Bandit Learning-based Online User Clustering and Selection for Cellular Networks

Isfar Tariq*§, Kartik Patel*, Thomas Novlan†, Salam Akoum†, Milap Majmundar†,
Gustavo de Veciana*, Sanjay Shakkottai*
**The University of Texas at Austin*, Austin, USA
{isfar, kartikpatel, deveciana, sanjay.shakkottai}@utexas.edu
†*AT&T Research Labs*, Austin, USA
{thomas_novlan, salam_akoum, milap_majmundar}@labs.att.com

***Abstract*—Current wireless networks employ sophisticated multi-user transmission techniques to fully utilize the physical layer resources for data transmission. At the MAC layer, these techniques rely on a semi-static map that translates the channel quality of users to the potential transmission rate (more precisely, a map from the Channel Quality Index to the Modulation and Coding Scheme) for user selection and scheduling decisions. However, such a static map does not adapt to the actual deployment scenario and can lead to large performance losses. Furthermore, adaptively learning this map can be inefficient, particularly when there are a large number of users. In this work, we make this learning efficient by clustering users. Specifically, we develop an online learning approach that jointly clusters users and channel-states, and learns the associated rate regions of each cluster. This approach generates a scenario-specific map that replaces the static map that is currently used in practice. Furthermore, we show that our learning algorithm achieves sub-linear regret when compared to an omniscient genie. Next, we develop a user selection algorithm for multi-user scheduling using the learned user-clusters and associated rate regions. Our algorithms are validated on the WiNGS simulator from AT&T Labs, that implements the PHY/MAC stack and simulates the channel. We show that our algorithm can efficiently learn user clusters and the rate regions associated with the user sets for any observed channel state. Moreover, our simulations show that a deployment-scenario-specific map significantly outperforms the current static map approach for resource allocation at the MAC layer.**

***Index Terms*—User clustering, Online learning, User selection, Cellular networks, Scheduling**

I. Introduction

The fifth-generation (5G) wireless technology uses sophisticated physical layer techniques such as Multi-User (MU) transmission to support high-speed communication to a large number of mobile devices. However, efficient multi-user transmission requires optimal multi-user scheduling which, in turn, requires learning and optimizing an unknown high-dimensional combinatorial function, solution of which depends on the channel observations (such as signal strength) at each time slot.

This optimization can be made efficient by exploiting the underlying low-dimensionality of the problem; namely, many users in a network may have "similar" channel distributions and optimal rate regions associated to scheduled user-sets. If such similarity between the users can be identified, then the network only needs to learn the mappings from the channel-states to the associated rate regions for each *user-cluster*. We also emphasize that the user clustering provides an important additional benefit: in the case of dynamic user-settings, any new user entering the system only needs to be categorized into a cluster, and any previously learned knowledge about the clusters can be used for the new user.

In this work, we design a user-clustering algorithm that clusters the users based on the "similarities" in the mappings from the channel-states to the associated rate regions. Using these mappings for clustering (as opposed to location, device types etc.) allows inferring the relevant information such as optimal rates and channel-state distributions across users in the same cluster. Note that learning these mappings from channel-states to the rate region itself is a high-complexity problem even for small number of users. An online epoch greedy-based strategy has been proposed in [1] that learns the optimal mapping from the channel-states to the rate regions. However, the strategy presented therein scales polynomially with the number of users in the system, thus making learning inefficient in the settings with the large user-sets. Hence, in this work, we build our user-clustering algorithm on top of the strategy in [1] and extend its design to enable the support for user-clusters. This allows our user-clustering algorithm to simultaneously learn these mappings and exploit the similarities in them to find optimal user-clusters. Additionally, we also develop a cluster -based user-selection algorithm that selects the optimal user-pairs and relative priorities for the transmission after observing the channel in each time-slot.

We summarize our main contributions below:

Joint user-channel clustering and rate scheduling algorithm: We design a user clustering model for MU scheduling and propose an *online* epoch-greedy strategy that can learn these user clusters and schedule them according to their channel-state partitions and capacity regions. Specifically, this epoch-greedy strategy simultaneously balances between four tasks: (1) learning channel-state partitions, (2) learning user clusters, (3) learning rate regions for scheduled cluster-sets and observed channel-state partitions, and (4) exploiting the gathered information to maximize the transmission rate. We

[§] The author is currently working at AT&T Research Labs. The work presented in the paper was completed during his Ph.D. at UT Austin.

ISBN 978-3-903176-49-2 © 2022 IFIP

then derive a theoretical guarantee on the regret, a measure of performance of our online strategy when compared to an omniscient genie. We show that the regret of our algorithm scales as $\mathcal{O}(T^{2/3}\log T)$ with time T, implying that it can learn and exploit the knowledge about the underlying environment and it converges to the optimal strategy over time.

User selection algorithm: We then propose an online user-selection algorithm that schedules user-sets and allocates relative priorities to each user depending on the observed channel states. This algorithm balances the task of exploring the user-sets to gather knowledge about the underlying environment and exploiting it to maximize the overall transmission rates. Given the cluster of users, the user-selection algorithm only requires learning the optimal combinations of different user clusters instead of large combinations of user-sets, which makes it feasible in a real-time deployment setting.

Numerical results: We use Wireless Next-Generation Simulator (WiNGS) - developed within AT&T Labs to perform extensive simulations [1]. Using this detailed simulation system that implements and processes packets through the 5G (NR) PHY, MAC, RLC, and PDCP sublayers (the channel itself is simulated), we study our proposed algorithms. We show that our proposed algorithms can learn the optimal user selections and identify the user clusters, their channel-state partitions, and the associated capacity regions. Moreover, we show that the user clustering-based strategies achieve better performance compared to static hand-tuned policies and [1]; especially in the scenarios with higher number of users and a larger channel-state space.

A. Related Work

Over the past few years, several machine learning ideas have been utilized to optimize a wide variety of areas of MU transmission scheduling such as user selection [2], beamforming [3], precoding [4], power allocation [5], interference mitigation [6] and receiver design [7]. Since naively searching the optimal user-set for transmission leads to a combinatorial explosion in complexity [8]–[11], the main application of machine learning for user selection is to cluster similar users to simplify the user selection decision. The clustering of users are based on different criteria, such as channel distribution [12], [13], user locations [14], user capacity [15] or signal strength [16]. In this work, we use a lower level criteria for clustering - the similarity in the channel-state distribution and associated rate regions.

We also emphasize that the majority of the work on user clustering requires strong assumptions such as the availability of the perfect channel state information of the users at the base-station [13], [17], a fixed set of users [12], [15] or a known number of user clusters [12], [15], [16]. There do exist some work that uses weaker assumptions; for instance, [13] proposes a user clustering algorithm for the setting with dynamic users and [14] proposes a user clustering algorithm with an unknown number of clusters. However, we are not aware of any studies that provide theoretical guarantees on the convergence of their algorithm for user clustering in MU-MIMO setting. In this work, our proposed algorithm *does not* assume perfect channel-state information or a known number of user clusters, and provides a theoretical guarantee on the regret compared to a genie policy.

In our previous work [1], [18], we have proposed online algorithm that learns the channel-state distributions and associated rate regions. In this work, we build on the model in [1] and propose an online user clustering algorithm that learns the channel-state distribution and associated rate regions and uses them for user clustering. We also present a robust version of our algorithm for a dynamic set of users. Similar to [1], we employ the contextual multi-armed bandit framework [19] for the design of our joint user-channel clustering and rate scheduling algorithm.

Notation: The set $[n] = \{1, 2, ..., n\}$. $\mathbb{R}^n_+ = \{v \in \mathbb{R}^n : v_i \geq 0, \forall i \in [n]\}$. For $C \in \mathbb{R}_+, \mathcal{B}_n(C) = \{v \in \mathbb{R}^n_+ : ||v||_2 \leq C\}$. $\mathbb{1}\{.\}$ denotes the indicator function. For a region $\mathcal{S}$ in a space, $|\mathcal{S}|$ defines the volume of the region.

II. System Model and Definitions

The system model extends the model in [1] by introducing the user clusters and redefining the system parameters for the user clusters. We consider a time-slotted cellular network containing a central network scheduler and n users indexed by $\mathcal{U} = [n]$. We assume that the set of users $\mathcal{U}$ can be clustered into L non-overlapping clusters denoted by $\mathcal{U}_1$,$\mathcal{U}_2$,...,$\mathcal{U}_L$ where $L << n$ and $|\mathcal{U}_i| \geq 1, \forall i \in [L]$. The *user clusters* are defined such that users in the same cluster provide similar performance under similar channel conditions. We formally define user clusters later in this section.

A. Channel states, scheduling, rate allocations

At the beginning of each time-slot t, the scheduler receives a *channel-state* estimate $\mathbf{q}(t) = [q_1(t), q_2(t), \ldots, q_n(t)]$ from a time-invariant distribution $f_{\mathcal{U}}$, where $q_i(t) \in \mathcal{Q} \subset \mathbb{R}^d_+$ denotes the channel-state of user i. We denote by $\mathcal{P} = \mathcal{Q}^n$ the set of all possible channel-states. In practice, the Channel Quality Indicators (CQI) reported by each user or the signal energy measured by the physical layer can be used as a channel-state.

Based on the observed channel-state $\mathbf{q}(t)$, the scheduler chooses a *user-pair* $A(t) = (i, j) \subset \mathcal{U}^2$ for multi-user communication in t-th time-slot. [1] We denote the channel-state of the user-pair A by $\mathbf{q}_A(t) = [q_i(t), q_j(t)]$ when $A = (i, j)$. Furthermore, we denote by f_A the marginal distribution of channel-states $\mathbf{q}_A$ and by $\mathcal{P}_A \subset \mathcal{Q}^2$ the set of all possible channel-states $\mathbf{q}_A$. Without loss of generality, we assume $i < j$ when denoting the user-pair $A = (i, j)$.

After scheduling the user-pair $A(t) = (i, j)$, the scheduler allocates the *rate vector* $\mathbf{r}_{A(t)}(t) = [r_i(t), r_j(t)] \in \mathcal{B}_2(C)$, where $r_i(t)$ denotes the rate of data transmission to user i during t-th time slot and $C \in \mathbb{R}_+$ is a constant which depends on the environment. We define the *capacity region* of a channel-state $\mathbf{q}_{A(t)}$ as the set of rate vectors $\mathbf{r}_{A(t)}(t)$

[1] We limit the discussion in this paper to two user scheduling for notational clarity. Our work can be extended to schedule multiple users in each time-slot.

that results in successful transmission with high probability. In practice, allocating the rate vector is equivalent to choosing an appropriate Modulation Coding Scheme (MCS) for each user in the scheduled user pair. The capacity region is defined by the highest feasible MCS such that neither user in the user-pair have significant errors during the transmission.

B. Channel-state partitions, capacity regions

For any user-pair $A = (i, j)$ where $i, j \in [n]$, the channel-state space $\mathcal{P}_A$ can be partitioned into K_A sets denoted by $\mathcal{P}_A^1, \mathcal{P}_A^2, \dots, \mathcal{P}_A^{K_A}$. We further define $\mathcal{C}_A^k$ to be the capacity region of the channel-state partition $\mathbf{q}_A \in \mathcal{P}_A^k$ for all $k \in [K_A]$. We assume that the capacity regions $\mathcal{C}_A^k$ are convex polytopes and for any rate vectors $\mathbf{x}, \mathbf{y}$, if $\mathbf{x} \leq \mathbf{y}$ (element-wise) and $\mathbf{y} \in \mathcal{C}_A^k$, then $\mathbf{x} \in \mathcal{C}_A^k, \forall k \in [K_A], \forall A$. We define K such that $K_A \leq K, \forall A$ and assume $K \ll |\mathcal{P}|$ if $\mathcal{Q}$ is discrete and finite. Furthermore, we assume that the channel-states $\mathbf{q}_A$ in each partition $\mathcal{P}_A^k$ are observed sufficiently often for any scheduled user-pair A and $k \in [K_A]$.

Assumption 1 (Class probabilities)**.** *We assume that for any given user-pair A, $\mathbb{P}(\mathbf{Q}_A \in \mathcal{P}_A^k) > \beta \ = O\left(\frac{1}{K_A}\right)\ \forall\ k \in [K_A]$, where $\mathbf{Q}_A \in \mathcal{P}_A$ is a random variable with distribution f_A capturing variability in the system.*

Similar to [1], we consider that the capacity regions of different channel-state partitions are well-separated.

Assumption 2 (Separability for channel-state partitions)**.** *We assume the for any given user-pair A and $k, m \in [K_A], k \neq m$, the capacity regions $\mathcal{C}_A^k$ and $\mathcal{C}_A^m$ are well separated, i.e.*

$$d(\mathcal{C}_A^k, \mathcal{C}_A^m) \triangleq \frac{|(\mathcal{C}_A^k \setminus \mathcal{C}_A^m) \cup (\mathcal{C}_A^m \setminus \mathcal{C}_A^k)|}{|\mathcal{B}_2(C)|} \geq \lambda > 0. \quad (1)$$

Intuitively, the parameter λ describes the fraction of the non-overlapping volume of capacity regions $\mathcal{C}_A^k$ and $\mathcal{C}_A^m$ for all $k, m \in [K_A], k \neq m$.

C. User clusters

We define the user clusters in terms of the similarity in the channel-state partitions and the associated rate regions.

Definition 1. *We say the users i, i' are in the same cluster $\mathcal{U}_a$, if the following conditions are true. For any user $j \in \mathcal{U}_b$, and the user-pairs $A = (i, j), A' = (i', j)$,*

1) *The channel distributions of user-pairs A and A', i.e. f_A and $f_{A'}$, are identical.*
2) *The number of partitions of the channel-state spaces $\mathcal{P}_A$ and $\mathcal{P}_{A'}$ are same, i.e. $K_A = K_{A'}$.*
3) *The partitions of the channel-state spaces are identical, i.e. $\forall k \in [K_A]$, $\mathcal{P}_A^k = \mathcal{P}_{A'}^k$.*
4) *The capacity regions of the channel-state partitions are identical, i.e. $\forall k \in [K_A], \mathcal{C}_A^k = \mathcal{C}_{A'}^k$.*

The definition of the user-cluster implies that the channel-state partitions and associated capacity regions depend on the user-cluster and *not* individual users. Thus, the scheduler only needs to learn and explore channel-state partitions of at most $L(L-1)/2$ user cluster-pairs instead of $n(n-1)/2$ unique user-pairs, significantly reducing the exploration.

Since the channel-state spaces, their partitions, and associated capacity regions are identical for the users in the same user clusters, we define the following notations for convenience. We denote the set of user cluster-pair by $\mathcal{A}_{\{a,b\}} = \{(i, j) | i \in \mathcal{U}_a,\ j \in \mathcal{U}_b\}$ where $a, b \in [L]$ and the set of all user pairs as $\mathcal{A} = \bigcup_{\forall a,b \in [L]} \mathcal{A}_{\{a,b\}}$. For any user pairs $A, A' \in \mathcal{A}_{\{a,b\}}$, we define the channel-state space as $\overline{\mathcal{P}}_{\mathcal{A}_{\{a,b\}}} \triangleq \mathcal{P}_A = \mathcal{P}_{A'}$, their partitions as $\overline{\mathcal{P}}^k_{\mathcal{A}_{\{a,b\}}} \triangleq \mathcal{P}^k_A = \mathcal{P}^k_{A'}$, associated capacity regions as $\bar{\mathcal{C}}^k_{\mathcal{A}_{\{a,b\}}} \triangleq \mathcal{C}^k_A = \mathcal{C}^k_{A'}$ and number of partition as $\overline{K}_{\mathcal{A}_{\{a,b\}}} \triangleq K_A = K_{A'}$. Throughout this paper, we use i, j for users, k, m for channel-state partitions, and a, b, c for user clusters in subscript or superscript of different parameters for better clarity. We use $\mathcal{A}_{\{\cdot,\cdot\}}$ to denote a general user cluster-pair.

In the following, we assume that the users in different clusters have distinct "characteristics".

Assumption 3 (Separability for different user clusters)**.** *Consider three user clusters $\mathcal{U}_a$, $\mathcal{U}_b$, and $\mathcal{U}_c$ where $b \neq c$. Denote by $\mathcal{A}_{\{a,b\}}, \mathcal{A}_{\{a,c\}}$ the user cluster-pairs. Let $\mathbf{q}_1 \in \overline{\mathcal{P}}^{k_1}_{\mathcal{A}_{\{a,b\}}}$ be a random variable from distribution $f_{\mathcal{A}_{\{a,b\}}}$ and $\mathbf{q}_2 \in \overline{\mathcal{P}}^{k_2}_{\mathcal{A}_{\{a,c\}}}$ be a random variable from distribution $f_{\mathcal{A}_{\{a,c\}}}$. Then*

1) $\overline{K}_{\mathcal{A}_{\{a,b\}}} \neq \overline{K}_{\mathcal{A}_{\{a,c\}}}$.
2) *If $\overline{K}_{\mathcal{A}_{\{a,b\}}} = \overline{K}_{\mathcal{A}_{\{a,c\}}}$, then for all permutations $\Delta_1,\ \Delta_2$ there exist some $k \in [1, K]$, $\Delta_1(k) = k_1$ and $\Delta_2(k) = k_2$ such that* $\mathbb{P}(\mathbf{q}_1 \in \overline{\mathcal{P}}^{k_2}_{\mathcal{A}_{\{a,c\}}})) \leq \gamma$, *and* $\mathbb{P}(\mathbf{q}_2 \in \overline{\mathcal{P}}^{k_1}_{\mathcal{A}_{\{a,b\}}})) \leq \gamma$.
3) *If there exist a permutation $\Delta_1,\ \Delta_2$, where for all $k \in [1, K]$, $\Delta_1(k) = k_1$ and $\Delta_2(k) = k_2$, we have* $\mathbb{P}(\mathbf{q}_1 \in \overline{\mathcal{P}}^{k_2}_{\mathcal{A}_{\{a,c\}}})) \geq \gamma$, *or* $\mathbb{P}(\mathbf{q}_2 \in \overline{\mathcal{P}}^{k_1}_{\mathcal{A}_{\{a,b\}}})) \geq \gamma$, *then there exists a value of k such that*

$$d(\bar{\mathcal{C}}^{k_1}_{\mathcal{A}_{\{a,b\}}}, \bar{\mathcal{C}}^{k_2}_{\mathcal{A}_{\{a,c\}}}) \geq \hat{\lambda} > 0. \quad (2)$$

The Assumption 3 states that for any two distinct user cluster pairs, i.e. $\mathcal{A}_{\{a,b\}}$ and $\mathcal{A}_{\{a,c\}}$, either their number of channel-state partitions are different. If not, then their channel-state partitions are sufficiently different. If their channel-state partitions are also similar, then their associated capacity regions of these user cluster-pairs must be well-separated.

D. Environment and channel-state classifiers

1) Environment: The *environment* is defined as the true user clusters mappings $\mathcal{U}_a,\ \forall\ a\ \in\ [L]$, the channel-state space $\{\overline{\mathcal{P}}_{\mathcal{A}_{\{\cdot,\cdot\}}}\}$, its partitions of user cluster pairs $\{\overline{\mathcal{P}}^k_{\mathcal{A}_{\{\cdot,\cdot\}}}\}_{k \in [\overline{K}_{\mathcal{A}_{\{\cdot,\cdot\}}}]}$ and the associated capacity regions $\{\bar{\mathcal{C}}^k_{\mathcal{A}_{\{\cdot,\cdot\}}}\}_{k \in [\overline{K}_{\mathcal{A}_{\{\cdot,\cdot\}}}]}$ for all user cluster-pairs $\mathcal{A}_{\{\cdot,\cdot\}}$. Note that the environment defines the "ground truth", typically, dependent on the deployment setting.

2) Index functions: In a given environment, the *index functions* map a channel-state to the index of its channel-state partition. Formally, we define index functions as follows:

Definition 2 (Index functions)**.** *Consider the scheduled user-pair $A = (i, j)$ with $i \in \mathcal{U}_a$ and $j \in \mathcal{U}_b$, and the user cluster-pair $\mathcal{A}_{\{a,b\}}$. We denote the index function of the user-pair A by $\mathcal{I}_A(\mathbf{q})$, such that $\mathbf{q} \in \mathcal{P}_A^{\mathcal{I}_A(\mathbf{q})}$. We also define the index function of the cluster-pair $\mathcal{A}_{\{a,b\}}$, denoted by $\bar{\mathcal{I}}_{\mathcal{A}_{\{a,b\}}}(\mathbf{q})$ where $\bar{\mathcal{I}}_{\mathcal{A}_{\{a,b\}}}(\mathbf{q}) \triangleq \mathcal{I}_A(\mathbf{q})$ if $A \in \mathcal{A}_{\{a,b\}}$.*

3) Classifiers: We assume that a set $\hat{\Pi}$ of binary classifiers/experts that maps channel-states to $\{0, 1\}$ is accessible to the algorithm. Similar to the prior works in [1], [20], we assume that out of all available experts in set $\hat{\Pi}$, at least one expert can represent the true behavior of the underlying environment by correctly partitioning the channel-state space.

Assumption 4 (Realizable setting)**.** *For $\kappa \subseteq [K_A]$, we define $\hat{\mathcal{I}}_A^{\kappa}(\mathbf{q}) = \mathbb{1}\{\mathcal{I}_A(\mathbf{q}) \in \kappa\}$ as a binary function. Then, we assume that we have access of class of binary classifier/experts $\hat{\Pi}$ such that $\hat{\mathcal{I}}_A^{\kappa}(.) \in \hat{\Pi}$ for all user-pairs A and $\kappa \subseteq [K_A]$. Furthermore, we assume that our class of experts have the VC dimension of V [21].*

Notably, Assumption 4 only implies that for any user pair, the binary classifiers (i.e., experts) can classify between κ and $[K_A] \backslash \kappa$. We require combining $\hat{\mathcal{I}}_A^{\kappa}$ for several different values of κ to recover the true index function $\mathcal{I}_A$ of a user-pair A. However, it is an algorithmic challenge to find the correct experts and efficiently compose them together to learn the index function $\mathcal{I}_A$ in an online setting.

E. Noise Model

After every transmission, the system receives feedback from the underlying environment. We define the feedback as the success of the transmission to a user-pair A (and equivalent user cluster-pair $\mathcal{A}_{\{\cdot,\cdot\}}$), with the observed channel-state $\mathbf{q}$, and the allocated rate vector $\mathbf{r} \in \mathbb{R}^2$. We model the feedback as a random variable $Y(\mathbf{q}, \mathbf{r}, A) \in \{0, 1\}$, where $Y(\mathbf{q}, \mathbf{r}, A) = 1$ signifies a successful transmission. We assume that $Y(\mathbf{q}, \mathbf{r}, A)$ is an i.i.d. random variable distributed as

$$\mathbb{P}(Y(\mathbf{q}, \mathbf{r}, A) = 1) = \begin{cases} 1 - \rho_{\mathcal{A}_{\{\cdot,\cdot\}}}(\mathbf{q}, \mathbf{r}), & \text{if } \mathbf{r} \in \bar{\mathcal{C}}_{\mathcal{A}_{\{\cdot,\cdot\}}}^{\bar{\mathcal{I}}_{\mathcal{A}_{\{\cdot,\cdot\}}}(\mathbf{q})}, \\ \rho_{\mathcal{A}_{\{\cdot,\cdot\}}}(\mathbf{q}, \mathbf{r}), & \text{if } \mathbf{r} \notin \bar{\mathcal{C}}_{\mathcal{A}_{\{\cdot,\cdot\}}}^{\bar{\mathcal{I}}_{\mathcal{A}_{\{\cdot,\cdot\}}}(\mathbf{q})}. \end{cases}$$

The term $\rho_{\mathcal{A}_{\{\cdot,\cdot\}}}(\mathbf{q}, \mathbf{r})$, called the *noise parameter*, defines the noise during the transmission at the rate $\mathbf{r}$ to the users in user cluster-pair $\mathcal{A}_{\{\cdot,\cdot\}}$ when observed channel-state is $\mathbf{q}$. We take the following assumption about the noise parameter.

Assumption 5 (Noise rate)**.** *We assume that*

1) $\rho_{\mathcal{A}_{\{\cdot,\cdot\}}}(\mathbf{q}, \mathbf{r}) \leq \rho < 1/8, \ \forall \ \mathbf{q}, \mathbf{r}, \mathcal{A}_{\{\cdot,\cdot\}}$.
2) *the channel-states from the same partition $\bar{\mathcal{P}}_{\mathcal{A}_{\{\cdot,\cdot\}}}^k$ introduces similar noise in the transmission. Specifically, for $\mathbf{p}, \mathbf{q} \in \bar{\mathcal{P}}_{\mathcal{A}_{\{\cdot,\cdot\}}}^k$, $\rho_{\mathcal{A}_{\{\cdot,\cdot\}}}(\mathbf{p}, \mathbf{r}) = \rho_{\mathcal{A}_{\{\cdot,\cdot\}}}(\mathbf{q}, \mathbf{r})$. We define $\rho_{\mathcal{A}_{\{\cdot,\cdot\}}}^k(\mathbf{r}) \triangleq \rho_{\mathcal{A}_{\{\cdot,\cdot\}}}(\mathbf{q}, \mathbf{r})$, where $k = \bar{\mathcal{I}}_{\mathcal{A}_{\{\cdot,\cdot\}}}(\mathbf{q})$ for notational convenience.*

F. Direction vectors and system evolution

1) Direction vectors and objective: We define a *direction vector*, denoted by $\mathbf{d} \in \mathbb{R}_+^2, ||\mathbf{d}||_1 = 1$, that controls the data rate ratio between the scheduled user-pairs. We denote the set of the D direction vectors by $\mathcal{D} = \{\mathbf{d}_1, \mathbf{d}_2, \ldots, \mathbf{d}_D\}$ and assume the set $\mathcal{D}$ is fixed for the entire duration of the algorithm. In practice, the direction vector can define the relative priorities of users that may be dependent on the current queue states or Quality of Service (QoS) requirements.

The main objective for the scheduler is to learn the maximum possible rate along these directions for all user-cluster pairs and their channel-state partitions. Specifically, for a chosen direction vector $\mathbf{d}$ and observed channel-state $\mathbf{q}$, the scheduler should find the rate vector $\mathbf{r}_A = g(\mathbf{q}, \mathbf{d}, A)\mathbf{d}$ where $g(\mathbf{q}, \mathbf{d}, A)$ denotes the highest scalar such that $\mathbf{r}_A \in \mathcal{C}_A^{\mathcal{I}_A(\mathbf{q})}$. Formally, $g(\mathbf{q}, \mathbf{d}, A) = \arg\max \left\{ \lambda > 0 | \lambda \mathbf{d} \in \mathcal{C}_A^{\mathcal{I}_A(\mathbf{q})} \right\}$.

2) System evolution: At each time-slot t (that contains multiple physical layer time-slots), the system executes the following steps:

1) It observes channel-state $\mathbf{q}(t)$ from the distribution $f_{\mathcal{U}}$.
2) The system schedules a user-pair $A(t)$ and a direction vector $\mathbf{d}(t) \in \mathcal{D}$.
3) The system then schedules the rate $\mathbf{r}(t)$ for transmission to the user-pair $A(t)$.
4) After transmission, the system receives binary feedback from the environment denoted by $Y(\mathbf{q}(t), \mathbf{r}(t), A(t))$, indicating whether the transmission was a success.

Note that the system requires the knowledge of the underlying environment to schedule the highest achievable rate for all observed channel-states, user pairs, and direction vectors. In this paper, we first propose an online algorithm in Section III that jointly clusters the users and channel-states and learns the associated capacity regions. Thus, the scheduler can schedule the highest possible rate in step 3. We then design the user selection algorithm in Section IV that uses the learned estimate of the environment to schedule the optimal user-pairs and direction vectors in step 2.

G. Reward function and regret

Before discussing the algorithm, we first describe the reward function and the regret definition used in the analysis of our algorithms.

1) Expected reward function: For any user-pair A, channel-state $\mathbf{q}$, rate vector $\mathbf{r}$, direction vector $\mathbf{d}$, and received feedback Y, we define the reward $\mu(\mathbf{q}, \mathbf{r}, \mathbf{d}, A)$ as

$$\mu(\mathbf{q}, \mathbf{r}, \mathbf{d}, A) = |\mathbf{r}| Y(\mathbf{q}, \mathbf{r}, A) \mathbb{1}\{\mathbf{r} \cdot \mathbf{d} = |\mathbf{r}|\}. \tag{3}$$

Thus, if the rate vectors are aligned with the direction vectors, then the expected reward received as a function of the chosen magnitude of the rate vectors, i.e. h, can be written as $f_{\mathbf{d}, \mathcal{I}_A(\mathbf{q}), A}(h) = h\mathbb{E}[Y(\mathbf{q}, h\mathbf{d}, A)]$. The reward function $f_{\mathbf{d},k,A}(h)$ characterizes the expected data rate when the rate vector $h\mathbf{d}$ is scheduled for channel-states $\mathbf{q} \in \mathcal{P}_A^k$ for user-pair A. Let $h_{\mathbf{d},k,A}^*$ denote the optimal choice of h that maximizes the expected data rate $f_{\mathbf{d},k,A}(h)$. Then, $f_{\mathbf{d},k,A}^* = f_{\mathbf{d},k,A}(h_{\mathbf{d},k,A}^*)$ denotes the maximum average rate achievable by a *genie policy* (with full knowledge of the environment) in the direction $\mathbf{d}$. We make the following assumption on the noise model that requires the maximum expected rate $f_{\mathbf{d},k,A}(h)$ be achieved by the rate vector $\mathbf{r} = h\mathbf{d}$ located at the boundary of the associated capacity region $\mathcal{C}_A^k$ along the direction $\mathbf{d}$.

Assumption 6. *Let $\hat{h}^*_{\mathbf{d},k,A} = \max_h\{h|h\mathbf{d} \in \mathcal{C}_A^k\}$. Then, the noise parameter $\rho_A^k(\mathbf{r})$ is such that $\hat{h}^*_{\mathbf{d},k,A} = h^*_{\mathbf{d},k,A}$, $\forall A \in \mathcal{A}, k \in [K_A], d \in \mathcal{D}$.*

2) Regret: Let $\mathbf{q}(t), A(t), \mathbf{d}(t), \mathbf{r}(t)$ denote the respective quantities at time t. Then the cumulative regret of the algorithm in T time slots is defined as

$$R(T) = \sum_{t=1}^{T}(f^*_{\mathbf{d}(t),\mathcal{I}_{A(t)}(\mathbf{q}(t)),A(t)} - \mathbb{E}[\mu(\mathbf{q}(t),\mathbf{r}(t),\mathbf{d}(t),A(t))]) \tag{4}$$

The regret compares the performance of an online policy with the *genie policy*. Our objective is to design an algorithm that incurs a regret that scales sub-linearly with T for all large enough T. It implies that our algorithm learns the environment over time and converges to the genie policy.

III. Joint user-channel clustering and rate scheduling algorithm

We now describe our first contribution – an online user clustering algorithm for multi-user rate scheduling. This algorithm modifies the framework of learning the channel-state partitions and associated capacity regions used in [1] and fits in step 3 of the system evolution discussed in Section II-F2. In the following, we first provide a brief overview of the complete algorithm, and discuss the user-clustering algorithm. We then derive the regret bound on the complete algorithm.

A. Algorithm overview

The structure of the algorithm is similar to the epoch greedy strategy in [1]. An important algorithmic idea is that we can learn about the unknown environment by simply scheduling a rate vector $\mathbf{r}$ (for each user-pair A) for many channel-state realizations $\mathbf{q}$, and observing the corresponding feedback $Y(\mathbf{q}, \mathbf{r}, A)$. Specifically, these observations are used to find a classifier $\pi \in \hat{\Pi}$ such that it can classify the observed channel-states $\mathbf{q}$ into two partitions: (1) $\mathcal{P}_A^* = \{\mathbf{q} : Y(\mathbf{q}, \mathbf{r}, A) = 1\}$, (2) $(\mathcal{P}_A^*)^c$. By repeating the same process for a fix set of rate points, we get a set of binary classifiers. The binary classifiers are then composed together to create a binary tree, called *classification tree* $\mathcal{T}_A$, such that each leaf of the tree correspond to a channel-state partition indexed by $\mathcal{I}_A(\mathbf{q})$ with high probability. Furthermore, these channel-state classifiers provide an estimate of the user channel-state partition and capacity region. Therefore, by constructing and comparing such classifiers for different user pairs A we can identify the user clusters.

The algorithm starts with an initialization phase that consists of two stages: (1) initializing channel-state classifiers for all user-pairs A, (2) identifying user-clusters based on the estimated channel-state classifiers. After the initialization phase, the algorithm proceeds in epochs. Each epoch consists of (1) *class explore* stage wherein the algorithm improves the estimates of the classifiers for all user cluster-pairs, (2) *capacity explore* stage wherein the algorithm learns capacity regions associated to the channel-state partitions of each user-cluster pair, and (3) *exploitation* stage wherein the algorithm uses the estimated environment, i.e. user clusters, the channel-state partitions, and associated capacity regions to schedule transmission rate optimally.

Among these stages, the initialization of the channel-state classifiers, class exploration, capacity exploration and exploitation stages are similar to that described in [1]; with appropriate modifications to use the user-clusters pairs instead of user-pairs. In the following, we describe our major contribution, the theoretically provable user-clustering algorithm while the complete algorithm and the details of the remaining stages of the algorithm is given in [22].

B. User clustering

After building the channel classifiers for all user-pairs, we determine the user clusters $\mathcal{U}_1, \ldots, \mathcal{U}_{\mathcal{L}}$ and initialize the user cluster classifiers. To find the user clusters, we use the channel classifier π_A's (and classification tree $\mathcal{T}_A$'s) for all the user pairs A generated during the initialization stage. Our algorithm works in three stages:

1) Computing the similarity: We start by computing the similarity between the users by comparing the similarity between the classifiers π_A for different user pairs. Specifically, to determine if users i and j are similar, we compare the classifiers $\pi_{(i',i)}$ and $\pi_{(i',j)}$ for all $i' \in \mathcal{U}\backslash\{i, j\}$. The classifiers $\pi_{(i',i)}$ and $\pi_{(i',j)}$ are similar if they satisfy the following criteria:

1) The number of leaves for classifier $\pi_{(i',i)}$ and $\pi_{(i',j)}$ are the same, i.e. $K_{(i',i)} = K_{(i',j)}$.
2) Let $\bar{n}^k_{(i',i)}$ be the number of channel-states stored at leaf $\mathcal{L}^k_{(i',i)}$ of classifier $\pi_{(i',i)}$. Let $\hat{n}^k_{(i',j)}$ be the number of channel-states at leaf $\mathcal{L}^k_{(i',i)}$ that are classified to leaf $\mathcal{L}^k_{(i',j)}$ using classifier $\pi_{(i',j)}$. Then, $\forall\ k \in [K_{(i',i)}]$, $\frac{\hat{n}^k_{(i',j)}}{\bar{n}^k_{(i',i)}} \geq \frac{3+\gamma}{4}$, and $\frac{\hat{n}^k_{(i',i)}}{\bar{n}^k_{(i',j)}} \geq \frac{3+\gamma}{4}$.
3) The rate vectors at all nodes for classifiers $\pi_{(i',i)}$ and $\pi_{(i',j)}$ are the same, i.e. the rate vector that spilt the node $\mathcal{N}^k_{(i',i)}$ and $\mathcal{N}^k_{(i',j)}$ are the same for all $k \in [K_{(i',i)} - 1]$.

We construct a similarity matrix in which the value at (i, j) indicates if the user i and j are similar.

2) Iterative clustering: In this part, we first define a single user cluster $\mathcal{U}_1$ containing the user 1 and place all the other users in set $\mathcal{U}'$. We then iteratively add the users from set $\mathcal{U}'$ to the cluster $\mathcal{U}_1$ if they are similar to all users in $\mathcal{U}_1$. If there are still any unassigned users left in set $\mathcal{U}'$, then we create a new cluster $\mathcal{U}_2$ consisting of a random unassigned user from set $\mathcal{U}'$ and iteratively add the unassigned users to the new cluster if they are similar to the users in $\mathcal{U}_2$. We repeat this process of cluster creation and assignment until all the users are assigned to the clusters.

3) Channel-state classifiers for user-cluster pairs: Finally, we generate the channel-state classifiers for all user-cluster pairs in the system by following the procedure of initializing channel-state classifiers (see [22]). We first create the dataset $\overline{S}^i_{\mathcal{A}_{\{a,b\}}}$ by collecting the data sampled for all user pair $A \in \mathcal{A}_{\{a,b\}}$. We then use the dataset $\overline{S}^i_{\mathcal{A}_{\{a,b\}}}$ to build the classifier $\overline{\pi}^i_{\mathcal{A}_{\{a,b\}}}$ for user-cluster pair $\mathcal{A}_{\{a,b\}}$.

C. Theoretical Result

We now provide the main theoretical result for our algorithm. Given that Assumptions 1-6 are satisfied, the cumulative regret bound for the proposed algorithm is given as follows.

Theorem 1. *Given Assumptions 1-6, the regret of the proposed algorithm scales as*

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

with probability of at least $1 - \xi KDL^2\delta$ *where* $\xi < 15$.

Proof. The detailed proof is given in [22]. □

Even though the regret of the proposed algorithm scales similarly to [1] with respect to T, D and K, the major reduction in the learning complexity is emphasized by the factor of $L^{2/3}$ that shows the dependence on the number of user-clusters and *not* on the number of users.

IV. User selection algorithm

In this section, we discuss our second major contribution: the user selection algorithm, a heuristic approach to select the user-pairs and direction vectors for scheduling based on the estimated environment and observed channel-state. Note that this algorithm fits in step 2 of the system evolution as discussed in Section II-F2. However, for the simplicity, we describe the user-selection algorithm as a standalone algorithm, assuming it is used after the user-clusters and channel-state classifiers have been estimated during the initialization phase of the joint user-channel clustering and rate scheduling algorithm. We discuss the details of the joint implementation of both algorithms in [22].

Algorithm 1 User selection algorithm based on user clustering

```
1: Input: Channel-state classifiers $\overline{\pi}_{\mathcal{A}_{\{\cdot,\cdot\}}}$, Epoch $\overline{E}_{\mathcal{A}_{\{\cdot,\cdot\}},k,\mathbf{d}}$
   and reward $\overline{G}_{\mathcal{A}_{\{\cdot,\cdot\}},k,\mathbf{d}}$ $\forall\mathcal{A}_{\{\cdot,\cdot\}}, k \in \overline{K}_{\mathcal{A}_{\{\cdot,\cdot\}}}, \mathbf{d} \in \mathcal{D}$.
2: Set weights $\overline{W}_{\mathcal{A}_{\{\cdot,\cdot\}},k,\mathbf{d}} = w_0$, $\forall\mathcal{A}_{\{\cdot,\cdot\}}, k, \mathbf{d}$.
3: For all user pair $A$ identify channel-state partition index
   $\bar{\mathbf{k}}^* = \{\bar{k}^*_A\}$ where $\bar{k}^*_A$ is determined using channel-state
   classifiers $\overline{\pi}_{\mathcal{A}_{\{\cdot,\cdot\}}}$, when $A \in \mathcal{A}_{\{\cdot,\cdot\}}$
4: Set $\overline{G}^* = \arg\max_{A,\mathbf{d}}(\overline{G}_{\mathcal{A}_{\{\cdot,\cdot\}},\bar{k}^*_A,\mathbf{d}})$ where $A \in \mathcal{A}_{\{\cdot,\cdot\}}$.
5: $g_n = |\{(A,d) : \overline{G}_{\mathcal{A}_{\{\cdot,\cdot\}},\bar{k}^*_A,\mathbf{d}} = \overline{G}^*, A \in \mathcal{A}_{\{\cdot,\cdot\}}, d \in \mathcal{D}\}|$.
6: for all user pairs $A$ and direction vector $\mathbf{d}$ do
7:     if $\overline{G}_{\mathcal{A}_{\{\cdot,\cdot\}},\bar{k}^*_A,\mathbf{d}} == \overline{G}^*$ then
8:         Set $W^*_{A,\mathbf{d}} = \min\left(\overline{W}_{\mathcal{A}_{\{\cdot,\cdot\}},\bar{\mathbf{k}}^*_A,\mathbf{d}}, w_0 g_n\right)$
9:     else if $\overline{E}_{\mathcal{A}_{\{\cdot,\cdot\}},\bar{k}^*_A,\mathbf{d}} > \zeta_1$ & $\overline{G}_{\mathcal{A}_{\{\cdot,\cdot\}},\bar{k}^*_A,\mathbf{d}} < \zeta_2\overline{G}^*$ then
10:        Set $W^*_{A,\mathbf{d}} = \infty$
11:    else
12:        Set $W^*_{A,\mathbf{d}} = \overline{W}_{\mathcal{A}_{\{\cdot,\cdot\}},\bar{\mathbf{k}}^*_A,\mathbf{d}}$
13: Sample $A, \mathbf{d}$ with probability $p_{A,\mathbf{d}} \propto 1/W^*_{A,\mathbf{d}}$.
14: if $\overline{E}_{\mathcal{A}_{\{\cdot,\cdot\}},\mathbf{k}^*,\mathbf{d}} > \zeta_3$ then $\overline{W}_{\mathcal{A}_{\{\cdot,\cdot\}},\mathbf{k}^*,\mathbf{d}} = \overline{W}_{\mathcal{A}_{\{\cdot,\cdot\}},\mathbf{k}^*,\mathbf{d}} + 1$
```

The key idea behind the user-selection algorithm is to find the optimal balance between exploitation, i.e. prioritizing user pairs and direction vectors with high throughput, and exploration, i.e. scheduling user pairs to further improve the estimate of the environment. The pseudo-code for the algorithm for user selection is given in Algorithm 1.

The algorithm starts with initializing the weights $\overline{W}_{\mathcal{A}_{\{\cdot,\cdot\}},k,\mathbf{d}}$ = w_0, for all user cluster pairs $\mathcal{A}_{\{\cdot,\cdot\}}$, channel-state partitions $k \in [\overline{K}_{\mathcal{A}_{\{\cdot,\cdot\}}}]$ and direction vector $\mathbf{d} \in \mathcal{D}$ in the system. At each time step, it uses 4 features: (1) the observed channel-state $\mathbf{q}(t)$, (2) the estimated channel-state classifiers $\overline{\pi}_{\mathcal{A}_{\{\cdot,\cdot\}}}$ generated during the user-cluster initialization phase (see Section III-B), (3) the epoch $\overline{E}_{\mathcal{A}_{\{\cdot,\cdot\}},k,\mathbf{d}}$, from user clustering algorithm, and (4) the reward $\overline{G}_{\mathcal{A}_{\{\cdot,\cdot\}},k,\mathbf{d}}$, the current estimate of the maximum achievable rate.

The algorithm first finds the channel-state partition index of all cluster-pairs $\bar{\mathbf{k}}^* = \{\bar{k}^*_A\}$ using the classifier $\overline{\pi}_{\mathcal{A}_{\{\cdot,\cdot\}}}$ when $A \in \mathcal{A}_{\{\cdot,\cdot\}}$. Using these channel-state partitions indices $\bar{\mathbf{k}}^*$ and the estimated throughput $\overline{G}_{\mathcal{A}_{\{\cdot,\cdot\}},k,\mathbf{d}}$, the algorithm then determines the maximum potential reward $\overline{G}^*$. Then the weights $W^*_{A,\mathbf{d}}$ for different user pairs and direction vectors are set such that the algorithm is incentivized to schedule the user-pairs with optimal rewards, and it significantly penalizes the users who have poor rewards even after long term explorations. Based on the updated weights, the algorithm generates the probability distribution $p_{A,\mathbf{d}}$ on the user-pairs and direction vectors and samples a random user-pair A and the direction vector d from the distribution. Finally, the algorithm updates the weight $\overline{W}_{\mathcal{A}_{\{\cdot,\cdot\}},\mathbf{k}^*,\mathbf{d}}$ by one, for the $(\mathcal{A}_{\{\cdot,\cdot\}}, \mathbf{k}^*, \mathbf{d})$-triplet that have been sufficiently explored. The threshold on epochs is designed so that the users and direction vectors are sampled uniformly at first, as we have a poor estimate of their capacity regions. However, after multiple epochs, when a sufficiently good estimates of their rewards are available, users pair and direction vectors are selected based on their weights to prioritize users pairs and direction vectors with better rewards.

Note that the $\overline{E}_{\mathcal{A}_{\{\cdot,\cdot\}},k,\mathbf{d}}$ is updated at each epoch of the joint user-channel clustering and rate scheduling algorithm and the reward $\overline{G}_{\mathcal{A}_{\{\cdot,\cdot\}},k,\mathbf{d}}$ is updated after every capacity explore stage of that algorithm.

V. Simulation Results

We now present the simulation results of our algorithms on the cellular network simulator, called *WiNGS*, developed within AT&T Labs. WiNGS synthetically generates data packets and pass them through a full-fledged implementation of PDCP, RLC, MAC, and PHY layer protocols. WiNGS then uses wireless channel models to simulate the transmission and provide feedback. More details about the simulator is given in [1]. In the following, we first provide a brief overview of our simulation settings before describing the simulation results. For additional results, please refer to [22].

A. Simulation setting

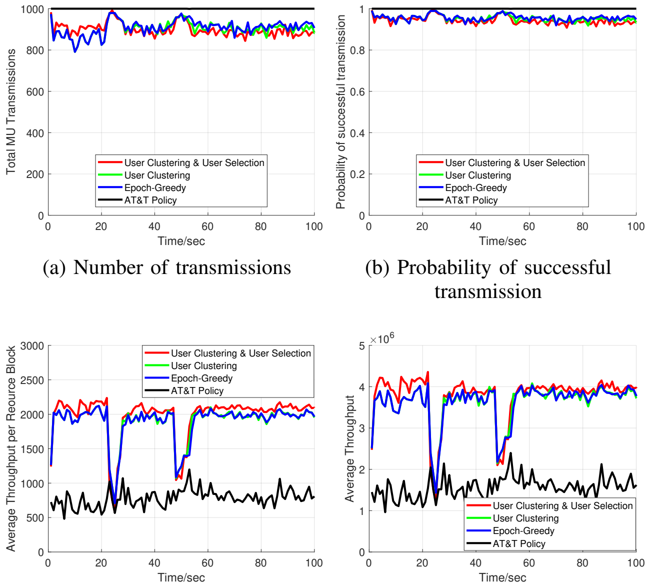
The WiNGS simulator operates in discrete time steps of size 1 ms as follows: First, it provides the set of users with non-empty queues and corresponding user metrics (consisting of CQI, set of pairable users and their MU-SINR, MIMO rank, etc.) at each time step. It then selects the primary user from this user-set based on a proportionally fair rule and the secondary user from the set of the candidate pairable users using our user selection algorithm (in Algorithm 1). Finally, it allocates the resources (chooses MCS indices) for MU-MIMO transmission according to the joint user-channel clustering and rate scheduling algorithm. After each transmission, it receives binary feedback indicating whether the transmission was successful. For the implementation, we used a robust version of our algorithm for the simulations that can schedule the users and learn the environment dynamically as new users arrive in the system at a later time. We have given details of the modifications in [22].

We assume a network with a single base-station and evaluate our algorithm with respect to the baseline AT&T scheduler and the algorithm in [1]. We consider that the user data arrives according to an FTP traffic model, where each user receives a packet of size 10 Mb with an exponential arrival rate of 0.1 packets per sec. Furthermore, we use the threshold value of 0.9 during the capacity explore phase of the algorithm based on the typical service requirement of wireless carriers. We use naive Bayes (implemented in MATLAB) as the class of experts for our simulations. To build the classifier, we test $l_0 = 20$ different rate vectors, and each rate vector is scheduled $l_1 = 30$ times for any given user-pair. For $n_0 = 3000$ time slots, the default AT&T policy is used for scheduling decisions. We exclude these time slots from our results for better clarity.

We consider several different scenarios with varying user dynamics (described later). The number of users, user-clusters, and their locations vary across scenarios, thus, evaluating the algorithms across different channel and interference environments. We test 3 different parameter settings for the AT&T policies for each scenario and plot the results of the parameter settings with the best throughput in each scenario. We consider $D = 5$ direction vectors for our algorithm and the epoch greedy algorithm from [1]. For the epoch greedy algorithm and the user clustering algorithm without the user selection, the simulator selects the direction vector that is most aligned with the direction of the MCS scheduled. For the joint user clustering and user selection algorithm, the direction vector is selected according to Algorithm 1.

B. Results

1) Scenario 1: For the first evaluation, we consider a system with $n = 6$ users and the users belonging to 2 distinct cluster of size 3. At the beginning of the simulation, only a single user from each cluster is activated and receives data from the base station. At the time $t = 20$ second, an additional user from each cluster activates and starts receiving data from the base station. Finally, at time $t = 45$, the remaining 2 users join their corresponding clusters, and all 6 users continue receiving data until the end of the simulation. Figure 1 summarizes the results.



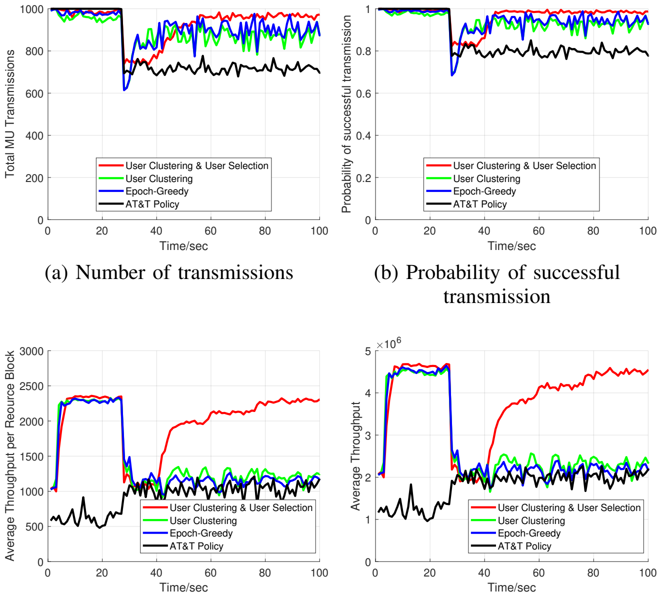
(a) Number of transmissions

(b) Probability of successful transmission

(c) Throughput per resource block

(d) Overall throughput

Fig. 1. Performance comparison for Scenario 1 with 6 users: The overall throughput achieved by our algorithm is 100% more than the throughput achieved by the current state-of-art algorithm.



(a) Number of transmissions

(b) Probability of successful transmission

(c) Throughput per resource block

(d) Overall throughput

Fig. 2. Performance comparison for Scenario 2 with 9 users: The overall throughput achieved by our algorithm is 120% more than the throughput achieved by the current state-of-art.

2) Scenario 2: For the second evaluation, we consider a system with $n = 9$ users where the users belong to 3 distinct cluster of size 3. At the start of the scenario, only 2 users from each cluster are activated and receive the data from the base station. At the time $t = 25$ sec, the final user from each cluster joins the system and begins receiving data. Figure 2 summarizes the results.

C. Summary of results

In all scenarios, we observe that our algorithms learn the environment pretty quickly (10 sec for a 6 user system, 20 sec for a 9 user system). Moreover, in all scenarios, our algorithms match or outperform the current baseline MU scheduling policies used by AT&T as well as the epoch-greedy algorithm proposed in [1]. We also observe that, given the dynamic scenarios, the static AT&T policies require periodic hand-tuning to achieve scenario-specific optimal results. In contrast, the epoch-greedy strategy from [1] may converge to sub-optimal performance. Thus, the user clustering and user-selection algorithms are better as they require less time to learn the environment and ensure better service quality to users. The results show that these algorithms can learn the user clusters and the optimal mappings from the channel-states to MCS for different user cluster pairs in an online manner.

Finally, we compare the benefit of using the user selection algorithm by comparing the performance of the user-clustering algorithm with and without the user selection algorithm. From Fig. 2, we observe a significant increase in the throughput by the user clustering algorithm with the user selection algorithm as compared to the user clustering algorithm without the user selection. This is because the user selection algorithm selects the user-pair according to the estimate of the environment, where fractions of times the optimal user-pair are scheduled for any given channel-state increases as knowledge of the environment gets more accurate. Therefore, by optimally selecting both the user-pairs and their corresponding rate vector, the joint user clustering and the user selection algorithm can achieve significantly higher throughput.

VI. Acknowledgements

This work was supported by NSF grants CNS-1910112, CNS-2107037 and IIS-2112471, Army Futures Command Grant W911NF-19-2-0333, and the Wireless Networking and Communications Group Industrial Affiliates Program.

VII. Conclusion

In this paper, we proposed two online algorithms for MU-MIMO systems with large and dynamic user-sets: (a) a user clustering algorithm based on the similarity in the channel-state distributions and associated rate regions and (b) the heuristic-based user selection algorithm. We derived the theoretical guarantee for the user clustering algorithm and showed that it achieves a sub-linear regret. Additionally, our user selection algorithm is designed to explore and exploit the environment to schedule the optimal user-pair and direction based on the knowledge about the user clusters. Finally, we validated the performance through simulations on state-of-the-art AT&T WiNGS simulator and showed greatly improved performance with respect to the current baselines. ;Furthermore, the user-cluster-based online algorithms significantly reduce learning complexity compared to those without user-clusters, especially in the scenarios with dynamic user-sets.

References

[1] I. Tariq, R. Sen, T. Novlan, S. Akoum, M. Majmundar, G. de Veciana, and S. Shakkottai, “Auto-tuning for cellular scheduling through bandit-learning and low-dimensional clustering,” *IEEE/ACM Trans. Netw.*, vol. 29, no. 5, pp. 1933–1947, 2021.

[2] K. He, Z. Wang, D. Li, F. Zhu, and L. Fan, “Ultra-reliable MU-MIMO detector based on deep learning for 5G/B5G-enabled iot,” *Physical Commun.*, vol. 43, p. 101181, 2020.

[3] A. Klautau, P. Batista, N. González-Prelcic, Y. Wang, and R. W. Heath, “5G MIMO data for machine learning: Application to beam-selection using deep learning,” in *2018 Information Theory and Applications Workshop (ITA)*. IEEE, 2018, pp. 1–9.

[4] M. Chai, S. Tang, M. Zhao, and W. Zhou, “HPNet: a compressed neural network for robust hybrid precoding in multi-user massive MIMO systems,” in *2020 IEEE Global Commun. Conf.* IEEE, 2020, pp. 1–7.

[5] M. Elsayed, K. Shimotakahara, and M. Erol-Kantarci, “Machine learning-based inter-beam inter-cell interference mitigation in mmwave,” in *2020 IEEE Int. Conf. Commun.* IEEE, 2020, pp. 1–6.

[6] C. Ge, S. Xia, Q. Chen, and F. Adachi, “Reinforcement learning-based interference coordination for distributed MU-MIMO,” in *2021 24th Int. Symp. Wireless Pers. Multimedia Commun.* IEEE, 2021, pp. 1–6.

[7] M. Goutay, F. A. Aoudia, J. Hoydis, and J.-M. Gorce, “Machine learning for MU-MIMO receive processing in OFDM systems,” *IEEE J. Sel. Areas Commun.*, vol. 39, no. 8, pp. 2318–2332, 2021.

[8] 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 Process. Mag.*, vol. 24, no. 5, pp. 36–46, 2007.

[9] E. Castaneda, A. Silva, A. Gameiro, and M. Kountouris, “An overview on resource allocation techniques for multi-user MIMO systems,” *IEEE Commun. Surv. & Tut.*, vol. 19, no. 1, pp. 239–284, 2016.

[10] K. Ko and J. Lee, “Multiuser MIMO user selection based on chordal distance,” *IEEE Trans. Commun.*, vol. 60, no. 3, pp. 649–654, 2012.

[11] R. Tian, Y. Liang, X. Tan, and T. Li, “Overlapping user grouping in IoT oriented massive MIMO systems,” *IEEE Access*, vol. 5, 2017.

[12] F. H. C. Neto and T. F. Maciel, “SDMA grouping based on unsupervised learning for multi-user MIMO systems,” *J. Commun. and Inf. Syst.*, vol. 35, no. 1, pp. 124–132, 2020.

[13] D. Marasinghe, N. Jayaweera, N. Rajatheva, and M. Latva-Aho, “Hierarchical user clustering for mmwave-NOMA systems,” in *2020 2nd 6G Wireless Summit (6G SUMMIT)*. IEEE, 2020, pp. 1–5.

[14] J. Ren, Z. Wang, M. Xu, F. Fang, and Z. Ding, “Unsupervised user clustering in non-orthogonal multiple access,” in *2019 IEEE Int. Conf. Acoust., Speech and Signal Process.* IEEE, 2019, pp. 3332–3336.

[15] J. Jiang, J. Chen, Y. Xie, H. Lei, and L. Zheng, “Modified-pbil based user selection for multi-user massive MIMO systems with massive connectivity,” in *IEEE Conf. on Comput. Commun. Workshops (INFOCOM WKSHPS)*. IEEE, 2020, pp. 1260–1265.

[16] Y. Yang, Y. Li, K. Li, S. Zhao, R. Chen, J. Wang, and S. Ci, “DECCO: Deep-learning enabled coverage and capacity optimization for massive MIMO systems,” *IEEE Access*, vol. 6, pp. 23 361–23 371, 2018.

[17] W. V. Mauricio, T. F. Maciel, A. Klein, and F. R. M. Lima, “Learning-based scheduling: Contextual bandits for massive MIMO systems,” in *2020 IEEE Int. Conf. Commun. Workshops*. IEEE, 2020, pp. 1–6.

[18] I. Tariq, R. Sen, G. de Veciana, and S. Shakkottai, “Online channel-state clustering and multiuser capacity learning for wireless scheduling,” in *IEEE Conf. Comput. Commun.* IEEE, 2019, pp. 136–144.

[19] J. Langford and T. Zhang, “The epoch-greedy algorithm for multi-armed bandits with side information,” in *Adv. in Neural Inf. Process. Syst.*, 2008, pp. 817–824.

[20] A. Agarwal, M. Dudík, S. Kale, J. Langford, and R. Schapire, “Contextual bandit learning with predictable rewards,” in *Artif. Intell. and Statist.*, 2012, pp. 19–26.

[21] 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.

[22] I. Tariq, K. Patel, T. Novlan, S. Akoum, M. Majmundar, G. de Veciana, and S. Shakkottai, “Bandit learning-based online user clustering and selection for cellular networks,” Technical Report, UT Austin, May 2022.