Helix: A RAN Slicing Based Scheduling Framework for Massive MIMO Networks

QING AN, Rice University, USA
DIVYANSHU PANDEY, Rice University, USA
RAHMAN DOOST-MOHAMMADY, Rice University, USA
ASHUTOSH SABHARWAL, Rice University, USA
SRINIVAS SHAKKOTTAI, Texas A&M University, USA

An important aspect of 5G networks is the development of Radio Access Network (RAN) slicing, a concept wherein the virtualized infrastructure of wireless networks is subdivided into slices (or enterprises), tailored to fulfill specific use-cases. A key focus in this context is the efficient radio resource allocation to meet various enterprises' service-level agreements (SLAs). In this work, we introduce Helix: a channel-aware and SLA-aware RAN slicing framework for massive multiple input multiple output (MIMO) networks where resource allocation extends to incorporate the spatial dimension available through beamforming. Essentially, the same time-frequency resource block (RB) can be shared across multiple users through multiple antennas. Notably, certain enterprises, particularly those operating critical infrastructure, necessitate dedicated RB allocation, denoted as private networks, to ensure security. Conversely, some enterprises would allow resource sharing with others in the public network to maintain network performance while minimizing capital expenditure. Building upon this understanding, Helix comprises scheduling schemes under both scenarios: where different slices share the same set of RBs, and where they require exclusivity of allocated RBs. We validate the efficacy of our proposed schedulers through simulation by utilizing a channel data set collected from a real-world massive MIMO testbed. Our assessments demonstrate that resource sharing across slices using our approach can lead up to 60.9% reduction in RB usage compared to other approaches. Moreover, our proposed schedulers exhibit significantly enhanced operational efficiency, with significantly faster running time compared to exhaustive greedy approaches while meeting the stringent 5G sub-millisecond-level latency requirement.

CCS Concepts: • **Networks** → **Network resources allocation**.

Additional Key Words and Phrases: RAN slicing, Resource scheduling, Massive MIMO networks

ACM Reference Format:
Qing An, Divyanshu Pandey, Rahman Doost-Mohammady, Ashutosh Sabharwal, and Srinivas Shakkottai. 2024. Helix: A RAN Slicing Based Scheduling Framework for Massive MIMO Networks. *Proc. ACM Netw.* 2, CoNEXT4, Article 27 (December 2024), 24 pages. https://doi.org/10.1145/3696399

1 INTRODUCTION

Massive MIMO has been an important part of the 5th Generation (5G) mobile network rollout, which has been in progress since the early 2020s. This technology significantly enhances overall network throughput by leveraging large-order multi-user MIMO (MU-MIMO) enabled through advanced beamforming techniques. However, the capital expenditure (CAPEX) associated with

Authors' Contact Information: Qing An, qa4@rice.edu, Rice University, USA; Divyanshu Pandey, dp76@rice.edu, Rice University, USA; Rahman Doost-Mohammady, doost@rice.edu, Rice University, USA; Ashutosh Sabharwal, ashu@rice.edu, Rice University, USA; Srinivas Shakkottai, sshakkot@tamu.edu, Texas A&M University, USA.

ACM 2834-5509/2024/12-ART27
https://doi.org/10.1145/3696399

deploying Massive MIMO arrays is considerably higher than the lower-order MIMO base stations (BS) used in 4G and earlier 5G networks. This increased CAPEX necessitates meticulous planning by mobile network operators (MNOs) to ensure a balanced cost-benefit ratio for Massive MIMO deployment. Therefore, meeting as many user service requirements as possible with limited physical infrastructure is a critical challenge for MNOs. Standing at the customers' points, based on their required types of services, such as throughput, latency or massive connectivity, they have diverse bandwidth demands. Meeting their network needs with the least amount of resources and costs is their top concern. Network slicing emerges as a solution to this challenge by enabling the partitioning and sharing of network resources across multiple slices or enterprises on a single physical infrastructure. Each enterprise typically has distinct use cases. For instance, entities with critical infrastructure or mission-critical applications, such as hospital and army networks, prioritize security and prefer not to share radio resources with others. These enterprises opt for private networks, which are accessible only to a specific set of devices. Furthermore, in order to save costs, most enterprises choose dependent private networks [45], wherein MNOs allocate spectrum to the enterprise based on cost and spectrum availability and are responsible for network deployment and maintenance. Conversely, enterprises in sectors such as retail and home networks, which do not have significant security concerns, are more willing to share radio resources with others to fulfill their service-level agreements (SLAs) cost-effectively. Such enterprises can be deployed in public networks, where they share the same resource blocks (RBs) with users from multiple enterprises.

Previous work [36, 43] proposed the allocation of different RBs in LTE and 5G to various slices. Recent advancements, such as RadioSaber [18], have introduced a channel-aware allocation of RBs to each slice to maximize network throughput. RadioSaber's architecture includes an inter-slice scheduler that polls different slices about their channel quality on each resource block and allocates them to slices that can achieve the highest rates. This approach separates the user allocation decision-making from the overall slice decision-making, allowing each intra-slice scheduler to manage its own users on the allocated RBs. However, Designing channel-aware RAN slicing for massive MIMO networks presents significantly greater challenges than Single-Input Single-Output (SISO) systems considered in earlier works, such as RadioSaber. These challenges are two-fold: First, beamforming for multiple users must account for channel correlation to avoid high inter-user interference and degraded throughput. Consequently, schedulers must consider user correlation and cannot rely solely on the best channel quality within each slice. Second, the computational complexity of separate inter- and intra-slice scheduling grows exponentially with the number of RBs, slices, and UEs, making it infeasible to meet 5G's sub-millisecond latency requirements. Thus, a scheduling framework is much needed wherein the intra- and inter-slice scheduling are performed jointly while still maintaining each individual slice's SLA guarantees and privacy requirements. In addition to being channel-aware, the scheduler must also be SLA-aware. This implies that the allocation of RBs should be guided by the specific SLAs of the enterprise to utilize the minimal number of RBs necessary to satisfy the enterprise's SLA requirements. By minimizing RB usage to meet SLA requirements, additional resources are freed for more users or other transmission needs, such as control and sensing signals. This approach benefits both enterprises and network operators.

In this paper, we present Helix, a RAN slicing-based scheduling framework for massive MIMO networks which is both SLA-aware and channel-aware. The framework architecture is depicted in Fig. 1. Helix not only allocates RBs to slices and users in an SLA-aware and channel-aware manner but also extends the allocation to spatial resources (beams), thereby going beyond the frequency-time resources considered in earlier works. Helix exploits the opportunity to allocate the same RB to multiple UEs simultaneously by leveraging multiple spatial dimensions available in massive MIMO. Two distinct use-case scenarios emerge in such cases. First, when each slice does not want an RB allocated to itself to be shared with UEs of other slices, such as in private networks

where data security and privacy are crucial. Second, when slices only want assurance of their SLAs being met while being acceptable of resource sharing across slices, such as in public networks. Henceforth, we refer to the former as the RB-orthogonal case and the latter as the RB-sharing case. The orthogonality of RBs ensures that any given RB is allocated exclusively to a specific slice in any scheduling instant, and multiple UEs within that same slice can only use it. RB sharing ensures that UEs from multiple slices can simultaneously be allocated the same RB. In this work, we propose scheduling algorithms for both RB orthogonal and RB sharing. Since our primary objective is cost-saving for operators and enterprises, the scheduling problem posed in this work aims to minimize the number of resource blocks required to meet the SLA guarantees of each slice. The main contributions of the paper are as follows:

- First framework for massive MIMO RAN slicing with SLA guarantees with the goal of optimizing resource usage, e.g. RBs.
- RB-orthogonal and sharing algorithms are proposed with different levels of slice autonomy.
- The proposed algorithms offer near-optimal performance with significantly lower complexity compared to exhaustive search methods, meeting 5G latency requirements.
- An exhaustive evaluation of the proposed schedulers on a real-world massive MIMO channel dataset under different network size configurations, correlation cases, mobility scenarios, and SLA constraints highlights the trade-offs between different schemes.

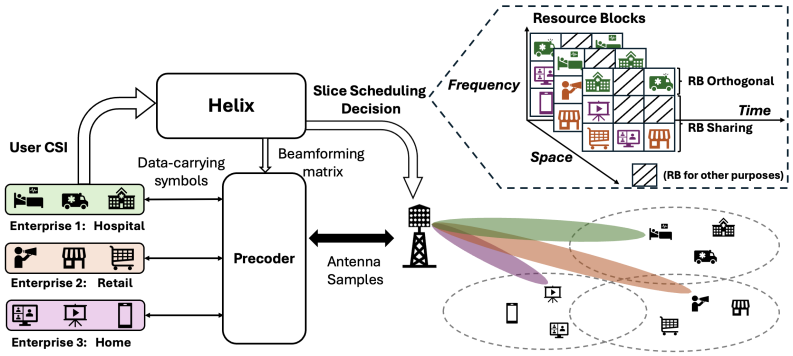


Fig. 1. RAN Slicing based scheduling architecture in massive MIMO networks.

2 BACKGROUND AND RELATED WORK

In this section, we provide a brief background on massive MIMO and its adoption in the 5G standard. We also discuss the problem of RAN slicing and its challenge as it pertains to massive MIMO. A literature review of RAN slicing and massive MIMO resource scheduling is also presented.

Massive MIMO beamforming. In theory, massive MIMO is referred to a cellular base station with an unlimited number of antennas. In [7], it is shown that with M antennas at the base station and N single-antenna users, as the ratio M/N goes to infinity, and when the channel is considered Gaussian, the beamforming matrix that maps the users' data to antenna is simply the conjugate of the $M \times N$ channel matrix $\mathbf{H}$ to a scale. In practice, however, there is a limit to the number of antennas at a massive MIMO base station. The current deployment of massive MIMO in 5G is limited to 64 to 128 antennas [12]. The beamforming operation is typically done through Zero Forcing (ZF) or its regularized variant. The ZF beamformer is calculated as: $\mathbf{W} = \mathbf{H}(\mathbf{H}^H\mathbf{H})^{-1}$. The channel matrix is estimated through uplink pilot signals transmitted by each UE to the base station.

In the time-division duplex (TDD) regime, the same estimated channel matrix is often used for both uplink and downlink due to channel reciprocity.

Numerology of 5G Frame Structure. 5G cellular networks operate based on OFDM signaling, where data is transmitted across frequency subcarriers. The 5G frame format includes 10 ms frames with slots of typically 1 ms length and 14 OFDM symbols [40]. The number of subcarriers in each OFDM symbol is a function of the system bandwidth. The subcarriers across all OFDM symbols of a slot are grouped into resource blocks (RB), each containing 12 subcarriers. For e.g., in a 100 MHz 5G system, there are 273 resource blocks. A special pilot signal known as the sounding reference signal (SRS) is used to obtain the channel estimate for beamforming in massive MIMO [34]. Through the channel estimate obtained from the SRS pilots, the massive MIMO 5G base station can schedule multiple users in each RB for data transmission or reception through multi-user beamforming. Thus, resources are scheduled across all time, frequency, and spatial domains.

Massive MIMO Resource Scheduling. The resource scheduling problem in massive MIMO is significantly more complex than in SISO systems. In the SISO case, the scheduler can allocate an RB to a user with the highest rate. In massive MIMO, however, the rate-optimal allocation of multiple users to each RB depends on how much rate the groups of users can collectively achieve. Therefore, a rate-optimal scheduler should solve a combinatorial problem that considers all combinations of users. The same goes with the proportional-fair or max-min fair algorithms that try to provide fairness along with maximizing rate. Such combinatorial problems are provably NP-hard and clearly not feasible to run in massive MIMO networks of 5G, which require very low latency in scheduling decisions [17]. Many related works propose sub-optimal heuristic-based MU-MIMO schedulers [4, 13, 24, 25, 41]. While they try to strike a balance between complexity and performance, their complexity does not scale to large networks or they significantly underperform the optimal scheduling policies. With the aid of prompt inference of AI models, several ML-models are proposed [10, 14, 15, 26, 30, 42, 47], which leverage deep reinforcement learning for optimal user selection, yet their applicability across diverse traffic scenarios remains limited. This limitation arises from the need for extensive training datasets to ensure the model can generalize effectively across varying environments and conditions [53]. Traditional schedulers focus on individual user requirements, often overlooking groups with similar QoS needs. In 5G, there is increasing emphasis on addressing enterprise-level QoS [48]. Unlike conventional uniform resource allocation, Helix enables network customization for different service types, supporting RB orthogonality or sharing to meet diverse 5G performance demands.

Massive MIMO RAN Slicing. RAN slicing has emerged as a prominent strategy for virtualizing radio resources, enhancing cost and power efficiency by accommodating multiple service slices within the network [37]. Network Virtualization Substrate (NVS) [36] allocates all RBs within a TTI to a single slice, employing a weighted round-robin approach to meet each slice's target throughput requirements specified in the SLA. However, NVS lacks channel awareness, leading to suboptimal resource allocation decisions. Advanced RAN virtualization frameworks like Orion and Scope [9, 23] build upon NVS, aiming to enhance resource allocation efficiency. RadioSaber [18] further refines this approach by incorporating channel awareness into the slice-level scheduler, optimizing RB allocation based on users' Channel Quality Indicators (CQIs). Ensuring slice-level service quality is a primary concern across various RAN slicing proposals [5, 11, 39]. Recent efforts, such as Zipper [6], a ML-based algorithm to compute SLA-compliant schedules in real-time, albeit with a focus on application-level service assurance. Despite these advancements, existing RAN slicing frameworks predominantly focus on single-antenna systems and commonly employ slot-based and RB-based slicing methodologies [18, 23, 36]. In summary, the slicing management framework guarantees SLA compliance by allocating resources such as slots or RBs to different slices based on metrics like average throughput or latency. Similar concepts can be used in massive MIMO

RAN slicing. However, spatial streams represent a third dimension, alongside time and frequency dimensions. While allocating spatial streams in each RB to different slices enhances cost/power efficiency by resource reuse, spatial stream slicing is less effective than slot and RB slicing due to residual inter-user interference based on the effectiveness of the beamforming procedure. This interference can impact operator services. Additionally, slices utilizing spatial streams in shared RBs must employ a common precoder, conceding some functionality to the slice management framework. In §3, we discuss massive MIMO RAN slicing and the design space in more detail.

Schedulers in Other Domains. While extensive research exists on scheduler design in areas such as cloud services [16, 20, 28, 54] and AI/ML model training [29, 44, 52], scheduling in 5G MIMO networks faces unique challenges due to the dynamic nature of wireless channels, affected by fading, interference, and mobility. 5G must also accommodate diverse traffic types (eMBB, URLLC, mMTC) with varying QoS needs and make real-time decisions under strict sub-millisecond latency. In contrast, cloud and AI/ML schedulers operate in controlled environments with flexible timing constraints (several milliseconds to seconds) [19]. Thus, the latency requirements make Helix design more challenging as opposed to the schedulers used in other domains.

3 PROBLEM OVERVIEW

Based on the enterprises' policy regarding sharing of radio resources and the types of services they request, RB allocation can be classified into two mechanisms: RB-orthogonal and RB-sharing. In the RB-orthogonal, slices prefer exclusive use of radio resource blocks without sharing, a scenario typically encountered in private networks. Conversely, the RB-sharing involves slices that are willing to share RBs with others to fulfill SLAs cost-effectively, which is common in public networks. MNOs allocate spectrum to these two sets of slices based on cost considerations and radio resource availability. The Helix framework provides scheduling solutions for both allocation mechanisms, thus it supports the operation of both private and public 5G network functionalities on a single base station by employing either cooperative or exclusive resource allocation across slices.

3.1 RB-Orthogonal: Problem Formulation

In this case, any given RB can only be allocated to users from the same slice. This constraints the optimal usage of each RB; however, it might be necessary due to security and privacy issues that stem from resource sharing across different slices.

Inter- and Intra-slice scheduler : Within the RB-Orthogonal case, we can have two situations: First, separate inter- and intra-slice schedulers make decisions at slice and user levels, respectively. To do this, the intra-slice scheduler needs to let the inter-slice scheduler know which user it intends to serve if given a certain RB, depending on its own private policy. The inter-slice scheduler allocates RBs to various slices, while the intra-slice scheduler assigns these RBs to individual users within each slice. Second, where there is a single scheduler that directly takes decisions at the user level while maintaining the slice SLAs. Note that having a separation between inter- and intra-slice schedulers allows a level of private autonomy for each slice (enterprise) to decide its own scheduling policy. The intra-slice scheduler can select users based on any customizable policy such as Proportional Fairness (PF) or Maximum Rate (MR). The inter-slice scheduler only schedules a slice depending on the response to its query from the slice and does not dictate the scheduling at the user level. However, responding to inter-slice queries often requires an exhaustive intra-slice search. Thus, computationally, it is more efficient to let a single scheduler directly take a decision at the user level. The latter approach can also accommodate various scheduling policies concurrently.

Let $\mathcal{B}, \mathcal{S}$ denote the set of RBs and slices respectively. Every slice $s \in \mathcal{S}$ contains a distinct set of users denoted as $\mathcal{K}_s$ such that $\sum_s |\mathcal{K}_s| = N$, where N is total number of users. Let us denote $x_k^{b,t}$ and

$x_s^{b,t}$ as the user-level and slice-level binary decision variables for user k on RB b respectively, such that $x_s^{b,t} = 1, \exists\, k \in \mathcal{K}_s$, s.t. $x_k^{b,t} = 1$. Subsequently, the scheduling problem for the RB-orthogonal case can be written as the following optimization problem:

$$\min_{x\in\{0,1\}} \mathbb{E}_t\left[\sum_{s\in\mathcal{S}}\sum_{b\in\mathcal{B}} x_s^{b,t}\right] \quad \text{s.t.} \sum_{s\in\mathcal{S}} x_s^{b,t} \le 1, \forall b, \forall t \quad \text{and} \quad \mathbb{E}_t\left[\sum_{b\in\mathcal{B}} r_s^{b,t}\right] \ge \gamma_s, \forall s \tag{1}$$

where the optimization variable x denotes $x_s^{b,t}$ if we have a separate inter- and intra-slice scheduler, and x denotes $x_k^{b,t}$ when there is a single scheduler taking decisions at the user level. The objective function remains the same in both cases. The entity $r_s^{b,t}$ is the achieved data rate of slice s on RB b at TTI t (detailed derivation is given in Appendix A) and γ_s is the SLA in terms of minimum throughput guarantee for slice s on average across several TTIs. The notation $\mathbb{E}_t[\cdot]$ denotes average across t. Further, at any given TTI instant t', SLA deficit of slice s can be computed as:

$$\Delta_s^{t'} = \begin{cases} \gamma_s, & \text{if } t' = 0 \\ \max\left\{0,\ (t'+1)\gamma_s - \sum_{t=0}^{t'-1}\sum_{b\in\mathcal{B}} r_s^{b,t}\right\}, & \text{if } t' > 0 \end{cases} \tag{2}$$

The objective function in Eq. (1) is to minimize the average allocated resource blocks to all the slices across time. The first constraint in Eq. (1) imposes the orthogonality of RB allocation, and the second constraint ensures that the SLA requirement is met for all slices. This optimization problem can be abstracted as binary integer programming (BIP), which is NP-complete. Heuristic methods like Branch and Bound (B&B)[31] have been proposed, but they face high computational complexity, especially with increasing binary variables and constraints[49].

3.2 RB-Sharing: Problem Formulation

RB-sharing maximizes the utilization of RBs by sharing it with users from multiple slices. Thus, there is no notion of a separation between inter- and intra-slice schedulers in this case. The corresponding optimization problem for the RB-sharing case can be formulated as:

$$\min_{x_k^{b,t}\in\{0,1\}} \mathbb{E}_t\left[\sum_{b\in\mathcal{B}} \min(1, \sum_{k=1}^{N} x_k^{b,t})\right] \quad \text{s.t.} \quad \mathbb{E}_t\left[\sum_{b\in\mathcal{B}} r_s^{b,t}\right] \ge \gamma_s, \forall s. \tag{3}$$

This objective function is also a combinatorial problem and NP-hard. It is more computationally complicated than RB-orthogonal because user selection on each RB extends from users within the slice to all users in the network, which makes the search space exponentially larger.

Note that in this work we consider two extreme cases of resource sharing - one where each slice wants exclusivity of resources allocated (RB orthogonal), and another where all slices can share the resources (RB sharing). However, in practice, a subset of slices being served by an MNO may allow sharing while another subset may want exclusive access to resources. Thus, the MNO can employ a mix of RB sharing and orthogonal approaches depending on the policies of the enterprises it serves.

4 HELIX DESIGN

To minimize the cost for both operators and enterprises, we align the objectives of scheduler design in both RB-orthogonal and RB-sharing scenarios to focus on fulfilling SLAs of slices while minimizing the consumption of resource blocks. Consequently, the scheduler must be both channel-aware and SLA-aware, as discussed in §1. Additionally, maintaining low computational complexity is another design criterion, as emphasized in §3, such that the scheduler can be implemented within each TTI. Consequently, we propose Helix that includes scheduling algorithms for both

RB-orthogonal and RB-sharing. An overview of implemented scheduler algorithms in this work is depicted in Fig. 2. Notably, resource allocation in SISO networks relies on RB-orthogonality, as each RB is only occupied by a single user. Therefore, in §5, we adapt [18] for MIMO networks, utilizing it as a baseline in the RB-orthogonal scenario.

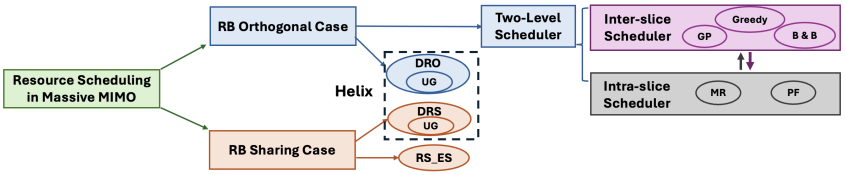


Fig. 2. Overview of implemented scheduling algorithms.

4.1 User Grouping (UG)

Since Helix aims to schedule multiple users on the same RB simultaneously by using different spatial streams, an appropriate grouping of users based on inter-user correlation is required to guide the scheduler design. Thus, as a prelude to our proposed scheduling algorithms, we first present a user-grouping approach. Suppose $\mathcal{N}$ denotes the set of all users. In that case, the objective of user-grouping is to partition $\mathcal{N}$ into disjoint sets $\{\mathcal{N}_1, \mathcal{N}_2, \ldots, \mathcal{N}_\eta\}$ such that η is minimum and users within any $\mathcal{N}_i$ have low correlation. The inter-user correlation between user i and j can be calculated using the CSI vectors as stated in [50]. Accordingly, a size $N \times N$ binary user correlation matrix $\mathbf{G}$ can be generated by defining a correlation threshold c_{th} [50]. The determination of the threshold is discussed in §5.1. If $c_{i,j} > c_{th}$, the (i, j)th element of the correlation matrix, $\mathbf{G}_{i,j} = 1$ otherwise, $\mathbf{G}_{i,j} = 0$. The diagonal values of this matrix are all set to 0. This binary user correlation matrix can then be represented as a graph where each vertex represents a user and a 1 in the $(i, j)th$ index of the correlation matrix corresponds to an edge between the ith and jth vertices of the graph. Since we want to form groups of users with low correlation, the graph representation allows us to solve user grouping as a graph coloring problem [33]. Graph coloring involves assigning colors to the vertices of a graph using a minimum number of colors such that no two adjacent vertices share the same color. The vertices (users) with the same colors can be grouped together, the number of colors denote the number of user groups η. Graph coloring is a well-researched topic for which several algorithms are proposed in the literature [8, 46]. With the help of parallel computing techniques, these algorithms are efficient and scalable to handle thousands of vertices. Since user grouping is a function of correlation statistics, it may not significantly vary across different b and t specially with slow time-varying channels. Hence, depending on practical considerations, computing such user grouping only once, or updating it only after several TTIs should suffice.

4.2 Scheduler for RB-Orthogonal case

In this section, we propose two algorithms: Greedy Plus (GP) and Delta Algorithm for the RB-Orthogonal (DRO). Greedy Plus (GP) is an inter-slice scheduler that operates separately from the intra-slice scheduler. DRO directly takes scheduling decisions at the user level without separating inter- and intra-slice schedulers, thereby avoiding an exhaustive search by any intermediary. Note that since Eq. (1) can be formulated as a binary integer programming (BIP) problem, we employ Gurobi optimizer [27] to numerically provide an optimal solution using the Branch and Bound (B&B) [31]. The complexity of solvers such as Gurobi is too high to meet the stringent latency requirements of 5G networks. Consequently, we will use the optimal solution generated by Gurobi only to evaluate our algorithm in §5 as a benchmark method.

4.2.1 Greedy and Greedy Plus (GP) Approaches . To make the scheduling process channel-aware, the intra-slice scheduler responds to the following query by the inter-slice scheduler: which user will receive a given RB if it is allocated to the slice by the inter-slice scheduler? Based on the response from such a query, the inter-slice scheduler in [18] greedily allocates RBs while prioritizing slices with favorable channel quality and ceases allocation once the RB quota is met. Such a greedy inter-slice scheduler, as proposed in [18], makes locally optimal decisions at each step, which may fail to satisfy the long-term SLAs.

An improvement of the Greedy approach, which we refer as Greedy Plus (GP), recognizes average SLA deficits in Eq. (2) and allocate the most suitable RBs to each slice to meet SLAs. Thus GP is cognizant of scenarios where assigning the RB that greedily provides the highest data rate to a user is not recommended if the corresponding SLA deficit is too small. GP sorts RBs based on achievable data rates within each slice rather than globally and starts allocation with the slice with the largest SLA deficits. After each RB allocation, it updates the SLA deficits using Eq. (2) and removes the allocated RB to a slice from the sorting lists of other slices. The scheduling of a slice will conclude once its SLA is met. An illustrative example is presented in Fig 3 showing that Greedy approach can fail to meet the SLA of each slice while over-serve a few slices with good channel. GP ensures no SLA violations, and performs similar to Gurobi.

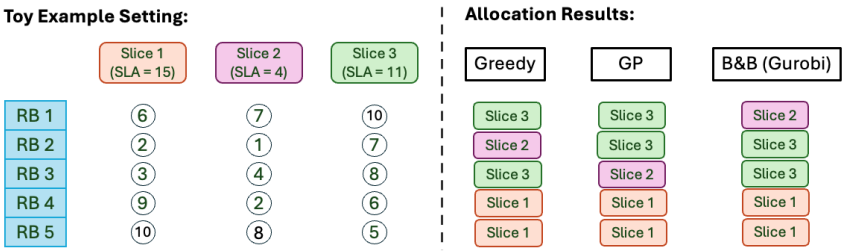


Fig. 3. Illustrative Example Comparing Greedy, Greedy Plus, and Gurobi: Consider 3 slices and 5 RBs. The achieved data rates by intra-slice schedulers are shown in circles. *Greedy* - $RB5 \rightarrow S1$, $RB1 \rightarrow S3$, $RB4 \rightarrow S1$, after which $S1$ has exceeded its SLA. Then, $RB3 \rightarrow S3$, $RB2 \rightarrow S2$. *GP* - Sort RBs in descending order within each slice, then serve the slice with largest SLA, thus $RB5 \rightarrow S1$, remove RB5 from lists of other slices and update SLA deficit of S1. Next S3 has largest deficit, thus $RB1 \rightarrow S3$. Further, $RB4 \rightarrow S1$, completing S1's SLA requirement. Further, $RB5 \rightarrow S1$, $RB3 \rightarrow S2$, and $RB2 \rightarrow S3$. *Gurobi*-employs Branch and Bound method.

4.2.2 Delta RB Orthogonal (DRO). Performace of any scheduling scheme with separate inter- and intra-slice schedulers depends on how quickly the intra-slice scheduler can respond to inter-slice scheduler's query regarding the best user. In SISO networks, this is feasible within a short time since the best user can be found using CQI [18]. However, in MIMO networks, CQI is not as reliable because CQI acquisitions for beamformed users are independent, meaning each user may not be aware of the number of other MU-MIMO layers being used. Consequently, CQI feedback lacks context, which is even more problematic in massive MIMO networks [2]. To accurately estimate the achieved data rate of each potential user set on an individual RB, Shannon capacity formula [51] can be used. We propose a low-complexity method that identifies the user combination with the highest potential to achieve the maximum rate on each RB without any exhaustive search.

Our proposed approach selects a user set by evaluating user channel gain and inter-user correlation, ensuring that RBs are allocated to users with low correlation and good channel quality. To this end, user-grouping proposed in §4.1 can be used. Additionally, we prioritize scheduling for slices with larger SLA deficits. Let us refer to the set of all slices with non-zero SLA deficit as active

slice, denote it as $\mathcal{S}^t_{act}$, and the union of the set of all users in $\mathcal{S}^t_{act}$ as $\mathcal{K}_{act}$. Then we partition this set into two sub-sets of users $\mathcal{K}^t_{act} = G^t_l \cup G^t_s$ defined as:

$$G^t_l = \{i : \Delta^t_{slice(i)} \geq \Delta^t_{avg}\} \text{ and } G^t_s = \{j : \Delta^t_{slice(j)} < \Delta^t_{avg}\}, \text{ where } \Delta^t_{avg} = \frac{\sum_{s \in \mathcal{S}_{act}} \Delta^t_s}{|\mathcal{S}^t_{act}|}. \quad (4)$$

The function $slice(i)$ denotes the slice of user i, and the quantity Δ^t_{avg}, denotes the average of all non-zero Δ^t_s from Eq. (2). Further, G^t_l and G^t_s denote subset of slices with more than and less than average SLA deficit respectively, making the users within G^t_l the higher priority set of users. From G^t_l, we pick the user-RB (k', b') pair with the highest channel gain as:

$$(k', b') = \underset{k \in G^t_l, b \in \mathcal{B}}{\arg\max} \quad ||\mathbf{h}^{b,t}_k||^2. \quad (5)$$

This allows us to initiate allocation from RB b' and user $k' \in G^t_l$ where $slice(k') = s'$. Given that this resource allocation occurs in massive MIMO networks, and we want to ensure RB orthogonality, we must also select other users from the same slice s'. As discussed in [7], increasing the number of scheduled users K in massive MIMO is not always beneficial, and there is an optimal M/K ratio for maximizing spectral efficiency. While our proposed algorithm applies to any generic $K < N$ value, for implementation, K can be determined following the guidelines in [7]. Thus, after finding a user k' in slice s' to be served, we need to find $K-1$ additional users, which should have a low inter-user correlation from slice s' to schedule with user k' on RB b'. Therefore, we group users into clusters where users exhibit low correlation with each other using the technique proposed in §4.1. We then identify the group containing user u' and select the top $K-1$ users with the highest channel gain from the same slice s' for scheduling. If the group contains less than K users, we find user j with the highest channel gain on RB b' from other groups and continue selecting uncorrelated users from user j's group until we have total K users or all users in slice s' are selected. This approach may introduce some inter-user interference, but increasing the number of scheduled users (as long as it remains below K) will compensate with higher data rates. After selecting the user set for scheduling on RB b', we estimate the total achieved rate of slice s' using the Shannon capacity equation [51] and update $\Delta^t_{s'}$. The scheduler continues updating SLA deficits to allocate RBs and user sets until all slice SLAs are fulfilled. A flowchart illustration of Helix algorithm is shown in Fig. 4 and the detailed algorithmic implementation is provided in Appendix B.

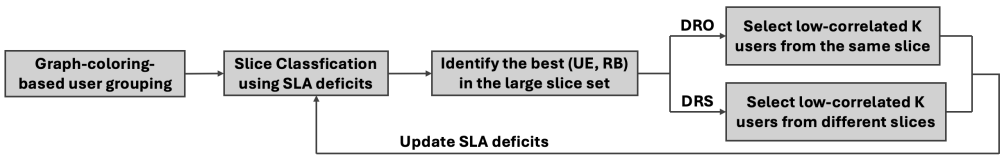


Fig. 4. Flowchart illustration of Helix.

4.3 RB Sharing Case

RB-Orthogonal does not provide an efficient solution when the users of a single slice are high-correlated. For example, in cluttered environments where users from the same slice are concentrated in a single location, such as in a crowded shopping mall, high inter-user interference occurs due to their strong correlation, leading to reduced system performance. Besides, in scenarios with sparse slices that contain very few active users (i.e., the number of users is fewer than K), such as resident networks in late night hours, exclusively assigning RB to these slices results in inefficient

resource utilization. The RB sharing approach addresses these issues by enabling radio resources to be shared among users from all slices, thus enhancing RB utilization. This method operates in a fully centralized manner and collaboratively determines the optimal user combination across slices. While RB sharing can achieve superior resource saving compared to the RB-orthogonal approach, it incurs a higher computational overhead due to the increased complexity of user and RB combinations as outlined in §3. Consequently, an exhaustive search for the optimal solution in a large-size network is impractical, so we propose a scheduler with low computational complexity.

4.3.1 Delta RB Sharing (DRS). Building on the principles of DRO, we introduce a low-complexity approach tailored for the RB sharing scenario. The core concept involves prioritizing slices with substantial SLA deficits and allocating RBs to users with high channel quality and low correlation. Initially, we classify slices based on the average SLA deficit Δ_{avg} of active slices, similar to the RB-orthogonal scenario. Next, we identify the user-RB pair with the highest channel gain to initiate the allocation process. We group users as explained in §4.1 and identify the user group containing the selected user and pick additional $K-1$ users in descending order of channel gain. Unlike the RB-orthogonal scenario, where users must be selected from the same slice, RB-sharing allows users to be selected from any slice. However, users from slices with larger deficits are given higher priority, and users from slices with smaller deficits are only selected if insufficient users are available in the larger deficit group. SLA deficit update follows the same procedure as in the RB-orthogonal scenario, with sequential RB allocation continuing until all SLAs are met, as detailed in Algorithm 1.

4.3.2 Exhaustive search in RB-Sharing (RS_ES). To compare the performance of DRS in RB sharing with a benchmark method, we implement a near-optimal method known as RS_ES, which performs a user-level exhaustive search. Specifically, RS_ES first identifies the initial (k', b') user-RB pair, similar to DRS. However, for selecting the remaining $K-1$ scheduled users on the same RB, unlike selecting for a fixed user group, RS_ES conducts an exhaustive search among all users to find the optimal set of UEs. It then updates the SLA deficit following the same procedure as in DRS, picks the next best RB-user pair, and continues the process till all SLAs are met. Note that performing an exhaustive search without picking an initial user-RB pair would be a computationally infeasible approach even for small-scale networks since the search space increases exponentially with both the number of resource blocks and the number of users. Thus, RS_ES provides a near-optimal solution by partly introducing user-level exhaustive search within the DRS approach.

4.4 RB Parallelism

In both DRO and DRS, RB allocation occurs sequentially because the outcomes of previous RB allocations adjust the SLA deficits, thereby affecting the priority of slices in subsequent allocations. This process determines whether a slice is categorized into a large or small delta group for the next allocation round. However, this sequential allocation mechanism leads to increased execution time, potentially violating the stringent latency requirements of 5G networks, particularly in RB-demanding cases (i.e. tight SLA constraints). However, because we use average SLA deficits to do slice classification, a single RB allocation result most often doesn't significantly impact the average deficits, and thus, the slice classification remains unchanged. This allows us to allocate multiple RBs using the same slice classification. This observation enables us to perform multiple RB allocations in parallel, thereby reducing execution time.

More specifically, in Algo 1, by finding multiple user-RB pairs (k', b') in step 7, steps 8 to 24 can be parallelly executed for all such pairs. The number of user-RB pairs we pick would determine the number of parallel threads required. One possible way to pick this degree of parallelism can be determined by the ratio of total SLA deficits of the current TTI to the average achieved data rate per

RB in the previous TTI. The degree of parallelism can be dynamically adapted as per changing SLA-deficits and execution time requirements. We will evaluate the effectiveness of RB parallelism in §5, demonstrating its impact on reducing running time with only a minor performance degradation.

4.5 Helix Choice of Algorithms

Depending on the use case, Helix will choose one of its core algorithms, DRS or DRO, to minimize the number of used RBs while satisfying the SLAs. All three proposed algorithms for both RB-orthogonal (GP and DRO) and RB-sharing cases (DRS) are SLA-aware extensions of a greedy approach, wherein at their core lies greedy allocation of RBs. In all three, we assign the most suitable RBs that achieve the largest possible rate, thereby lowering the SLA deficit by employing as few RBs as possible, which is our objective function as outlined in Eq. (1). This is done while being cognizant of the SLA deficit of each slice. Due to the use of exhaustive search, GP clearly does not scale well in massive MIMO networks and can only be considered in small-scale MIMO or SISO networks. Therefore, we primarily present it here as a baseline for performance comparison. In contrast, the DRO and DRS algorithms are specifically designed for massive MIMO networks. DRO is applied in RB-orthogonal cases, while DRS is tailored for RB-sharing cases.

5 EXPERIMENTS

In this section, we comprehensively evaluate Helix in massive MIMO-based trace-driven simulation under various network configurations. The highlights of our evaluation are as follows:

- In RB-orthogonal case, GP's performance is near-optimal and better than the Greedy algorithm (§5.2) in small and medium-size networks.
- DRO achieves comparable performance to GP and is scalable to real-world networks. (§5.2).
- DRS provides near-optimal solution and consumes fewer RBs than any orthogonality-based schedulers due to the cooperative sharing of resources (§5.2).
- The benefits of our proposed scheduler hold in different network configurations (small, medium, and real-world size) and scenarios (static and mobility) (§5.2).
- Our proposed schedulers can support diverse scheduling policies and SLAs (§5.2.5).
- DRO and DRS can meet 5G sub-millisecond latency requirements through RB parallelism.(§5.2.6).

5.1 Experimental Setup

To evaluate our schemes under realistic massive MIMO deployments, we leverage a publicly available real-world channel dataset [22] from the RENEW wireless testbed [21]. This dataset encompasses practical considerations such as multipath reflections, hardware impairments, and noise. The raw traces, collected on the 2.4 GHz Wi-Fi ISM band with a 20 MHz bandwidth and 52 OFDM subcarriers, contain received 802.11 Long Training Symbols (LTS). Channel matrices were subsequently extracted through channel estimation. The dataset represents a scenario with a BS equipped with 64 antennas serving 225 single-antenna UEs positioned at various locations. Channel measurements were conducted for both Line-of-Sight (LOS) and Non-Line-of-Sight (NLOS) propagation conditions. The LOS component comprises four clusters, while the NLOS component comprises five clusters. Each cluster approximates a circular shape with a diameter of approximately 4 meters. The specific locations of the massive MIMO BS and user clusters are depicted Fig. 9a in Appendix C. Within each cluster, user channels were measured at 25 uniformly distributed locations and Fig. 9b shows the average inter-user correlation among clusters. It is obvious that users within the same cluster exhibit high correlation, whereas those in different clusters show low correlation, particularly in LoS clusters. The correlation threshold for user grouping in §4.1 is a critical factor in managing the complexity of the grouping process, as it directly impacts the

Table 1. Overview of implemented experiment configurations. (LC is static low-correlated scenario, HC is static high-correlated scenario, SM is slow-mobility, FM is fast-mobility and PF is Proportional Fairness)

Network Size	Scenario	BS	RB	UE (Physical Clusters)	Slice	K
Small Size	LC	64	52	16 (4 LoS)	4	3, 8
	HC	64	52	16 (4 LoS)	4	3, 8
Medium Size	LC	64	52	80 (4 LoS)	8	8, 16
	HC	64	52	80 (4 LoS)	8	8, 16
	HC + PF	64	52	80 (2 LoS + 2 NLoS)	8	16
Real-World Size	LC	64	52	200 (4 LoS + 4 NLoS)	8	16
	HC	64	52	200 (4 LoS + 4 NLoS)	8	16
	SM	64	52	200 (4 LoS + 4 NLoS)	8	16
	FM	64	52	200 (4 LoS + 4 NLoS)	8	16

Network Size	SLA Stringency	Throughput SLA (Mbps)
Small-size network	Loose	[51.9, 46.2, 50, 53.8]
	Tight	[90.4, 84.6, 88.5, 92.3]
Medium and Real-world size network	Loose	[16.7, 46.4, 42.3, 51.7, 19.2, 50.5, 48.1, 53.6]
	Tight	[55.8, 84.2, 80.8, 91.2, 57.7, 88.3, 86.5, 92.4]

number of resulting groups. A stricter threshold increases the number of user groups but provides better assurance that users with high correlation are not scheduled simultaneously. In our Helix evaluations, we adopt a threshold of 0.5, which is commonly employed in prior works [22, 50]. In our comparative evaluation, we consider all schedulers introduced in §4: the Greedy algorithm, Greedy Plus (GP), optimal solution through Gurobi, DRO, DRS, and RS_ES. For the RB-orthogonal case, the Greedy algorithm serves as a baseline adopted by state-of-the-art (SOTA) work [18], while the Gurobi optimizer provides the upper performance bound. For RB-sharing, since the exponential combinatorial space associated with UEs, slices, and RBs makes it infeasible to obtain an optimal solution in large-size networks, RS_ES serves as the baseline method for comparison. All experiment configurations are listed in Table. 1. We establish two static channel correlation scenarios: high-correlated (HC) and low-correlated. In the high-correlated scenario, users within a slice are deliberately chosen from the same LoS cluster, resulting in high inter-user correlation (reported to exceed 0.8 in [22]). Conversely, for the low-correlated (LC) scenario, users are positioned in distinct clusters within a slice, which ensures that inter-user correlation remains below 0.2 in the low-correlated case. To comprehensively assess performance, we also define two distinct sets of SLAs (loose and tight) corresponding to each scenario. For mobility channels, we consider both slow-mobility (SM) such as pedestrian speed and fast-mobility (FM) scenarios. Following the methodology outlined in [22], since channels of different spots in each cluster are measured continuously, it allows us to simulate a user moving randomly at low speed within a cluster. Similarly, channels across different clusters can be used to emulate high-speed mobility.

5.2 Performance Evaluation

We evaluate the performance of our proposed schedulers through various network sizes, differing stringencies of SLAs, and diverse correlation scenarios. In 5G, SLAs are negotiated by enterprise and service-provider based on radio resource availability, CAPEX and type of services. We set loose and tight SLAs based on channel dataset following the instructions of [35]. For a fair comparison, all algorithms receive identical channels generated from real-world traces as input. We assess scheduler performance using the average number of allocated RBs across all TTIs while meeting SLAs. This metric indicates an algorithm's ability to satisfy SLAs with minimal RB utilization. Error bars are also incorporated into each plot to visualize the standard deviation in allocated RBs across TTIs. First, we employ throughput as the SLA to evaluate scheduler performance in a static scenario. Subsequently, we demonstrate our scheduler's superiority to support a variety of

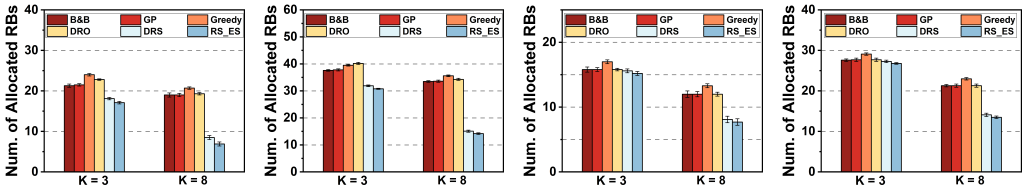


Fig. 5. Scheduler performance in small-size network (a) static high-correlated and loose SLA (b) static high-correlated and tight SLA (c)static low-correlated and loose SLA (d) static low-correlated and tight SLA.

scheduling policies and SLAs, while achieving strong performance in mobility scenarios. Finally, we evaluate the running time of the proposed schedulers, highlighting the ability of DRO and DRS to deliver scheduling decisions within milliseconds.

5.2.1 Small-Size Network. We consider a massive MIMO network with 64 BS antennas, serving 16 users distributed across 4 slices, utilizing 52 RBs. Fig. 5a and 5b illustrate the number of RBs consumed under loose and tight SLA constraints within the high-correlated scenario. Evidently, GP outperforms Greedy in terms of RB utilization while being very close to the optimal solution obtained through B&B. DRO exhibits slightly worse performance than GP (i.e. consuming one additional RB per TTI), but avoids the exhaustive search required by GP. This difference stems from the fact that B&B identifies a globally optimal solution by accounting for both inter-slice and intra-slice scheduling. In contrast, GP applies a greedy algorithm for inter-slice scheduling and utilizes exhaustive searches for intra-slice scheduling to achieve near-optimal performance, whereas DRO adopts a sub-optimal intra-slice scheduling approach to mitigate computational complexity. As a result, both GP and B&B incur substantial latency due to their computational overhead (see §5.2.6), while DRO, despite its faster execution, allocates more RBs than GP and B&B when satisfying SLAs. Notably, DRS obtains near-optimal performance compared to RS_ES and consumes much fewer RBs than any RB-orthogonal schedulers due the RB sharing opportunities among slices. Specifically, when $K = 3$, DRS achieves the best performance with a reduction in RB consumption of 25% and 19.2% for loose and tight SLAs, respectively, compared to the Greedy algorithm. This advantage is further amplified to 58.9% and 57.6% when $K = 8$. This behavior can be attributed to large K emulating a user-sparse scenario ($\mathcal{K}_s < K$), where DRS can effectively share redundant resources with users from other slices, maximizing the utilization. Fig. 5c and 5d depict the performance of schedulers in the low-correlated scenario. In this case as well, DRS and DRO are able to achieve near-optimal performance in their respective scenarios. However, the advantage of RB sharing among slices is less pronounced compared to the high-correlated case. This is because, with low-correlated users, the impact of selecting users from the same slice (RB-orthogonality) or different slices (RB-sharing) is minimal. In contrast, for the high-correlated scenario, imposing RB-orthogonality introduces significant inter-user interference, leading to performance degradation.

5.2.2 Medium-Size Network. We configure the network with 64 BS antennas, 52 RBs, and 80 users distributed into 8 slices, with each slice containing 10 users. However, RS_ES is not implementable in this scale because of high complexity (i.e. $O(\binom{80}{K})$). Fig. 6 shows similar trends as in the small-size network: (1) GP performs near optimally and consistently outperforms the Greedy algorithm across all K, SLA constraints, and correlation conditions. (2) DRO, though slightly below GP due to its

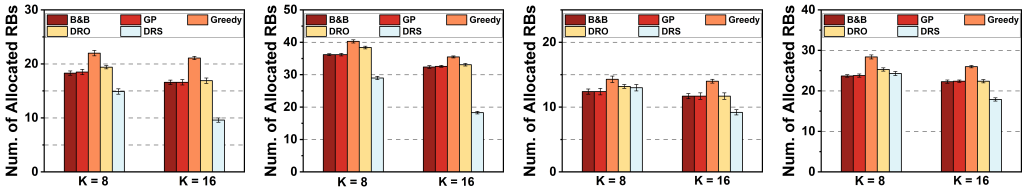


Fig. 6. Scheduler performance in medium-size network (a) static high-correlated and loose SLA (b) static high-correlated and tight SLA (c)static low-correlated and loose SLA (d) static low-correlated and tight SLA.

approximation, offers significantly lower complexity. (3) DRS demonstrates the benefits of RB sharing, especially as K increases and in highly correlated scenarios.

5.2.3 Real-World-Size Network. To conduct a realistic evaluation, we expand the network size to encompass 200 UEs with various channel conditions, originating from four LoS clusters and four NLoS clusters, while keeping all other configurations unchanged. Following the established topology of the clusters, we configure both high-correlated and low-correlated scenarios as previously applied to different network sizes. To make it more realistic, unlike the previous even distribution of users among slices, we generate a random set of numbers (i.e. [10, 12, 18, 20, 25, 33, 45, 37] with a mean of 25) to determine the number of users in each slice. With $K = 16$, our experiments include both sparsely populated ($|\mathcal{K}_s| < K$) and densely populated ($|\mathcal{K}_s| > K$) slices. For this network size, the Greedy algorithm, GP, and Gurobi are infeasible due to their high complexity. Consequently, as depicted in Fig. 7a, we only present the performance of DRO and DRS to show its scalability.

5.2.4 Mobility Scenarios. A comparison of the schedulers' performance in mobile scenarios is illustrated in Fig. 7b. It is noteworthy that the results for the low-speed mobility scenario are similar to the static high-correlated scenario in Fig. 7a, as users moving within the same cluster still remain highly correlated. Similarly, the results for high-speed mobility are comparable to those in the static low-correlated scenario, albeit with larger standard deviations than static scenario.

5.2.5 Diverse Scheduling Policies. As illustrated in §4, channel gain serves as a key indicator of user channel quality and affects scheduling decisions to fulfill throughput SLAs. However, our schedulers can accommodate a variety of scheduling policies and SLA types by substituting channel gain with alternative performance metrics. In this evaluation, we utilize the proportional fairness metric in place of channel gain to implement the PF scheduling policy, thereby ensuring SLAs are met concerning both throughput and inter-user rate fairness. In this work, we employ Jain's Fairness Index (JFI) [32] to indicate rate fairness, which is a number between 0 to 1 and JFI=1 represents perfect fairness. PF metric is expressed as $\hat{g}_k^{b,t}/\hat{R}_k^t$, where $\hat{g}_k^{b,t}$ denotes normalized channel gain of user k on RB b at TTI t and $\hat{R}_k^t$ indicates normalized accumulated rate of user k by TTI t. It is essential to note that the normalization factors are the maximum channel gain and the maximum accumulated rate in s where $slice(k) = s$. This is because only intra-slice fairness needs to be ensured, as guaranteeing global user fairness is impractical due to the varying SLAs across slices. To provide a thorough comparison, we conducted experiments in a medium-sized network with 80 UEs, incorporating the Greedy algorithm, GP, the optimal solution generated by the Gurobi optimizer (B&B), DRO, and DRS. As illustrated in Fig. 7c and 7d, after replacing MR policy with PF, DRS is able to achieve an outstanding JFI but only uses additional 0.5 RBs in average per TTI compared to DRS with MR scheduling policy. Similarly, to support a variety of SLAs, Helix can be

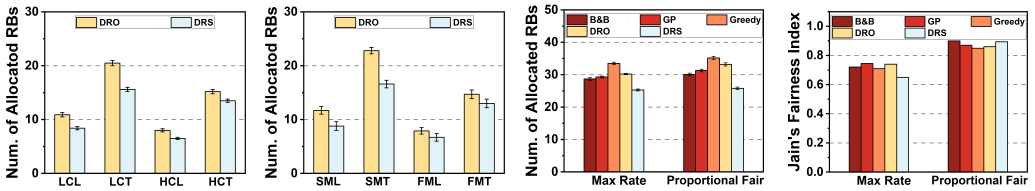
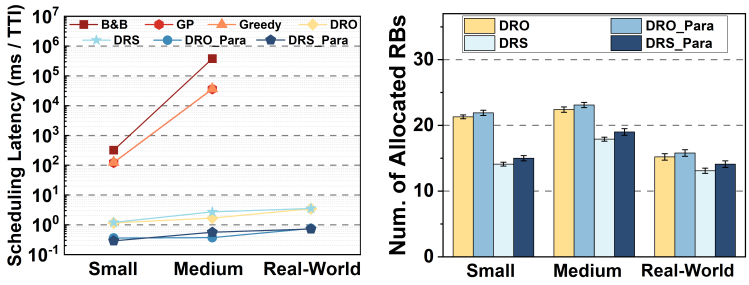


Fig. 7. Scheduler Performance: (a) and (b) real-world-size network with static and mobility scenario respectively. (c) Diverse scheduling polices in static medium-size network. (d) Jain's Fairness Index for Diverse scheduling polices in static medium-size network. (LCL:low-correlated with loose SLA, HCT: high-correlated with tight SLA, SML: slow-mobility with loose SLA, FMT: fast-mobility with tight SLA)

adapted by replacing the channel gain with other performance metrics, such as buffer occupancy for latency-sensitive SLAs, extending its functionality beyond just throughput.

5.2.6 Computational Complexity Analysis. As discussed in §3 and §4, both the Greedy algorithm and GP necessitate an exhaustive search by the intra-slice scheduler. This approach is feasible within SISO networks due to the availability of rate estimation via CQI utilizing a look-up table. However, in MIMO networks, computing rate for all user combinations within each slice on each RB is infeasible within a TTI. Conducting an exhaustive search for $|\mathcal{S}|$ slice across all $|\mathcal{B}|$ RBs (assuming each slice contains the same $|\mathcal{K}_s|$ users) incurs a time complexity of $O(\binom{|\mathcal{K}_s|}{K} \times |\mathcal{B}| \times |\mathcal{S}| \times K)$ for rate estimation. In contrast, approaches such as DRO or DRS compute rates only once for selected K users after resource allocation decision have been made, thereby significantly reducing computational overhead. The primary computational bottleneck lies in user grouping and the sorting of channel gains within each user group, detailed in §4, with a time complexity logarithmically dependent on the user group's size. Furthermore, as delineated in §4.1 and §4.4, user grouping and slice sorting require only periodic execution over multiple TTIs. Due to the inherent optimization potential and flexibility of these methods, the computational complexity of DRO and DRS depends on implementation details and adjustable parameters like user grouping thresholds and the degree of RB parallelism. We show their competitive time efficiency through a detailed runtime analysis.

To evaluate the running time of proposed schedulers, we implement them using a single Intel Xeon core. Fig. 8a shows the scheduling time of GP and Greedy algorithm increases exponentially with the network size because of the exhaustive search that the intra-slice scheduler has to perform. B&B does an exhaustive search at both inter-slice and intra-slice levels to find the optimal solution. Even though it has been accelerated by the Gurobi optimizer, it still takes the longest time to obtain scheduling decisions. In contrast, DRS and DRO grow linearly with the help of an approximate approach. However, as discussed in 4, because the outcomes of previous RB allocations adjust the SLA deficit, RB allocation in DRO and DRS occurs sequentially. It results in a scheduling time of slightly more than 1 ms for both DRS and DRO, which is unfavorable in real-world networks. Therefore, we adopt RB parallelism strategy as discussed in §4.4 with DRS and DRO (labelled as DRS_Para and DRO_Para) to do allocation on multiple RBs in parallel. By doing this, DRS_Para and DRO_Para will spend a little more radio resources than DRS and DRO (i.e. up to 1.1 and 0.7 more RBs per TTI in average) in various network sizes but can significantly reduce the running time by 5 times so as to meet the stringent 5G latency requirement as Fig. 8 shown.



(a) (b)

Fig. 8. (a) Scheduling latency comparison in various sizes of networks and (b) RB consumption comparison between w/ and w/o RB parallelism.

6 DISCUSSION AND FUTURE WORK

Timely and comprehensive channel information. Helix operates under the premise of obtaining precise and thorough channel data from users to do user grouping. This assumption is common in channel-aware RAN slicing literature [6, 18, 36, 43], wherein, obtaining CSI is challenging, especially in large networks with many users. However, Helix can achieve strong performance even with partial and dynamic CSI, as its primary use is for user grouping. As long as the channel remains relatively stable, user grouping is unaffected by minor fluctuations, as discussed in §4.1. Thus, frequent CSI updates are unnecessary for Helix's scheduling; they are only needed when significant changes in user correlation occur. Future work will enhance Helix by incorporating dynamic mechanisms to detect correlation changes and request CSI updates as required.

Heterogenous SLAs. Although Helix's capability to support multiple scheduling policies and different types of SLAs is outlined in §5.2.5, Helix in its current form, only addresses rate as the main metric for the SLA of different slices. One particular future direction for the design of Helix, could consider heterogeneous SLA parameters including rate, packet latency and buffer size.

Ultra-Large-Scale Networks. In practical scenarios, an urban macro 5G cell can support hundreds to over a thousand active users concurrently [3, 38], which presents significant challenges for resource scheduling due to increased complexity. To accommodate a higher number of users, Helix leverages RB parallelism, as outlined in §4.4, enabling it to complete allocations within sub-millisecond intervals. However, this approach requires additional computational resources. Future work will focus on evaluating Helix's performance in such large-scale network environments.

7 CONCLUSION

This paper introduces Helix, a RAN slicing framework for massive MIMO networks that optimizes resource scheduling through spatial-time slicing via beamforming. Unlike traditional MIMO schedulers maximizing throughput, Helix prioritizes enterprise-specific SLA with minimal RBs. It supports cooperative and exclusive resource allocation, enabling operators to choose resource sharing (DRS) or exclusive allocation (DRO) based on privacy and traffic needs. Evaluations using real-world data demonstrate near-optimal performance and efficiency across various scenarios.

ACKNOWLEDGMENTS

We thank the anonymous CoNEXT reviewers and our shepherd for their invaluable feedbacks. This work was funded in part by National Science Foundation Grant 2312978, 2120447, 1827940, 2120363, 2106993 and 2148313. All opinions and findings are of the authors.

REFERENCES

[1] 2024. RENEW: Reconfigurable Eco-system for Next-generation End-to-end Wireless. https://renew.rice.edu/dataset-iuc.html. Open Access.

[2] Qing An, Mehdi Zafari, Chris Dick, Santiago Segarra, Ashutosh Sabharwal, and Rahman Doost-Mohammady. 2023. ML-Based Feedback-Free Adaptive MCS Selection for Massive Multi-User MIMO. In *2023 57th Asilomar Conference on Signals, Systems, and Computers*. 157–161. https://doi.org/10.1109/IEEECONF59524.2023.10476866

[3] Jeffrey G. Andrews, Stefano Buzzi, Wan Choi, Stephen V. Hanly, Angel Lozano, Anthony C. K. Soong, and Jianzhong Charlie Zhang. 2014. What Will 5G Be? *IEEE Journal on Selected Areas in Communications* 32, 6 (2014), 1065–1082. https://doi.org/10.1109/JSAC.2014.2328098

[4] M. Andrews, K. Kumaran, K. Ramanan, A. Stolyar, P. Whiting, and R. Vijayakumar. 2001. Providing quality of service over a shared wireless link. *IEEE Communications Magazine* 39, 2 (2001), 150–154. https://doi.org/10.1109/35.900644

[5] Sihem Bakri, Pantelis A. Frangoudis, Adlen Ksentini, and Maha Bouaziz. 2021. Data-Driven RAN Slicing Mechanisms for 5G and Beyond. *IEEE Transactions on Network and Service Management* 18, 4 (2021), 4654–4668. https://doi.org/10.1109/TNSM.2021.3098193

[6] Arjun Balasingam, Manikanta Kotaru, and Victor Bahl. 2024. Application-Level Service Assurance with 5G RAN Slicing. In *2024 Networked Systems Design and Implementation*. USENIX. https://www.microsoft.com/en-us/research/publication/application-level-service-assurance-with-5g-ran-slicing/

[7] Emil Björnson, Erik G. Larsson, and Thomas L. Marzetta. 2016. Massive MIMO: ten myths and one critical question. *IEEE Communications Magazine* 54, 2 (2016), 114–123. https://doi.org/10.1109/MCOM.2016.7402270

[8] Erik G. Boman, Doruk Bozdağ, Umit Catalyurek, Assefaw H. Gebremedhin, and Fredrik Manne. 2005. A Scalable Parallel Graph Coloring Algorithm for Distributed Memory Computers. In *Euro-Par 2005 Parallel Processing*, José C. Cunha and Pedro D. Medeiros (Eds.). Springer Berlin Heidelberg, Berlin, Heidelberg, 241–251.

[9] Leonardo Bonati, Salvatore D'Oro, Stefano Basagni, and Tommaso Melodia. 2021. SCOPE: an open and softwarized prototyping platform for NextG systems. In *Proceedings of the 19th Annual International Conference on Mobile Systems, Applications, and Services* (Virtual Event, Wisconsin) *(MobiSys '21)*. Association for Computing Machinery, New York, NY, USA, 415–426. https://doi.org/10.1145/3458864.3466863

[10] Gaojing Bu and Jing Jiang. 2019. Reinforcement Learning-Based User Scheduling and Resource Allocation for Massive MU-MIMO System. In *2019 IEEE/CIC International Conference on Communications in China (ICCC)*. 641–646. https://doi.org/10.1109/ICCChina.2019.8855949

[11] Chia-Yu Chang and Navid Nikaein. 2018. RAN Runtime Slicing System for Flexible and Dynamic Service Execution Environment. *IEEE Access* 6 (2018), 34018–34042. https://doi.org/10.1109/ACCESS.2018.2847610

[12] Robin Chataut and Robert Akl. 2020. Massive MIMO Systems for 5G and beyond Networks—Overview, Recent Trends, Challenges, and Future Research Direction. *Sensors* 20, 10 (2020). https://doi.org/10.3390/s20102753

[13] Cheng-Ming Chen, Qing Wang, Abdo Gaber, Andrea P. Guevara, and Sofie Pollin. 2020. User Scheduling and Antenna Topology in Dense Massive MIMO Networks: An Experimental Study. *IEEE Transactions on Wireless Communications* 19, 9 (2020), 6210–6223. https://doi.org/10.1109/TWC.2020.3001224

[14] Hongchao Chen, Yupu Liu, Zhe Zheng, Huiyang Wang, Xiaohui Liang, Yi Zhao, and Junwei Ren. 2021. Joint User Scheduling and Transmit Precoder Selection Based on DDPG for Uplink Multi-User MIMO Systems. In *2021 IEEE 94th Vehicular Technology Conference (VTC2021-Fall)*. IEEE, 1–5.

[15] Liang Chen, Fanglei Sun, Kai Li, Ruiqing Chen, Yang Yang, and Jun Wang. 2021. Deep Reinforcement Learning for Resource Allocation in Massive MIMO. In *2021 29th European Signal Processing Conference (EUSIPCO)*. 1611–1615. https://doi.org/10.23919/EUSIPCO54536.2021.9616054

[16] Shuang Chen, Christina Delimitrou, and José F. Martínez. 2019. PARTIES: QoS-Aware Resource Partitioning for Multiple Interactive Services. In *Proceedings of the Twenty-Fourth International Conference on Architectural Support for Programming Languages and Operating Systems* (Providence, RI, USA) *(ASPLOS '19)*. Association for Computing Machinery, New York, NY, USA, 107–120. https://doi.org/10.1145/3297858.3304005

[17] Yongce Chen, Yubo Wu, Y. Thomas Hou, and Wenjing Lou. 2021. mCore: Achieving Sub-millisecond Scheduling for 5G MU-MIMO Systems. In *IEEE INFOCOM 2021 - IEEE Conference on Computer Communications*. 1–10. https://doi.org/10.1109/INFOCOM42981.2021.9488684

[18] Yongzhou Chen, Ruihao Yao, Haitham Hassanieh, and Radhika Mittal. 2023. Channel-Aware 5G RAN Slicing with Customizable Schedulers. In *20th USENIX Symposium on Networked Systems Design and Implementation (NSDI 23)*. USENIX Association, Boston, MA, 1767–1782. https://www.usenix.org/conference/nsdi23/presentation/chen-yongzhou

[19] Arnab Choudhury, Yang Wang, Tuomas Pelkonen, Kutta Srinivasan, Abha Jain, Shenghao Lin, Delia David, Siavash Soleimanifard, Michael Chen, Abhishek Yadav, Ritesh Tijoriwala, Denis Samoylov, and Chunqiang Tang. 2024. MAST: Global Scheduling of ML Training across Geo-Distributed Datacenters at Hyperscale. In *18th USENIX Symposium on Operating Systems Design and Implementation (OSDI 24)*. USENIX Association, Santa Clara, CA, 563–580. https://www.usenix.org/conference/osdi24/presentation/choudhury

[20] Christina Delimitrou and Christos Kozyrakis. 2014. Quasar: resource-efficient and QoS-aware cluster management. *SIGPLAN Not.* 49, 4 (feb 2014), 127–144. https://doi.org/10.1145/2644865.2541941

[21] Rahman Doost-Mohammady, Oscar Bejarano, Lin Zhong, Joseph R. Cavallaro, Edward Knightly, Z. Morley Mao, Wei Wayne Li, Xuemin Chen, and Ashutosh Sabharwal. 2018. RENEW: Programmable and Observable Massive MIMO Networks. In *2018 52nd Asilomar Conference on Signals, Systems, and Computers*. 1654–1658. https://doi.org/10.1109/ACSSC.2018.8645391

[22] Xu Du and Ashutosh Sabharwal. 2022. Massive MIMO Channels With Inter-User Angle Correlation: Open-Access Dataset, Analysis and Measurement-Based Validation. *IEEE Transactions on Vehicular Technology* 71, 2 (2022), 1602–1616. https://doi.org/10.1109/TVT.2021.3131606

[23] Xenofon Foukas, Mahesh K. Marina, and Kimon Kontovasilis. 2017. Orion: RAN Slicing for a Flexible and Cost-Effective Multi-Service Mobile Network Architecture. In *Proceedings of the 23rd Annual International Conference on Mobile Computing and Networking* (Snowbird, Utah, USA) *(MobiCom '17)*. Association for Computing Machinery, New York, NY, USA, 127–140. https://doi.org/10.1145/3117811.3117831

[24] Mikael Gidlund and J.-C. Laneri. 2008. Scheduling Algorithms for 3GPP Long-Term Evolution Systems: From a Quality of Service Perspective. In *2008 IEEE 10th International Symposium on Spread Spectrum Techniques and Applications*. 118–123. https://doi.org/10.1109/ISSSTA.2008.28

[25] Mukesh Kumar Giluka, Nitish Rajoria, Ashish C. Kulkarni, Vanlin Sathya, and Bheemarjuna Reddy Tamma. 2014. Class based dynamic priority scheduling for uplink to support M2M communications in LTE. In *2014 IEEE World Forum on Internet of Things (WF-IoT)*. 313–317. https://doi.org/10.1109/WF-IoT.2014.6803179

[26] Xiaojun Guo, Ziyi Li, Pengyu Liu, Rudan Yan, Yingying Han, Xiaojun Hei, and Guohui Zhong. 2020. A Novel User Selection Massive MIMO Scheduling Algorithm via Real Time DDPG. In *GLOBECOM 2020 - 2020 IEEE Global Communications Conference*. 1–6. https://doi.org/10.1109/GLOBECOM42002.2020.9322383

[27] Gurobi Optimization, LLC. 2023. Gurobi Optimizer Reference Manual. https://www.gurobi.com

[28] Benjamin Hindman, Andy Konwinski, Matei Zaharia, Ali Ghodsi, Anthony D. Joseph, Randy Katz, Scott Shenker, and Ion Stoica. 2011. Mesos: a platform for fine-grained resource sharing in the data center. In *Proceedings of the 8th USENIX Conference on Networked Systems Design and Implementation* (Boston, MA) *(NSDI'11)*. USENIX Association, USA, 295–308.

[29] Qinghao Hu, Meng Zhang, Peng Sun, Yonggang Wen, and Tianwei Zhang. 2023. Lucid: A Non-intrusive, Scalable and Interpretable Scheduler for Deep Learning Training Jobs. In *Proceedings of the 28th ACM International Conference on Architectural Support for Programming Languages and Operating Systems, Volume 2* (Vancouver, BC, Canada) *(ASPLOS 2023)*. Association for Computing Machinery, New York, NY, USA, 457–472. https://doi.org/10.1145/3575693.3575705

[30] Chih-Wei Huang, Ibrahim Althamary, Yen-Cheng Chou, Hong-Yunn Chen, and Cheng-Fu Chou. 2023. A DRL-Based Automated Algorithm Selection Framework for Cross-Layer QoS-Aware Scheduling and Antenna Allocation in Massive MIMO Systems. *IEEE Access* 11 (2023), 13243–13256. https://doi.org/10.1109/ACCESS.2023.3243068

[31] Lingying Huang, Xiaomeng Chen, Wei Huo, Jiazheng Wang, Fan Zhang, Bo Bai, and Ling Shi. 2021. Branch and Bound in Mixed Integer Linear Programming Problems: A Survey of Techniques and Trends. arXiv:2111.06257 [cs.LG]

[32] R. Jain, D. Chiu, and W. Hawe. 1998. A Quantitative Measure Of Fairness And Discrimination For Resource Allocation In Shared Computer Systems. arXiv:cs/9809099 [cs.NI]

[33] Tommy R Jensen and Bjarne Toft. 2011. *Graph coloring problems*. John Wiley & Sons.

[34] Alexander Kalachikov and Alexander Stenin. 2021. Performance Evaluation of the SRS Based MIMO Channel Estimation on 5G NR Open Source Channel Model. In *2021 IEEE 22nd International Conference of Young Professionals in Electron Devices and Materials (EDM)*. 124–127. https://doi.org/10.1109/EDM52169.2021.9507598

[35] Evgenia Kapassa, Marios Touloupou, and Dimosthenis Kyriazis. 2018. SLAs in 5G: A Complete Framework Facilitating VNF- and NS- Tailored SLAs Management. In *2018 32nd International Conference on Advanced Information Networking and Applications Workshops (WAINA)*. 469–474. https://doi.org/10.1109/WAINA.2018.00130

[36] Ravi Kokku, Rajesh Mahindra, Honghai Zhang, and Sampath Rangarajan. 2012. NVS: A Substrate for Virtualizing Wireless Resources in Cellular Networks. *IEEE/ACM Transactions on Networking* 20, 5 (2012), 1333–1346. https://doi.org/10.1109/TNET.2011.2179063

[37] Adlen Ksentini and Navid Nikaein. 2017. Toward Enforcing Network Slicing on RAN: Flexibility and Resources Abstraction. *IEEE Communications Magazine* 55, 6 (2017), 102–108. https://doi.org/10.1109/MCOM.2017.1601119

[38] Erik G. Larsson, Ove Edfors, Fredrik Tufvesson, and Thomas L. Marzetta. 2014. Massive MIMO for next generation wireless systems. *IEEE Communications Magazine* 52, 2 (2014), 186–195. https://doi.org/10.1109/MCOM.2014.6736761

[39] Junling Li, Weisen Shi, Peng Yang, Qiang Ye, Xuemin Sherman Shen, Xu Li, and Jaya Rao. 2020. A Hierarchical Soft RAN Slicing Framework for Differentiated Service Provisioning. *IEEE Wireless Communications* 27, 6 (2020), 90–97. https://doi.org/10.1109/MWC.001.2000010

[40] Shao-Yu Lien, Shin-Lin Shieh, Yenming Huang, Borching Su, Yung-Lin Hsu, and Hung-Yu Wei. 2017. 5G New Radio: Waveform, Frame Structure, Multiple Access, and Initial Access. *IEEE Communications Magazine* 55 (2017), 64–71.

https://api.semanticscholar.org/CorpusID:1660972

[41] Haijing Liu, Hui Gao, Shaoshi Yang, and Tiejun Lv. 2017. Low-Complexity Downlink User Selection for Massive MIMO Systems. *IEEE Systems Journal* 11, 2 (2017), 1072–1083. https://doi.org/10.1109/JSYST.2015.2422475

[42] Victor Hugo L Lopes, Cleverson Veloso Nahum, Ryan M Dreifuerst, Pedro Batista, Aldebaro Klautau, Kleber Vieira Cardoso, and Robert W Heath. 2022. Deep Reinforcement Learning-Based Scheduling for Multiband Massive MIMO. *IEEE Access* 10 (2022), 125509–125525.

[43] Akihiro Nakao, Ping Du, Yoshiaki Kiriha, Fabrizio Granelli, Anteneh Atumo Gebremariam, Tarik Taleb, and Miloud Bagaa. 2017. End-to-end Network Slicing for 5G Mobile Networks. *Journal of Information Processing* 25 (2017), 153–163. https://doi.org/10.2197/ipsjjip.25.153

[44] Deepak Narayanan, Keshav Santhanam, Fiodar Kazhamiaka, Amar Phanishayee, and Matei Zaharia. 2020. {Heterogeneity-Aware} cluster scheduling policies for deep learning workloads. In *14th USENIX Symposium on Operating Systems Design and Implementation (OSDI 20)*. 481–498.

[45] Samsung Technical White Paper. 2021. *Private Networks Vol.2 Architectures and Features for Industrial Scenarios*. White Paper. Samsung. https://images.samsung.com/is/content/samsung/assets/global/business/networks/insights/white-papers/1026-private-networks-vol-2-architectures-and-features-for-industrial-scenarios/Private_Network_Vol.2_Network_Architecture_and_Features_for_Industrial_Scenarios.pdf

[46] Georgios Rokos, Gerard Gorman, and Paul H J Kelly. 2015. A Fast and Scalable Graph Coloring Algorithm for Multi-core and Many-core Architectures. arXiv:1505.04086 [cs.DC]

[47] Junchao Shi, Wenjin Wang, Jiaheng Wang, and Xiqi Gao. 2018. Machine Learning Assisted User-scheduling Method for Massive MIMO System. In *2018 10th International Conference on Wireless Communications and Signal Processing (WCSP)*. 1–6. https://doi.org/10.1109/WCSP.2018.8555722

[48] Prashant Subedi, Abeer Alsadoon, P. W. C. Prasad, Sabih Rehman, Nabil Giweli, Muhammad Imran, and Samrah Arif. 2021. Network slicing: a next generation 5G perspective. *EURASIP J. Wirel. Commun. Netw.* 2021, 1 (apr 2021), 26 pages. https://doi.org/10.1186/s13638-021-01983-7

[49] Ninad Thakoor, Venkat Devarajan, and Jean Gao. 2009. Computation complexity of branch-and-bound model selection. In *2009 IEEE 12th International Conference on Computer Vision*. 1895–1900. https://doi.org/10.1109/ICCV.2009.5459420

[50] Hong Yang. 2018. User Scheduling in Massive MIMO. In *2018 IEEE 19th International Workshop on Signal Processing Advances in Wireless Communications (SPAWC)*. 1–5. https://doi.org/10.1109/SPAWC.2018.8445907

[51] Hong Yang and Thomas L. Marzetta. 2017. Massive MIMO With Max-Min Power Control in Line-of-Sight Propagation Environment. *IEEE Transactions on Communications* 65, 11 (2017), 4685–4693. https://doi.org/10.1109/TCOMM.2017.2725262

[52] Peifeng Yu and Mosharaf Chowdhury. 2020. Fine-grained GPU sharing primitives for deep learning applications. *Proceedings of Machine Learning and Systems* 2 (2020), 98–111.

[53] Chaoyun Zhang, Paul Patras, and Hamed Haddadi. 2019. Deep Learning in Mobile and Wireless Networking: A Survey. *IEEE Communications Surveys and Tutorials* 21, 3 (2019), 2224–2287. https://doi.org/10.1109/COMST.2019.2904897

[54] Yanqi Zhang, Weizhe Hua, Zhuangzhuang Zhou, G. Edward Suh, and Christina Delimitrou. 2021. Sinan: ML-based and QoS-aware resource management for cloud microservices. In *Proceedings of the 26th ACM International Conference on Architectural Support for Programming Languages and Operating Systems* (Virtual, USA) *(ASPLOS '21)*. Association for Computing Machinery, New York, NY, USA, 167–181. https://doi.org/10.1145/3445814.3446693

A ACHIEVABLE DATA RATE DERIVATION

Consider a massive MIMO base station (BS) with M antennas, serving N single-antenna users. The BS employs OFDM and performs Multi-User MIMO transmission and reception to K users, where $K < M$ and $K \leq N$. For the uplink transmission, the signal model can be specified as

$$\mathbf{y} = \mathbf{H}\mathbf{u} + \mathbf{n}, \tag{6}$$

where $\mathbf{y}$ represents the $M \times 1$ received signal vector at the base station. The matrix $\mathbf{H}$ is the $M \times K$ channel matrix, and $\mathbf{u}$ denotes the $K \times 1$ vector of symbols transmitted by the users. Additionally, $\mathbf{n}$ is the $M \times 1$ received noise vector, which follows a circularly symmetric complex Gaussian distribution, $\mathbf{n} \sim \mathcal{CN}(0, \sigma^2\mathbf{I})$, where σ^2 is the noise variance and $\mathbf{I}$ is the identity matrix. It is important to note that the value of K can vary in each TTI depending on the channel conditions and is bounded by a maximum value $K_{\max}$ [7]. We assume the BS uses zero-forcing for beamforming, for which the beamformer matrix is specified as

$$\mathbf{W} = \mathbf{H}(\mathbf{H}^H\mathbf{H})^{-1}. \tag{7}$$

If the set of available resource blocks is denoted by $\mathcal{B}$, the BS performs receive beamforming on the received signal at each resource block $b \in \mathcal{B}$ to estimate the transmit symbol vector on RB b at TTI t, denoted as $\hat{\mathbf{u}}^{b,t}$, and calculated using

$$\hat{\mathbf{u}}^{b,t} = \left(\mathbf{W}^{b,t}\right)^H \mathbf{y}^{b,t}. \tag{8}$$

The entities $\mathbf{W}^{b,t}$ and $\mathbf{y}^{b,t}$ denote the receive beamformer matrix and the received signal corresponding to resource block b at TTI t. Based on Eq. 8, the output of the receive beamformer corresponding to the transmit signal by user k on RB b, can be expressed as

$$\hat{\mathbf{u}}_k^{b,t} = \left(\mathbf{w}_k^{b,t}\right)^H \mathbf{y}^{b,t} = (\mathbf{w}_k^{b,t})^H \mathbf{h}_k^{b,t} x_k^{b,t} u_k^{b,t} + \sum_{i=1, i\neq k}^{K} (\mathbf{w}_k^{b,t})^H \mathbf{h}_i^{b,t} x_i^{b,t} u_i^{b,t} + (\mathbf{w}_k^{b,t})^H \mathbf{n} \tag{9}$$

where $(\mathbf{w}_k^{b,t})^H$ is the $1 \times M$ complex beamforming vector of user k on RB b at TTI t, $\mathbf{h}_k^{b,t}$ is the $M \times 1$ complex channel vector of user k on RB b at TTI t, $u_k^{b,t}$ is the transmitted symbol of user k on RB b at TTI t and $x_k^{b,t} \in \{0, 1\}$ is a binary variable indicating whether or not RB b is scheduled to user k by the BS at TTI t. The first term $(\mathbf{w}_k^{b,t})^H \mathbf{h}_k^{b,t} x_k^{b,t} u_k^{b,t}$ in Eq. (9) is desired signal for user k on RB b at TTI t, the second term $\sum_{i=1, i\neq k}^{K} (\mathbf{w}_k^{b,t})^H \mathbf{h}_i^{b,t} x_i^{b,t} u_i^{b,t}$ represents the interference from other concurrently scheduled users and the last term is noise. Accordingly, assuming the CSI is known and the signals transmitted by different users are independent and with unit power, we can obtain signal-to-interference-plus-noise ratio (SINR) of user k on RB b at TTI t as follows:

$$\mathrm{SINR}_k^{b,t} = \frac{\left|(\mathbf{w}_k^{b,t})^H \mathbf{h}_k^{b,t} x_k^{b,t}\right|^2}{\sum_{i=1, i\neq k}^{K} \left|(\mathbf{w}_k^{b,t})^H \mathbf{h}_i^{b,t} x_i^{b,t}\right|^2 + \left|\mathbf{w}_k^{b,t} \sigma^2\right|^2} \tag{10}$$

Plugging Eq. (10) into Shannon's capacity formula, the achievable data rate of user k on RB b at TTI t is obtained by:

$$r_k^{b,t} = \mathbf{BW}^{b,t} \log_2(1 + \mathrm{SINR}_k^{b,t}) \tag{11}$$

where $\mathbf{BW}^{b,t}$ is bandwidth of RB b and total achievable data rate of slice s on RB b at TTI t is

$$r_s^{b,t} = \sum_{k \in \mathcal{K}_s} r_k^{b,t} \tag{12}$$

where $\mathcal{K}_s$ denotes the set of users corresponding to slice s.

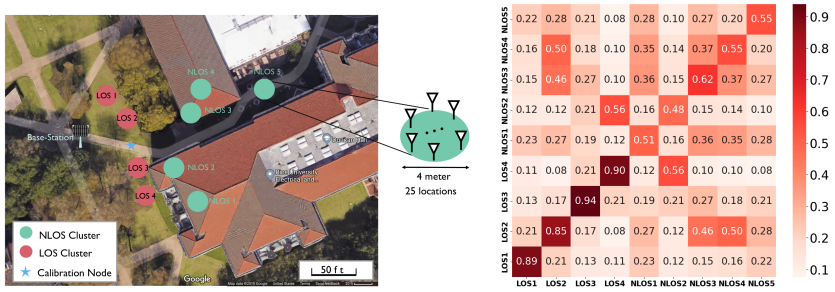
B HELIX ALGORITHM

Algorithm 1 Helix: Algorithmic Description of DRO and DRS approaches

Input: RB set $\mathcal{B}$; Slice set $\mathcal{S}$; Channel $\mathbf{H}^{b,t}$; Users K; TTIs T; Decision to run DRO or DRS.
Output: Average number of allocated RBs

$RB\_alloc\_total \leftarrow 0$ ▷ Total number of allocated RBs;
for $t = 0; t < T; t++$ **do**
  $\{\mathcal{N}_1, \mathcal{N}_2, ...\mathcal{N}_\eta\}^{b,t} \longleftarrow user\_grouping(\mathbf{H}^{b,t})$ ▷ user grouping as per §4.1
  Update $\Delta_s^t, \forall s$ using Eq. (2) and $\mathcal{B}^t \leftarrow \mathcal{B}$ ▷ update SLA deficits
  while $(\exists \Delta_s^t > 0)$ and $(len(\mathcal{B}^t) > 0)$ **do**
    sel_UE $\leftarrow []$ ▷ Empty selected UE list
    Find G_l^t, G_s^t, and (k', b') using Eq. (4) and (5) ▷ large, small Δ groups, best User-RB pair
    Identify $\mathcal{N}_i^{b',t}$ such that $k' \in \mathcal{N}_i^{b',t}$ ▷ set of low-correlated users with user k'
    if DRO **then**
      $G_l^t \leftarrow \{k : slice(k) = slice(k')\}$ ▷ change G_l^t to set of all users in slice with user k'
    end if
    while $len(\text{sel\_UE}) < K$ **do**
      Set $\mathcal{K}_{large} \leftarrow \{k : k \in (\mathcal{N}_i^{b',t} \cap G_l^t)\}$ ▷ set of low-correlated users within G_l^t
      Sort $\mathcal{K}_{large}$ in descending order of channel gain
      sel_UE.$append$ $\left(\mathcal{K}_l \left[0 : \min(K - len(\text{sel\_UE}), len(\mathcal{K}_{large})) - 1\right]\right)$
      if $(len(\text{sel\_UE}) < K$ & DRS$)$ **then**
        $\mathcal{K}_{small} \leftarrow \{k : k \in (\mathcal{N}_i^{b',t} \cap G_s^t\}$ ▷ Pick from small delta group if DRS
        Sort $\mathcal{K}_{small}$ in descending order of channel gain
        sel_UE.$append$ $(\mathcal{K}_s [0 : \min(K - len(\text{sel\_UE}), len(\mathcal{K}_{small})) - 1])$
      end if
      if $len(\text{sel\_UE}) < K$ **then**
        Find $k' \leftarrow \underset{k \in G_l^t, k \notin \mathcal{N}_i^{b',t}}{\arg\max} ||\mathbf{h}_k^{b',t}||$, and set $\mathcal{N}_i^{b',t} \leftarrow \mathcal{N}_j^{b',t}$ s.t. $k' \in \mathcal{N}_j^{b',t}$
      end if
    end while
    Update Δ_s^t using Eq. (2)
    $RB\_alloc\_total++$
    $\mathcal{B}^t \leftarrow \mathcal{B}^t - \{b'\}$ ▷ remove selected RB b' from available RB set at TTI t
  end while
end for
return RB_alloc_total/T

C TOPOLOGY AND INTER-USER CORRELATION HEATMAP OF REAL-WORLD DATASET



(a) (b)

Fig. 9. (a) Topology of real-world dataset collection [1, 22] and (b) Average inter-user channel correlation among clusters reproduced from [22].

D ARTIFACT APPENDIX

D.1 Abstract

This section provides a concise summary of the artifacts accompanying this paper. We offer tools for conducting follow-up experiments, along with raw data and analysis scripts, enabling the reproduction of the results and figures presented herein.

D.2 Artifact Check-list (meta-information)

- **Algorithm: B&B, Greedy, Greedy Plus (GP), DRO, DRS, DRO_para and DRS_para**
- **Data set: RENEW Multi-User MIMO Dataset**
- **Run-time environment: Python (We also provide C++ codes)**
- **Metrics: Number of allocated RBs and running time**
- **How much time is needed to complete experiments (approximately)?: 30 mins**
- **Publicly available?: Yes**

D.3 Description

D.3.1 How to access. All artifacts are available via the following public repository: https://github.com/qinganrice/Helix_CoNEXT24

D.3.2 Hardware dependencies. Intel Xeon CPU cores

D.3.3 Software dependencies. The minimal requirements to reproduce our figures are Python with Numpy, h5py, and Matplotlib. For C++ codes, g++ 8.0, gcc 8.0 or higher are required.

D.3.4 Data sets. Dataset used in this work is uploaded in Google drive: https://drive.google.com/drive/folders/1whLRTUP45xOK6VDah8BsqVsyQHyqCyfB?usp=sharing The original dataset can be downloaded from RENEW Dataset website: https://renew.rice.edu/dataset-iuc.html

D.4 Code Structure

We now provide a detailed overview of the repository's structure.

RB_orthgonal:

- **BB.py**: Branch and bound algorithm for small-size and medium-size networks
- **BB_PF.py**: Proportional fairness policy based B&B
- **GP.py**: Greedy Plus algorithm for small-size and medium-size networks
- **GP_PF.py**: Proportional fairness policy based GP
- **Greedy.py**: Greedy algorithm for small-size and medium-size networks
- **Greedy_PF.py**: Proportional fairness policy based Greedy
- **DRO.py**: DRO algorithm for small-size and medium-size networks
- **DRO_PF.py**: Proportional fairness policy based DRO
- **DRO_real.py**: DRO algorithm for real-world-size network and mobility scenario
- **DRO_para.py**: RB parallelism based DRO algorithm for real-world-size network
- **max_rate.py**: functions of MR policy for exhaustive research
- **prop_fairness.py**: functions of PF policy for exhaustive research

RB_sharing:

- **DRS.py**: DRS algorithm for small-size and medium-size networks
- **DRS_PF.py**: Proportional fairness policy based DRS
- **DRS_real.py**: DRS algorithm for real-world-size network and mobility scenario
- **DRS_para.py**: RB parallelism based DRS algorithm for real-world-size network
- **RS_ES.py**: Exhaustive search based RB sharing algorithm

C_CODE: C++ codes of DRO and DRS for real-world platform integration

D.5 Evaluation Workflow and Expected Results

In this subsection, we will provide implementation details for reproducing all plots of this work.

Figure 5

- Set the right file path to load dataset
- Set ***config = 16*** for small-size network experiment
- Set ***SEL_UE = 3 (or 8)*** for different **K**
- ***uncorr = 1*** is low-correlated case and ***uncorr = 0*** is high-correlated case
- ***loose_sla = 1*** is loose SLA and ***loose_sla = 0*** is tight SLA
- Run **B&B**, **Greedy**, **GP**, **DRO**, **DRS** and **RS_ES** to reproduce Fig.5 in the paper

Figure 6

- Set the right file path to load dataset
- Set ***config = 80*** for medium-size network experiment
- Set ***SEL_UE = 8 (or 16)*** for different **K**
- ***uncorr = 1*** is low-correlated case and ***uncorr = 0*** is high-correlated case
- ***loose_sla = 1*** is loose SLA and ***loose_sla = 0*** is tight SLA
- Run **B&B**, **Greedy**, **GP**, **DRO**, and **DRS** to reproduce Fig.6 in the paper

Figure 7 (a)

- Set ***static = 1*** to load static scenario dataset
- Set ***config = 200*** for real-world-size network experiment
- ***uncorr = 1*** is low-correlated case and ***uncorr = 0*** is high-correlated case
- ***loose_sla = 1*** is loose SLA and ***loose_sla = 0*** is tight SLA
- Run **DRO_real**, and **DRS_real** to reproduce Fig.7 (a) in the paper

Figure 7 (b)

- Set ***static = 0 (or 2)*** to load slow (high) mobility scenario dataset
- Set ***config = 200*** for real-world-size network experiment
- Set ***uncorr = 0*** high-correlated case
- ***loose_sla = 1*** is loose SLA and ***loose_sla = 0*** is tight SLA
- Run **DRO_real**, and **DRS_real** to reproduce Fig.7 (b) in the paper

Figure 7 (c) and (d)

- Set the right file path to load dataset
- Set ***config = 80*** for medium-size network experiment
- Run **DRO_PF** and **DRS_PF** to reproduce results of **Proportional Fair** Fig.7 (c) and (d) in the paper
- For **Max Rate** results, it can be obtained from previous evaluations

Figure 8

- Set the right file path to load dataset
- Set ***config = 200*** for real-world-size network experiment (Take the most complicated case as example)
- ***new_rb_para = 1*** is RB parallelism and ***new_rb_para = 0*** is non-RB-parallelism
- Run **DRO_para** and **DRS_para** to reproduce **Scheduling latency** and **Num. of allocated RBs** in Fig. 8
- Other results of smaller size networks can be obtained similarly or from previous evaluations

Received June 2024; revised September 2024; accepted October 2024