Technical Report: Real-time MU-MIMO Beamforming with Limited CSI Data Samples

Shaoran Li† Nan Jiang‡ Chengzhang Li† Y. Thomas Hou† Wenjing Lou† Weijun Xie ‡
† Virginia Tech, Blacksburg, VA
‡ Georgia Tech, Atlanta, GA

***Abstract*—MU-MIMO beamforming is a key technology for 5G and Next-G networks and it requires Channel State Information (CSI). But in practice, CSI is bound to have uncertainty and only limited data samples are available to derive a beamforming solution. Further, an MU-MIMO beamforming solution must be derived in a millisecond to be useful for 5G in real-time. This paper addresses these issues by developing a real-time beamforming solution with limited CSI data samples. We present ReDBeam—a real-time data-driven beamforming solution for MU-MIMO. The main contribution of ReDBeam is a parallel algorithm and a GPU-based implementation that delivers an MU-MIMO beamforming solution within 1 millisecond while offering a probabilistic guarantee of data rates and minimizing power consumption associated with the beamforming solution. ReDBeam is purposefully designed to take advantage of the vast parallel processing capability offered by Commercial Off-The-Shelf (COTS) GPU. Through extensive experiments, we show that ReDBeam can meet the 1 millisecond real-time requirement and is orders of magnitude faster than other state-of-the-art algorithms for the same problem. ReDBeam conclusively demonstrates that MU-MIMO beamforming with a performance guarantee can be achieved in real-time when using only limited CSI data samples.**

I. Introduction

MU-MIMO is becoming ubiquitous in 5G and next-G networks because it is capable of increasing spectral efficiency and enhancing connectivity [1]–[3]. In MU-MIMO, the Base Station (BS) can transmit (or receive) different data streams to (or from) multiple User Equipments (UEs) simultaneously on the same spectrum [4]. Due to mutual interference among the UEs' data streams, beamforming is necessary to increase the received signal power and suppress interference. It is well-known that Channel State Information (CSI) is needed to derive a beamforming solution that ensures transmit signals are precoded in the correct directions.

There is a large body of works on MU-MIMO beamforming that assumes CSI as a given constant (see, e.g., [5]–[7]). However, such an assumption is unrealistic as there is always some discrepancy between the obtained CSI and the actual CSI. Such a discrepancy is due to a number of factors, such as channel estimation errors [8], [9], limited feedback [10], [11], and hardware imbalance [12], [13], among others. Therefore, a practical MU-MIMO beamforming solution must address *uncertainty* in CSI.

Among the existing works that have addressed CSI uncertainty, they can be categorized into two branches: *model-based* and *data-driven* approaches. Under the model-based approach, CSI is assumed to follow or observe some known distributions [14]–[18], channel statistics [19]–[21], or worst-case boundaries [22], [23]. These works typically offer a neat mathematical formulation and subsequently propose solutions with certain performance guarantees. However, these works are limited to their assumed models. With increased complexity in the operating environments for 5G and next-G networks, these assumed models are likely to lose their efficacy. On the other hand, the data-driven approaches (a.k.a. model-free) directly use CSI data samples to derive a beamforming solution. The prevailing examples of this approach are learning-based solutions (see, e.g., [24]–[26]). Data-driven solutions are highly adaptive to a wide range of scenarios and can easily meet real-time requirements. The problems with this approach are its requirement of a large amount of past CSI data samples for training and a lack of theoretical performance guarantee.

Recently, a new approach called D^{2}BF was proposed by [27] that combines the strengths of both model-based and data-driven approaches. Like the data-driven approaches, D^{2}BF works with CSI data samples to derive an MU-MIMO beamforming solution. But D^{2}BF can directly work with limited real-time CSI data samples without a complicated offline training process. Another attractive feature of D^{2}BF is that it can offer probabilistic performance guarantees to the UEs based on mathematical formulation and solutions, just like model-based approaches (which cannot be offered by existing data-driven approaches). The only limitation of D^{2}BF is its computational complexity, which cannot meet the stringent real-time requirement. By "real-time", we mean a beamforming solution must be derived on the order of one Transmission Time Interval (TTI) so that it can be used in time.

In this paper, we investigate this novel data-driven approach in [27] and address the real-time challenge that has been hindering its potentials. We consider the most common 5G-NR numerology 0, which has a TTI of 1 ms [28]. Therefore, we will use this 1 ms as the real-time requirement of our data-driven MU-MIMO beamforming solution. The main contributions of this paper are summarized as the following:

- We address the key limitation in a novel data-driven approach in [27] for MU-MIMO beamforming. This approach is extremely appealing as it combines the best features of state-of-the-art data-driven approaches (i.e., limited data samples) and model-based approaches (i.e., rigorous mathematical models, performance guarantees) without their pitfalls (e.g., a training process in data-driven approach and assumed channel knowledge in model-based approach). The only limitation of this new

approach is how to make it work in real-time. It is especially important to address this "real-time challenge" for 5G and next-G networks, where the available time for computation is limited to 1 ms.

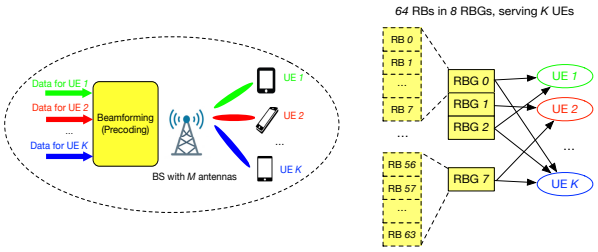
- We proposed a novel solution called ReDBeam, for Real-time Data-Driven Beamforming. The goals of RedBeam are: (i) derive a beamforming solution within 1 ms and (ii) minimize power consumption at the BS associated with the beamforming solution while guaranteeing probabilistic data rates for the UEs. ReDBeam is a parallel algorithm and is purposefully designed to take advantage of the vast parallel processing capability offered by Commercial Off-The-Shelf (COTS) GPU. The main ideas of ReDBeam are: (i) identify a promising subspace and generate a large number of initial solutions within this search space (ii) perform a local search for each initial solution in parallel to ensure feasibility and minimum power consumption, and (iii) choose the best feasible solution as the final solution.
- We implement ReDBeam on an NVIDIA V100 GPU hardware with CUDA programming and optimize our hardware implementation to minimize the total time consumption. Specifically, we implement ReDBeam in three parts, with each part as a kernel consisting of many small and independent steps to fit into the COTS GPU. We optimize each kernel by properly allocating GPU threads into thread blocks and efficiently choosing the small steps for each GPU thread to minimize execution time. To reduce memory access time, we use shared memory inside each kernel to store the temporary results and only use global memory to exchange results between different kernels.
- Through extensive experiments, we show that ReDBeam can meet the 1 ms timing requirement and this timing performance has effectively addressed the real-time challenge associated with D^{2}BF, which requires several orders of magnitude more computation time. Further, ReDBeam can guarantee probabilistic SINR thresholds (equivalent to data rates) for the UEs and is very close to D^{2}BF while consuming only slightly more power than D^{2}BF. When compared to a model-based algorithm (Gaussian Approximation), ReDBeam offers significantly better performance in running time and power consumption. Our successful implementation of RedBeam conclusively demonstrates that MU-MIMO beamforming with performance guarantee can be done in real-time using limited CSI data samples.

II. System Model and Mathematical Formulation

In this section, we introduce our system model and formulate our optimization problem. Table I lists the notations used in this paper.

A. MU-MIMO Beamforming

Figure 1 shows a 5G cell that employs MU-MIMO scheme to serve a group of UEs. Denote M as the number of antennas of the BS and denote K as the number of UEs. Denote $\mathcal{K} = \{1, 2, 3, \cdots, K\}$ as the set of K UEs. Without loss of generality, we assume that each UE only has one antenna. We consider downlink in this work but the proposed solution can be easily extended to the uplink case as well.



(a) MU-MIMO setting (b) RBGs for UEs

Fig. 1. Downlink MU-MIMO in a 5G cell.

TABLE I
Notations

Symbol	Definition
$\mathbb{C}^{M\times 1}$	The set of all complex $M \times 1$ column vectors
G	Number of RBGs
$\mathcal{G}$	The set of RBGs, i.e., $\mathcal{G} = \{1, 2, \cdots, G\}$
$\mathbf{h}_{(g,i)}$	An $M \times 1$ complex column vector including the CSI from the BS to UE i on RBG g
$\hat{\mathbf{h}}_{(g,i)}(n)$	The n-th data sample of $\mathbf{h}_{(g,i)}$
K	Number of UEs
$\mathcal{K}$	The set of UEs, i.e., $\mathcal{K} = \{1, 2, 3, \cdots, K\}$
$\mathcal{K}_g$	The subset of UEs from $\mathcal{K}$ scheduled on RBG g
$\lvert\mathcal{K}_g\rvert$	Number of UEs scheduled on RBG g
L	Number of start points in ReDBeam
M	Number of antennas at the BS
N	Number of data samples for each $\mathbf{h}_{(g,i)}$
$P^{\max}$	Maximum power budget of the BS on all RBGs
$\mathbb{P}_{\mathbf{h}_i}$	True but unknown distribution of $\mathbf{h}_{(g,i)}$
$\mathbf{w}_{(g,i)}$	Beamforming vector for UE i on RBG g, an $M \times 1$ complex column vector, i.e., $\mathbf{w}_i \in \mathbb{C}^{M\times 1}$
$\mathbf{w}^{\ell}_{(g,i)}$	Precoding vector for UE i in the ℓ-th start point
γ_i	Actual SINR at UE i
γ_i^{req}	Required SINR threshold at UE i
ϵ_i	Risk level for UE i
λ^{ℓ}	Scaling factor for the ℓ-th start point
σ_i^2	Power of the thermal noise at UE i

Following 5G terminology, the time domain and frequency domain are slotted into Transmission Time Intervals (TTIs) and sub-carriers. As defined in 5G-NR [28], 12 sub-carriers in one TTI are called one Resource Block (RB). To reduce the scheduling overhead, the BS can group multiple RBs into an RB Group (RBG) and use RBG as the granularity for scheduling. The number of RBs in one RBG can be $2, 4, 8$ or 16 [29]. In Fig. 1, 64 RBs are grouped into 8 RBGs (with 8 RBs in an RBG). Each RBG serves a subset (multiple) of UEs; each UE can receive from multiple RBGs. Denote $\mathcal{G} = \{1, 2, \cdots, G\}$ as the set of G RBGs at the BS. For RBG $g \in \mathcal{G}$, denote $\mathcal{K}_g$ as the subset of UEs that are selected to receive data from RBG g. Similar to existing works (e.g., [15], [16], [27]), we assume that $\mathcal{K}_g$ for each g is given *a priori*.

Based on the MU-MIMO scheme, UEs in $\mathcal{K}_g$ will simultaneously receive different data streams from the BS. So we need to design a unique precoding vector for each active UE

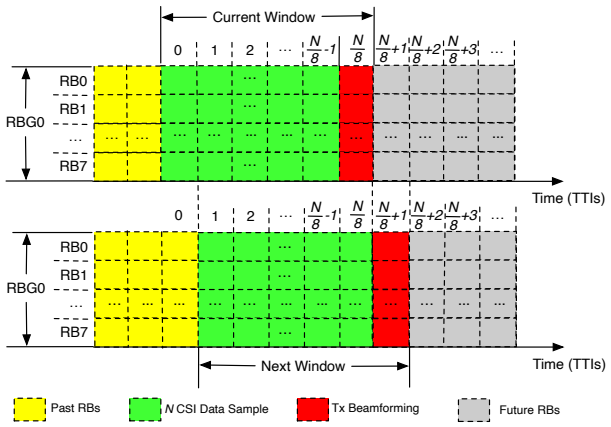


Fig. 2. Sliding window for CSI data samples.

on an RBG g. Denote $\mathbf{w}_{(g,i)}$ (an $M \times 1$ complex column vector) as the precoding vector for UE i on RBG g. These $\mathbf{w}_{(g,i)}$'s should be optimized by the BS before being applied to the downlink transmission. Specifically, $\mathbf{w}_{(g,i)}$ needs to satisfy certain constraints or requirements, such as the BS's total power budget on all RBGs and UEs' data rate requirements. Optimizing $\mathbf{w}_{(g,i)}$ requires knowledge of CSI from the BS to the UEs. As discussed in Section I, channel uncertainty is inevitable, thus CSI is intrinsically random. In this work, we will show how to use a small amount of CSI data samples to accomplish this big task.

B. CSI Data Samples from Sliding Window

Denote $\mathbf{h}_{(g,i)}$ (an $M \times 1$ complex column vector) as the CSI from the BS to UE i on RBG g. In a real-world 5G cell, such $\mathbf{h}_{(g,i)}$'s should be estimated during a channel sounding process. Channel sounding can be performed on each RB, then the estimated $\mathbf{h}_{(g,i)}$ on the RBs from the recent time slots can be collected as the data samples of $\mathbf{h}_{(g,i)}$. That is, a limited number of CSI data samples per $\mathbf{h}_{(g,i)}$ are available. Note that we do not require any other knowledge such as distributions, which is a much stronger requirement and is typically unavailable in practice.

Now we elaborate on how to obtain a limited number of CSI data samples and perform MU-MIMO beamforming using a sliding window mechanism. Figure 2 illustrates the idea, where each small rectangle represents an RB and each RBG consists of 8 RBs (i.e., $G = 8$). Each window covers $(N/G + 1)$ TTIs and has $N + G$ RBs. We can estimate the CSI from each UE on each RB, as is commonly assumed in MU-MIMO beamforming literature [4], [5], [30]. So we will use the N data samples collected in the most recent N/G TTIs (green) to design precoding vectors for the G RBs in the upcoming TTI (red). Given that these CSI samples are from neighboring RBs (either in frequency or TTIs), we assume the CSI in two neighboring windows follows the same (unknown) distribution. We will design beamforming vectors $\mathbf{w}_{(g,i)}$ solely based on these N CSI data samples. Under the sliding window mechanism shown in Fig. 2, we need to design an MU-MIMO beamforming solution within one TTI. Under the most common 5G numerology 0, one TTI is 1 ms, so we must obtain the precoding vectors within 1 ms. This is the real-time requirement for our MU-MIMO beamforming problem.

For our MU-MIMO beamforming problem, we use a small N, typically on the order of tens. When N becomes larger (more CSI data samples), we would expect a more accurate channel. But it also increases complexity (due to a larger problem size of MU-MIMO beamforming) with likely marginal improvement. On the other hand, when N becomes smaller, we may not have sufficient CSI data samples to address channel uncertainty in MU-MIMO beamforming and will likely experience poor performance. So the goal is to use the smallest possible N to design a real-time MU-MIMO beamforming solution (in 1 ms) so that the UEs' data rate requirements can be met. We note that this sliding window is a general form of the widely used "block-fading" model [31], where CSI is assumed to be constant on each block (a group of RBs) but is completely independent on different blocks. The main difference here is that the CSI is a random variable in our setting and we have no prior knowledge of its distribution.

Denote $\mathbb{P}_{\mathbf{h}_{(g,i)}}$ as the probability density function (PDF) of the unknown distribution of $\mathbf{h}_{(g,i)}$, i.e., $\mathbf{h}_{(g,i)} \sim \mathbb{P}_{\mathbf{h}_{(g,i)}}$. Then we have the N data samples of $\mathbf{h}_{(g,i)}$ drawn from the unknown distribution $\mathbb{P}_{\mathbf{h}_{(g,i)}}$. We denote this knowledge as:

$$\text{Unknown distribution: } \mathbf{h}_{(g,i)} \sim \mathbb{P}_{\mathbf{h}_{(g,i)}} \text{ , } N \text{ samples .} \quad (1)$$

C. Problem Formulation

We consider two requirements for the precoding vectors $\mathbf{w}_{(g,i)}$. The first is the power budget. Denote $P^{\max}$ as the power budget at the BS. Then the total transmission power from the BS over all RBGs (to all UEs) cannot exceed $P^{\max}$, i.e.,

$$\sum_{g \in \mathcal{G}} \sum_{i \in \mathcal{K}_g} ||\mathbf{w}_{(g,i)}||_2^2 \leq P^{\max} , \quad (2)$$

where $|| \cdot ||_2$ is the L_2-norm.

The second is on UE's service requirement and we assume each UE has a data rate requirement. Meeting this data rate requirement is equivalent to meeting an SINR threshold [16] for the given RBG bandwidth. When a UE receives data from multiple RBGs, it must use the same Modulation and Coding Scheme (MCS) on all its allocated RBGs per 5G standards [29]. So it must use the same SINR threshold on all its RBGs. Denote γ_i^{req} as the SINR threshold for UE i, which is a given constant for each MU-MIMO beamforming instance. Denote $\gamma_{(g,i)}$ as the actual SINR at UE i, which depends on the uncertain CSI $\mathbf{h}_{(g,i)}$ and the precoding vectors $\mathbf{w}_{(g,i)}$:

$$\gamma_{(g,i)} = \frac{|(\mathbf{w}_{(g,i)})^H \mathbf{h}_{(g,i)}|^2}{\sum_{k \in \mathcal{K}_g}^{k \neq i} |(\mathbf{w}_{(g,k)})^H \mathbf{h}_{(g,i)}|^2 + \sigma_i^2} \quad (i \in \mathcal{K}_g, g \in \mathcal{G}) , \quad (3)$$

where $(\cdot)^H$ denotes conjugate transpose. σ_i^2 is the power of thermal noise at UE i.

Due to channel uncertainty, the CSI $\mathbf{h}_{(g,i)}$'s should be modeled as random variables (based on N data samples).

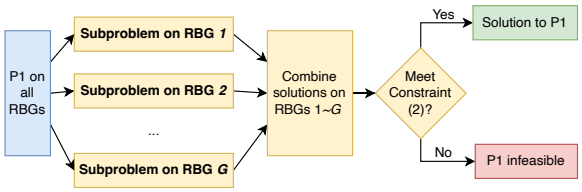


Fig. 3. Recover the solution to P1.

Consequently, the actual SINRs $\gamma_{(g,i)}$ are also random variables. To cope with such randomness, we employ probabilistic constraint for $\gamma_{(g,i)}$ as follows:

$$\mathbb{P}\left\{\gamma_{(g,i)} \geq \gamma_i^{\text{req}}\right\} \geq 1-\epsilon_i \qquad (i \in \mathcal{K}_g, g \in \mathcal{G}) \;, \tag{4}$$

where $\mathbb{P}\{\cdot\}$ denotes the probability function, ϵ_i is called *risk level* and is the upper bound of γ_i^{req} violation probability for UE i. Constraints (4) state that the actual SINR $\gamma_{(g,i)}$ on RBG g should be greater or equal than the required SINR threshold γ_i^{req} with a probability at least $1-\epsilon_i$. Clearly, a larger ϵ_i means a higher tolerance of SINR threshold violation and a larger optimization space. In the special case when $\epsilon_i = 0$, constraints (4) become deterministic as $\gamma_{(g,i)} \geq \gamma_i^{\text{req}}$ always holds.

Substituting (3) into (4), we have

$$\mathbb{P}\left\{\frac{|(\mathbf{w}_{(g,i)})^H \mathbf{h}_{(g,i)}|^2}{\gamma_i^{\text{req}}} - \sum_{k\in\mathcal{K}_g}^{k\neq i} |(\mathbf{w}_{(g,k)})^H \mathbf{h}_{(g,i)}|^2 \geq \sigma_i^2\right\} \geq 1-\epsilon_i \qquad (i \in \mathcal{K}_g,\ g \in \mathcal{G}) \;. \tag{5}$$

In this work, we are interested in minimizing the BS's power consumption while meeting the UEs' probabilistic data rate requirements. Our problem can be stated as follows:

$$\begin{aligned}(\text{P1}) \ \min_{\mathbf{w}_{(g,i)}} \ & \sum_{g\in\mathcal{G}}\sum_{i\in\mathcal{K}_g} ||\mathbf{w}_{(g,i)}||_2^2 \\ \text{s.t.} \ & \text{BS power budget (2)} \;, \\ & \text{Probabilistic SINR guarantees (5)} \;, \\ & \text{CSI data samples with unknown distribution (1)} \;, \\ & \mathbf{w}_{(g,i)} \in \mathbb{C}^{M\times 1} \;,\end{aligned}$$

where $\mathbb{C}^{M\times 1}$ is the set of all complex $M \times 1$ column vectors.

Since P1 is a *chance-constrained program*, we will first reformulate it into a deterministic optimization problem. The reformulation of P1 mainly focuses on the probabilistic SINR guarantees (5) using N CSI data samples (1). In [27], the authors showed that P1 can be decomposed into G parallel subproblems, where each subproblem corresponds to MU-MIMO beamforming on one RBG. Then each subproblem can be equivalently reformulated into a *deterministic* problem based on the empirical distribution of $\mathbf{h}_{(g,i)}$ from its N data samples using ∞-Wasserstein ambiguity set [32]. Figure 3 shows how we can recover the solution to P1 after we solve all subproblems. This solution recovery process in Figure 3 does not introduce relaxation errors and has negligible computation efforts (a summation of G real numbers and a comparison with $P^{\max}$). Therefore, we only need to solve the G subproblems.

Since the G subproblems are independent and can be solved in parallel, we will develop a solution to solve one subproblem w.r.t. g ($g \in \mathcal{G}$), given as:

$$\begin{aligned}(\text{P2}) \ \min_{\mathbf{w}_{(g,i)}} \ & \sum_{i\in\mathcal{K}_g} ||\mathbf{w}_{(g,i)}||_2^2 \\ \text{s.t.} \ & \frac{1}{N}\cdot\sum_{n=1}^{N} \mathbb{I}\Big\{\hat{f}\big(\mathbf{w}_{(g,i)}, \hat{\mathbf{h}}_{(g,i)}(n)\big) \geq \sigma_i^2\Big\} \geq 1-\epsilon_i \quad (i \in \mathcal{K}_g) \;, \\ & \mathbf{w}_{(g,i)} \in \mathbb{C}^{M\times 1} \quad (i \in \mathcal{K}_g) \;,\end{aligned}$$

where $\hat{f}\big(\mathbf{w}_{(g,i)}, \hat{\mathbf{h}}_{(g,i)}(n)\big)$ is defined as

$$\begin{aligned}&\hat{f}\big(\mathbf{w}_{(g,i)}, \hat{\mathbf{h}}_{(g,i)}(n)\big) = \\ &\Big\{\min_{\mathbf{c}_i} \ \Big(\frac{|(\mathbf{w}_{(g,i)})^H \mathbf{c}_i|^2}{\gamma_i^{\text{req}}} - \sum_{k\in\mathcal{K}_g}^{k\neq i} |(\mathbf{w}_{(g,k)})^H \mathbf{c}_i|^2\Big) \\ &\quad \text{s.t.} \ \ ||\mathbf{c}_i - \hat{\mathbf{h}}_{(g,i)}(n)||_2 \leq \theta_{(g,i)} \ , \ \mathbf{c}_i \in \mathbb{C}^{M\times 1}\Big\}\end{aligned} \tag{6}$$

In P2, $\hat{\mathbf{h}}_{(g,i)}(n)$ is the n-th data sample of $\mathbf{h}_{(g,i)}$, $\mathbb{I}(\cdot)$ is the indicator function, and $\theta_{(g,i)}$ is a small constant that represents the search space (distance) of $\mathbf{c}_i$ around each CSI data sample [27], [32], [33]. Intuitively, $\theta_{(g,i)}$ should be larger when N CSI data samples deviate further from the true (but unknown) distribution. We will show a simple approach to set $\theta_{(g,i)}$ based on the N CSI data samples of the current window in Section V. Note that the unknown CSI $\mathbf{h}_{(g,i)}$ in P1 has been replaced by N CSI data sample $\hat{\mathbf{h}}_{(g,i)}(n)$ in P2. Thus, P2 is a deterministic optimization problem.

Technical Challenges P2 has an interesting mathematical structure. In P2, an optimization problem is nested within the constraint, through the calculation of $\hat{f}\big(\mathbf{w}_{(g,i)}, \hat{\mathbf{h}}_{(g,i)}(n)\big)$. It is a non-convex Quadratically Constrained Quadratic Program (QCQP). For a given $\mathbf{w}_{(g,i)}$, we need to solve this QCQP by finding an optimal $\mathbf{c}_i$ that minimizes the difference between two terms. Note that there are a total of $|\mathcal{K}_g| \cdot N$ $\hat{f}(\mathbf{w}_{(g,i)}, \hat{\mathbf{h}}_{(g,i)}(n))$ in P2 where $|\mathcal{K}_g|$ is the number of UEs served on RBG g. Then we plug these $|\mathcal{K}_g| \cdot N$ objective values into the first set of constraints with indicator functions $\mathbb{I}(\cdot)$ to check whether the given $\mathbf{w}_{(g,i)}$ is a feasible solution to P2 or not. That is, checking the feasibility for a given $\mathbf{w}_{(g,i)}$ requires solving $|\mathcal{K}_g| \cdot N$ non-convex QCQPs.

Second, it is not clear how to optimize $\mathbf{w}_{(g,i)}$ considering the complicated mathematical structures of P2. Iterative algorithms for MU-MIMO (e.g., [5], [6], [27]) require substantial computation time and cannot meet our real-time requirement (1 ms). In contrast, the goal of this paper is to develop a real-time solution with low complexity that sufficiently explores the search space for $\mathbf{w}_{(g,i)}$ (all $M \times 1$ complex column vectors).

III. ReDBeam: Real-time Data-Driven Beamforming

In this section, we present ReDBeam—a Real-time Data-Driven Beamforming solution to P2. In Section IV, we present a GPU implementation of ReDBeam.

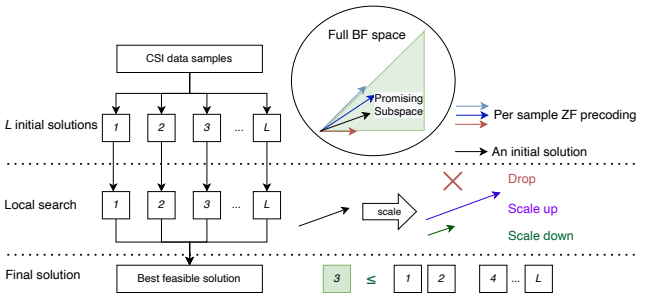


Fig. 4. Key steps of ReDBeam.

By design, ReDBeam is a parallel algorithm that is suitable for implementation on COTS GPU. Figure 4 shows the three key steps of ReDBeam. First, it generates a sufficiently large number (denoted as L) of initial solutions in a promising search space. We call the search space "promising" because it is generated as a cone based on Zero-Forcing (ZF) precoding of the CSI data samples. Clearly, there is no guarantee of feasibility or good performance for any of these L initial solutions. So in the second step, we employ a scaling-based local search to find feasible solutions based on these L initial solutions. Then in the final step, we find the best feasible solution by comparing their achieved objective values. The one with the minimum objective value will be chosen as our solution to P2. Although the main idea of ReDBeam is easy to understand, how to do the first and second steps is non-trivial.

A. Generating A Population of Initial Solutions

In this step, we generate a sufficiently large number (L) of initial solutions within a promising search space. The "promising search space" is a subspace formed by some basis vectors and it should contain many feasible beamforming vectors with satisfactory performance. It is possible that the optimal beamforming solution falls outside of this search space but it is not a big issue, as long as we can find a good solution within this space.

A Promising Search Space For ease of exposition, we drop the subscript g when there is no confusion. The original search space for $\mathbf{w}_i$ is $\mathbb{C}^{M\times 1}$, which is too large. To narrow down this search space, we observe that a promising direction for $\mathbf{w}_i \in \mathbb{C}^{M\times 1}$ should increase the received power and suppress the received interference for UE i. Based on this observation, we identify a promising search space to be a cone whose basis vectors are derived from ZF precoding based on CSI data samples. ZF precoding is widely used in practice due to its low complexity and good performance [30].

Denote these N basis vectors as $\mathbf{v}_i(n)$, $n = 1, 2, \cdots, N$. Each $\mathbf{v}_i(n)$ is an $M \times 1$ complex column vector, which is the ZF precoding vector for UE i under the n-th CSI data sample. For a given n, the calculation of $\mathbf{v}_i(n)$, $i \in \mathcal{K}_g$ is based on the CSI data samples $\hat{\mathbf{h}}_i(n)$, $i \in \mathcal{K}_g$, i.e., the CSI data samples from the UEs in $\mathcal{K}_g$. Define an $M \times |\mathcal{K}_g|$ matrix $\hat{\mathbf{H}}(n)$ as:

$$\hat{\mathbf{H}}(n) = \left[\hat{\mathbf{h}}_1(n) \quad \hat{\mathbf{h}}_2(n) \quad \cdots \quad \hat{\mathbf{h}}_{|\mathcal{K}_g|}(n)\right],$$

where the i-th column of $\hat{\mathbf{H}}(n)$ is the CSI data sample for UE i. We can calculate the ZF precoding vectors $\mathbf{v}_i(n)$'s based on $\hat{\mathbf{H}}(n)$ following the deterministic CSI model. Since there are N $\hat{\mathbf{H}}(n)$'s, we can calculate the ZF precoding vectors for each $\hat{\mathbf{H}}(n)$ in parallel for $n = 1, 2, \cdots, N$.

In ZF precoding, $\mathbf{v}_i(n)$ is related to the Moore-Penrose pseudo-inverse of $\hat{\mathbf{H}}(n)$. Due to the zero interference property of ZF precoding, the received SINR at UE i is $|\left(\mathbf{v}_i(n)\right)^H \hat{\mathbf{h}}_i(n)|^2/\sigma_i^2$ and we would like it to be no smaller than the SINR threshold γ_i^{req}. Denote a $|\mathcal{K}_g| \times M$ complex matrix $\hat{\mathbf{H}}(n)^\dagger$ as the Moore-Penrose pseudo-inverse of $\hat{\mathbf{H}}(n)$. Denote $\mathbf{u}_i(n)$, a $M\times 1$ complex column vector, as the complex conjugate of the i-th row of $\hat{\mathbf{H}}(n)^\dagger$, i.e.,

$$\hat{\mathbf{H}}(n)^\dagger = \left[\mathbf{u}_1(n) \quad \mathbf{u}_2(n) \quad \cdots \quad \mathbf{u}_{|\mathcal{K}_g|}(n)\right]^H.$$

Therefore, based on the fact that $\left(\mathbf{u}_i(n)\right)^H \hat{\mathbf{h}}_i(n) = 1$, $\mathbf{v}_i(n)$ is given as:

$$\mathbf{v}_i(n) = \sigma_i \sqrt{\gamma_i^{\text{req}}} \cdot \mathbf{u}_i(n) \quad (i \in \mathcal{K}_g,\ n = 1, 2, \cdots, N), \quad (7)$$

which means that $\mathbf{v}_i(n)$ follows the same direction as $\mathbf{u}_i(n)$.

Clearly, the main computation complexity for calculating $\mathbf{v}_i(n)$ in (7) is the calculation of $\hat{\mathbf{H}}(n)^\dagger$. There are many existing methods to calculate $\hat{\mathbf{H}}(n)^\dagger$ and in ReDBeam, we calculate $\hat{\mathbf{H}}(n)^\dagger$ based on QR decomposition and forward/backward substitutions since some computations in this approach can be done in parallel [34].

After calculating (7), we have N sets of precoding vectors. We define a cone formed by these N precoding vectors, in which we will search for $\mathbf{w}_i$, i.e.,

$$\mathbf{w}_i \in \left\{\mathbf{e}\colon \mathbf{e} = \sum_{n=1}^{N} \alpha_i(n)\mathbf{v}_i(n),\ \alpha_i(n) \geq 0\right\} \ (i \in \mathcal{K}_g), \quad (8)$$

where each vector inside this cone is a linear combination of the N basis vectors with $\alpha_i(n) \geq 0$, $i \in \mathcal{K}_g$.

Sampling Now we have a promising search space for $\mathbf{w}_i$ but it still contains an infinite number of complex $M \times 1$ column vectors. To further narrow down the search space, we generate L initial solutions (through sampling) inside this cone.

Denote the ℓ-th initial solution as $\mathbf{z}_i^\ell$ where $\ell = 1, 2, 3, \cdots, L$. To generate $\mathbf{z}_i^\ell$, we choose each coefficient $\alpha_i^\ell(n)$, $n = 1, 2, \cdots, N$ in (8) following a uniform distribution between $[0, 1]$. Then we scale the $\alpha_i^\ell(n)$'s proportionally so that their sum is normalized to 1, i.e., $\sum_{n=1}^{N} \alpha_i^\ell(n) = 1$. This means each initial solution $\mathbf{z}_i^\ell = \sum_{n=1}^{N} \alpha_i^\ell(n)\mathbf{v}_i(n)$. Clearly, finding the L initial solutions can be done in parallel since they are independent of each other.

B. Finding Good Solutions via Local Search

Now we have L initial solutions $\mathbf{z}_i^\ell$, $\ell = 1, 2, \cdots, L$, $i \in \mathcal{K}_g$. However, these L initial solutions neither guarantee feasibility nor good performance. So we will perform a local search on each of these L initial solutions so that i) each new solution is feasible (if possible), and ii) the objective function of P2 is minimized.

Main Idea With 1 ms real-time requirement, the local search must be simple and fast, with as few steps as possible. With this in mind, we limit our local search only to the scaling of the length (or norm) of $\mathbf{z}_i^\ell$, i.e., without creating new directions. Denote $\mathbf{w}_i^\ell$ as the solution for P2 after scaling of $\mathbf{z}_i^\ell$. For each initial solution $\mathbf{z}_i^\ell$, $i \in \mathcal{K}_g$, $\ell = 1, 2, \cdots, L$, we scale it with a factor of $\lambda^\ell > 0$ uniformly for all $i \in \mathcal{K}_g$ to obtain $\mathbf{w}_i^\ell$, i.e.,

$$\mathbf{w}_i^\ell = \lambda^\ell \cdot \mathbf{z}_i^\ell \quad (i \in \mathcal{K}_g) \ . \tag{9}$$

With the scaling in (9), the objective function of P2 becomes $\sum_{i\in\mathcal{K}_g} ||\lambda^\ell \cdot \mathbf{z}_i^\ell||_2^2$, which is $(\lambda^\ell)^2 \cdot \sum_{i\in\mathcal{K}_g} ||\mathbf{z}_i^\ell||_2^2$. Since $\sum_{i\in\mathcal{K}_g} ||\mathbf{z}_i^\ell||_2^2$ is a constant when $\mathbf{z}_i^\ell$'s are given, the objective of P2 can be replaced by $\min \ \lambda^\ell$. Further, based on the definition of $\hat{f}(\mathbf{w}_i^\ell, \hat{\mathbf{h}}_i(n))$ in P2, we have

$$\hat{f}(\mathbf{w}_i^\ell, \hat{\mathbf{h}}_i(n)) = \hat{f}(\lambda^\ell \mathbf{z}_i^\ell, \hat{\mathbf{h}}_i(n)) = (\lambda^\ell)^2 \hat{f}\big(\mathbf{z}_i^\ell, \hat{\mathbf{h}}_i(n)\big) \ . \tag{10}$$

Therefore, with given $\mathbf{z}_i^\ell$'s, we can rewrite P2 as follows:

$$\text{(P3)} \ \min \lambda^\ell$$
$$\text{s.t.} \sum_{n=1}^{N} \mathbb{I}\Big\{(\lambda^\ell)^2 \hat{f}\big(\mathbf{z}_i^\ell, \hat{\mathbf{h}}_i(n)\big) \geq \sigma_i^2\Big\} \geq N \cdot (1-\epsilon_i) \quad (i \in \mathcal{K}_g)$$
$$\text{Definition of } \hat{f}\big(\mathbf{z}_i^\ell, \hat{\mathbf{h}}_i(n)\big) \text{ in (6)}, \ \lambda^\ell > 0 \ .$$

Clearly, the main difficulty of P3 is $\hat{f}\big(\mathbf{z}_i^\ell, \hat{\mathbf{h}}_i(n)\big)$, which involves non-convex QCQPs as defined in (6). In the following paragraphs, we first show how to calculate $\hat{f}\big(\mathbf{z}_i^\ell, \hat{\mathbf{h}}_i(n)\big)$. Then we show how to find λ^ℓ in P3.

Calculation of $\hat{f}\big(\mathbf{z}_i^\ell, \hat{\mathbf{h}}_i(n)\big)$ In P3, there are $N \cdot |\mathcal{K}_g|$ terms of $\hat{f}\big(\mathbf{z}_i^\ell, \hat{\mathbf{h}}_i(n)\big)$'s. Since these terms are independent of each other, we can solve them in parallel. Based on (6), for a specific $\hat{f}\big(\mathbf{z}_i^\ell, \hat{\mathbf{h}}_i(n)\big)$, we need to solve

$$\text{(P4)} \ \min_{\mathbf{c}_i} \quad \Big(\frac{|(\mathbf{z}_i^\ell)^H \mathbf{c}_i|^2}{\gamma_i^{\text{req}}} - \sum_{k\in\mathcal{K}_g}^{k\neq i} |(\mathbf{z}_k^\ell)^H \mathbf{c}_i|^2\Big)$$
$$\text{s.t.} \quad ||\mathbf{c}_i - \hat{\mathbf{h}}_i(n)||_2 \leq \theta_i \ .$$

Unfortunately, P4 is a non-convex QCQP [35], which is NP-hard in general. Therefore, we will find a lower bound for the optimal objective value of P4 and use it for $\hat{f}\big(\mathbf{z}_i^\ell, \hat{\mathbf{h}}_i(n)\big)$ in P3. This will lead to a slightly larger λ^ℓ in the objective value in P3. Nevertheless, all constraints will remain satisfied except the objective (power consumption) may be slightly higher, which is the nature of a satisfactory solution.

To obtain a lower bound for P4, we relax its objective function by separating the $|\mathcal{K}_g|$ terms, i.e.,

$$\min_{\mathbf{c}_i} \ \frac{|(\mathbf{z}_i^\ell)^H \mathbf{c}_i|^2}{\gamma_i^{\text{req}}} - \sum_{k\in\mathcal{K}_g}^{k\neq i} \max_{\mathbf{c}_i} |(\mathbf{z}_k^\ell)^H \mathbf{c}_i|^2 \tag{11}$$

The gap of this lower bound depends on the parameters $\mathbf{z}_i^\ell, i \in \mathcal{K}_g$ and $\hat{\mathbf{h}}_i(n)$. Note that the first term is related to the received signal while the other $(|\mathcal{K}_g| - 1)$ terms are related to interference. Since the initial solution $\mathbf{z}_i^\ell$ is chosen from our promising space based on ZF precoding, the interference terms are usually small. Given that $|\mathcal{K}_g|$ (number of UEs per RBG) is typically $2 \sim 4$, the terms in the summation are small.

Based on the $|\mathcal{K}_g|$ terms in (11), P4 can be decomposed into two sets of independent optimization problems P4-A and P4-B as follows:

$$\text{(P4-A)} \quad \min_{\mathbf{c}_i} \quad \frac{|(\mathbf{z}_i^\ell)^H \mathbf{c}_i|^2}{\gamma_i^{\text{req}}}$$
$$\text{s.t.} \quad ||\mathbf{c}_i - \hat{\mathbf{h}}_i(n)||_2 \leq \theta_i \ ,$$

and for $k \in \mathcal{K}_g, \ k \neq i$,

$$\text{(P4-B)} \quad \max_{\mathbf{c}_i} \quad |(\mathbf{z}_k^\ell)^H \mathbf{c}_i|^2$$
$$\text{s.t.} \quad ||\mathbf{c}_i - \hat{\mathbf{h}}_i(n)||_2 \leq \theta_i \ .$$

Both (P4-A) and (P4-B) have only one decision variable term in their objective functions and have closed-form solutions. Denote $d_i^\ell(n)$ as the optimal objective for (P4-A) and $d_k^\ell(n), k \in \mathcal{K}_g, k \neq i$ as the the optimal objective for (P4-B) respectively. Then $d_i^\ell(n)$ and $d_k^\ell(n), k \in \mathcal{K}_g, k \neq i$ are given as follows:

$$d_i^\ell(n) = \frac{1}{\gamma_i^{\text{req}}} \left(\max\left\{0, \left(|(\mathbf{z}_i^\ell)^H \hat{\mathbf{h}}_i(n)| - \theta_i \cdot ||\mathbf{z}_i^\ell||_2\right)^2\right\}\right) \ ,$$
$$d_k^\ell(n) = \left(|(\mathbf{z}_k^\ell)^H \hat{\mathbf{h}}_i(n)| + \theta_i \cdot ||\mathbf{z}_k^\ell||_2\right)^2 \quad (k \in \mathcal{K}_g, k \neq i) \ .$$

Clearly, $d_i^\ell(n)$ and $d_k^\ell(n)$ can be calculated in parallel. Therefore, we obtain a lower bound for P4's objective function (i.e., $\hat{f}\big(\mathbf{z}_i^\ell, \hat{\mathbf{h}}_i(n)\big)$) as follows:

$$d_i^\ell(n) - \sum_{k\in\mathcal{K}_g}^{k\neq i} d_k^\ell(n) \quad (i \in \mathcal{K}_g, \ n = 1, 2, \cdots, N) \ . \tag{12}$$

Solution to P3 Substituting $\hat{f}\big(\mathbf{z}_i^\ell, \hat{\mathbf{h}}_i(n)\big)$ with the lower bound in (12), we can rewrite the constraints in P3 as

$$\sum_{n=1}^{N} \mathbb{I}\Big\{(\lambda^\ell)^2 \Big(d_i^\ell(n) - \sum_{k\in\mathcal{K}_g}^{k\neq i} d_k^\ell(n)\Big) \geq \sigma_i^2\Big\} \geq N \cdot (1-\epsilon_i) \quad (i \in \mathcal{K}_g) \ . \tag{13}$$

There are $|\mathcal{K}_g|$ constraints in (13). Denote β_i^ℓ as the minimum λ^l to satisfy the i-th constraint of (13). β_i^ℓ can be easily found by sorting N real numbers in non-decreasing order and set β_i^ℓ as the $(1-\epsilon_i)$-quantile, i.e., the $\lceil N \cdot (1-\epsilon_i) \rceil$-th element after sorting. Specifically, there are two cases:

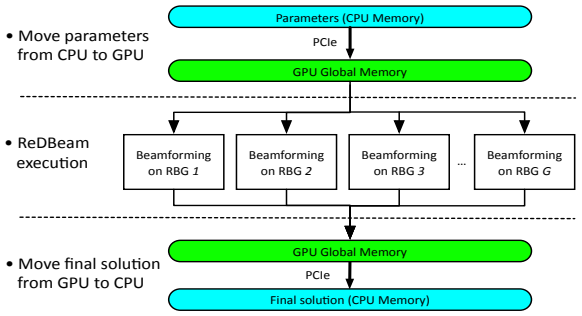
i) If for all $i \in \mathcal{K}_g$, $\big(d_i^\ell(n) - \sum_{k\in\mathcal{K}_g}^{k\neq i} d_k^\ell(n)\big) > 0$ holds for at least $\lceil N \cdot (1-\epsilon_i) \rceil$ CSI data samples, then β_i^ℓ is a positive number. To satisfy all constraints in (13), we simply set λ^ℓ as

$$\lambda^\ell = \max_{i\in\mathcal{K}_g} \ \beta_i^\ell \ . \tag{14}$$

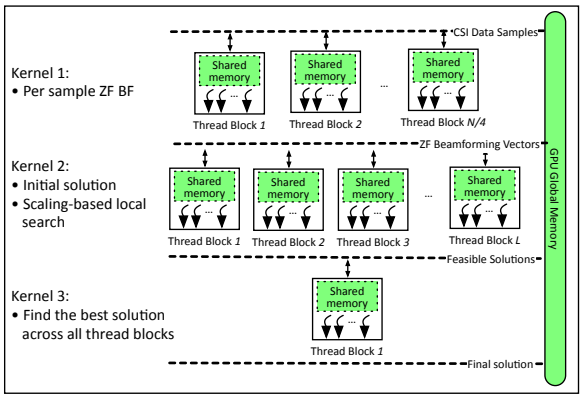
Using λ^ℓ from (14) to (9), we have a feasible solution as

$$\mathbf{w}_i^\ell = \Big(\max_{i\in\mathcal{K}_g} \ \beta_i^\ell\Big) \cdot \sum_{n=1}^{N} \alpha_i^\ell(n) \mathbf{v}_i(n) \quad (i \in \mathcal{K}_g) \ . \tag{15}$$

ii) Otherwise, i.e., for some $i \in \mathcal{K}_g$, $\big(d_i^\ell(n) - \sum_{k\in\mathcal{K}_g}^{k\neq i} d_k^\ell(n)\big) \leq 0$ holds for at least $\lceil N \cdot \epsilon_i \rceil$



(a) A flow chart of GPU Implementation



(b) Beamforming on an RBG

Fig. 5. A GPU Implementation of ReDBeam.

CSI data samples, then both β_i^ℓ and λ^ℓ do not exist as there is no feasible solution to (13) under this initial solution $\mathbf{z}_i^\ell$. We can simply drop this initial solution $\mathbf{z}_i^\ell$.

C. Finding the Final Solution

After the local search, we have at most L feasible solutions, we can calculate their objectives (i.e., $\sum_{i\in\mathcal{K}_g}||\mathbf{w}_i^\ell||_2^2$) in parallel. Then we compare these objectives and find the one with the smallest objective (since P2 is a minimization problem). The feasible solution $\mathbf{w}_i^\ell$ associated with this smallest objective value is our final solution to P2 on RBG g.

IV. GPU Implementation

In this section, we present a GPU implementation of ReDBeam. The goal is to ensure the running time of ReDBeam is within 1 ms time requirement.

A. Overview

We choose GPU to implement our proposed ReDBeam due to its massive parallel computing capability. A flow chart of our GPU Implementation of ReDBeam is given in Fig. 5(a). It consists of two data transfers and one algorithm execution. All three parts count toward the total time consumption. These two data transfers are necessary since we are using a COTS GPU. To reduce these data transfer times, we use asynchronous data transfer between CPU and GPU. We also overlap data transfers and kernel executions between different streams [36] .

In Fig. 5(a), there are G streams and each stream calculates the beamforming solution on a specific RBG and is independent of other streams. The details of each stream are given in Fig. 5(b), which corresponds to a P2 instance. We design three kernels and kernel 2 is most involved—it generates an initial solution $\mathbf{z}_i^\ell$ and immediately performs the local search to obtain $\mathbf{w}_i^\ell$, which reduces the overall time consumption.

To meet the real-time requirement (1 ms), we need to efficiently allocate the available resources on a GPU, such as *threads* for computation and *memory* for data storage. A *thread* is the minimum processing unit for algorithm execution and threads are grouped into *thread blocks* to execute *kernels*. Each thread block will be allocated to a Streaming Multiprocessor (SM) for execution and the number of SMs depends on the COTS GPU hardware, e.g., 80 in our NVIDIA V100 GPU. Threads can run in parallel or sequence and we need to properly program the operations of the threads to correctly execute ReDBeam while reducing the execution time.

Further, optimizing *memory* management can reduce memory access time. In our GPU implementation, we mainly use *global memory* and *shared memory*. Global memory has a large volume (e.g., several gigabytes) and can exchange data with external platforms (i.e., CPU or other GPUs) while shared memory has a faster access time but a smaller volume (e.g., 48 kilobytes per SM in our NVIDIA V100 GPU) and no external access outside of a thread block. Thus, shared memory is more suitable for repeatedly accessed data.

B. Details of Three Kernels

As shown in Fig. 5(b), there are three kernels in our implementation. Each kernel reads data from GPU global memory, processes the data, and then stores the results back in the GPU global memory for later use.

Kernel 1 Kernel 1 is for the calculation of ZF precoding vectors $\mathbf{v}_i(n)$ based on CSI data samples $\hat{\mathbf{h}}_i(n)$, $i \in \mathcal{K}_g$, $n = 1, 2, \cdots, N$. Since there are N sets of CSI data samples, we need to calculate N sets of ZF precoding vectors in parallel. Calculating one set of ZF precoding vectors only requires $M|\mathcal{K}_g|$ threads, which can be smaller than the number of threads on an SM. For example, we use $M = 8$ and $|\mathcal{K}_g| = 2$ in our case study. Then $M|\mathcal{K}_g| = 16$, which is smaller than the number of threads per SM in our NVIDIA V100 GPU. This means some of the threads will be idle. To maximize the parallel computing capability of a GPU so that all threads on an SM are utilized, we will use one thread block to calculate multiple sets of ZF precoding vectors. In our GPU implementation, each thread block calculates 4 sets of ZF precoding vectors using $4M|\mathcal{K}_g|$ threads[1].

To further reduce the time consumption, we will enhance parallelization when calculating each set of ZF precoding vectors based on QR decomposition, forward/backward substitutions, and (7). For instance, we need to perform QR decomposition of $\hat{\mathbf{H}}(n)$ (an $M \times |\mathcal{K}_g|$ complex matrix) that requires the calculation of an $M \times |\mathcal{K}_g|$ complex upper triangular matrix $\mathbf{R}(n)$. With the help of $M|\mathcal{K}_g|$ threads, we can calculate each column of $\mathbf{R}(n)$ at the same time

[1]This number of ZF precoding vectors per thread block can be easily changed by operators based on network size.

instead of calculating the elements of $\mathbf{R}(n)$ sequentially, which reduces the number of iterations and leads to a smaller time consumption. Similar ideas of parallelization are also applied to the forward/backward substitution and (7).

In terms of memory management, all temporary results such as $\mathbf{R}(n)$ are stored in the shared memory to reduce memory access time since they are frequently used in kernel 1 but are not needed in other kernels. The ZF precoding vectors $\mathbf{v}_i(n)$ will be stored to the GPU global memory for later use.

Kernel 2 Each kernel 2 generates an initial solution $\mathbf{z}_i^\ell$ from (8), finds the scaling factor λ^ℓ by (14) and applies λ^ℓ to obtain $\mathbf{w}_i^\ell = \lambda^\ell \cdot \mathbf{z}_i^\ell$ by (9). Since there are L initial solutions, we use L thread blocks and each thread block has $N|\mathcal{K}_g|$ threads.

The core step of generating an initial solution is to calculate $|\mathcal{K}_g|M$ sums of N complex numbers with randomly generated $\alpha_i(n)$. Note that the promising search space (8) will be implicitly included during this process. A common technique to reduce execution time in GPU implementation is parallel reduction, which is suitable for comparison or summation over a large number of terms [37]. For a sum of N numbers, we need $\log_2(N)$ iterations and $N/2$ threads. Since timing is our main concern while we have sufficient threads on GPU, we employ parallel reduction technique to calculate an initial solution $\mathbf{z}_i^\ell$, $i \in \mathcal{K}_g$.

For the lower bounds in (12), we need to calculate $N|\mathcal{K}_g|^2$ times of multiplication of two $M \times 1$ complex vectors in the form of $(\mathbf{z}_k^\ell)^H \cdot \hat{\mathbf{h}}_i(n)$. So each thread will compute one such term and $N|\mathcal{K}_g|$ terms can be calculated at the same time, which is an advantage of GPU parallelization compared to sequential processing. Then we can easily calculate $d_i^\ell(n) - \sum_{k\in\mathcal{K}_g}^{k\neq i} d_k^\ell(n)$ based on (12).

To find λ^ℓ, we can directly sort N numbers and then check the sorted numbers to see whether we employ (15) to obtain $\mathbf{w}_i^\ell$ or drop this initial solution $\mathbf{z}_i$. Specifically, we need to perform $|\mathcal{K}_g|$ times of sorting of N real numbers. We employ a parallel sorting algorithm called odd-even sorting, which uses $\lfloor N/2 \rfloor$ threads and N iterations to sort N numbers. Then λ^ℓ can be easily found by comparing $|\mathcal{K}_g|$ real numbers or we declare it does not exist.

If λ^ℓ is found, we only need to multiply a real number λ^ℓ to $|\mathcal{K}_g|$ $M \times 1$ complex vectors $\mathbf{z}_i^\ell$, $i \in \mathcal{K}_g$. To reduce computation time, we use $2M|\mathcal{K}_g|$ threads,[2] where the first $|\mathcal{K}_g|M$ threads are for the real part of $\mathbf{w}_i^\ell$, $i \in \mathcal{K}_g$ and the remaining $|\mathcal{K}_g|M$ threads are for the imaginary part of $\mathbf{w}_i^\ell$, $i \in \mathcal{K}_g$. Then we can calculate the objective of this feasible solution using parallel reduction.

Since we use one thread block to generate an initial solution and perform the local search in ReDBeam, each thread block only takes $\mathbf{v}_i(n)$ as input and outputs a feasible solution $\mathbf{w}_i^\ell$, $i \in \mathcal{K}_g$ or declare infeasible of its initial solution. To reduce memory access time, all intermediate results in kernel 2 are stored in the shared memory, which includes $\mathbf{z}_i^\ell$, $d_i^\ell(n)$, $d_k^\ell(n)$, β_i^ℓ, and λ^ℓ. The output from L thread blocks has at most L feasible solutions $\mathbf{w}_i^\ell$, $i \in \mathcal{K}_g$, and their objective values, which will be stored to GPU global memory.

[2]For our problem, we typically have $N|\mathcal{K}_g|^2 > 2M|\mathcal{K}_g|$. So there are sufficient threads in each thread block.

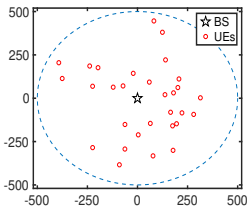


Fig. 6. Topology of a 5G cell with 30 UEs.

Kernel 3 The goal of kernel 3 is to find the best feasible solution for a P2 instance, which requires a comparison among at most L objective values (real numbers). Based on parallel reduction, we use 1 thread block with $\lceil L/2 \rceil$ threads to compare these L objective values. Since L is a large number and the objective values will be frequently accessed during comparisons, we first copy the objective values from GPU global memory to the shared memory before we perform parallel reduction. After comparison, the best feasible solution is identified and transferred to the CPU memory.

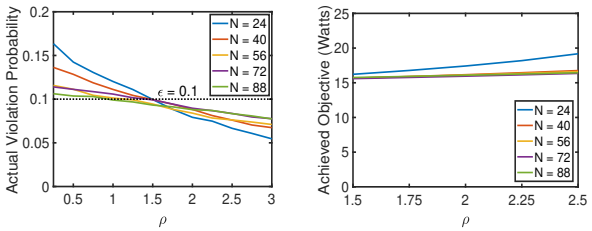
V. Experimental Results

In this section, we evaluate the performance of ReDBeam. We implement ReDBeam using CUDA 11.2 Toolkit on an NVIDIA Tesla V100, which has 5120 CUDA cores. We will focus on ReDBeam's running time and performance.

A. Parameter Settings

We consider a 5G cell with a 500-meter radius with a topology shown in Fig. 6. There are $K = 30$ UEs randomly distributed inside the cell. We assume the BS has 8 antennas (i.e., $M = 8$). Following Fig. 1, the BS has $G = 8$ RBGs and each RBG consists of 8 RBs. Under 5G numerology 0, the sub-carrier spacing is set to 15 kHz [28]. We set the number of UEs per RBG $|\mathcal{K}_g| = 2$. The BS has a power budget $P^{\max} = 46$ dBm for all 8 RBGs and the thermal noise σ_i^2 is set to -150 dBm/Hz for all UEs. For the required SINR threshold γ_i^{req}, we set it according to Shannon Theorem, $(500/s_i) = \log_2(1 + \gamma_i^{\text{req}})$, where 500 is the cell radius and s_i is the distance between UE i and the BS (both in meters) [38], which means $\gamma_i^{\text{req}} = 2^{(500/s_i)} - 1$. Further, we found that it is sufficient to set $L = 650$ (number of initial solutions) in ReDBeam for our setting.

For the wireless channel, we consider the path-loss model and fast fading. Note that the channel model described here is used only for generating parameters in our numerical studies. ReDBeam only relies on the CSI data samples and is blindfolded to any knowledge of distribution information. The path-loss between UE i and the BS is modeled by $PL_i = 38 + 30 \times \log_{10}(s_i)$ (in dB) [39]. For fast fading, we employ Rician fading with a 10 dB Rician factor [40], which is a common model for correlated RBs. In addition to the channel variation, we also need to include the CSI estimation errors during the collection of CSI data samples. Therefore, we employ a truncated Gaussian distribution to simulate the CSI



(a) Violation probability (b) Objective value

Fig. 7. Impact of ρ and N when $M = 8$, $|\mathcal{K}_g| = 2$.

estimation errors [13], [14]. Specifically, we use 0 as the mean and 0.1 as the variance for the original Gaussian distribution and then truncate it at three times its standard deviation.

For each setting below, we perform 50 runs and all results shown represent the average. Since $|\mathcal{K}_g| = 2$, in each run, we randomly pick 2 UEs from the 30 UEs for each RBG. Since we have 8 RBGs, there can be at most 16 active UEs at the same time. Given that one UE can be served by multiple RBGs, the number of active UEs may be lower than 16.

B. How to Set $\theta_{(g,i)}$ and N

We first discuss how to choose N and $\theta_{(g,i)}$ simultaneously in ReDBeam since these two parameters are coupled together. Intuitively, a larger N means more channel knowledge and will likely lead to better performance (at the cost of higher complexity). On the other hand, $\theta_{(g,i)}$ represents the difference between the true (but unknown) distribution and the N CSI data samples. Therefore, $\theta_{(g,i)}$ should be smaller when N increases. However, if $\theta_{(g,i)}$ is chosen too small, then the probabilistic data rate guarantees for the UEs in a solution may not hold. On the other hand, if $\theta_{(g,i)}$ is chosen too large, we will use more transmission powers at the BS than necessary [27]. Therefore, choosing an overly small value is more detrimental, so we can choose $\theta_{(g,i)}$ to be relatively large to be conservative (ensure the probabilistic performance guarantee holds).

We propose to choose $\theta_{(g,i)}$ based on fast heuristics for each window before executing ReDBeam. For each window and a UE $i \in \mathcal{K}_g$, $g \in \mathcal{G}$, we have N CSI data samples $\hat{\mathbf{h}}_{(g,i)}(n)$. Then we choose $\theta_{(g,i)}$ based on a constant factor and the estimated standard derivation from the N CSI data samples, i.e.,

$$\theta_{(g,i)} = \frac{\rho}{N} \cdot \sqrt{\frac{1}{N-1} \sum_{n=1}^{N} \left(||\hat{\mathbf{h}}_{(g,i)}(n) - \frac{\sum_{n=1}^{N} \hat{\mathbf{h}}_{(g,i)}(n)}{N}||_2^2 \right)} \quad (i \in \mathcal{K}_g,\ g \in \mathcal{G}) \ . \tag{16}$$

In (16), ρ is the constant factor we need to choose and the square root term is the unbiased estimation of standard derivation [41]. The use of ρ/N is because $\theta_{(g,i)}$ is related to the neighboring region of each CSI data sample. Once ρ is given, $\theta_{(g,i)}$ can be easily calculated based on the N CSI data samples in the current window.

To find a proper ρ and N, we set $\epsilon_i = 0.1$ for all $i \in \mathcal{K}$ and simulate ReDBeam under both $0.25 \leq \rho \leq 3$ and $24 \leq N \leq 88$. The actual violation probabilities and achieved objectives w.r.t. ρ and N are shown in Fig. 7. As shown in Fig. 7(a), it is sufficient to choose $\rho \in [1.5, 2.5]$ in all cases of N. Then we zoom into these ρ values and study the achieved objectives as shown in Fig. 7(b). As shown in Fig. 7(b), the objective only increases slightly w.r.t. ρ. Taking both Fig. 7(a) and Fig. 7(b) into account, we conclude that it is prudent to choose $N = 40$ and $\rho = 2$ to calculate $\theta_{(g,i)}$ in (16).

The above process shows how to set N and $\theta_{(g,i)}$ for different network settings and they can be dynamically adjusted during run-time through continuous tracking of the violation probabilities at the UEs. As for time consumption, in each window, before executing ReDBeam in Fig. 5, we will use $G|\mathcal{K}_g|$ thread blocks to calculate these $\theta_{(g,i)}$'s in parallel and then use these $\theta_{(g,i)}$'s in ReDBeam to derive a beamforming solution. This step should be counted toward the overall time consumption.

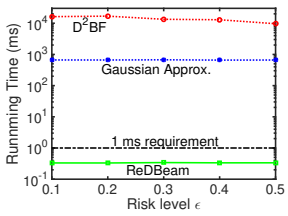
C. A Case Study

Now we evaluate ReDBeam and compare its performance to other algorithms. We set $M = 8$, $|\mathcal{K}_g| = 2$, $L = 650$, $N = 40$, and $\rho = 2$ based on Section V-B. For comparison, we consider two other benchmarks. The first one is D^{2}BF [27], which is the state-of-the-art solution, which solves each P2 through convex approximation and Semidefinite Programming (SDP). The second one is Gaussian Approximation [14], which assumes CSI follows a complex Gaussian distribution. Both D^{2}BF and Gaussian Approximation are CPU-based solutions. We implement them with commercial solver MOSEK 9.2.38 using MATLAB R2017b on Intel Xeon E5-2687w v4.

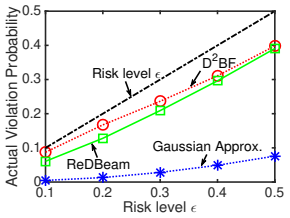
Figure 8 shows the running time, actual threshold violation probabilities, and the achieved objectives for ReDBeam, D^{2}BF, and Gaussian Approximation. As shown in Fig. 8(a), the time consumption of ReDBeam is under the 1 ms timing requirement under all risk levels. Further, we see the running time of ReDBeam is rather independent of ϵ because the running time depends on the number of steps for each thread, which is independent of ϵ. On the other hand, none of the other two solutions (D^{2}BF and Gaussian Approximation) can meet the 1 ms timing requirement. Specifically, Gaussian Approximation requires $\sim 10^2$ ms while D^{2}BF requires $\sim 10^4$ ms.

Figure 8(b) shows the threshold violation probabilities of ReDBeam are lower than the target risk level ϵ. D^{2}BF has a slightly better performance than ReDBeam since it is the state-of-the-art solution to P2. Gaussian Approximation has the lowest threshold violation probabilities due to its conservativeness.

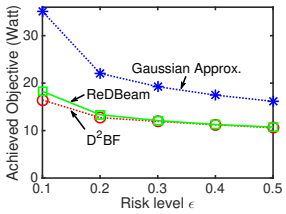
Figure 8(c) shows that the objective value achieved by ReDBeam is very close to (slightly higher than) that of D^{2}BF, from 1.4% (when $\epsilon = 0.5$) to 11% (when $\epsilon = 0.1$). This demonstrates the superb performance of ReDBeam. In Figure 8(c), Gaussian Approximation offers the worst performance (as it uses the most transmission power), which is consistent with its conservativeness demonstrated in Fig. 8(b). In general, the closer the actual violation probabilities to



(a) Running Time



(b) Violation probability



(c) Objective value

Fig. 8. Performance of ReDBeam.

the risk level ϵ (in Fig. 8(b)), the less power is needed (in Fig. 8(c)).

We also conduct experiments with varying M and $|\mathcal{K}_g|$ and found that our ReDBeam can meet the 1 ms real-time requirement up to a network setting of $M = 18$, $|\mathcal{K}_g| = 4$, and $G = 12$ (i.e., serving up-to 48 UEs simultaneously), which is sufficient for real-world scenarios. All other observations are consistent with the above discussion.

VI. Conclusions

We investigated a novel data-driven MU-MIMO beamforming approach that only uses limited CSI data samples. We focused on the fundamental technical challenge facing this approach—whether or not it could be done in real-time (i.e., 1 ms for 5G). We presented ReDBeam, a real-time MU-MIMO beamforming solution that offers performance guarantees (in terms of probabilistic data rate requirements) and minimizes power consumption. The key idea of ReDBeam is to employ GPU's massive parallel computing capability (both algorithm design and GPU implementation) to solve the beamforming problem on each RBG in parallel and combine them as the final solution. For each RBG, ReDBeam first generates a population of initial solutions from a promising subspace derived from ZF precoding; then it employs local search to ensure feasibility and improve objective value; and last, it finds the one with the best objective value as the final solution. For GPU implementation, we optimized threads allocations and memory management to minimize the total time consumption. Experiment results showed that ReDBeam can deliver an MU-MIMO beamforming solution within 1 ms while minimizing BS's power consumption and meeting the UEs' probabilistic data rate requirements.

References

[1] J. Laneman, D. Tse, and G. Wornell, "Cooperative diversity in wireless networks: Efficient protocols and outage behavior," *IEEE Trans. Information Theory*, vol. 50, no. 12, pp. 3062–3080, Dec. 2004.

[2] E. Castañeda, A. Silva, A. Gameiro, and M. Kountouris, "An overview on resource allocation techniques for multi-user MIMO systems," *IEEE Commun. Surveys & Tutorials*, vol. 19, no. 1, pp. 239–284, First Quarter 2017.

[3] M. Shafi, A. F. Molisch, P. J. Smith, T. Haustein, P. Zhu, P. De Silva, F. Tufvesson, A. Benjebbour, and G. Wunder, "5G: A tutorial overview of standards, trials, challenges, deployment, and practice," *IEEE J. Selected Areas in Commun.*, vol. 35, no. 6, pp. 1201–1221, June 2017.

[4] I. Ahmed, H. Khammari, A. Shahid, A. Musa, K. S. Kim, E. De Poorter, and I. Moerman, "A survey on hybrid beamforming techniques in 5G: Architecture and system model perspectives," *IEEE Commun. Surveys & Tutorials*, vol. 20, no. 4, pp. 3060–3097, Fourth Quarter 2018.

[5] M. Schubert and H. Boche, "Solution of the multiuser downlink beamforming problem with individual SINR constraints," *IEEE Trans. Vehicular Technology*, vol. 53, no. 1, pp. 18–28, Jan. 2004.

[6] G. Scutari, D. P. Palomar, and S. Barbarossa, "The MIMO iterative waterfilling algorithm," *IEEE Trans. Signal Processing*, vol. 57, no. 5, pp. 1917–1935, May 2009.

[7] K. Zheng, L. Zhao, J. Mei, B. Shao, W. Xiang, and L. Hanzo, "Survey of large-scale MIMO systems," *IEEE Commun. Surveys & Tutorials*, vol. 17, no. 3, pp. 1738–1760, Third Quarter 2015.

[8] Y. Wu, R. H. Louie, and M. R. McKay, "Analysis and design of wireless ad hoc networks with channel estimation errors," *IEEE Trans. Signal Processing*, vol. 61, no. 6, pp. 1447–1459, Mar. 2013.

[9] Y. Liu, Z. Tan, H. Hu, L. J. Cimini, and G. Y. Li, "Channel estimation for OFDM," *IEEE Commun. Surveys & Tutorials*, vol. 16, no. 4, pp. 1891–1908, Fourth Quarter 2014.

[10] X. Rao and V. K. Lau, "Distributed compressive CSIT estimation and feedback for FDD multi-user massive MIMO systems," *IEEE Trans. Signal Processing*, vol. 62, no. 12, pp. 3261–3271, June 2014.

[11] Z. Jiang, A. F. Molisch, G. Caire, and Z. Niu, "Achievable rates of FDD massive MIMO systems with spatial channel correlation," *IEEE Trans. Wireless Commun.*, vol. 14, no. 5, pp. 2868–2882, May 2015.

[12] X. Jiang and F. Kaltenberger, "Channel reciprocity calibration in TDD hybrid beamforming massive MIMO systems," *IEEE Journal of Selected Topics in Signal Processing*, vol. 12, no. 3, pp. 422–431, June 2018.

[13] D. Mi, M. Dianati, L. Zhang, S. Muhaidat, and R. Tafazolli, "Massive MIMO performance with imperfect channel reciprocity and channel estimation error," *IEEE Trans. Commun.*, vol. 65, no. 9, pp. 3734–3749, Sept. 2017.

[14] K.-Y. Wang, A. M.-C. So, T.-H. Chang, W.-K. Ma, and C.-Y. Chi, "Outage constrained robust transmit optimization for multiuser MISO downlinks: Tractable approximations by conic optimization," *IEEE Trans. Signal Processing*, vol. 62, no. 21, pp. 5690–5705, Nov. 2014.

[15] M. B. Shenouda, T. N. Davidson, and L. Lampe, "Outage-based design of robust Tomlinson–Harashima transceivers for the MISO downlink with QoS requirements," *Elsevier Signal Processing*, vol. 93, no. 12, pp. 3341–3352, Dec. 2013.

[16] Y. Shi, J. Zhang, and K. B. Letaief, "Optimal stochastic coordinated beamforming for wireless cooperative networks with CSI uncertainty," *IEEE Trans. Signal Processing*, vol. 63, no. 4, pp. 960–973, Feb. 2014.

[17] C. Pan, H. Ren, M. Elkashlan, A. Nallanathan, and L. Hanzo, "Robust beamforming design for ultra-dense user-centric C-RAN in the face of realistic pilot contamination and limited feedback," *IEEE Trans. Wireless Commun.*, vol. 18, no. 2, pp. 780–795, Feb. 2019.

[18] J. Xu and R. Zhang, "A general design framework for mimo wireless energy transfer with limited feedback," *IEEE Trans. Signal Processing*, vol. 64, no. 10, pp. 2475–2488, May 2016.

[19] H. Liu, X. Yuan, and Y. J. Zhang, "Statistical beamforming for FDD downlink massive MIMO via spatial information extraction and beam selection," *IEEE Trans. Wireless Commun.*, vol. 19, no. 7, pp. 4617–4631, July 2020.

[20] X. Li, S. Jin, X. Gao, and R. W. Heath, "Three-dimensional beamforming for large-scale FD-MIMO systems exploiting statistical channel state information," *IEEE Trans. Vehicular Technology*, vol. 65, no. 11, pp. 8992–9005, Nov. 2016.

[21] B. K. Chalise, S. Shahbazpanahi, A. Czylwik, and A. B. Gershman, "Robust downlink beamforming based on outage probability specifications," *IEEE Trans. Wireless Commun.*, vol. 6, no. 10, pp. 3498–3503, Oct. 2007.

[22] E. Song, Q. Shi, M. Sanjabi, R.-Y. Sun, and Z.-Q. Luo, "Robust SINR-constrained MISO downlink beamforming: When is semidefinite programming relaxation tight?" *EURASIP J. Wireless Commun. and Networking*, vol. 2012, no. 1, pp. 1–11, Aug. 2012.

[23] M. B. Shenouda and T. N. Davidson, "Nonlinear and linear broadcasting with QoS requirements: Tractable approaches for bounded channel uncertainties," *IEEE Trans. Signal Processing*, vol. 57, no. 5, pp. 1936–1947, May 2009.

[24] J. Zhang, M. You, G. Zheng, I. Krikidis, and L. Zhao, "Model-driven learning for generic MIMO downlink beamforming with uplink channel information," *IEEE Trans. Wireless Commun.*, vol. 21, no. 4, pp. 2368–2382, Apr. 2022.

[25] M. Alrabeiah, Y. Zhang, and A. Alkhateeb, "Neural networks based beam codebooks: Learning mmwave massive mimo beams that adapt to deployment and hardware," *IEEE Trans. Commun.*, vol. 70, no. 6, pp. 3818–3833, June 2022.

[26] H. Zhu, Q. Wu, X.-J. Wu, Q. Fan, P. Fan, and J. Wang, "Decentralized power allocation for MIMO-NOMA vehicular edge computing based on deep reinforcement learning," *IEEE Internet of Things Journal*, vol. 9, no. 14, pp. 12 770–12 782, June 2022.

[27] S. Li, N. Jiang, Y. Chen, Y. T. Hou, W. Lou, and W. Xie, "D^2BF—Data-driven beamforming in MU-MIMO with channel estimation uncertainty," in *Proc. IEEE INFOCOM 2022*, May 2022, pp. 120–129, Virtual Conference, May 2–5, 2022.

[28] 3GPP, *TS 38.211: 5G; NR; Physical channels and modulation*, Jan. 2023, version 17.4.0. Available: https://portal.3gpp.org/desktopmodules/Specifications/SpecificationDetails.aspx?specificationId=3213.

[29] ——, *TS 38.214: 5G; NR; Physical layer procedures for data*, Jan. 2023, version 17.4.0. Available: https://portal.3gpp.org/desktopmodules/Specifications/SpecificationDetails.aspx?specificationId=3216.

[30] H. Yang and T. L. Marzetta, "Performance of conjugate and zero-forcing beamforming in large-scale antenna systems," *IEEE J. Selected Areas in Commun.*, vol. 31, no. 2, pp. 172–179, Feb. 2013.

[31] W. Yang, G. Durisi, and E. Riegler, "On the capacity of large-MIMO block-fading channels," *IEEE J. Selected Areas in Commun.*, vol. 31, no. 2, pp. 117–132, Feb. 2013.

[32] W. Xie, "On distributionally robust chance constrained programs with Wasserstein distance," *Mathematical Programming*, vol. 186, no. 1, pp. 115–155, Mar. 2021.

[33] N. Jiang and W. Xie, "ALSO-X and ALSO-X+: Better convex approximations for chance constrained programs," *Operations Research*, vol. 70, no. 6, pp. 3581–3600, Feb. 2022.

[34] R. C.-H. Chang, C.-H. Lin, K.-H. Lin, C.-L. Huang, and F.-C. Chen, "Iterative QR decomposition architecture using the modified Gram–Schmidt algorithm for MIMO systems," *IEEE Trans. Circuits and Systems I: Regular Papers*, vol. 57, no. 5, pp. 1095–1102, May 2010.

[35] C. Chen, A. Atamtürk, and S. S. Oren, "A spatial branch-and-cut method for nonconvex QCQP with bounded complex variables," *Mathematical Programming*, vol. 165, no. 2, pp. 549–577, July 2017.

[36] M. Harris. (2012, Dec.) How to overlap data transfers in CUDA C/C++. Available: https://devblogs.nvidia.com/how-overlap-data-transfers-cuda-cc/ (Last accessed: Mar. 2023).

[37] ——. (2007) Optimizing parallel reduction in CUDA. Available: https://developer.download.nvidia.com/assets/cuda/files/reduction.pdf (Last accessed: Mar. 2023).

[38] J. Wang, A. Jin, D. Shi, L. Wang, H. Shen, D. Wu, L. Hu, L. Gu, L. Lu, Y. Chen, J. Wang, Y. Saito, A. Benjebbour, and Y. Kishiyama, "Spectral efficiency improvement with 5G technologies: Results from field tests," *IEEE J. Selected Areas in Commun.*, vol. 35, no. 8, pp. 1867–1875, Aug. 2017.

[39] 3GPP, *TR 36.931: Radio Frequency (RF) requirements for LTE Pico Node B*, Apr. 2022, version 17.0.0. Available: https://portal.3gpp.org/desktopmodules/Specifications/SpecificationDetails.aspx?specificationId=2589.

[40] X. Li, S. Jin, H. A. Suraweera, J. Hou, and X. Gao, "Statistical 3-D beamforming for large-scale MIMO downlink systems over Rician fading channels," *IEEE Trans. Commun.*, vol. 64, no. 4, pp. 1529–1543, Apr. 2016.

[41] D. C. Montgomery and G. C. Runger, *Applied Statistics and Probability for Engineers*. John Wiley & Sons, 2010, Chapter 7, pp. 224–226.