Online Channel-state Clustering And Multiuser Capacity Learning For Wireless Scheduling

Isfar Tariq, Rajat Sen, Gustavo de Veciana and Sanjay Shakkottai
Department of Electrical and Computer Engineering
The University of Texas at Austin, USA
Email: isfartariq@gmail.com, rajat.sen@utexas.edu, gustavo@ece.utexas.edu, shakkott@mail.utexas.edu

***Abstract*—In this paper we propose an online algorithm for clustering channel-states and learning the associated achievable multiuser rates. Our motivation stems from the complexity of multiuser scheduling. For instance, MU-MIMO scheduling involves the selection of a user subset and associated rate selection each time-slot for varying channel states (the vector of quantized channels matrices for each of the users) — a complex integer optimization problem that is different for each channel state. Instead, our algorithm clusters the collection of channel states to a much lower dimension, and for each cluster provides achievable multiuser capacity trade-offs, which can be used for user and rate selection.**

Our algorithm uses a bandit approach, where it learns both the unknown partitions of the channel-state space (channel-state clustering) as well as the capacity region for each cluster along a pre-specified set of directions, by observing the success/failure of the scheduling decisions (e.g. through packet loss). We propose an epoch-greedy learning algorithm that achieves a sub-linear regret, given access to a class of classifying functions over the channel-state space. Finally, we empirically validate the performance of our algorithm through simulations.

***Index Terms*—Online Learning, Bandit Algorithms, Wireless Networks, Scheduling, Capacity Region**

I. Introduction

Wireless cellular networks have become increasingly more complex to operate – the aggregate number of parameters available for optimization at various layers can range in the thousands (e.g. MIMO antenna weights, power levels, coding and modulation rates, and frequency/sub-frame allocation to users), and the choice of which depend on the channel-states of the users[1]. Thus, when scheduling users (e.g. in MU-MIMO scheduling [1]), a *channel-state dependent* combinatorial optimization problem needs to be solved each time-slot, where a subset of users need to be selected, and transmission rates and power levels *jointly* determined for each of these users from among the allowable parameters. This problem however has a latent low-dimensionality that can be exploited, namely that for channel-states that are "near" each other, the optimal solution (user and rate selection) is likely to be the same. Thus, if we cluster channel-states, and determine the effective rate region trade-offs for each cluster, these cluster-dependent rate regions can be used for user and rate selection, and thus significantly reduce the complexity of user and rate scheduling.

[1]In a MIMO setting, the channel-state for *each* of the users is the channel H matrix, and in practice the base-station would have access to an approximation of this (e.g. quantized version).

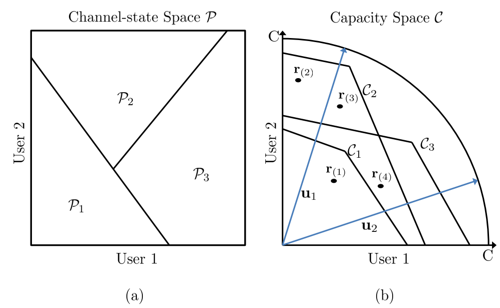


Fig. 1: An illustrative example of the channel-state space $\mathcal{P}$ and the corresponding capacity classes for $n = 2$ users, $K = 3$ capacity classes and $d = 1$. $\{\mathbf{r}_{(i)}\}_{i\in[4]}$ are different rate vectors that can be scheduled. The vectors $\{\mathbf{u}_i\}_{i\in[2]}$ correspond to the directions along which we need to maximize user rates.

However, these clusters are unlikely to be universal, meaning that different scenarios (e.g. indoor, outdoor urban, outdoor rural) would lead to different channel-state clusterings. Indeed, it is further likely that the clusters will also vary with the time of day depending on different loading/use-case scenarios. This then, motivates an *online* clustering and multi-user rate region learning approach, which is the focus of this paper.

Main Contributions: We consider a system where the channel-state space $\mathcal{P}$ clusters into K (unknown) classes, with a corresponding multiuser rate-region for each class. Our goal is to develop *online* strategies that can learn clusterings of different channel-states that have similar multiuser rate regions along with the boundaries of these regions. Simultaneously while learning, we need to schedule users based on the observed channel-states to maximize the user rates along pre-specified directions (see Figure 1(a) precise definition is given in Section III). Our contributions are:

(i) We propose an *epoch-greedy* bandit algorithm for our problem setting. The algorithm assumes access to a class of experts/classifying functions $\hat{\Pi}$, where an expert in $\hat{\Pi}$ is a mapping from the channel-state space to $\{0, 1\}$. We also assume that the class of experts is rich enough, such that there exists a set of functions, which when composed together can yield a function from the channel-state space to $\{1, 2, ..., K\}$

which correctly identifies the class in which each channel-state belongs in. Similar assumptions have been made in the realizable setting in stochastic contextual bandits [2]. Our approach achieves a balance between three objectives: (i) *Class Explore*- learning the clustering of the channel-state space using the class of experts and feedback obtained by scheduling different rates in an exploratory manner (ii) *Capacity Explore*- learning the boundaries of the capacity regions in the specified directions for the different channel-state region clusters (iii) *Exploit* - finally, exploiting the knowledge learned, by scheduling the rate vector of maximum possible magnitude in the specified direction, that lies within the capacity region corresponding to the channel-state observed in a time-slot.

(ii) We define a notion of *cumulative regret* for our problem. The *regret* in our setting is the difference between the total effective rate obtained by a learning policy in T time-slots and the total rate obtained by a *genie policy* which knows the capacity clusters and the corresponding capacity regions and given a channel-state, always schedules the rate vector of maximum possible magnitude in the specified direction, which lies within the capacity region corresponding to the channel-state. We provide a rigorous definition of regret for our problem in Equation (4). We analyze our algorithm and prove that it has a regret scaling of $\mathcal{O}(T^{2/3}\log T)$ at time T.

Finally, we circle back to one of our motivations – understanding the channel-state-dependent capacity regions. Note that since our algorithms focus on optimizing along a pre-specified set of directions, the resulting capacity region that can be constructed for each channel-state class will be an approximation (because we can potentially miss some of the faces of the capacity region). However, if the capacity regions are "nice", then the direction vectors can be designed in order to get an almost exact estimate of the capacity regions. For instance in [3], it has been shown that convex polytopes formed by the intersection R half-spaces (the hyper-planes should have rational coefficients) and for which the vertex enumeration problem is efficient [4], can be learned with $O(\text{poly}(R, d'))$ *noiseless* membership queries, where d' is the dimension of the space.

II. Related Work

Over the last few decades, there has been a lot of work on opportunistic scheduling for wireless networks. This has led to a powerful framework of algorithms that utilize channel feedback and the queue lengths to achieve objectives like system stability, optimization of a utility function or average delay [5]. In the setting of multi-user MIMO wireless networks (MU-MIMO), scheduling algorithms need to optimize over user selection, beamforming (antenna weight selection), power allocations, physical layer modulation and coding parameters [1], [6], [7]. Here, the user selection sub-problem (choosing a subset of users for transmission from among all the possible users) renders leads to a combinatorial explosion in complexity, and several approximations have been used as guidelines for complexity reduction [8]–[10].

We approach dimensionality reduction through online clustering, and our algorithmic approach is related to the contextual multi-armed bandit problem [11]–[13]. The stochastic contextual bandits with experts problem [2], [12], [14], [15] is especially relevant to our problem. This problem has been studied in the literature starting with the epoch-greedy policy in [12] leading to the more powerful and essentially statistically optimal policies in [2], [14], [15]. Our problem is somewhat similar to this setting as the channel-states observed is analogous to the context and the feedback received after scheduling a rate vector is similar to the stochastic reward observed after pulling an arm. We also assume access to a class of experts that map the space of channel-states to $\{1, 2, ..., K\}$, where K is the number of capacity classes. However, it should be noted that the feedback received in our setting is much more challenging, as it does not provide direct information about the capacity classes unlike the rewards received from the arms in contextual bandits, which directly reflects the utility of that arm under the given context. Moreover, in our problem there is an additional task of learning the boundary of the capacity regions, even after the clustering of the channel-state region into K classes has been learned.

In the context of learning the capacity regions, there is a line of related work on learning convex polytopes which are formed by the intersection of a finite number of half-spaces, from noiseless membership queries [3], [16]. In [3] binary search type strategies have been used to provide efficient algorithms for learning a class of convex polytopes that are formed by the intersection of half-spaces defined by hyper-planes with rational coefficients and for which the vertex enumeration problem can be solved efficiently.

III. System Model and Definitions

We consider a discrete time scheduling system with n users and a single scheduler. At each time t, the scheduler observes a channel-state vector $\mathbf{q}(t) = \{\mathbf{q}_1(t), \mathbf{q}_2(t), \ldots, \mathbf{q}_n(t)\}$ where $\mathbf{q}_i(t) \in \mathcal{Q}^d$ is the channel-state for user $i \in [n]$, where $[n] \triangleq \{1, 2, .., n\}$. The set $\mathcal{Q}$ can be a bounded subset of $\mathbb{R}$ or a discrete alphabet set. We denote the set of all channel-state vectors as $\mathcal{P}(= (\mathcal{Q}^d)^n)$. At any time t we observe the channel-state vector $\mathbf{q}$ from a time-invariant distribution $f_{\mathcal{Q}}$ over $\mathcal{P}$ (this distribution depends on the wireless channel between the user and the base-station).

Scheduling a rate vector: Corresponding to each channel-state, there is a unique capacity region that the system can support. The capacity region corresponding to a channel-state is defined as the set of all user rate vectors $\mathbf{r} \in \mathbb{R}_+^n$ that can be achieved with probability close to one, potentially by time-sharing. Strictly speaking, we are really considering the rate region, i.e., the set of user rate vectors that are achievable using the available physical layer strategies at the base-station (convex hull of the data rates that be generated using the available physical layer coding/modulation/antenna-beamforming choices), as opposed to an information-theoretic characterization. We however use the term capacity region instead of rate region for clarity of description.

In our subsequent discussion, when using the phrase "schedule a rate vector $\mathbf{r}$", it means that we notify the PHY/MAC parameter selection algorithm that $\mathbf{r}$ needs to be scheduled. Then, this algorithm tries to achieve the rate $\mathbf{r}$ potentially by time-sharing among various allowable physical layer rates, and over a block of several physical layer time-slots, in which the channel-state remains the same. Finally, at the end of this time-share block, a notification is received which tells us whether the requested rate $\mathbf{r}$ is achieved or not. Therefore, in the subsequent discussion we use 'time-slot' as an abstraction for one trial by the PHY/MAC parameter selection algorithm to achieve a rate over a block of physical layer time-slots. Note that we use a finite length block of physical layer time-slots to judge whether a rate $\mathbf{r}$ can be achieved and therefore the notification is bound to be noisy. This noise is captured in our *noise model* which is described later in this section.

Channel-state Partitions and Capacity Regions: We assume that the channel-state space $\mathcal{P}$ can be partitioned into K *sets* denoted by $\mathcal{P}_1, \mathcal{P}_2, \ldots, \mathcal{P}_K$ with their corresponding unique capacity regions $\mathcal{C}_1, \mathcal{C}_2, \ldots, \mathcal{C}_K$ respectively [2] such that for any $\mathbf{q} \in \mathcal{P}_i$, the capacity region is $\mathcal{C}_i$. In the case where $\mathcal{Q}$ is discrete and finite, it is reasonable to assume that $K \ll |\mathcal{P}| = |\mathcal{Q}|^{nd}$. The capacity regions $\mathcal{C}_1, \mathcal{C}_2, \ldots, \mathcal{C}_K \subseteq \mathcal{C} \subset \mathbb{R}^n$ are convex polytopes that lie in the positive quadrant. Further, for non-negative vectors $\mathbf{x}, \mathbf{y}$, if $\mathbf{x} \leq \mathbf{y}$ (element-wise) and $\mathbf{y} \in \mathcal{C}_i$, then $\mathbf{x} \in \mathcal{C}_i$ for any $i \in [K]$. We also assume that all the capacity regions lie inside the positive quadrant of the ball with radius C centered at the origin, i.e $\mathcal{C}_i \subset \mathcal{B}(0, C)^+$ for all $i \in [K]$. Here, $\mathcal{B}(0, x) = \{\mathbf{u} \in \mathbb{R}^n : \|\mathbf{u}\|_2 \leq x\}$ and $\mathcal{A}^+$ denotes the subset of $\mathcal{A}$ that lies in the positive quadrant.

We provide an illustrative example in Fig. 1, with $n = 2$ users and $K = 3$ capacity classes. In our example each user provides a one-dimensional channel-state vector, therefore the dimensions of both $\mathcal{P}$ and $\mathcal{C}$ are two. The partitions of the channel-state space $\mathcal{P}$ is shown in Fig. 1(a), which correspond to K different capacity regions in Fig. 1(b). We shall define an index function relating any channel-state vector $\mathbf{q} \in \mathcal{P}$ to the channel-state partition and the capacity region as follows.

Definition 1 (Index function $\mathcal{I}(.)$)**.** *Given a channel-state vector* $\mathbf{q}$*,* $\mathcal{I}(\mathbf{q})$ *is the index of the element of the partition that contains* $\mathbf{q}$*, i.e.* $\mathbf{q} \in \mathcal{P}_{\mathcal{I}(\mathbf{q})}$.

The following assumption states that channel-state's from each element of the partition are observed sufficiently often.

Assumption 1 (Class Probabilities)**.** *We assume that* $\mathbb{P}(\mathcal{I}(\mathbf{Q}) = i) > \beta \ = O\left(\frac{1}{K}\right) \ \forall \ i \in [K]$*, where* $\mathbf{Q} \in \mathcal{P}$ *is a random variable with distribution* f_Q *capturing variability in the system.*

[2]Our theoretical guarantees require the channel-state regions corresponding to the different capacity regions be disjoint, however our algorithm can also handle cases where the channel-state classes are not disjoint.

Separation of Capacity Regions: We assume that the capacity regions are sufficiently different from each other. For instance, in Fig. 1 if $\mathcal{C}_1$ and $\mathcal{C}_2$ were almost identical to each other, then it would be better to merge $\mathcal{P}_1, \mathcal{P}_2$ and treat it as a system with $K = 2$. The following assumption says that for any two capacity regions $\mathcal{C}_i, \mathcal{C}_j$ a sufficient fraction of the volume lies outside of their intersection.

Assumption 2 (Separability)**.** *We assume the capacity regions are well separated, i.e., for all* $i, j \in [K]$

$$d(\mathcal{C}_i, \mathcal{C}_j) \triangleq \frac{|(\mathcal{C}_i \setminus \mathcal{C}_j) \cup (\mathcal{C}_j \setminus \mathcal{C}_i)|}{|\mathcal{B}(0, C)^+|} \geq \lambda > 0,$$

where $|\mathcal{A}|$ *denotes the volume of the set* $\mathcal{A}$*.*

Noise Model: Let $Y(\mathbf{q}, \mathbf{r}) \in \{0, 1\}$ denote a random variable modeling the observed notification when a rate $\mathbf{r} \in \mathbb{R}^n$ is scheduled when the observed channel-state vector is $\mathbf{q}$. Here, $Y(\mathbf{q}, \mathbf{r}) = 1$ signifies a successful transmission and $Y(\mathbf{q}, \mathbf{r}) = 0$ signifies a failure to achieve that rate vector . The success or failure to transmit a rate vector $\mathbf{r}$ under channel-state $\mathbf{q}$ is assumed to be an i.i.d random variable $Y(\mathbf{q}, \mathbf{r})$ with distribution given by

$$\mathbb{P}(Y(\mathbf{q}, \mathbf{r}) = 1) = \begin{cases} 1 - \rho(\mathbf{q}, \mathbf{r}), & \text{if } \mathbf{r} \in \mathcal{C}_{\mathcal{I}(\mathbf{q})}, \\ \rho(\mathbf{q}, \mathbf{r}), & \text{if } \mathbf{r} \in \mathcal{B}(0, C)^+ \setminus \mathcal{C}_{\mathcal{I}(\mathbf{q})}, \\ 0, & \text{otherwise.} \end{cases}$$

where $\rho(\mathbf{q}, \mathbf{r})$ can be viewed as a noise parameter (essentially the packet error rate) which depends on the channel-state $\mathbf{q}$ and the rate $\mathbf{r}$.

Assumption 3 (Noise Rate)**.** *We assume that* $\rho(\mathbf{q}, \mathbf{r}) \leq \rho < 1/8, \ \forall \ \mathbf{q}, \mathbf{r}$*. We further assume that for all* $\mathbf{p}, \mathbf{q}$ *and* i *such that* $\mathbf{p}, \mathbf{q} \in \mathcal{P}_i$*,* $\rho(\mathbf{p}, \mathbf{r}) = \rho(\mathbf{q}, \mathbf{r})$*. For notational convenience, for all* $\mathbf{q} \in \mathcal{P}_i$*, let* $\rho_i(\mathbf{r}) \triangleq \rho(\mathbf{q}, \mathbf{r}) = \rho_{\mathcal{I}(q)}(\mathbf{r})$*.*

Given a channel-state and the corresponding capacity region, when $\mathbf{r}$ approaches the boundary of the capacity region (from inside) the probability of successful transmission is close to 1 but decreases slightly near the boundary. The success probability drops significantly after $\mathbf{r}$ crosses the boundary (there is a discontinuous jump in success probability at the boundary). After crossing the boundary, $\rho(\mathbf{q}, \mathbf{r})$ decreases till $|\mathbf{r}| = C$, beyond which $\rho(\mathbf{q}, \mathbf{r}) = 0$.

Bandit Feedback and Objectives: Let $\mathcal{U} = \{\mathbf{u}_1, ..., \mathbf{u}_D\}$ be a set of unit vectors such that $\mathbf{u}_i \in \mathbb{R}^n_+$. This set is fixed a priori. The broad objective is to discover the maximum possible service rates in these directions, given a particular channel-state. Since we use 'time-slot' as an abstraction for a block of several physical layer time-slots where a rate vector $\mathbf{r}$ is attempted to be scheduled potentially by time-sharing. Therefore, a wide-range of direction vectors within the capacity region can be supported.

Concurrently with the channel-state, a direction vector $\mathbf{u}$ is chosen uniformly at random from the set $\mathcal{U}$. The task is to schedule a rate vector within the capacity region $\mathcal{C}_{\mathcal{I}(\mathbf{q})}$, of maximum possible magnitude in the direction $\mathbf{u}$. In other words, we would ideally like to schedule a rate vector $c\mathbf{u}$ such that

$$c(\mathbf{q}) = \arg\max_{d}\{d|d\mathbf{u} \in \mathcal{C}_{\mathcal{I}(\mathbf{q})}\}.$$

The precise order of events at a given time-step is as follows:

- A channel-state $\mathbf{q}(t)$ from the distribution $f_{\mathcal{Q}}$ is observed. A direction $\mathbf{u}(t)$ drawn uniformly at random from $\mathcal{U}$ is also specified.
- The policy optionally selects a magnitude $c(\mathbf{q}(t), \mathbf{u}(t)) \in [0, C]$ to be scheduled in the direction $\mathbf{u}(t)$ and the rate vector $\mathbf{r}(t) = c(\mathbf{q}(t), \mathbf{u}(t))\mathbf{u}(t)$ is scheduled. On the other hand, the policy may choose any other rate vector $\mathbf{r}(t)$ that does not lie in the specified direction. In this case the reward obtained is zero in the time-step[3].
- A notification $Y(\mathbf{q}(t), \mathbf{r}(t)) \in \{0, 1\}$ is then observed.

Expected Reward Function: Recall $Y(\mathbf{q}, \mathbf{r})$ is the notification received for transmitting rate vector $\mathbf{r}$ when the observed channel-state was $\mathbf{q}$. Let us define the reward $r(\mathbf{q}, \mathbf{r}, \mathbf{u})$ for a rate vector $\mathbf{r}$, channel-state $\mathbf{q}$ and direction vector $\mathbf{u}$ to be

$$r(\mathbf{q}, \mathbf{r}, \mathbf{u}) = |\mathbf{r}| \mathbb{1}\{\mathbf{r}.\mathbf{u} = |\mathbf{r}|\} Y(\mathbf{q}, \mathbf{r}), \tag{1}$$

where $\mathbb{1}\{\}$ is the indicator function.

Note that for any $\mathbf{p}, \mathbf{q} \in \mathcal{P}_i$, we have $\mathbb{E}[r(\mathbf{q}, \mathbf{r}, \mathbf{u})] = \mathbb{E}[r(\mathbf{p}, \mathbf{r}, \mathbf{u})] \triangleq \mathbb{E}[r_i(\mathbf{r}, \mathbf{u})]$. Therefore, we define the expected reward function $f_{\mathbf{u},i}(c)$ for direction vector $\mathbf{u}$, capacity region $\mathcal{C}_i$ and magnitude c, as follows:

$$f_{\mathbf{u},i}(c) = \mathbb{E}[r_i(c\mathbf{u}, \mathbf{u})]. \tag{2}$$

The function $f_{\mathbf{u},i}(c)$ is the expected rate achieved if we schedule a rate vector $c\mathbf{u}$ when the channel-state observed belongs to capacity class i. It can be evaluated as follows,

$$f_{\mathbf{u},i}(c) = \begin{cases} c(1 - \rho_i(c\mathbf{u})), & \text{if } c\mathbf{u} \in \mathcal{C}_i, \\ c\rho_i(c\mathbf{u}), & \text{if } c\mathbf{u} \in \mathcal{B}(0, C)^+ \setminus \mathcal{C}_i, \\ 0, & \text{otherwise.} \end{cases} \tag{3}$$

Since we have assumed that $\rho_i(\mathbf{r}) < \frac{1}{8} \; \forall \; \mathbf{r}$ therefore $f_{\mathbf{u},i}(c)$ is a discontinuous function of c, and the discontinuity is located at the point where a ray in the direction $\mathbf{u}$ meets the boundary of $\mathcal{C}_i$. We make the following assumption on the expected rate function.

Assumption 4 (Maxima of Rate Function)**.** *Let us define*

$$\hat{c}_{\mathbf{u},i} = \arg\max_{c} f_{\mathbf{u},i}(c)$$

and $c^*_{\mathbf{u},i} = \max_c\{c|c\mathbf{u} \in \mathcal{C}_i\}$. *We assume that the noise function* $\rho_i(\mathbf{r})$ *is such that* $c^*_{\mathbf{u},i} = \hat{c}_{\mathbf{u},i}$.

This assumption basically implies that for all $i \in [K]$ the maximum of the rate function $f_{\mathbf{u},i}(c)$ is achieved at the point where a ray in the direction $\mathbf{u}$ meets the boundary of $\mathcal{C}_i$.

Class of Experts: We assume access to a class of binary experts/classifiers $\hat{\Pi}$, where each expert $\hat{\pi} \in \hat{\Pi}$ is a function mapping the space of channel-states to $\{0, 1\}$ i.e $\hat{\pi} : \mathcal{P} \to \{0, 1\}$.

Assumption 5 (Classifying Functions)**.** *Let* κ *be a proper subset of* $[K]$. *Let us define the following binary function* $\hat{\mathcal{I}}_\kappa(\mathbf{q}) = \sum_{i\in\kappa} \mathbb{1}\{\mathbf{q} \in \mathcal{P}_i\}$. *We assume that the set of binary experts/classifiers* $\hat{\Pi}$ *is such that for all* $\kappa \subset [K]$, $\hat{\mathcal{I}}_\kappa(.) \in \hat{\Pi}$. *We further assume that the VC dimension [17] of our class of experts is* V.

The above assumption states that the binary functions from $\mathcal{P}$ to $\{0, 1\}$ that are induced by labeling the channel-state's belonging to a set $\kappa \subset [K]$ of capacity classes as 1 and the rest as 0, are a part of our class of experts, for all such proper subsets κ. Consider the example exhibited in Figure 1. Suppose $\kappa = \{1, 2\}$. Then, $\hat{\mathcal{I}}_\kappa(\mathbf{q})$ divides $\mathcal{P}$ into two regions $\mathcal{P}_1 \cup \mathcal{P}_2$ and $\mathcal{P}_3$. Note that both these regions can be represented as the intersection of at most two half-spaces, as the boundaries of the partitions are linear. This is true for all such proper subsets κ. Therefore, if our class of binary classifiers contains all the separators that are intersections of at most two half-spaces, then Assumption 5 is valid.

Note that $\hat{\mathcal{I}}_\kappa(\mathbf{q})$ for different κ's can be composed together to recover $\mathcal{I}(\mathbf{q})$. In the example in Figure 1, $\hat{\mathcal{I}}_{[2,3]}(\mathbf{q})$ differentiates class 1 from 2, 3 and $\hat{\mathcal{I}}_{[3]}(\mathbf{q})$ separates class 3 from the rest. Given a channel-state $\mathbf{q}$, if for instance $\hat{\mathcal{I}}_{[3]}(\mathbf{q}) = 0$ and $\hat{\mathcal{I}}_{[2,3]}(\mathbf{q}) = 1$ then we can infer that $\mathcal{I}(\mathbf{q}) = 2$. Therefore, Assumption 5 basically implies that there exists a group of binary functions in $\hat{\Pi}$, which when composed together can yield the true index function. Note that this is similar to the *realizable setting* in the contextual bandits with experts problem [2], where it is assumed that the true behavior of the system can be represented by one of the expert function. However, finding the correct expert in an online setting is an algorithmic challenge.

Definition of Regret: The main objective is to minimize regret when compared to a genie strategy which knows the index function $\mathcal{I}$ and the capacity regions $\mathcal{C}_i$'s. Let $\mathbf{r}(t)$ be the rate vector selected by a policy, at time t. Then the regret of the policy till time T is given by:

$$R(T) = \sum_{t=1}^{T} \big(f_{\mathbf{u}(t),\mathcal{I}(\mathbf{q}(t))}(\hat{c}_{\mathbf{u}(t),\mathcal{I}(\mathbf{q}(t))}) - \mathbb{E}\left[r_{\mathcal{I}(\mathbf{q}(t))}(\mathbf{r}(t), \mathbf{u}(t)) \right] \big) \tag{4}$$

where $\mathbf{q}(t)$,$\mathbf{u}(t)$ are the channel-state vector and direction vector at time t, respectively. Note, that $f_{\mathbf{u}(t),\mathcal{I}(\mathbf{q}(t))}(\hat{c}_{\mathbf{u}(t),\mathcal{I}(\mathbf{q}(t))})$ is the maximum average rate that can be achieved in the

[3]Note that this is a conservative estimate of the reward. In general, there is some non-zero value in scheduling any rate vector in the capacity region corresponding to the observed channel-state. However, our theoretical guarantees will be under this conservative reward model, and in practice the performance observed will only be better.

direction $\mathbf{u}(t)$, by a *genie policy* that knows the capacity classes and the boundaries of the capacity regions. The regret measures the sub-optimality of the policy in question with respect to the *genie policy*, in an expected sense. The goal is to design a policy that yields $R(T)$ that is sub-linear in T, for all times T large enough. This basically implies that the policy keeps learning the system as time progresses.

IV. Algorithm

The algorithm is structured as an *epoch-greedy* strategy [12]. One key algorithmic idea is that if a rate vector $\mathbf{r}$ is scheduled for several different observed channel-state's $\mathbf{q}$, then the success notifications $Y(\mathbf{q}, \mathbf{r})$ provide useful information that can be leveraged using the class of binary experts $\hat{\Pi}$ to obtain a binary classifier that separates the channel-state space $\mathcal{P}$ into two regions $\mathcal{P}_*$ and $\mathcal{P}_*^c$, where $\mathcal{P}_* = \{\mathbf{q} \in \mathcal{P} : \mathbf{r} \in \mathcal{C}_{\mathcal{I}(\mathbf{q})}\}$. A carefully chosen set of rate points can then be used to form a group of binary classifying functions, which when composed together yields a mapping $\pi : \mathcal{P} \to [K]$, which is identical to $\mathcal{I}(\mathbf{q})$ with high probability.

The algorithm starts with an initialization phase and then proceeds in *epochs*. In initialization phase the algorithm constructs π by building a tree of binary classifiers which is then used to classify the channel-state points into K different classes. This stage is referred to as *initializing classifier* π. After building π, the algorithm runs in epochs similar to epoch-greedy policies for contextual bandits. At the beginning of each epoch, there is a *class explore* stage corresponding to improving the accuracy of classifier π. This is followed by a *capacity explore* stage aimed at learning the capacity regions of the K different channel-state partitions, in the given directions $\mathcal{U}$. The last stage in an epoch is the *exploitation* stage where we deduce the correct capacity class of the observed channel-state vector using π and then schedule the optimal rate vector according to the current belief about the boundary of the corresponding capacity region. An illustrative pseudo-code of our algorithm is shown in Algorithm 1, while a more detailed pseudo-code can be found as Algorithm 4. We will explain each of the stages/phases in more detail in subsequent sections.

A. Initializing Classifier π

The first stage of the algorithm is to initialize the mapping (multi-class classifier) π used to classify the different channel-state's into the K different classes. This mapping consists of $K - 1$ binary experts from our class of experts, which are composed together in a *tree-like* structure, in order to yield the mapping π.

The detailed pseudo-code for this phase is provided as Algorithm 2. In the beginning of this phase, for several time-slots the channel-state's are observed and stored, while not making any scheduling decisions (for instance, the scheduler is allowed to proceed in its default behavior). This process is continued until we observe n_0 distinct channel-state vectors, which are essentially n_0 distinct i.i.d random variables sampled from f_Q.

Algorithm 1 Epoch-greedy algorithm for online capacity class learning and rate allocation

Initialize classifier π, by observing t_0 channel-state's, scheduling corresponding carefully designed rate vectors and observing the notifications. ***(Initializing Classifier)***
Epoch: $l = 1$. Time: $t = t_0$.
while $t \leq T$ **do**
  Update the classifier π by observing channel-state $\mathbf{q}$, scheduling a carefully chosen rate point $\mathbf{r}$, and using the notification $Y(\mathbf{q}, \mathbf{r})$. This is repeated $K - 1$ times. ***(Class explore)***
  Learn the boundaries of the K capacity regions in the directions $\mathcal{U}$, by scheduling carefully chosen rate points and using the current π. A total of $\alpha(l)$ rate points are scheduled in this part of the epoch. ***(Capacity explore)***
  Schedule next $s(l)$ rate points optimally using π and the learned boundaries. ***(Exploit)***
  Let $t = t + K - 1 + \alpha(l) + s(l)$ and $l = l + 1$.
end while

Then we begin initializing the tree-structure which is detailed in steps 2-20 of Algorithm 2. Note that in each iteration of the while loop in step 2 of Algorithm 2, a rate point is randomly selected and then for the following l_0 time-slots irrespective of the channel-state observed, this rate point is scheduled. The feedback observed helps us in building a binary classification data-set that can be used to train a classifier $\hat{\pi} \in \hat{\Pi}$ which can differentiate all $\mathbf{q} \in \mathcal{P}$ such that $\mathbf{r} \in \mathcal{C}_{\mathcal{I}_\mathbf{q}}$ from the rest. We assume that the classifiers are trained in step 11 using empirical risk minimization (ERM) with the $0 - 1$ loss function. Therefore, we have that:

$$\hat{\pi}_i = \underset{\hat{\pi} \in \hat{\Pi}}{\operatorname{argmin}} \frac{1}{|S_i|} \sum_{(\mathbf{q}, y) \in S_i} \mathbb{1}\{\hat{\pi}(\mathbf{q}) \neq y\}.$$

At any point in time, an internal node $\mathcal{N}_i$ in the tree stores the triplet $(\hat{\pi}_i, \mathbf{r_i}, S_i)$ where $\hat{\pi}_i$ is the expert obtained by ERM over the examples $\{(\mathbf{q}, y)\}$ stored in S_i which were in turn obtained by scheduling the rate $\mathbf{r}_i$ for l_0 time-slots. A leaf of the tree $\mathcal{L}_i$ stores a subset of the initial n_0 channel-state points. In each iteration of the while loop, the classifier trained using the data collected by scheduling the current randomly chosen rate point, is only retained if it can split at least one of the current leaf nodes in the tree reliably into two distinct partitions. This is achieved by the check in step 14 of Algorithm 2. The while loop continues to iterate until the tree has $K - 1$ internal nodes.

In order to illustrate this phase, let us consider a system as shown in Figure 1 with $r = 2$ users, such that the channel-states can be partitioned into $K = 3$ classes $\mathcal{P}_1$, $\mathcal{P}_2$ and $\mathcal{P}_3$ with capacity regions $\mathcal{C}_1$, $\mathcal{C}_2$ and $\mathcal{C}_3$ respectively.

For, simplicity let $\mathbf{r}_{(1)}, .., \mathbf{r}_{(4)}$ be the first four rate points that are randomly chosen in step 3 of Algorithm 2, in that order (see Fig. 1). Since, $\mathbf{r}_{(1)}$ is a rate point that lies in all the capacity regions, the corresponding classifier $\hat{\pi}_1$ formed

Algorithm 2 Initializing the Classifier Tree

1: Schedule arbitrary rate vectors for the first n_0 channel-state vectors observed. Let $i = 1$ and form a tree $\mathcal{T}$ where the root contains the n_0 initial channel-state points. There are no other nodes in the tree. Set $i = 1$.
2: **while** $i < K - 1$ **do**
3:     Randomly select a rate point $\mathbf{r}$.
4:     $S_i = \{\}$
5:     **for** $l = 1 : l_0$ **do**
6:         Let $\mathbf{q}$ be the observed channel-state at time-step t.
7:         Schedule rate $\mathbf{r}$. *(Class Explore)*
8:         Let $y \in \{0, 1\}$ be the notification received. Add $(\mathbf{q}, y)$ to S_i.
9:         Set $t = t + 1$.
10:     **end for**
11:     Construct a binary classifier $\hat{\pi}_i$ by empirical risk minimization (ERM) over S_i, over the expert set $\hat{\Pi}$.
12:     **for** all leaves j of $\mathcal{T}$ **do**
13:         Classify the channel-state at leaf j according to the classifier $\hat{\pi}_i$. Let n_j be the number of channel-state points at leaf j.
14:         **if** $\frac{n_0\beta}{2} <$ number of leaf channel-state classified as $0 < n_j - \frac{n_0\beta}{2}$ **then**
15:             Make leaf j into a parent of two new leaves where the left leaf has all the channel-state's classified as 1 and the right has all the channel-state's classified as 0.
16:             $i = i + 1$
17:             Break
18:         **end if**
19:     **end for**
20: **end while**

using that data collected in step 8, will classify most of the n_0 channel-state points as 1. Therefore, this will not split the current leaf node (the root node with n_0 initial channel-state vectors) into any partitions. Hence, the classifier and the rate point is discarded and the value of the iterator i remains unchanged. The tree remains the same with one leaf node as shown in Fig. 2(a)-(b).

In the next iteration of the while loop, the randomly chosen rate point is $\mathbf{r}_{(2)}$. The data collected using $\mathbf{r}_{(2)}$ is used to train a classifier $\hat{\pi}_1$, which classifies most points in class $\mathcal{P}_2$ as 1, while classifying most points outside of $\mathcal{P}_2$ as zero [4]. This point splits the n_0 channel-state points in the current leaf node into two partitions. Therefore, the classifier is retained. An internal node $\mathcal{N}_1 = \{\mathbf{r}_1, \hat{\pi}_1, S_1\}$ is formed where $\mathbf{r}_1 = \mathbf{r}_{(2)}$. Moreover, two leaf nodes are formed where $\mathcal{L}_1$ is a leaf corresponding to all the n_0 channel-state vectors that are labeled as 1 by $\hat{\pi}_1$ and $\mathcal{L}_2$ contains the rest. This is illustrated in Fig. 2(c).

[4]Note that this is just an initialization of the classifier and moreover the feedback received from scheduling is noisy. Therefore, the binary classifiers trained will not be fully accurate. However, n_0 and l_0 are designed to be large enough such that with high probability the tree structure is correct.

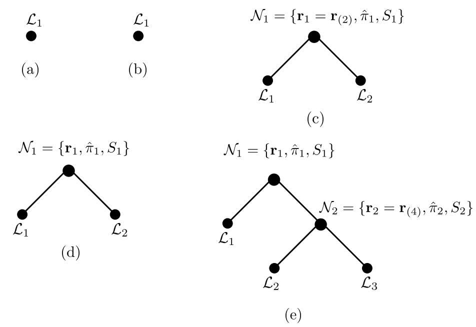


Fig. 2: Construction of a classification tree which represents the final initial classifier π that maps $\mathcal{P} \to [K]$ corresponding to channel-state class structure in Fig. 1.

In the next iteration, the rate point $\mathbf{r}_{(3)}$ is chosen, which will effectively yield the same classifier as the one corresponding to $\mathbf{r}_{(2)}$. Therefore, this classifier will be insufficient to split any of the leaves in Fig. 2(c). Thus the value of i is unchanged and the tree remains the same as shown in Fig. 2(d).

Finally, the rate point $\mathbf{r}_{(4)}$ is chosen. The classifier $\hat{\pi}_2$ corresponding to this point ideally distinguishes between points lying in $\mathcal{P}_1$ from those outside of $\mathcal{P}_1$. Thus, this new classifier can split the points in leaf $\mathcal{L}_2$ of the tree in Fig. 2(c), into two nodes, as shown in Fig. 2(d). This leads us to our final classifying tree π. Ideally (ignoring classification errors), a channel-state point belonging to $\mathcal{P}_1$,$\mathcal{P}_2$ and $\mathcal{P}_3$ will land in $\mathcal{L}_3$, $\mathcal{L}_1$ and $\mathcal{L}_2$ respectively.

The parameters n_0, l_0 have been chosen in order to ensure that w.h.p a correct classifying tree is obtained. The following lemma formalizes this claim.

Lemma 1. *Let* $n_0 \geq \frac{24K}{\beta^2}\log\left(\frac{2\log(\frac{1}{\delta})+K}{\delta\lambda}\right)$ *and* l_0 *is large enough such that* $\frac{1}{1-2\rho}\sqrt{\frac{V}{l_0}} + \sqrt{\frac{2\log\left(\frac{l_0^2}{\delta}\right)}{l_0}} < \frac{\beta}{4K}$ *and* $l_0 > \sqrt{\left(\frac{2\log(\frac{1}{\delta})+K-1}{K\lambda}\right)}$. *Then with probability at least* $1 - 3K\delta$, *the loop in step* 2 *of Algorithm* 2 *is terminated after at most* $\frac{2\log(\frac{1}{\delta})+K-1}{\lambda}$ *iterations and further a correct classifying tree structure is obtained.*

B. Class explore

After the classification tree is initialized, the algorithm proceeds in epochs and the structure of the tree remains unchanged. The first few time-slots in each epoch are dedicated to improving the accuracy of the classifiers $\hat{\pi}_i$'s stored in the internal nodes of the tree $\mathcal{N}_i$'s. We name this part of an epoch *class explore*. The *class explore* phase in an epoch consists of $K-1$ time-steps $t_1, ..., t_{K-1}$. At time-step t_i, let the channel-state observed be $\mathbf{q}_i$. After the channel-state is observed, the rate vector $\mathbf{r}_i$ stored in the internal node $\mathcal{N}_i$ is scheduled and a notification y_i is received The data-sample $(\mathbf{q}_i, y_i)$ is added to the set S_i and $\hat{\pi}_i$ is updated through ERM over the updated set S_i. This is performed for all $i = 1, 2, ..., K-1$. This phase is detailed in steps 7 -14 of Algorithm 4. The basic idea is to obtain one more training sample for each of the classifiers stored in the internal nodes, at the beginning of each epoch, thereby improving the classification accuracy of the global classier $\pi : \mathcal{P} \to [K]$. The following lemma provides an upper bound for the classification error of the global classifier $\hat{\pi}$ at the beginning of epoch l.

Lemma 2. *At the end of the class explore phase in epoch l with probability at least $1-(K-1)\frac{\delta}{(l+l_0)^2}$ we have*

$$
\begin{aligned}
&\mathbb{P}(\pi(\mathbf{Q}) \neq \mathcal{I}(\mathbf{Q})) \\
\leq &(K-1)\left(\left(\frac{1}{1-2\rho}\right)\sqrt{\frac{V}{l_0+l}}+\sqrt{\frac{2\log\left(\frac{(l_0+l)^2}{\delta}\right)}{l_0+l}}\right) \\
\triangleq &(K-1)\epsilon(l_0+l,\delta),
\end{aligned}
$$

where the probability is over the randomness in $\mathbf{Q} \sim f_{\mathcal{Q}}$ and the randomness in π due to the random samples in the training set.

C. Capacity explore

In each epoch, the class explore phase is followed by a few time-slots dedicated to *capacity explore*. This phase is described as steps 16-22 in Algorithm 4. It is aimed towards learning the boundaries of the K capacity classes in the directions $\mathcal{U}$. Note that there are K possible capacity classes and $D = |\mathcal{U}|$ direction vectors to explore. In the capacity explore phase of epoch l, for $\alpha(l,\delta)$ time-slots we observe the channel-state vectors, direction vectors and schedule carefully designed rate vectors to learn the capacity region. We set $\alpha(l,\delta) = \frac{2D}{\beta}\left(\frac{16}{1-2\rho}\right)^2 \log\left(\frac{l^2}{\delta}\right)$.

We initialize $C_{k,\mathbf{u}}[0] = 0$ and $C_{k,\mathbf{u}}[1] = C$ for all $k \in [K]$ and $\mathbf{u} \in \mathcal{U}$ at the start of the algorithm. $C_{k,\mathbf{u}}[0]$ is a lower bound for $c^*_{\mathbf{u},i}$ and $C_{k,\mathbf{u}}[1]$ is an upper bound for $c^*_{\mathbf{u},i}$, and these values are updated after the capacity explore phase in every epoch.

Algorithm 3 Capacity explore update

for $\forall\ k \in [K]$ and $\mathbf{u} \in \mathcal{U}$ **do**
  if $m_{k,\mathbf{u}} > \frac{1}{2}$ **then**
    $C_{k,\mathbf{u}}[0] = \frac{C_{k,\mathbf{u}}[0]+C_{k,\mathbf{u}}[1]}{2}$
  else if $m_{k,\mathbf{u}} < \frac{1}{2}$ **then**
    $C_{k,\mathbf{u}}[1] = \frac{C_{k,\mathbf{u}}[0]+C_{k,\mathbf{u}}[1]}{2}$
  end if
end for

Let $\tau_{l,k,\mathbf{u}}$ be the set of time-slots in which the channel-state $\mathbf{q}$ observed is such that $\pi(\mathbf{q}) = k$ and the direction vector specified is $\mathbf{u}$, in the capacity explore phase of epoch l. In all these time-slots, the rate $\frac{C_{k,\mathbf{u}}[0]+C_{k,\mathbf{u}}[1]}{2}\mathbf{u}$ is scheduled. $m_{k,\mathbf{u}}$ denotes the empirical mean of the success rates in scheduling the above rate vectors. The lower and upper bounds $C_{k,\mathbf{u}}[0]$ and $C_{k,\mathbf{u}}[1]$ are then updated depending on the value of $m_{k,\mathbf{u}}$ for all $k, \mathbf{u}$. The update procedure is detailed in Algorithm 3, which is similar to a traditional binary search procedure for searching the boundary of the capacity regions in the given directions $\mathcal{U}$ (see also [3]).

Algorithm 4 Online rate allocation from channel-state and service data

1: Initialize empty sets $S_i = \{\}$ for $i \in [K]$.
2: Initialize a single node tree $\mathcal{T}$ where the node contains n_0 different channel-state points.
3: Initialize capacity rate $C_{k,\mathbf{u}}[0] = 0$ and $C_{k,\mathbf{u}}[1] = C$ for all $k \in [K]$ and $\mathbf{u} \in \mathcal{U}$.
4: Initialize classifier π using Algorithm 2.
5: Set $t = t_0$ (time index) and $l = 1$ (epoch index).
6: **while** $t \leq T$ **do**
7:   **for** $i = 0 : K-1$ **do**
8:     $\mathbf{r}_i$ is the rate vector stored in node $\mathcal{N}_i$.
9:     Let $\mathbf{q}$ be the channel-state observed at time step t.
10:     Schedule rate $\mathbf{r}_i$. *(Class Explore)*
11:     Let $y \in \{0,1\}$ be the notification received. Add $(\mathbf{q}, y)$ to S_i.
12:     Set $t = t+1$.
13:     Update the classifier $\hat{\pi}_i$ in $\mathcal{N}_i$.
14:   **end for**
15:   Let the empirical means of success rate be $m_{k,\mathbf{u}} = 0$ for all $k \in [K]$ and $\mathbf{u} \in \mathcal{U}$.
16:   **for** $s = 1 : \alpha(\delta, l)$ **do**
17:     Observe $(\mathbf{q}, \mathbf{u})$.
18:     Let $k = \pi(\mathbf{q})$.
19:     Schedule rate vector $\left(\frac{C_{k,\mathbf{u}}[0]+C_{k,\mathbf{u}}[1]}{2}\right)\mathbf{u}$. *(Capacity Explore)*
20:     Update $m_{k,\mathbf{u}}$ according to received notification y.
21:     Set $t = t+1$.
22:   **end for**
23:   Update C and $\hat{S}$ according to Algorithm 3.
24:   **for** $s = 1 : s(l)$ **do**
25:     Observe $(\mathbf{q}, \mathbf{u})$.
26:     Let $k = \pi(\mathbf{q})$.
27:     Schedule rate vector $C_{k,\mathbf{u}}[0]\mathbf{u}$. *(Exploit)*
28:     Let $t = t+1$.
29:   **end for**
30:   $l = l+1$.
31: **end while**

D. Exploitation

In every epoch, after the *exploration* phases, the overwhelming majority of time-slots are dedicated to *exploitation*. The *exploitation phase* in epoch l consists of $s(l) = O(\sqrt{l})$ time-slots. In each of these time-slots, a channel-state $\mathbf{q}$ is observed and a direction vector $\mathbf{u}$ is specified. The class $k = \pi(\mathbf{q})$ is identified according to our current global classifier and the rate vector $C_{k,\mathbf{u}}[0]\mathbf{u}$ is scheduled. This phase is detailed as steps 24 - 29 in Algorithm 4.

Remark 1. *Algorithm 4 satisfies our regret bound in Theorem 1. However, there are few low-probability failure events that can affect the working of the algorithm in all future time-*

steps. For instance, the initial classifier tree-structure may be incorrect, which happens with low probability as shown in Lemma 1. Moreover, at any epoch the binary search can take an incorrect decision, which can also happen with a very low-probability. We can generalize the discussion to a more robust algorithm that can detect such low-probability failure states and correct them online. Thus, we provide a more robust version of our algorithm in Appendix A. In our simulations in Section VI we use the robust version of the algorithm.

V. Regret Bound

In this section, we provide our main theoretical result which provides a cumulative regret bound for Algorithm 4, when Assumptions 1-5 are satisfied.

Theorem 1. *Under Assumptions 1-5, with probability at least $1 - O(KD\delta)$, Algorithm 4 achieves a regret bound of,*

$$R(T) = \mathcal{O}\left(T^{2/3}\log\left(\frac{1}{\delta}\right)\left(D\log T + K + \sqrt{V}\right)\right),$$

at time T.

Theorem 1 has been stated in more detail in the Appendix F, where the explicit dependence on the various problem parameters has been specified.

Discussion: Theorem 1 states that the regret of Algorithm 4 scales as $\mathcal{O}(T^{2/3}\log T)$ as a function of time. The scaling is linear with respect to the number of classes K and the number of direction vectors D. It scales as $\sqrt{V}$ in terms of the VC dimension of the class of experts. For a finite class of experts $\hat{\Pi}$, the VC dimensions is $\mathcal{O}(\log N)$, where $N = |\hat{\Pi}|$ is the number of experts.

It should be noted that *epoch-greedy* algorithms in bandit settings generally have a regret scaling of $O(T^{2/3})$ in the problem independent setting, because of explicit exploration. For instance, the epoch-greedy strategy in [12] has a similar regret scaling for the problem of stochastic contextual bandits with experts. However, we would like to highlight that our problem setting is significantly more complicated than the usual contextual bandits with experts problem, as in a contextual bandit setting when an arm is pulled under a context, we get a direct feedback about the reward of that arm under that context. However, in our problem setting when a channel-state is observed and a rate vector is scheduled, the received feedback only gives us a partial noisy feedback about the possible capacity class in which the channel-state belongs. The quality of the feedback also depends on the choice of the rate points. Further in our problem setting, even after the capacity classes are learned there is an additional task to recover the boundaries of the corresponding capacity regions. Therefore, the epoch-greedy algorithm proposed in this paper is a first step towards analyzing this setting, and we leave the study of algorithms with implicit exploration that can potentially yield $O(\sqrt{T})$ regret bound as future work.

VI. Simulation Results

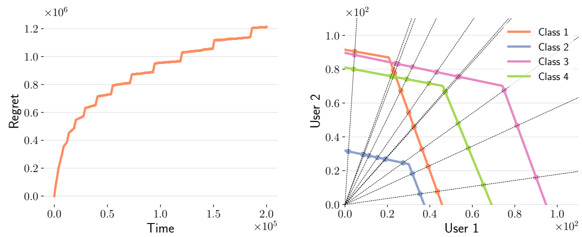
In this section we perform empirical simulations of our algorithm on synthetic data-sets with different parameter settings. In all our simulations, we use the robust version of our algorithm. The class of experts/classifying functions, used in our simulations, is the K-nearest-neighbor classifier implementation in scikit-learn [18], where the number of nearest neighbors used is determined through cross-validation. Since our algorithm is *robust*, we set $n_0 = 1000K$, $l_0 = 50K$, $\alpha(l, \delta) = 40KD$ and $s(l) = 200l^2$, in all our simulations. The number of direction vectors in set $\mathcal{U}$ is kept constant at $D = 10$ in all our simulations.

Synthetic System Model: We simulate our algorithm under various regimes where the number of capacity classes (K), the dimension of the channel-state feedback from each user (d), the number of users (n) are set to different values. The channel-state vectors that are in the channel-state class $k \in [K]$ are generated from a multi-variate normal distribution with mean $\mathbf{m}_k \in \mathbb{R}^{nd}$ and covariance matrix $\Sigma_k \in \mathbb{R}^{nd\times nd}$. The means and covariances for the different classes are randomly generated and held fixed over the course of an experiment. We ensure that $\|\mathbf{m}_i - \mathbf{m}_j\|_2 \geq 2(\|\Sigma_i\|_{2,2} + \|\Sigma_j\|_{2,2})$ for all $i, j \in [K]$, where $\|M\|_{2,2}$ denotes the spectral norm of a matrix M. Note that in our experiments the capacity classes are not disjoint partitions, but can have overlaps with low-probability. In our simulations, we have $\mathbb{P}(\mathcal{I}(\mathbf{Q}) = k) = 1/K$ for all $k \in [K]$, i.e. the capacity classes are equally likely. At each time step, a capacity class is selected uniformly at random and the channel-state vector observed is a random vector drawn from the corresponding Gaussian distribution.

The capacity region for each channel-state class is a convex set in the positive quadrant of $\mathbb{R}^n$ which is constructed as an intersection of n hyperplanes, where n is also the number of users in the system. The parameters of the hyperplanes are selected randomly for each class, while also ensuring that Assumption 2 is satisfied. The noise model of the system is such that

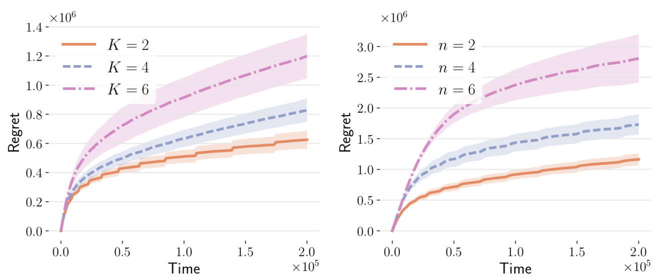
$$\rho_i(c\mathbf{u}) = \begin{cases} 1 - 0.1\left(\frac{c}{c^*_{\mathbf{u},k}}\right), & \text{if } c \leq c^*_{\mathbf{u},k}, \\ 0.1\left(2 - \frac{c}{c^*_{\mathbf{u},k}}\right), & \text{if } c > c^*_{\mathbf{u},k}, \\ 0, & \text{otherwise.} \end{cases} \quad (5)$$

Results: We first plot the results for one sample path (one run of our algorithm) for a system with $K = 4$ capacity classes and $n = 2$ users each providing a feedback in $d = 4$ dimensions. The regret is plotted versus time in Fig. 3a. We can see that the regret is clearly sub-linear. The simulations were carried out till 2×10^5 time-slots, out of which only about 25k time-slots were used for exploration. In Fig. 3b, we plot the boundary points obtained at the end of the run, in the $D = 10$ pre-specified directions. We can see that in only about 25k exploration time-slots, the system effectively learns the boundary points.



(a) (b)

Fig. 3: One run of our algorithm (one sample path) for a system with $n = 2$ users having $d = 4$ dimensional feedback vectors. The number of capacity classes is $K = 4$. In (a), we plot the regret as a function of time. In (b), we plot the boundaries of the capacity regions obtained in the $D = 10$ directions which were specified.



(a) (b)

Fig. 4: Variation in regret for different settings of K (number of capapcity classes) and n (number of users). In (a), we plot the regret of the algorithm for $K = 2, 4, 6$ for fixed values of $n = 2$, $d = 4$ and $D = 10$. In (b), we plot the regret for $n = 2, 4, 6$ for fixed values of $K = 4$, $d = 4$ and $D = 10$. The confidence region of one standard deviation is shown in faded colors.

In Fig. 4a, we *plot the dependence of regret on the number of capacity classes* K, when all other parameters are held constant. All the plots are obtained by averaging over 100 simulations and the corresponding confidence regions are also plotted. It can be seen that the regret increases almost linearly with K, as predicted by our theoretical results.

In Fig. 4b, we *plot the dependence of regret on the number of users* n, when all other parameters are held constant. All the plots are obtained by averaging over 100 simulations and the corresponding confidence regions are also plotted. The effect of n on the regret is more severe as compared to the effect due to K. This is because when the number of users is increased the effect on the hardness is two-folds: (i) the dimensionality of the channel-state space increases and therefore the classification problem is harder, (ii) the dimensions of the rate region increases and therefore the capacity explore phase becomes more difficult.

Acknowledgements

This work was partially supported by NSF grants CNS-1718089 and CNS-1718089, the Wireless Networking and Communciations Group Industrial Affiliates Program, and the US DoT supported D-STOP Tier 1 University Transportation Center. Sanjay Shakkottai thanks researchers at AT&T Labs (Drs. A. Ghosh, S. Akoum and T. Novlan) for valuable discusssions and feedback on the model.

References

[1] Y. Du and G. de Veciana, "Wireless networks without edges: Dynamic radio resource clustering and user scheduling," in *Proc. IEEE INFOCOM*, April 2014, pp. 1–9.

[2] A. Agarwal, M. Dudík, S. Kale, J. Langford, and R. Schapire, "Contextual bandit learning with predictable rewards," in *Artificial Intelligence and Statistics*, 2012, pp. 19–26.

[3] P. W. Goldberg and S. Kwek, "The precision of query points as a resource for learning convex polytopes with membership queries." in *COLT*. Citeseer, 2000, pp. 225–235.

[4] D. Avis and K. Fukuda, "A pivoting algorithm for convex hulls and vertex enumeration of arrangements and polyhedra," *Discrete & Computational Geometry*, vol. 8, no. 3, pp. 295–313, 1992.

[5] R. Srikant and L. Ying, *Communication Networks: An Optimization, Control, and Stochastic Networks Perspective*. Cambridge University Press, 2014.

[6] G. Wunder, M. Kasparick, A. Stolyar, and H. Viswanathan, "Self-organizing distributed inter-cell beam coordination in cellular networks with best effort traffic," in *Proc. 8th Intl. Symposium on Modeling and Optimization in Mobile, Ad Hoc and Wireless Networks (WiOpt)*. IEEE, 2010, pp. 295–302.

[7] W. Yu, T. Kwon, and C. Shin, "Multicell coordination via joint scheduling, beamforming, and power spectrum adaptation," *IEEE Transactions on Wireless Communications*, vol. 12, no. 7, pp. 1–14, 2013.

[8] T. Yoo and A. Goldsmith, "On the optimality of multiantenna broadcast scheduling using zero-forcing beamforming," *IEEE Journal on selected areas in communications*, vol. 24, no. 3, pp. 528–541, 2006.

[9] X. Xie and X. Zhang, "Scalable user selection for mu-mimo networks," in *INFOCOM, 2014 Proceedings IEEE*. IEEE, 2014, pp. 808–816.

[10] D. Gesbert, M. Kountouris, R. W. Heath, C.-B. Chae, and T. Salzer, "From single user to multiuser communications: Shifting the mimo paradigm," *IEEE signal processing magazine*, vol. 24, no. 5, pp. 36–46, 2007.

[11] A. Slivkins, "Contextual bandits with similarity information," in *Proceedings of the 24th annual Conference On Learning Theory*, 2011, pp. 679–702.

[12] J. Langford and T. Zhang, "The epoch-greedy algorithm for multi-armed bandits with side information," in *Advances in neural information processing systems*, 2008, pp. 817–824.

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

[14] A. Agarwal, D. Hsu, S. Kale, J. Langford, L. Li, and R. Schapire, "Taming the monster: A fast and simple algorithm for contextual bandits," in *Intl. Conf. on Machine Learning*, 2014, pp. 1638–1646.

[15] M. Dudik, D. Hsu, S. Kale, N. Karampatziakis, J. Langford, L. Reyzin, and T. Zhang, "Efficient optimal learning for contextual bandits," *arXiv preprint arXiv:1106.2369*, 2011.

[16] S. Kwek and L. Pitt, "Pac learning intersections of halfspaces with membership queries," *Algorithmica*, vol. 22, no. 1-2, pp. 53–75, 1998.

[17] V. N. Vapnik and A. Y. Chervonenkis, "On the uniform convergence of relative frequencies of events to their probabilities," in *Measures of complexity*. Springer, 2015, pp. 11–30.

[18] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay, "Scikit-learn: Machine learning in Python," *Journal of Machine Learning Research*, vol. 12, pp. 2825–2830, 2011.

[19] A. K. Menon, B. van Rooyen, and N. Natarajan, "Learning from binary labels with instance-dependent corruption," *arXiv preprint arXiv:1605.00751*, 2016.

Appendix A
Robust Algorithm

Algorithm 4 achieves the regret bound in Theorem 1 with high probability. However, this algorithm has very low-probability failure events that can disrupt the working of the algorithm for all future time-slots. In this section, we introduce a more robust version of our algorithm as Algorithm 5. We talk about the changes required in each of the phases separately.

Initialization Phase: For the initialization of the classifier π previously the classifier could fail (with low probability) if the set of n_0 initial channel-states used to build the classifier was "bad" (it does not contain sufficient channel-state points from a particular class). The robust algorithm fixes this issue by reinitializing the set of n_0 channel-state points for the classifier π and rebuilding the classifier π from scratch if we are unable to build π by testing $\frac{2\log(\frac{1}{\delta})+K-1}{\lambda}$ rate vectors. The pseudo code for the robust initialization of the Classifier Tree is given in Algorithm 6.

Class Explore and Reinitializing π: While the problem of having a bad initial set of n_0 channel-states is fixed by using the robust initialization of the classification tree however the classifier π can still be incorrect due to a wrong sub classifier present in it. We should also note that as the epoch number l increase our classifier misclassification error decreases. We exploit this and reclassify the initial n_0 channel-states with the classifier at each epoch and check if the number of channel-states all leaves are greater than $\frac{n_0}{2\beta}$. If this condition does not hold then we would rebuild the classifier and reinitialize epoch number l, set S_i and $C_{k,u}$. The pseudo code for the robust algorithm is given in Algorithm 5.

The capacity explore phase for the robust algorithm is described as steps 27-35 in Algorithm 5. It is aimed at not only learning the boundaries of the K capacity classes in the directions $\mathcal{U}$ but also correcting for wrong decisions. We initialize $C_{k,\mathbf{u}}[0] = 0$ and $C_{k,\mathbf{u}}[1] = C$ for all $k \in [K]$ and $\mathbf{u} \in \mathcal{U}$ at the start of the algorithm same as previous algorithm.

However unlike the previous algorithm where we only tested for $\frac{C_{k,\mathbf{u}}[0]+C_{k,\mathbf{u}}[1]}{2}\mathbf{u}$ we would also test the rate vectors $C_{k,\mathbf{u}}[0]\mathbf{u}$ and $C_{k,\mathbf{u}}[1]\mathbf{u}$. This is because either $C_{k,\mathbf{u}}[0]\mathbf{u}$ or $C_{k,\mathbf{u}}[1]\mathbf{u}$ corresponds to the rate vector that was tested for epoch $l-1$. Therefore if a wrong update decision was made by the binary classification error in epoch $l-1$ then by scheduling this point again in epoch l we can detect and correct for the error. At any epoch not all capacity update decision are correct and not all of them are incorrect so determining the epoch number for next phase is hard. In order to simplify it we introduce a set $\hat{S} = \{\hat{S}_{k,\mathbf{u}}\}$ which is initialized to be equal to epoch number for all $k \in [K]$ and $\mathbf{u} \in \mathcal{U}$. We update this set according to the robust capacity explore algorithm and then the new epoch number is $\min(\hat{S})$.

So for the robust algorithm we initialize $m_{i,k,\mathbf{u}} = 0$ for all $i \in \{-1, 0, 1\}$, $k \in [K]$ and $\mathbf{u} \in \mathcal{U}$ at the start of each epoch where

---

Algorithm 5 Robust online rate allocation from channel-state and service data

1: Initialize empty sets $S_i = \{\}$ for $i \in [K]$
2: Initialize a single node tree $\mathcal{T}$ where the node contains n_0 different channel-state points
3: Initialize capacity rate $C_{k,\mathbf{u}}[0] = 0$, $C_{k,\mathbf{u}}[1] = C$ and stage number $\hat{S}_{k,n} = 1$ for all $k \in [K]$ and $\mathbf{u} \in \mathcal{U}$.
4: Initialize classifier π using Algorithm 6
5: Set $t = t_0$ (time index) and $l = 1$ (epoch index)
6: **while** $t \leq T$ **do**
7:     **for** $i = 1 : K-1$ **do**
8:         $\mathbf{r}_i$ is the rate vector stored in node $\mathcal{N}_i$
9:         Let $\mathbf{q}$ be the channel-state observed at time step t.
10:         Schedule rate $\mathbf{r}_i$. *(Class Explore)*
11:         Let $y \in \{0, 1\}$ be the notification received.
12:         Add $(\mathbf{q}, y)$ to S_i.
13:         Set $t = t + 1$.
14:         Update the classifier $\hat{\pi}_i$ in $\mathcal{N}_i$
15:     **end for**
16:     Reclassify the n_0 channel-state points at the root of classifier according to the update sub classifiers.
17:     **for** $i = 1 : K$ **do**
18:         **if** number of channel-state point that classify to leaf $i < \frac{n_0}{2\beta}$ **then**
19:             Reinitialize the single node tree $\mathcal{T}$ where the node contains n_0 different channel-state points
20:             Reinitialize $S_i = \{\}$ for $i \in [K]$
21:             Reinitialize $C_{k,\mathbf{u}}[0] = 0$, $C_{k,\mathbf{u}}[1] = C$ and stage number $\hat{S}_{k,n} = 1$ for all $k \in [K]$ and $\mathbf{u} \in \mathcal{U}$.
22:             Rebuild classifier π using Algorithm 6
23:             Epoch number $l = 1$
24:             Break
25:         **end if**
26:     **end for**
27:     Let the empirical means of success rate be $m_{i,k,\mathbf{u}} = 0$ for all $i \in \{-1, 0, 1\}$, $k \in \{1, 2\}$ and $\mathbf{u} \in \mathcal{U}$.
28:     **for** $s = 1 : 3\alpha(\delta, l)$ **do**
29:         Observe $(\mathbf{q}, \mathbf{u})$.
30:         Let $k = \pi_l(\mathbf{q})$ and $j = Uniform(\{-1, 0, 1\})$.
31:         Schedule rate vector *(Capacity Explore)* $\left(\frac{C_{k,\mathbf{u}}[0]+C_{k,\mathbf{u}}[1]}{2} + j \times \left(\frac{C_{k,\mathbf{u}}[0]-C_{k,\mathbf{u}}[1]}{2}\right)\right)\mathbf{u}$.
32:         Update $m_{j,k,\mathbf{u}}$.
33:         Set $t = t + 1$.
34:     **end for**
35:     Update C and $\hat{S}$ according to Algorithm 7
36:     $l = min(\hat{S})$
37:     **for** $s = 1 : s(l)$ **do**
38:         Observe $(\mathbf{q}, \mathbf{u})$.
39:         Let $k = \pi_l(\mathbf{q})$.
40:         Schedule rate vector $C_{k,\mathbf{u}}[0]\mathbf{u}$. *(Exploit)*
41:         Let $t = t + 1$.
42:     **end for**
43:     $l = l + 1$.
44: **end while**

Algorithm 6 Robust initialization of the Classifier Tree

Let $i = 1$ and for a tree $\mathcal{T}$ where the root contains n_0 initial channel-state points. There are no other nodes initially.
$x = 0$
while $i < K$ **do**
  if $x > \frac{2\log(\frac{1}{\delta})+K-1}{\lambda}$ **then**
    Reinitialize $i = 1$, $x = 0$ and tree $\mathcal{T}$ where the root contains n_0 new channel-state points. There are no other nodes initially.
    $t = t + n_0$
  end if
  Randomly select a capacity point $\mathbf{r}$
  $S_i = \{\}$
  for $l = 1 : l_0$ **do**
    Let $\mathbf{q}$ be the channel-state observed at this time-step.
    Schedule rate $\mathbf{r}$. *(Class Explore)*
    Let $y \in \{0, 1\}$ be the notification received. Add $(\mathbf{q}, y)$ to S_i.
    Set $t = t + 1$.
  end for
  Construct a binary classifier $\hat{\pi}_i$ by empirical risk minimization (ERM) over S_i, over the expert set $\hat{\Pi}$.
  for all leaves j of $\mathcal{T}$ **do**
    Classify the channel-state points at leaf j according to the classifier $\hat{\pi}_i$. Let n_j be the number of channel-state points at leaf j.
    if $\frac{n_0\beta}{2} <$ number of leaf channel-state classified as $0 < n_j - \frac{n_0\beta}{2}$ **then**
      Make leaf j into a parent of two new leaves where the left leaf has all the channel-state's classified as 1 and the right has all the channel-state's classified as 0.
      $i = i + 1$
      Break
    end if
  end for
  $x = x + 1$
end while

$m_{i,k,\mathbf{u}}$ is empirical mean success rate and $i = \{-1, 0, 1\}$ corresponds to rate vector $C_{k,\mathbf{u}}[1]\mathbf{u}$, $\frac{C_{k,\mathbf{u}}[0]+C_{k,\mathbf{u}}[1]}{2}\mathbf{u}$ and $C_{k,\mathbf{u}}[0]\mathbf{u}$ respectively. In the capacity explore phase we schedule $3\alpha(l, \delta)$ channel-state for epoch l in order to check and improve the upper or lower bound for $c^*_{\mathbf{u},i}$ at each epoch. For a given channel-state vector $\mathbf{q}$ and direction vector $\mathbf{u}$ we use the classifier to find the class $k = \pi(\mathbf{q})$ of the channel-state point. We then select $j = Uniform(\{-1, 0, 1\})$ and transmit the rate vector $\left(\frac{C_{k,\mathbf{u}}[0]+C_{k,\mathbf{u}}[1]}{2} + j \times \left(\frac{C_{k,\mathbf{u}}[0]-C_{k,\mathbf{u}}[1]}{2}\right)\right)\mathbf{u}$ and update $m_{j,k,\mathbf{u}}$. After all $3\alpha(l, \delta)$ channel-state points are transmitted we update the $C_{k,\mathbf{u}}[0]$, $C_{k,\mathbf{u}}[1]$ and $\hat{S}_{k,u}$for all $k \in [K]$ and $\mathbf{u} \in \mathcal{U}$ according to Algorithm 7, which is similar to a standard binary search procedure. The next epoch number is then $l = min(\hat{S})$.

Algorithm 7 Robust capacity explore update

for $\forall k \in [K]$ and $\mathbf{u} \in \mathcal{U}$ **do**
  if $m_{-1,k,\mathbf{u}} > \frac{1}{2}$ **then**
    $C_{k,\mathbf{u}}[1] = 2 \times C_{k,\mathbf{u}}[1] - C_{k,\mathbf{u}}[0]$
    $\hat{S}_{k,\mathbf{u}} = \hat{S}_{k,\mathbf{u}} - 1$
  else if $m_{1,k,\mathbf{u}} < \frac{1}{2}$ **then**
    $C_{k,\mathbf{u}}[0] = 2 \times C_{k,\mathbf{u}}[0] - C_{k,\mathbf{u}}[1]$
    $\hat{S}_{k,\mathbf{u}} = \hat{S}_{k,\mathbf{u}} - 1$
  else if $m_{0,k,\mathbf{u}} > \frac{1}{2}$ **then**
    $C_{k,\mathbf{u}}[0] = \frac{C_{k,\mathbf{u}}[0]+C_{k,\mathbf{u}}[1]}{2}$
    $\hat{S}_{k,\mathbf{u}} = \hat{S}_{k,\mathbf{u}} + 1$
  else if $m_{0,k,\mathbf{u}} < \frac{1}{2}$ **then**
    $C_{k,\mathbf{u}}[1] = \frac{C_{k,\mathbf{u}}[0]+C_{k,\mathbf{u}}[1]}{2}$
    $\hat{S}_{k,\mathbf{u}} = \hat{S}_{k,\mathbf{u}} + 1$
  end if
end for

APPENDIX B
PROOF SKETCH

In this section, we give a brief outline of the steps leading up to the regret bound in Theorem 1. The proof sketch is broken down into the following subsections, corresponding to the various phases of our algorithm.

Initialization phase: The key result for this phase has been provided as Lemma 1. The proof of Lemma 1 uses two subsequent Lemmas 3 and 4. In Lemma 3, we show that at most $\frac{2\log(\frac{1}{\delta})+K-1}{\lambda}$ randomly chosen rate points need to be tested in the while loop of Algorithm 2, in order to yield the correct structure of the classification tree. This is because Assumption 2 implies that there is sufficient probability of a randomly chosen rate vector to be lying at regions that can provide signal for separating the capacity regions. In Lemma 4, we show that for a training sets of size l_0 constructed by scheduling rate points at the internal nodes of the tree, is sufficient to drive the classification error at every internal node down to $\epsilon(l_0, \delta)$, with a probability of at least $1 - \frac{\delta}{l_0}$. Note that a key step in the proof of Lemma 4 is a well-known result from noisy classification theory [19]. This result crucially relies on Assumption 5.

Subsequently, it can be shown that with high probability, there are at least $n_0\beta(1 - \frac{1}{2K})$ samples from each channel-state class in the initial n_0 channel-state points selected in Algorithm 2, owing to Assumption 1. This is sufficient to show that even with at most $\epsilon(l_0, \delta)$ misclassification errors at the classifiers in the internal nodes of the tree, correct rate points are chosen to build the tree-structure. For more details, please refer to Appendix C.

Class explore phase: The main guarantees for the *class explore* phase in an epoch has been provided in Lemma 2. The proof of this lemma is fairly straightforward, and follows directly from the proof of Lemma 4 used in the proof of Lemma 1. At the end of the class explore phase in epoch

l, $l+l_0$ training samples have been collected in each internal node of the tree. Therefore, the classification error for the binary classifiers stored in each of the internal nodes of the tree goes down to $\epsilon(l+l_0,\delta)$ with a probability of at least $1-\delta/(l+l_0)^2$. A union bound over all the $K-1$ internal nodes yields the result.

Capacity explore stage: The next step is to analyze the *capacity explore stage* in an epoch. The idea is to prove that with high probability the binary search in Algorithm 3 succeeds in every epoch. In epoch l, we ensure that the binary search is *correct* with a probability of at least $1-\mathcal{O}(KD\delta/l^2)$. Therefore, if the error probabilities are added up over all the epochs, the resultant probability of error for the whole algorithm till any time horizon T will be at most $\mathcal{O}(KD\delta)$.

For the sake of intuition, let us consider a case where in epoch l, for a particular $k \in [K]$ and $\mathbf{u} \in \mathcal{U}$, the rate vector $(C_{k,\mathbf{u}}[0] + C_{k,\mathbf{u}}[1])\mathbf{u}/2$ lies inside the capacity region for class k. Then the binary search succeeds if after the capacity explore phase $C_{k,\mathbf{u}}[0]$ is updated to $(C_{k,\mathbf{u}}[0] + C_{k,\mathbf{u}}[1])/2$ and $C_{k,\mathbf{u}}[1]$ remains the same. Since, the value of $\alpha(l,\delta)$ is $\frac{2D}{\beta}\left(\frac{16}{1-2\rho}\right)^2 \log\left(\frac{l^2}{\delta}\right)$, we can prove that with high probability there are at least $\left(\frac{16}{1-2\rho}\right)^2 \log\left(\frac{l^2}{\delta}\right)$ time-slots in the class explore phase of epoch l, where the capacity class observed is k and the direction vector specified is $\mathbf{u}$. The empirical mean of the notifications received in these time-slot should ideally be close to 1. However, there are two possible errors that can occur in each of these time-slots: (i) the classifier π can mistakenly predict that the capacity class is not k, (ii) the notification received is 0 due to the noise in our system. We prove that even with these errors, $m_{k,\mathbf{u}}$ concentrates close to 1, within $\left(\frac{16}{1-2\rho}\right)^2 \log\left(\frac{l^2}{\delta}\right)$ time-steps and is greater than $1/2$ with a probability of at least $1-O(1/l^2)$. Therefore, the update made by Algorithm 3 is correct. An union bound over all $k, \mathbf{u}$ yields the desired result, which has been stated and proved in Appendix F. Note that given the binary search is correct in all the epochs, it is fairly straightforward to show that the estimates of the boundaries of the capacity regions converge to the true boundaries at a rate that is exponential in the number of epochs l. Therefore, the regret arising from these errors (scheduling a rate vector inside the capacity region that is away from the true boundary) is $O(1/2^l)$, which is dominated by the regret arising from classification error which is $O(1/\sqrt{l})$ in epoch l.

Exploit Phase: Since the vector $C_{k,\mathbf{u}}[0]\mathbf{u}$ is transmitted during the exploit phase for a given value of k and $\mathbf{u}$, $C_{k,\mathbf{u}}[0]$ is used as an estimate for the true optimal magnitude of the rate i.e $c^*_{k,\mathbf{u}}$ in that direction and for that capacity class. The difference between $C_{k,\mathbf{u}}[0]$ and $c^*_{k,\mathbf{u}}$ is at most $\frac{C}{2^l}$ for epoch l, with high probability. Note that the misclassification rate is $\epsilon(l_0+l,\delta)$ in epoch l. Moreover, the expected regret is $\frac{C}{2^l}$ if the channel-state class is identified correctly and at most $c^*_{k,\mathbf{u}} < C$ otherwise. Therefore it can be shown that the expected regret for the exploit stage is less than $2C\epsilon(l,\delta) \times s(l)$ where $s(l) = \sqrt{l}$. It should be noted that since $\epsilon(l,\delta) = \mathcal{O}(\frac{\log l}{\sqrt{l}})$ so by choosing $s(l) = \sqrt{l}$ we get $\mathcal{O}(\log l)$ regret for epoch l.

Regret Bound: For the final regret bound we observe that the length of epoch l is $K-1+\alpha(l,\delta)+s(l) = \mathcal{O}(\sqrt{l})$. This implies that the number of epochs till time T, L^* is $\mathcal{O}(T^{\frac{2}{3}})$. Finally for epoch l the expected regret is at most $(K-1+\alpha(l,\delta)+2\epsilon(l,\delta)s(l))C$. By adding up this regret for $l=1$ to L^* we get the required regret bound of $\mathcal{O}\left(T^{2/3}\log\left(\frac{1}{\delta}\right)\left(D\log T + K + \sqrt{V}\right)\right)$. The final regret bound has been stated and proved with the precise details in Theorem 2 in the Appendix.

APPENDIX C
INITIALIZATION PHASE

Before we provide the proof of Lemma 1, we state and prove two lemmas used for proof of Lemma 1.

Lemma 3. *With probability at least $1-\delta$, the classifier π is constructed by testing $\frac{2\log(\frac{1}{\delta})+K-1}{\lambda}$ different rate vectors.*

Proof. A rate vector $\mathbf{r}$ is defined to be **good** if the classifier $\hat{\pi}$ built using it splits a leaf in the tree $\mathcal{T}$. Let R be the set of different channel-state classes at leaf $\mathcal{L}$ in the tree and i and j be two channel-state classes in set R. Then a classifier $\hat{\pi}$ constructed using a good **good** rate vector will partition the K classes into set $\mathcal{K}$ and $\mathcal{K}^c$ such that $i \in \mathcal{K}$ and $j \in \mathcal{K}^c$ therefore classifier will split the set R at leaf $\mathcal{L}$. Since $d(C_i, C_j) \geq \lambda,\ \forall\, i,\ j$ and the rate vectors are selected uniformly at random so the probability of selecting a **good** rate vector that can split the set of channel-state classes R into I and J such that $i \in I$, $j \in J$, $I \cup J = R$ and $I \cap J = \emptyset$ is at least λ i.e.

$$\begin{aligned}
&\mathbb{P}(\text{rate vector } \mathbf{r} \text{ is a "good" rate vector})\\
\geq&\mathbb{P}(\hat{\pi} \text{ can split leaf } \mathcal{L} \text{ that contain muliple classes})\\
\geq&\mathbb{P}(\hat{\pi} \text{ can split the set of channel-state } R \text{ at leaf } \mathcal{L} \text{ into } I \text{ and } J)\\
\geq&\lambda
\end{aligned}$$

It should be noted that after $K-1$ such splits, all the channel-state regions can be distinguished since there will be K leaf node and the set of channel-state classes at each leaf node is distinct (due to construction of tree). Therefore each leaf node corresponds to a single and unique channel-state class.

Let $\mathbf{r}_i$ be the i^{th} rate vector tested and let X_i be the event that $\mathbf{r}_i$ can potentially split some leaf in the tree $\mathcal{T}$. Then X_i is a Bernoulli random variable with mean greater or equal than λ. Using multiplicative Chernoff bound

$$\mathbb{P}\left(\sum_{i=1}^{\frac{(K-1)(m+1)}{\lambda}} X_i < K-1\right) < e^{-\frac{m(K-1)}{2}} \tag{6}$$

Therefore with probability $1-\delta$, in $\frac{2\log(\frac{1}{\delta})+K-1}{\lambda}$ randomly selected rate vectors there at least $K-1$ **good** rate vectors. □

For, any rate point $\mathbf{r}$ let us define a mapping $\hat{\mathcal{I}}^{(\mathbf{r})}$ from $\mathcal{P} \to \{0,1\}$ such that,

$$\hat{\mathcal{I}}^{(\mathbf{r})}(\mathbf{q}) = \begin{cases} 1, & \text{if } \mathbf{r} \in \mathcal{C}_{\mathcal{I}(\mathbf{q})} \\ 0, & \text{otherwise} \end{cases}$$

.

Thus, if outputs observed by scheduling rate point $\mathbf{r}$ are used to build a binary classifier, an ideal classifier would try to learn the above mapping.

Lemma 4. *Let $\hat{\pi}_i$ be the classifying function returned by the oracle for a training set of l_0 channel-state's for rate vector $\mathbf{r}_i$. Note that $\hat{\mathcal{I}}_i \triangleq \hat{\mathcal{I}}^{(\mathbf{r}_i)}$ is the ideal classification function, that this classifier aims to approximate. Then with probability at least $1 - \frac{\delta}{l_0^2}$ we have,*

$$\mathbb{P}(\hat{\pi}_i(\mathbf{Q}) \neq \hat{\mathcal{I}}_i(\mathbf{Q})) \leq \frac{1}{1-2\rho}\sqrt{\frac{V}{l_0}} + \sqrt{\frac{2\log\left(\frac{l_0^2}{\delta}\right)}{l_0}} \triangleq \epsilon(l_0,\delta), \tag{7}$$

where the probability is over the randomness in $\mathbf{Q} \sim f_{\mathcal{Q}}$ and the randomness in $\hat{\pi}_i$ which was trained on the random training set $\mathcal{N}_i$.

Proof. Let $\mathcal{N}_i = \{(\mathbf{q}_1, y_1), ..., (\mathbf{q}_{l_0}, y_{l_0})\}$ be the set of channel-state's and observations from scheduling the point $\mathbf{r}_i$. Let us first define a classification problem in the absence of scheduling noise. Let us define a random variable $\mathbf{Q} \sim f_{\mathcal{Q}}$. Let us define $Y = \mathbb{1}\left(\hat{\mathcal{I}}_i(\mathbf{Q}) = 1\right)$. Let $\mathcal{D}$ be the joint distribution of $(\mathbf{Q}, Y)$. Note that $\mathcal{N}_i$ contains i.i.d samples from a distribution $\mathcal{D}$, however the label for a channel-state point $\mathbf{q}$ is flipped with probability $\rho(\mathbf{q}, \mathbf{r}_i)$. Let us name the distribution with flipped labels by $\tilde{\mathcal{D}}$. For any function $\hat{\pi} \in \hat{\Pi}$, let us define the risk with respect to $\mathcal{D}$ as follows:

$$\mathcal{R}_{\mathcal{D}}(\hat{\pi}) = \mathbb{E}_{\mathcal{D}}\left[\mathbb{1}\left(\hat{\pi}(\mathbf{q}) \neq Y\right)\right]. \tag{8}$$

Let $\hat{\pi}^*_{\tilde{\mathcal{D}}}$ be defined as follows:

$$\hat{\pi}^*_{\tilde{\mathcal{D}}} = \operatorname*{argmin}_{\hat{\pi} \in \hat{\Pi}} \frac{1}{l_0} \sum_{(\mathbf{q},y) \in \mathcal{N}_i} \mathbb{1}\left(\hat{\pi}(\mathbf{q}) \neq y\right).$$

A well-known result from noisy classification [19] states that even though we minimize the above loss function over samples drawn from the distribution $\tilde{\mathcal{D}}$, the function resulting from the minimization has good risk guarantees with respect to the non-noisy distribution $\mathcal{D}$. By Proposition 4 in [19], with at least probability $1-\delta$ we have:

$$\mathcal{R}_{\mathcal{D}}(\hat{\pi}^*_{\tilde{\mathcal{D}}}) - \mathcal{R}_{\mathcal{D}}(\hat{\pi}^*) \leq \frac{1}{1-2\rho}\sqrt{\frac{V}{l_0}} + \sqrt{\frac{2\log\left(\frac{l_0^2}{\delta}\right)}{l_0}} \tag{9}$$

Note $\mathcal{R}_{\mathcal{D}}(\hat{\pi}^*) = 0$ by assumption. This yields the required result. □

Remark 2. *Note that the above result from noisy classification is only true if the Bayes optimal classifier is in our class of classifying functions. This has been assumed in our experts setting.*

Proof of Lemma 1. Let E_1 be the event that in the step 2 in Algorithm 2 the classifier π is built by testing at most $\frac{2\log(\frac{1}{\delta})+K-1}{\lambda}$ different rate vectors $\mathbf{r}_i$ i.e $K-1$ of these rate vectors are sufficient to split the channel-state region into K correct partitions.

Let $\hat{\pi}_i$ be the classifier built using the rate vector $\mathbf{r}_i$. Let $E_2(i)$ be the event that the classifier $\hat{\pi}_i$ satisfies Eq. (7) in Lemma 4. Given E_1 and $E_2(i)$ for all $i \in \left[\frac{2\log(\frac{1}{\delta})+K-1}{\lambda}\right]$, at most $\frac{2\log(\frac{1}{\delta})+K-1}{\lambda}$ different classifiers are constructed and for each of the classifiers we have $\mathbb{P}(\hat{\pi}_i(q) \neq \hat{\mathcal{I}}_i(q)) < \epsilon(l_0, \delta)$.

Let $\hat{\pi}_i$ be the classifier used for classifying $\hat{n}$ channel-state points (at some leaf in the classification tree), where the channel-state points are selected at random from $\mathcal{F}$ according to distribution $f_{\mathcal{Q}}$. Let X_j be the indicator random variable that $\hat{\pi}_i(q_j) \neq \hat{\mathcal{I}}_i(q_j)$. Then given $E_2(i)$ the probability of misclassifying more then $2\hat{n}\epsilon(l_0,\delta)$ channel-state is

$$\mathbb{P}(\sum_{j=1}^{\hat{n}} X_j > 2\hat{n}\epsilon(l_0,\delta)) < e^{-\frac{\hat{n}\epsilon(l_0,\delta)}{3}}$$

Let $\hat{A}_i$ be the event that fewer than $n_0\beta\left(1-\frac{1}{2K}\right)$ channel-state point out of n_0 channel-state points at the root node belong to channel-state class i. Then according to multiplicative Chernoff bound

$$\mathbb{P}(\hat{A}_i) < e^{-\frac{n_0\beta}{8K^2}} \qquad \forall\ j \in [K]$$

Therefore by union bound

$$\mathbb{P}(\cup_i \hat{A}_i) < \sum_i \mathbb{P}(\hat{A}_i) < Ke^{-\frac{n_0\beta}{8K^2}}$$

Let $E_3(i,\hat{n})$ be the event that the classifier $\hat{\pi}_i$ classified $\hat{n}$ points with less then $2\hat{n}\epsilon(l_0,\delta)$ misclassification and E_4 be the event that all the channel-state classes have at least $n_0c\left(1-\frac{1}{2K}\right)$ points sampled, then

$$\begin{aligned} \mathbb{P}(E_3(i,\hat{n})|E_1, E_2(i)) &> 1 - e^{-\frac{\hat{n}\epsilon(l_0,\delta)}{3}} \\ \mathbb{P}(E_4|E_1, E_2(i)\ \forall\ i) &> 1 - Ke^{-\frac{n_0\beta}{8K^2}} \end{aligned} \tag{10}$$

Let E_5 be the event that all the classifier $\hat{\pi}_i\ \forall\ i \in \left[\frac{2\log(\frac{1}{\delta})+K-1}{\lambda}\right]$ classify $\hat{n}$ channel-state points in leaf $\mathcal{L}$ with less then $2\hat{n}\epsilon(l_0,\delta)$ misclassification for all leaves in the tree.

It should be noted that the classification tree π has at most K leaves. Therefore

$$\begin{aligned}
&\mathbb{P}(\bar{E}_5|E_1, E_2(i)\ \forall\ i)\\
<&\mathbb{P}\left(\bigcup_{i=1}^{\frac{2\log(\frac{1}{\delta})+K-1}{\lambda}} \bigcup_{leaves} \bar{E}_3(i,\hat{n})|E_1, E_2(i), E_4\ \forall\ i\right)\\
<&\sum_{i=1}^{\frac{2\log(\frac{1}{\delta})+K-1}{\lambda}} \sum_{leaves} \mathbb{P}(\bar{E}_3(i,\hat{n})|E_1, E_2(i), E_4\ \forall\ i)\\
<&\sum_{i=1}^{\frac{2\log(\frac{1}{\delta})+K-1}{\lambda}} \sum_{leaves} e^{-\frac{\hat{n}\epsilon(l_0,\delta)}{3}}\\
<&K\left(\frac{2\log(\frac{1}{\delta})+K-1}{\lambda}\right)e^{-\frac{\hat{n}\epsilon(l_0,\delta)}{3}} \qquad (11)
\end{aligned}$$

The above equations follow owing to the fact that at any point the number of leaves in the tree is less than K.

Let us select l_0 such that $\epsilon(l_0,\delta) < \frac{\beta}{4K}$ (i.e. l_0 is large enough so that $\frac{1}{1-2\rho}\sqrt{\frac{V}{l_0}} + \sqrt{\frac{2\log\left(\frac{l_0^2}{\delta}\right)}{l_0}} < \frac{\beta}{4K}$) i.e. for classifying $\hat{n}$ points, at most $\frac{\hat{n}\beta}{2K}$ points are mis-classified with high probability.

Let us suppose that event E_4 and E_5 hold. Since the tree has height of at most K (as there are only $K-1$ internal nodes) which means that the channel-state points of any channel-state class have to be classified by at most $K-1$ classifiers. As the largest value of $\hat{n}$ is n_0 at the root node so the number of misclassified points of channel-state class j after doing classification with $K-1$ different classifiers is at most $n_0\beta(\frac{K-1}{2K})$. Since the total number of channel-state points for channel-state class j is at least $n_0\beta(1-\frac{1}{2K})$ so after misclassification of $n_0\beta(\frac{K-1}{2K})$ points at least $\frac{n_0\beta}{2}$ points will be correctly classified.

Since the above statement hold for all channel-state classes therefore a leaf j is split into two if and only if the number of channel-state classified as 0 are more then $\frac{n_0\beta}{2}$ and less than $n_j - \frac{n_0\beta}{2}$ where n_j are number of channel-state points at the leaf.

Since the minimum value of $\hat{n}$ is $\frac{n_0\beta}{2}$ i.e. minimum number of channel-state points that can be at any leaf. Therefore

$$\begin{aligned}
&\mathbb{P}(\bar{E}_4, \bar{E}_5|E_1, E_2(i)\ \forall\ i)\\
<&\mathbb{P}(\bar{E}_4|E_1, E_2(i)\ \forall\ i) + \mathbb{P}(\bar{E}_5|E_1, E_2(i)\ \forall\ i)\\
<&K\left(\frac{2\log(\frac{1}{\delta})+K-1}{\lambda}\right)e^{-\frac{n_0\beta^2}{24K}} + Ke^{-\frac{n_0\beta}{8K^2}}\\
<&K\left(\frac{2\log(\frac{1}{\delta})+K}{\lambda}\right)e^{-\frac{n_0\beta^2}{24K}}\\
<&K\delta \qquad \text{for} \quad n_0 \geq \frac{24K}{\beta^2}\log\left(\frac{2\log(\frac{1}{\delta})+K}{\delta\lambda}\right) \qquad (12)
\end{aligned}$$

Let E_6 be the event that the classifier π is successfully build then

$$\begin{aligned}
\mathbb{P}(\bar{E}_6) = \mathbb{P}(\bar{E}_4, \bar{E}_5) <&K\delta + \delta + \left(\frac{2\log(\frac{1}{\delta})+K-1}{\lambda}\right)\frac{\delta}{l_0^2}\\
<&3K\delta \qquad (13)
\end{aligned}$$

since $\mathbb{P}(\bar{E}_1) = \delta$ and $\mathbb{P}(\bar{E}_2(i)) = \frac{\delta}{l_0^2}$ according to Lemma 3 and Lemma 4 respectively and $l_0 > \sqrt{\left(\frac{2\log(\frac{1}{\delta})+K-1}{K\lambda}\right)}$. □

APPENDIX D
CLASS EXPLORE PHASE

Outline of proof of lemma2. The proof is pretty straightforward. The classifier π is build using $K-1$ sub-classifiers each of which is trained on l_0+l points. So by using the result of Lemma 4 and taking union bound over all sub classifiers inside classifier π we get the above lemma. □

APPENDIX E
CAPACITY EXPLORE PHASE

The value of $\alpha(l,\delta)$ is set to be $\frac{2D}{\beta}\left(\frac{16}{1-2\rho}\right)^2\log\left(\frac{l^2}{\delta}\right)$. Let $T_{\mathbf{u},k,l}$ be the number of samples in step 18 of the Algorithm 4 in phase l, such that capacity class observed is k and the direction observed is $\mathbf{u}\in\mathcal{U}$. A simple application of multiplicative Chernoff bound yields,

$$\mathbb{P}\left(T_{\mathbf{u},k,l} \leq \left(\frac{16}{1-2\rho}\right)^2\log\left(\frac{l^2}{\delta}\right)\right) \leq \frac{\delta}{l^2}.$$

APPENDIX F
PUTTING IT TOGETHER: REGRET BOUND

Let $E_7(l)$ be the success event in lemma 2 for phase l. Let $E_8(l)$ be the event $\left\{T_{\mathbf{u},k,l} > \left(\frac{16}{1-2\rho}\right)^2\log\left(\frac{l^2}{\delta}\right)\right\}$ for all k and $\mathbf{u}$. Also, recall the event E_6 which is the success event in Lemma 1.

Lemma 5. *The events $E_7(l)$ and $E_8(l)$ are mutually independent. Moreover, we have,*

$$\begin{aligned}
&\mathbb{P}\left(E_6 \cap \left(\cap_{l=1}^{\infty}(E_7(l)\cap E_8(l))\right)\right)\\
\geq&1 - K(D+1)\sum_{l=1}^{\infty}\frac{\delta}{l^2} - 3K\delta. \qquad (14)
\end{aligned}$$

Note that $m_{\mathbf{u},k}$ are the means collected in step 20 of the algorithm 4. In stage l, we name $S_{\mathbf{u},k,l}$ as a success event which signifies whether $m_{\mathbf{u},k} > \frac{1}{2}$ when $1/2(C_{\mathbf{u},k}[0]+C_{\mathbf{u},k}[1])$ is within $\mathcal{C}_k$ or whether $m_{\mathbf{u},k} < \frac{1}{2}$ when $1/2(C_{\mathbf{u},k}[0]+C_{\mathbf{u},k}[1])$ is outside $\mathcal{C}_k$.

Lemma 6. *We have,*

$$\mathbb{P}\left(S^c_{\mathbf{u},k,l}|E_6\cap E_7(l)\cap E_8(l)\right) \leq \frac{\delta}{l^2}. \qquad (15)$$

This further implies that,

$$\mathbb{P}\left(E_6 \cap \left(\cap_{l=1}^{\infty}(E_7(l)\cap E_8(l)\cap_{k,\mathbf{u}} S_{k,\mathbf{u},l}\right)\right) \geq 1 - \kappa KD\delta \qquad (16)$$

where κ is a suitable constant.

Proof. Consider the scheduling instances $s = \{1, .., \alpha(l, \delta)\}$. Let $s_{(1)}, ..., s_{(T_{\mathbf{u},k,l})}$ be the indices of the time-slots where class is k and direction is $\mathbf{u}$. Let $d_{k,\mathbf{u}}$ be set as $(C_{k,\mathbf{u}}[0] + C_{k,\mathbf{u}}[1])/2$ and assume that $d_{k,\mathbf{u}}$ lies within $\mathcal{C}_k$. We have the following bounds,

$$m_{k,\mathbf{u}} \geq \frac{1}{T_{\mathbf{u},k,l}} \sum_{r=1}^{T_{\mathbf{u},k,l}} Y_{s(r)} \mathbb{1}\left(\pi_l(\mathbf{q}_{s(r)}) = k\right).$$

Now conditioned $\mathbb{E}\left[m_{k,\mathbf{u}} | E_7(l)\right] \geq (1-(K-1)\epsilon(l_0+l, \delta))(1-\rho)$. Note that the random variables in the above summation are i.i.d conditioned on $E_7(l)$. Therefore, by Chernoff's bound we have,

$$\mathbb{P}\left(m_{k,\mathbf{u}} < \mathbb{E}\left[m_{k,\mathbf{u}} | E_7(l)\right] - \gamma | E_7(l)\right) \leq \exp\left(-\gamma^2 \frac{T_{\mathbf{u},k,l}}{2}\right)$$

This yields the following,

$$\begin{aligned}
&\mathbb{P}\left(m_{k,\mathbf{u}} < (1-(K-1)\epsilon(l_0+l,\delta))(1-2\rho) - \gamma | E_7(l), E_8(l)\right) \\
&\leq \exp\left(-\gamma^2 \left(\frac{8}{1-2\rho}\right)^2 \log\left(\frac{l^2}{\delta}\right)\right)
\end{aligned}$$

Since $\epsilon(l_0+l, \delta) \leq \epsilon(l_0, \delta) \leq \frac{\beta}{4K} \leq \frac{1}{5(K-1)}$, therefore setting $\gamma = (1-2\rho)/8$ yields the following,

$$\mathbb{P}\left(m_{k,\mathbf{u}} < \frac{1}{2} \middle| E_7(l), E_8(l)\right) \leq \frac{\delta}{l^2}.$$

Similar results hold for the case when the scheduled rate vector lies outside the capacity region. Therefore, we have the following:

$$\mathbb{P}\left(\cup_{k,\mathbf{u}} S^c_{\mathbf{u},k,l} \middle| E_7(l), E_8(l), E_6\right) \leq \frac{KD\delta}{l^2}.$$

Therefore,

$$\begin{aligned}
\sum_l \mathbb{P}\left(\cup_{k,\mathbf{u}} S^c_{\mathbf{u},k,l} | E_6\right) &\leq \sum_l \left(\frac{KD\delta}{l^2} + \frac{\delta}{l^2} + \frac{(K-1)\delta}{(l_0+l)^2}\right) \\
&\leq \sum_l \left(\frac{KD\delta}{l^2} + \frac{K\delta}{l^2}\right) \\
\implies \mathbb{P}\left(\cup_{k,\mathbf{u}} S^c_{\mathbf{u},k,l}\right) &\leq \frac{\pi^2}{6} K(D+1)\delta + 3K\delta.
\end{aligned}$$

Combining this with lemma 5 we get the following,

$$\begin{aligned}
&\mathbb{P}\left(E_6 \cap \left(\cap_{l=l_0}^{\infty} (E_7(l) \cap E_8(l) \cap_{k,\mathbf{u}} S_{k,\mathbf{u},l})\right)\right) \\
&\geq 1 - \left(\frac{\pi^2}{3} K(D+1)\delta + 6K\delta\right) \\
&\geq 1 - \kappa K D \delta.
\end{aligned}$$

□

Theorem 2. *With probability at least $1 - \kappa K D\delta$ the regret of the algorithm is $\mathcal{O}\Big(CT^{2/3}\left(K + \frac{D}{\beta}\log\left(\frac{T}{\delta}\right) + \sqrt{V}\right) + \left(\frac{K^2}{\beta^2}\left(V + \log\left(\frac{1}{\delta}\right)\right)\right)\left(\frac{\log(\frac{1}{\delta})+K}{\lambda}\right)\Big)$.*

Proof. Let $E = E_6 \cap \left(\cap_{l=l_0}^{\infty}(E_7(l) \cap E_8(l) \cap_{k,\mathbf{u}} S_{k,\mathbf{u},l}\right)$. Given E, at time l we have the following:

$$|C_{k,\mathbf{u}}[0] - c^*_{\mathbf{u},k}| \leq \frac{2C}{2^l}. \tag{17}$$

Let us choose $s(l) = \sqrt{l}$. Let $\mu(l)$ be the expected regret on the exploit slots at time l. It is easy to see that given E,

$$\begin{aligned}
\mu(l) &\leq \left((1-\epsilon(l_0+l,\delta)) \times \frac{2C}{2^l} + \epsilon(l_0+l,\delta)C\right) \times \sqrt{l} \\
&\leq 2\epsilon(l,\delta)C\sqrt{l} = \frac{C}{1-2\rho}\sqrt{V} + C\sqrt{3\log\left(\frac{l^2}{\delta}\right)}
\end{aligned}$$

Let $\gamma(l)$ be the total expected regret at phase l. Then we have the following:

$$\begin{aligned}
\gamma(l) \leq& (K-1)C + \alpha(l,\delta)C + \frac{C}{1-2\rho}\sqrt{V} + C\sqrt{3\log\left(\frac{l^2}{\delta}\right)} \\
\leq& C\Big((K-1) + \frac{2D}{\beta}\left(\frac{16}{1-2\rho}\right)^2 \log\left(\frac{l^2}{\delta}\right) \\
&+ \frac{1}{1-2\rho}\sqrt{V} + \sqrt{3\log\left(\frac{l^2}{\delta}\right)}\Big) \\
\leq& C\left((K-1) + \frac{997D}{\beta}\log\left(\frac{l^2}{\delta}\right) + \frac{4}{3}\sqrt{V} + \sqrt{3\log\left(\frac{l^2}{\delta}\right)}\right)
\end{aligned} \tag{18}$$

since $\rho \leq \frac{1}{8}$.

Let L^* be the epoch after the end of T time-slots. It is easy to see that $L^* \leq T^{2/3}$. Therefore given E, the expected regret for T time-slots is given by,

$$\begin{aligned}
R(T) \leq& \sum_{l=0}^{L^*} C\left((K-1) + \frac{997D}{\beta}\log\left(\frac{l^2}{\delta}\right) + \frac{4}{3}\sqrt{V} + \sqrt{3\log\left(\frac{l^2}{\delta}\right)}\right) \\
&+ n_0 + l_0\left(\frac{2\log(\frac{1}{\delta}) + K - 1}{\lambda}\right) \\
\leq& \sum_{l=0}^{T^{\frac{2}{3}}} C\left(K + \frac{997D}{\beta}\log\left(\frac{T^{\frac{4}{3}}}{\delta}\right) + \frac{4}{3}\sqrt{V} + \sqrt{3\log\left(\frac{T^{\frac{4}{3}}}{\delta}\right)}\right) \\
&+ \frac{24K}{\beta^2}\log\left(\frac{2\log(\frac{1}{\delta}) + K}{\delta\lambda}\right) + l_0\left(\frac{2\log(\frac{1}{\delta}) + K - 1}{\lambda}\right) \\
\leq& \sum_{l=0}^{T^{\frac{2}{3}}} C\left(K + \frac{1000D}{\beta}\log\left(\frac{T^{\frac{4}{3}}}{\delta}\right) + \frac{4}{3}\sqrt{V}\right) \\
&+ \frac{24K}{\beta^2}\log\left(\frac{2\log(\frac{1}{\delta}) + K}{\delta\lambda}\right) + l_0\left(\frac{2\log(\frac{1}{\delta}) + K - 1}{\lambda}\right) \\
=& \mathcal{O}\Big(CT^{2/3}\left(K + \frac{D}{\beta}\log\left(\frac{T}{\delta}\right) + \sqrt{V}\right) \\
&+ \left(\frac{K^2}{\beta^2}\left(V + \log\left(\frac{1}{\delta}\right)\right)\right)\left(\frac{\log(\frac{1}{\delta}) + K}{\lambda}\right)\Big)
\end{aligned} \tag{19}$$

□