $O\left(\sqrt{|\mathcal{A}|_{max} T \log(T)}\right)$*, with a tighter regret bound than the standard finite MAB bandit problem due to* $|\mathcal{A}|_{max} \leq |\mathcal{A}|$*. In the alternative scenario, wherein all states are equally likely to be observed—a scenario that represents the most adverse conditions for regret as detailed in Appendix A—and with arms uniformly distributed across each state such that* $|\mathcal{A}|_{max} = |\mathcal{A}|/|\mathcal{S}|$*, the bound is refined to* $O\left(\sqrt{|\mathcal{A}| T \log(T)}\right)$*. This is improved over the* $O\left(\sqrt{M|\mathcal{A}| T \log(T)}\right)$ *bound posited by [22]. Please refer to Appendix A for comprehensive proof.*

B. Performance Guarantee of CNS-UCB

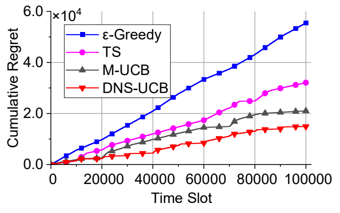
Then, we demonstrate the regret upper bound and the memory required to store arms of the CNS-UCB algorithm. For simplicity, we assume $\mathcal{X} = [0, 1]$ without loss of generality.

Theorem 2. *The regret upper bound of CNS-UCB is*

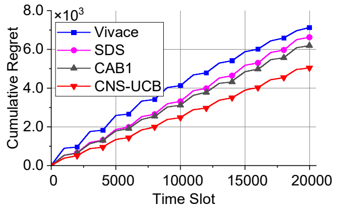
$$O\left(\left(L|\mathcal{S}|^{\frac{\beta}{2\beta+1}} + |\mathcal{S}|^{\frac{1}{2}}\right) T^{\frac{\beta+1}{2\beta+1}} \log^{\frac{\beta}{2\beta+1}}(T)\right) + O\left(CM\right), \tag{10}$$

with $\tilde{O}\left(T^{1/2\beta+1}\right)$ *memory required to store arms. Here,* β *and* L *are the constants of* ***uniformly local Lipschitz*** *condition, and the notation* $\tilde{O}(\cdot)$ *ignores logarithmic factors.*

Remark 2. *Besides the regret caused by delay, the regret is primarily due to two factors: 1) the gap between the feasible optimal arm in the discretized arm subset* $\mathcal{A}_s$ *and the theoretically optimal arm, which is bounded by* $O\left(L|\mathcal{S}|^{\frac{\beta}{2\beta+1}} T^{\frac{\beta+1}{2\beta+1}} \log^{\frac{\beta}{2\beta+1}}(T)\right)$*. 2) the gap between the chosen arm and the best arm in the current discretized arm subset* $\mathcal{A}_s$*, bounded by* $O\left(|\mathcal{S}|^{\frac{1}{2}} T^{\frac{\beta+1}{2\beta+1}} \log^{\frac{\beta}{2\beta+1}}(T)\right)$*. The memory required to store arms mainly depends on the discretization parameter D. In a typical scenario where* $\beta = 1$ *with* $\|a - a'\|_2 \leq \delta$*, the condition is written as* $|u_{s,t}(a) - u_{s,t}(a')| \leq L\|a - a'\|_2$*. In this case, the regret upper bound and the storage*



(a): Discrete Arm Space



(b): Continuous Arm Space

Fig. 2: Cumulative regret in two arm spaces.

memory requirement are approximately $\tilde{O}(T^{\frac{2}{3}})$, and $\tilde{O}(T^{\frac{1}{3}})$, matching stationary Lipschitz continuous bandits [23], [24]. Detailed proof is in Appendix B.

C. Performance Comparison

We present the numerical simulation results with maximum delay $C = 500$ time slots for both proposed algorithms.

Discrete Arm Space. There are a total of 4 states for simulations, each associated with a subset of 12 arms. The payoff of each arm is modeled as a Bernoulli random variable. We evaluate the proposed DNS-UCB algorithm against several benchmark algorithms, including 1) ϵ-Greedy, selecting the arm with the highest average payoff with probability $1-\epsilon$, and randomly tries other arms with probability ϵ [19]. 2) Thompson sampling (TS), a probabilistic Bayesian approach to the MAB problem [20]. 3) M-UCB, an adaptive MAB algorithm that incorporates change detection for the non-stationary bandit framework [22].

Continous Arm Space. We evaluate the proposed CNS-UCB algorithm against several benchmark algorithms, including 1) Vivace, an online gradient-ascent learning policy for making decisions [18]. 2) Sliding-decision-space approach (SDS), employing a padding function to search within a continuous range by increasing or decreasing the arm [5]. 3) CAB1, designed for the one-parameter continuum-armed bandit problem [11]. Although Vivace and SDS originally have different decision spaces, here we use their algvorithms to guide the decision (arm) selection from the same range for a fair comparison. The continuous arm space is set to the range $[1, R]$, where R is specified as 8000 for a large scale. To make dynamic payoff distributions, we set two states: $\bar{u}_s(a) = -(a-\frac{1}{8}R)^2$ when s is even and $\bar{u}_s(a) = -(a-\frac{3}{4}R)^2$ when s is odd, $\forall a \in [1, R]$. Then, the regret is normalized.

As shown in Fig. 2, the regret is determined by computing the difference between the payoff achieved by each algorithm and the payoff achieved by Hindsight, which analyzes the complete record of all time slots, offers the ground truth of payoffs and is not plotted. Fig. 2(a) and Fig. 2(b) demonstrate the effectiveness of our proposed algorithms in mitigating the regret of payoffs in both arm spaces.

V. OPTIMIZING NETWORK APPLICATIONS

This section presents the specific applications of our proposed DNS-UCB and CNS-UCB algorithms to network applications under two types of arm spaces. We describe the specific problem for each network application and map our framework to it. Then, we evaluate them with the comparison of existing state-of-the-art algorithms.

A. Wireless Network Channel Selection (Discrete Arms)

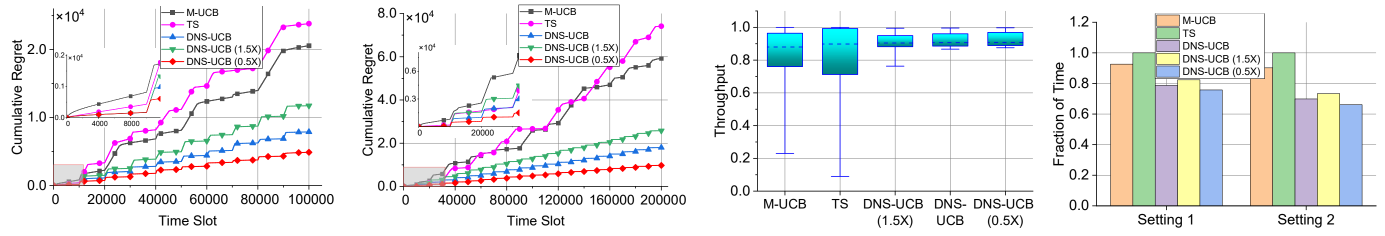
1) Model Mapping: Cognitive radio networks categorize users into two types of users. Primary users (PUs) have licensed access to a specific spectrum band and enjoy priority for network resources. The presence of high-priority PUs and the requirement that secondary users (SUs) should not interfere with them gives rise to the opportunistic spectrum access problem, which involves accessing the available spectrum [25].
Motivation. In the dynamic landscape of wireless cognitive radio networks, the crux of the challenge for SUs to efficiently discover and harness available spectrum resources lies in the SUs' ability to detect idle spectrum channels—a task that must be executed with precision to avoid disrupting the operations of PUs, who retain preferential rights to network resources. This intricate balance between utilization and coexistence underscores the critical need for innovative solutions that enable SUs to seamlessly integrate into the spectrum without impinging on the PUs' domain.

Formally, we consider a wireless cognitive radio network system consisting of $|\mathcal{A}|$ channels. At every time slot $t \in \mathcal{T}$, each spectrum channel $a \in \mathcal{A}$ is associated with an unknown normalized channel throughput $u_t(a) \in [0,1]$. An online learned channel selection policy $\boldsymbol{p}$ guides SUs in selecting channels with high throughput based on previously observed channel throughput. We map the proposed DNS-UCB algorithm to a spectrum channel management framework under a dynamic network environment, composed of spectrum mobility, spectrum sensing, and spectrum decision.
Spectrum Mobility. The PUs mainly utilize spectrum channels. The availability of spectrum channels, referred to as state s, changes over time due to the arrival and departure of PUs, resulting in state transitions. The number of different availability of spectrum channels is represented as $\mathcal{S}, \forall s \in \mathcal{S}$. If a PU arrives or departs at $g_i, i \in [M], g_i \in \mathcal{T}$, spectrum mobility is triggered, i.e., a state change.
Spectrum Sensing. A SU can sense the new state s_t after delay d_{g_i} at time slot t ($t = g_i + d_{g_i}$) by listening to the available spectrum channels through the wideband access technology and coordinated protocol [14], [26]. Hence, we use this available information as **"network-assist" signals**. Let $\eta_{s_t}(a)$ denote a binary variable equal to 1 if channel a is available at time slot t, and 0 otherwise under state s_t. A SU receives the listened result set $\mathcal{N}_{s_t} = \{\eta_{s_t}(1), \dots, \eta_{s_t}(|\mathcal{A}|)\}$ and updates the subset of channels $\mathcal{A}_{s_t}$ as follows:

$$\mathcal{A}_{s_t} = \{a \in \mathcal{A} \mid \eta_{s_t}(a) = 1, \forall \eta_{s_t}(a) \in \mathcal{N}_{s_t}\}. \qquad (11)$$

Spectrum Decision. To select an appropriate channel for transmission, a SU performs a channel-characteristic learning routine. This routine takes the user's local knowledge of throughput as input and selects one channel $a_t \in \mathcal{A}$ with the largest arm index, i.e., using Eq. (5) at time slot t.

2) Performance Evaluation: Utilizing the normalized channel throughput from the open available 4G trace dataset [27],



(a): Cumulative Regret (Setting 1) (b): Cumulative Regret (Setting 2) (c): Channel Throughput (d): Time Required

Fig. 3: Performance comparisons under the wireless network channel selection problem.

TABLE I: Example of Throughput of 6 Channels.

Channel Index	#1	#2	#3	#4	#5	#6
Channel Throughput	0.398	0.851	0.432	0.516	0.752	0.630

the channel throughput $u_t(a)$ of each channel $a \in \mathcal{A}$ is uniformly sampled from $[0, u(a)]$, where $u(a)$ takes values from the second row of Table I. If a channel is occupied by PUs, its channel throughput will be discounted by a factor $\gamma \in (0,1)$ due to the absolute priority of channels enjoyed by PUs. Different states represent various occupied channel configurations, such as channels 3 and 6 being occupied under state 1 in Table I. We simulate in two settings: Setting 1 with $T = 10^4$, $M = 10$, $|\mathcal{S}| = 4$, $|\mathcal{A}| = 12$, and $\gamma = \frac{1}{2}$; and Setting 2 with $T = 2 \times 10^4$, $M = 20$, $|\mathcal{S}| = 8$, $|\mathcal{A}| = 24$, and $\gamma = \frac{1}{3}$. In both settings, $|\mathcal{A}|/|\mathcal{S}|$ channels are not occupied by PUs under every state. The state changes every T/M time slot, with $C = 0.1T/M$ as an instance. Consistent with [28], we set the slot duration to 100 ms and adhere to the IEEE 802.11 Standard by defining the minislot duration as 20 μs.

We conduct an evaluation of the proposed DNS-UCB algorithm alongside the algorithms described in Section IV. For clarity in the graphical representation, we have omitted the ϵ-greedy algorithm with the poorest performance· from the figures. Since throughput is not a binary payoff, following [20], a binary value is generated from a Bernoulli trial, with the normalized throughput serving as the Bernoulli probability. Except for the general delay setting ($C = 0.1T/M$), we also simulate DNS-UCB with maximum delay $C = 0.05T/M$ and $C = 0.15T/M$, denoted as DNS-UCB (0.5×) and DNS-UCB (1.5×), to study the impact of network-assist signal delays.

Figure 3 shows the results of performance comparisons between different algorithms. As shown in Fig. 3(a), the cumulative regret of channel throughput for DNS-UCB accounts for 38.2%, 33.1% of M-UCB and TS, respectively. As time goes by in Fig. 3(b), the percentage further drops to 30.4%, 24.8%. Fig. 3(a) and Fig. 3(b) also demonstrate the impact of network-assisted signal reception delay between DNS-UCB series. To better understand the channel throughput distribution, Fig. 3(c) shows that the DNS-UCB series exhibit a generally high and stable channel throughput. Compared to DNS-UCB and DNS-UCB (0.5X), DNS-UCB (1.5X) indicates a more significant difference in the distribution of throughput due to its highest delay. Fig. 3(d) illustrates the time taken by each algorithm to achieve the same sum of channel throughput, where DNS-UCB (0.5X) requires the fewest time slots. Since the two settings have different orders of magnitude, the algorithm with the lowest throughput is used as the reference.

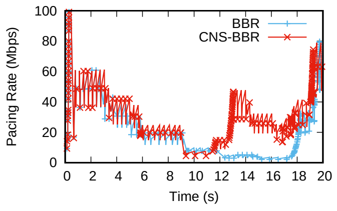
B. Rate-Based TCP Congestion Control (Continuous Arms)

1) Model Mapping: We elaborate the application of our CNS-UCB algorithm using a TCP congestion control scenario. We choose the popular BBR protocol [29] for illustration and show how to apply CNS-UCB to dynamically select proper values for the "highGain" parameter of BBR.

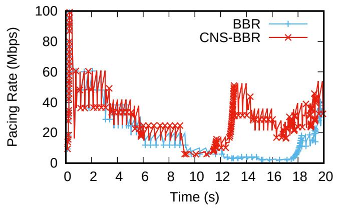
Motivation. BBR was originally designed to control long flows that last for many round-trip times (RTTs), and thus defined four phases for a flow, namely, Startup, Drain, ProbeBW, and ProbeRTT [29]. By successively adjusting (or pacing) the flow rates through multiple rounds and different phases, BBR can effectively control long flows to achieve good performance. However, there is an emerging trend that more and more flows become "*short flows*" which are finished within a few RTTs [30], due to higher network link speeds or new application patterns such as short videos. These short flows will significantly impair the effectiveness of BBR's successive rate control since they may only be alive in the Startup phase, making rate control in Startup more important. We notice that BBR's highGain parameter plays a critical role in the Startup phase as it regulates the initial pacing rate and influences the magnitude of bursts. However, BBR only uses a fixed value (≈ 2.89) for highGain across various network settings, leading to ineffective rate control in scenarios with many short flows.

We apply our CNS-UCB algorithm to guide the dynamic selection of the highGain values to improve BBR, named "CNS-BBR". The details of CNS-BBR are described as follows.

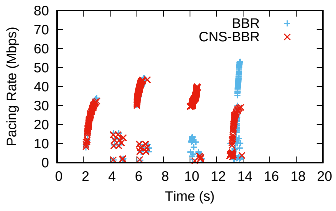
Network-Assist Signal. ECN is a widely supported signal for congestion notification in many modern routers, i.e., the RED active queue management (AQM) scheme [31], which is implemented in many network routers, supports ECN marking for packets when the queue length at the router buffer exceeds certain thresholds. We define ECN ratio as our **"network-assist" signal** used in CNS-BBR, which is expressed by $e_a(t) = \frac{ne_a(t)}{nt_a(t)}$. Here, $ne_a(t)$ and $nt_a(t)$ denote the number of ECN-marked packets and the number of total packets sent using arm a between round $[t, t+1]$, respectively. A sender running CNS-BBR can compute this information and then select a proper arm based on CNS-UCB.



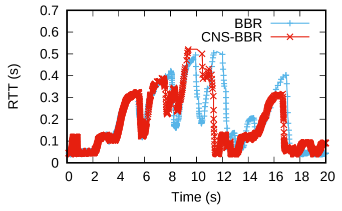
(a): Pacing Rate of Long Flow 1



(b): Pacing Rate of Long Flow 2

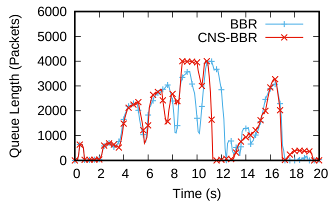


(c): Pacing Rates of Short Flows

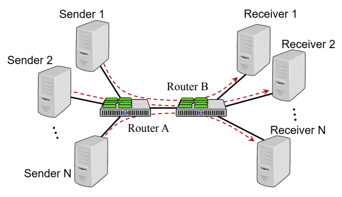


(d): RTT of Long Flow 1

Fig. 4: Performance comparisons for the rate-based TCP congestion control scenario.



(a): Queue Length at Router A



(b): Dumbbell Topology

Fig. 5: Queue dynamics and topology for the TCP scenario.

Reward (Payoff) Function. Note that smaller values of ECN ratio indicate less congested networks. Thus, we employ the following reward (payoff) function to evaluate a chosen arm in CNS-BBR: $r_a(t) = 1 - e_a(t)$, where $r_a(t)$ represents the reward value of the chosen arm a between round $[t, t+1]$.
Arm Setup. For handling more general scenarios containing both short and long flows, CNS-BBR dynamically selects a proper value for highGain from a continuous arm space [2, 3]. Utilizing the network-assist signal, the original arm space can further be narrowed to an arm subspace belonging to a specific state. For example, when the ECN ratio $e_a(t) < 2\%$ indicating a non-congested state, the value of highGain can be selected from the subspace $[2.5, 3]$. When $e_a(t) \geq 2\%$ indicating a congested state, the corresponding subspace is set to $[2, 2.5]$. Based on the level of ECN ratio, more states can be defined. Here, we use two states for simplicity.

2) *Performance Evaluation:* We use the ns-3 (version 3.38) packet-level simulator [32] to evaluate CNS-BBR by comparing it to the original BBR. The simulation topology is shown in Fig. 5(b), where sender i $(i = 1, 2, .., N,\ N = 10)$ transmits data to the corresponding receiver i through two routers under the typical dumbbell topology. Specifically, each of Sender 1 and Sender 2 establishes a long flow for transmission during the entire simulation. The remaining senders intermittently initiate short flows lasting for around one second. The link connecting two routers has the capacity of 100 Mbps and propagation delay of 10 ms, and all the other links are set to 1000 Mbps capacity and 5 ms delay. CNS-BBR or BBR is run on the sender side. The RED AQM [31] is employed by both routers with ECN enabled, and the minimum and maximum thresholds for ECN marking are set to 1000 and 3000 packets, respectively. Note that a sender can record past rewards related to the short flows initiated by itself, so as to guide the future selection for highGain in CNS-BBR.

The simulation results are presented in Fig. 4 and Fig. 5(a). From Fig. 4(a) and Fig. 4(b), we see BBR yields very low pacing rates (during $[12\,\mathrm{s}, 18\,\mathrm{s}]$) for the two long flows when they are mixed with bursty short flows, substantially lowering the overall throughput. In contrast, CNS-BBR significantly improves the pacing rates of the two long flows while mostly maintaining similar or lower RTT and queue length (refer Fig. 4(d) and Fig. 5(a)), by dynamically adjusting highGain based on CNS-UCB. Fig. 4(c) depicts the pacing rates of the short flows initiated by a specific Sender 3. One can observe that the pacing rate of the last short flow (around $[13\,\mathrm{s}, 14\,\mathrm{s}]$) goes much higher than that of the two long flows in BBR, so CNS-BBR significantly alleviates this issue.

VI. Related Work

In the field of decision-making, the MAB problem has been the subject of many research works. For instance, [1] proposes a combinatorial sleeping MAB under a stationary environment. However, the problem of non-stationary bandits remains an active area of research. One recent MAB variant in the non-stationary bandit is that the payoff distributions of arms remain constant over epochs, but change at unknown time instants, referred to as a "piece-wise stationary" process. [7], [22] utilize limited side observations on past payoff data and infer change occurrences. Besides the non-stationary bandit, continuous arm space brings new challenges due to limited memory and infinite arms. [24] extends such a framework into the adversarial setting.

In our network application of the cognitive radio network, [33] proposes a ϵ-greedy based algorithm to handle the opportunistic spectrum problem. [4] studies an MAB problem with minimum-guarantee constraints and applies it to select channels. [25] designs a hierarchical thompson sampling for multi-band radio channel selection. However, all these works do not consider dynamically monitoring spectrum availability and adapting to it. In TCP congestion control, the core component, congestion control, regulates data transmission rate to achieve congestion avoidance [17]. There are two primary methods: implicit congestion control and network-assist (explicit) congestion control. Implicit congestion control usually uses packet loss or delay to infer congestion, whereas network-assist congestion control directly uses signals from network devices like ECN to assess congestion status [15]. Besides traditional window-based schemes [17], rate-based schemes have been proposed to perform direct rate control such as BBR [29]. Moreover, online learning approach has recently been applied to TCP congestion control like PCC-Vivace [18] and rate control in RDMA networks [34]. However, these approaches may face challenges in dynamic network environments with many short flows which are finished within a few RTTs. Our work differs from the studies above in that we focus on the characteristics of specific network scenarios. We design a new general MAB framework, utilizing assistance from the network signal and integrating it into our algorithms to adapt to the time-varying environment. Then, we show that our protocols based on our MAB algorithms can better tackle the dynamic network conditions. In general, our designed approaches can cover a large range of network applications regardless of arm spaces.

VII. CONCLUSION

This paper proposes a novel approach for network application optimization by leveraging network-assist online learning in dynamic environments. To our best knowledge, this is the first work to employ network-assist signals in an online learning model to handle the dynamic environments and is integrated with both discrete and continuous arm space. To address the challenges posed by two distinct arm spaces, we develop DNS-UCB and CNS-UCB algorithms, which incorporate network-assist signals and provide provable *sub-linear* regret bounds. Furthermore, we apply both algorithms to real-world network applications, enhancing cognitive ratio network channel selection and rate-based TCP congestion control protocols. Extensive numerical and packet-level simulation results demonstrate the substantial benefits of network-assist state-based online learning in enhancing network performance.

ACKNOWLEDGEMENT

The work of John C.S. Lui is supported in part by the RGC GRF-14202923.

APPENDIX

A. Proof of Theorem 1

Proof. For simplicity, for all $s \in \mathcal{S}$, we define the "sub-payoff gap" Δ_a^s as the difference between the expected payoff $\bar{u}^s_{a^\star_s}$ of the optimal arm $a^\star_s$ and any arm $a \in \mathcal{A}_s$, i.e., $\Delta_a^s = \bar{u}^s_{a^\star_s} - \bar{u}^s_a$. The total regret, defined in Eq. (3), is divided into a separate regret instance for each state:

$$R_{p_s}(T_s) = \mathbb{E}\left[\sum_{\substack{t\in\mathcal{T}:\\ s_t=s}} \left(u_{s_t,t}(a^\star_{s_t}) - u_{s_t,t}(a_t^{p_{s_t}})\right)\right]. \tag{12}$$

Substituting Eq. (12) into Eq. (3) and applying Lemma 4.5 from [35], we derive the regret with signal delay as follows:

$$\begin{aligned} R_{\boldsymbol{p}}(T) &= \sum_{s\in\mathcal{S}} R_{T_s} + CM = \sum_{s\in\mathcal{S}}\sum_{a\in\mathcal{A}_s} \Delta_a^s \mathbb{E}\left[n_a^s(j)\right] + CM \\ &= \sum_{s\in\mathcal{S}}\left(\sum_{\substack{a\in\mathcal{A}_s\\ \Delta_a^s<\Delta^s}} \Delta_a^s\mathbb{E}\left[n_a^s(j)\right] + \sum_{\substack{a\in\mathcal{A}_s\\ \Delta_a^s\geq\Delta^s}} \Delta_a^s\mathbb{E}\left[n_a^s(j)\right]\right) + CM, \end{aligned} \tag{13}$$

where $\Delta^s = \sqrt{\frac{8|\mathcal{A}_s|\ln T_s}{T_s}}$, $\forall s \in \mathcal{S}$.

Due to $\mathbb{E}\left[\sum_{a\in\mathcal{A}_s} n_a^s(j)\right] = \mathbb{E}[T_s] = T_s$, we have:

$$\sum_{\substack{a\in\mathcal{A}_s\\ \Delta_a^s<\Delta^s}} \mathbb{E}\left[n_a^s(j)\right] = \mathbb{E}\left[\sum_{\substack{a\in\mathcal{A}_s\\ \Delta_a^s<\Delta^s}} n_a^s(j)\right] \leq T_s. \tag{14}$$

The proof of Theorem 1 in [19] yields the following inequality:

$$\mathbb{E}\left[n_a^s(j)\right] \leq \frac{8\ln j}{(\Delta_a^s)^2} + \underbrace{1+\frac{\pi^2}{3}}_{K}, \forall a \in \mathcal{A}_s, \forall s \in \mathcal{S}. \tag{15}$$

Plugging Eq. (14) and Eq. (15) into Eq. (13), we obtain:

$$\begin{aligned} R_{\boldsymbol{p}}(T) &\leq \sum_{s\in\mathcal{S}}\left(T_s\Delta^s + \sum_{\substack{a\in\mathcal{A}_s\\ \Delta_a^s\geq\Delta^s}}\left(\frac{8\ln T_s}{\Delta_a^s} + K\Delta_a^s\right)\right) + CM \\ &\leq \sum_{s\in\mathcal{S}}\left(T_s\Delta^s + \frac{8|\mathcal{A}_s|\ln T_s}{\Delta^s} + \sum_{a\in\mathcal{A}_s} K\Delta_a^s\right) + CM \\ &= \sum_{s\in\mathcal{S}}\left(4\sqrt{2|\mathcal{A}_s|T_s\ln T_s} + \sum_{a\in\mathcal{A}_s} K\Delta_a^s\right) + CM \\ &\leq \sum_{s\in\mathcal{S}}\left(4\sqrt{2\sum_{t=1}^{T}|\mathcal{A}|_{max}\mathbb{I}\left\{s_t=s\right\}\ln T}\right) \\ &\quad + \sum_{s\in\mathcal{S}}\sum_{a\in\mathcal{A}_s} K\Delta_a^s + CM. \end{aligned} \tag{16}$$

The inequality $\sum_{s\in\mathcal{S}}\sqrt{\sum_{t=1}^{T}\mathbb{I}\left\{s_t=s\right\}} \leq \sqrt{|\mathcal{S}|T}$ holds by Jensen's inequality, with equality when $\sum_{t=1}^{T}\mathbb{I}\left\{s_t=s\right\} = \frac{T}{|\mathcal{S}|}$. Plugging this into Eq. (16), we have $R_{\boldsymbol{p}}(T) \leq O\left(\sqrt{|\mathcal{S}||\mathcal{A}|_{max}T\log(T)}\right) + O\left(CM\right)$. □

B. Proof of Theorem 2

Proof. We first focus on the regret in the inner loop, denoted as $R_{\boldsymbol{p}}(H)$, where H is temporarily fixed. We renumber the H steps from 1 to H for brevity. Define $a^\star_s$ as the optimal arm in the continuous arm subspace $\mathcal{X}_s$, and a'_s as the optimal arm in the discretized arm subset $\mathcal{A}_s$ that is closest to $a^\star_s$. Due to Eq. (8) with $\mathcal{X}_s \subseteq \mathcal{X} = [0,1]$, we obtain: $|a'_s - a^*_s| \leq \frac{n_s-m_s}{D} \leq \frac{1}{D}$. Combing Eq. (2) and Eq. (7), we have:

$$\begin{aligned} &\mathbb{E}\left[\sum_{\substack{t\in[H]:\\ s_t=s}}\left(u_{s_t,t}(a^\star_{s_t}) - u_{s_t,t}(a'_{s_t})\right)\right] \\ &\leq \mathbb{E}\left[\sum_{\substack{t\in[H]:\\ s_t=s}} L\left\|a^\star_{s_t} - a'_{s_t}\right\|_2^\beta\right] \leq \frac{L\sum_{t=1}^{H}\mathbb{I}\left\{s_t=s\right\}}{D^\beta} \\ &\leq O\left(L\left(\sum_{t=1}^{H}\mathbb{I}\left\{s_t=s\right\}\right)^{\frac{\beta+1}{2\beta+1}}\log^{\frac{\beta}{2\beta+1}}\left(\sum_{t=1}^{H}\mathbb{I}\left\{s_t=s\right\}\right)\right), \end{aligned} \tag{17}$$

where the last term holds due to $(\frac{x_1}{\ln x_1})^{\frac{1}{2\beta+1}} \leq (\frac{x_2}{\ln x_2})^{\frac{1}{2\beta+1}}$ for $x_2 \geq x_1 \geq e$. Otherwise, we treat the rest regret as a constant. Hence, we have $\mathbb{E}\left[\sum_{t\in[H]:s_t=s}\left(u_{s_t,t}(a^\star_{s_t}) - u_{s_t,t}(a'_{s_t})\right)\right] \leq O\left(L\sum_{t=1}^{H}\mathbb{I}\{s_t = s\}^{\frac{\beta+1}{2\beta+1}}\log^{\frac{\beta}{2\beta+1}}(H)\right)$.

Next, we employ continuum-armed bandit techniques [11]. summing Eq. (17) for all $s \in \mathcal{S}$ and applying inequality $\sum_{s\in\mathcal{S}}\left(\sum_{t=1}^{H}\mathbb{I}\{s_t = s\}\right)^{\frac{\beta+1}{2\beta+1}} \leq |\mathcal{S}|^{\frac{\beta}{2\beta+1}}H^{\frac{\beta+1}{2\beta+1}}$ by Jensen's inequality, we obtain the following bound:

$$\begin{aligned}&\mathbb{E}\left[\sum_{s\in\mathcal{S}}\sum_{\substack{t\in[H]:\\ s_t=s}}\left(u_{s_t,t}(a^\star_{s_t}) - u_{s_t,t}(a'_{s_t})\right)\right]\\ &\leq O\left(L|\mathcal{S}|^{\frac{\beta}{2\beta+1}}H^{\frac{\beta+1}{2\beta+1}}\log^{\frac{\beta}{2\beta+1}}(H)\right).\end{aligned} \tag{18}$$

Similar to the proof in Theorem 1 with $|\mathcal{A}|_{max} = D$, we have:

$$\begin{aligned}&\mathbb{E}\left[\sum_{s\in\mathcal{S}}\sum_{\substack{t\in[H]:\\ s_t=s}}\left(u_{s_t,t}(a'_{s_t}) - u_{s_t,t}(a^{p_{s_t}}_{s_t})\right)\right]\\ &\leq O\left(\sqrt{|\mathcal{S}||\mathcal{A}|_{max}H\log H}\right) + O(CM)\\ &= O\left(\sqrt{|\mathcal{S}|}H^{\frac{\beta+1}{2\beta+1}}\log^{\frac{\beta}{2\beta+1}}(H)\right) + O(CM)\end{aligned} \tag{19}$$

Combining Eq. (18) and Eq. (19) bounds the inner loop regret:

$$\begin{aligned}R_{\boldsymbol{p}}(H) &= \mathbb{E}\left[\sum_{s\in\mathcal{S}}\sum_{\substack{t\in[H]:\\ s_t=s}}\left(u_{s_t,t}(a^\star_{s_t}) - u_{s_t,t}(a^{p_{s_t}}_{s_t})\right)\right]\\ &\leq O\left(\left(L|\mathcal{S}|^{\frac{\beta}{2\beta+1}} + |\mathcal{S}|^{\frac{1}{2}}\right)H^{\frac{\beta+1}{2\beta+1}}\log^{\frac{\beta}{2\beta+1}}(H)\right) + O(CM).\end{aligned} \tag{20}$$

Finally, following the proof of Theorem 3.1 in [11], we sum up this bound over all iterations of the inner loop, resulting in a geometric progression. The largest term of this progression is expressed as $O\left(\left(L|\mathcal{S}|^{\frac{\beta}{2\beta+1}} + |\mathcal{S}|^{\frac{1}{2}}\right)T^{\frac{\beta+1}{2\beta+1}}\log^{\frac{\beta}{2\beta+1}}(T)\right)$ regarding T. This completes the proof of Theorem 2. □

References

[1] F. Li, J. Liu, and B. Ji, "Combinatorial sleeping bandits with fairness constraints," in *Proc. of IEEE INFOCOM*. IEEE, 2019, pp. 1702–1710.

[2] X. Liu, J. Zuo, H. Xie, C. Joe-Wong, and J. C. Lui, "Variance-adaptive algorithm for probabilistic maximum coverage bandits with general feedback," in *IEEE INFOCOM 2023-IEEE Conference on Computer Communications*. IEEE, 2023, pp. 1–10.

[3] S. Wang *et al.*, "Delay-aware microservice coordination in mobile edge computing: A reinforcement learning approach," *IEEE Transactions on Mobile Computing*, vol. 20, no. 3, pp. 939–951, 2021.

[4] K. Cai, X. Liu, Y.-Z. J. Chen, and J. C. Lui, "An online learning approach to network application optimization with guarantee," in *Proc. of IEEE INFOCOM*, 2018, pp. 2006–2014.

[5] X. Nie *et al.*, "Dynamic tcp initial windows and congestion control schemes through reinforcement learning," *IEEE Journal on Selected Areas in Communications*, vol. 37, no. 6, pp. 1231–1247, 2019.

[6] P. Yang, N. Zhang, S. Zhang, L. Yu, J. Zhang, and X. Shen, "Content popularity prediction towards location-aware mobile edge caching," *IEEE Transactions on Multimedia*, vol. 21, no. 4, pp. 915–929, 2018.

[7] J. Y. Yu and S. Mannor, "Piecewise-stationary bandit problems with side observations," in *Proc. of ACM ICML*, 2009, pp. 1177–1184.

[8] K. Winstein and H. Balakrishnan, "Tcp ex machina: Computer-generated congestion control," *ACM SIGCOMM Computer Communication Review*, vol. 43, no. 4, pp. 123–134, 2013.

[9] Y. Ma *et al.*, "Fairness-guaranteed transcoding task assignment for viewer-assisted crowdsourced livecast services," in *Proc. of IEEE ICC*, 2021, pp. 1–6.

[10] X. Li, N. Zhao, Y. Sun, and F. R. Yu, "Interference alignment based on antenna selection with imperfect channel state information in cognitive radio networks," *IEEE Transactions on Vehicular Technology*, vol. 65, no. 7, pp. 5497–5511, 2015.

[11] R. Kleinberg, "Nearly tight bounds for the continuum-armed bandit problem," *Proc. of NeurIPS*, vol. 17, 2004.

[12] A. Agarwal, S. Khanna, and P. Patil, "A sharp memory-regret trade-off for multi-pass streaming bandits," in *Proc. of PMLR COLT*, 2022, pp. 1423–1462.

[13] O. Avner and S. Mannor, "Multi-user communication networks: A coordinated multi-armed bandit approach," *IEEE/ACM Transactions on Networking*, vol. 27, no. 6, pp. 2192–2207, 2019.

[14] I. F. Akyildiz, W.-Y. Lee, M. C. Vuran, and S. Mohanty, "A survey on spectrum management in cognitive radio networks," *IEEE Communications magazine*, vol. 46, no. 4, pp. 40–48, 2008.

[15] A. Sacco, M. Flocco, F. Esposito, and G. Marchetto, "Owl: Congestion control with partially invisible networks via reinforcement learning," in *Proc. of IEEE INFOCOM*, 2021, pp. 1–10.

[16] S. Sawant, R. Kumar, M. K. Hanawal, and S. J. Darak, "Learning to coordinate in a decentralized cognitive radio network in presence of jammers," *IEEE Transactions on Mobile Computing*, vol. 19, no. 11, pp. 2640–2655, 2019.

[17] H. Jiang *et al.*, "When machine learning meets congestion control: A survey and comparison," *Computer Networks*, vol. 192, p. 108033, 2021.

[18] M. Dong *et al.*, "Pcc vivace: Online-learning congestion control," in *Proc. of USENIX NSDI*, 2018, pp. 343–356.

[19] P. Auer, N. Cesa-Bianchi, and P. Fischer, "Finite-time analysis of the multiarmed bandit problem," *Mach. Learn.,*, vol. 47, pp. 235–256, 2002.

[20] S. Agrawal and N. Goyal, "Analysis of thompson sampling for the multi-armed bandit problem," in *Proc. of PMLR COLT*, 2012, pp. 39.1–39.26.

[21] R. Xie, X. Jia, and K. Wu, "Adaptive online decision method for initial congestion window in 5g mobile edge computing using deep reinforcement learning," *IEEE Journal on Selected Areas in Communications*, vol. 38, no. 2, pp. 389–403, 2019.

[22] Y. Cao, Z. Wen, B. Kveton, and Y. Xie, "Nearly optimal adaptive procedure with change detection for piecewise-stationary bandit," in *Proc. of PMLR AISTATS*, 2019, pp. 418–427.

[23] R. Kleinberg, A. Slivkins, and E. Upfal, "Bandits and experts in metric spaces," *Journal of the ACM (JACM)*, vol. 66, no. 4, pp. 1–77, 2019.

[24] C. Podimata and A. Slivkins, "Adaptive discretization for adversarial lipschitz bandits," in *Proc. of PMLR COLT*, 2021, pp. 3788–3805.

[25] J. Wigmore, B. Shrader, and E. Modiano, "Hierarchical thompson sampling for multi-band radio channel selection," in *2023 IFIP Networking Conference (IFIP Networking)*. IEEE, 2023, pp. 1–9.

[26] O. Avner and S. Mannor, "Multi-user lax communications: A multi-armed bandit approach," in *Proc. of IEEE INFOCOM*, 2016, pp. 1–9.

[27] D. Raca, J. J. Quinlan, A. H. Zahran, and C. J. Sreenan, "Beyond throughput: a 4g lte dataset with channel and context metrics," ser. MMSys '18. Association for Computing Machinery, 2018.

[28] L. Lai, H. El Gamal, H. Jiang, and H. V. Poor, "Cognitive medium access: Exploration, exploitation, and competition," *IEEE Transactions on Mobile Computing*, vol. 10, no. 2, pp. 239–253, 2010.

[29] N. Cardwell *et al.*, "Bbr: Congestion-based congestion control," *Communications of the ACM*, vol. 60, no. 2, pp. 58–66, 2017.

[30] S. Hu *et al.*, "Aeolus: A building block for proactive transport in datacenters," in *Proc. of ACM SIGCOMM*, 2020, pp. 422–434.

[31] S. Floyd and V. Jacobson, "Random early detection gateways for congestion avoidance," *IEEE/ACM Transactions on Networking*, vol. 1, no. 4, pp. 397–413, 1993.

[32] Network simulator 3. [Online]. Available: https://www.nsnam.org/

[33] O. Avner and S. Mannor, "Concurrent bandits and cognitive radio networks," in *Proc. of Springer ECML-PKDD*, 2014, pp. 66–81.

[34] J. Ye, D. Lin, K. Cai, C. Zhou, J. He, and J. C. Lui, "Data-driven rate control for rdma networks: A lightweight online learning approach," in *2023 IEEE 43rd International Conference on Distributed Computing Systems (ICDCS)*. IEEE, 2023, pp. 1–11.

[35] T. Lattimore and C. Szepesvári, *Bandit algorithms*. Cambridge University Press, 2020.