Dynamic Nested Clustering for Parallel PHY-Layer Processing in Cloud-RANs

Congmin Fan, Ying Jun (Angela) Zhang, and Xiaojun Yuan

Abstract

Featured by centralized processing and cloud based infrastructure, Cloud Radio Access Network (C-RAN) is a promising solution to achieve an unprecedented system capacity in future wireless cellular networks. The huge capacity gain mainly comes from the centralized and coordinated signal processing at the cloud server. However, full-scale coordination in a large-scale C-RAN requires the processing of very large channel matrices, leading to high computational complexity and channel estimation overhead. To resolve this challenge, we exploit the near-sparsity of large C-RAN channel matrices, and derive a unified theoretical framework for clustering and parallel processing. Based on the framework, we propose a dynamic nested clustering (DNC) algorithm for uplink signal detection. This algorithm allows flexible trade-offs between system performance and other critical parameters, such as computational complexity and channel estimation overhead. Moreover, the algorithm is amenable to parallel processing, and various parallel implementations are discussed for different types of data center architectures. With the proposed algorithm, we show that the computation time for the optimal linear detector can be reduced from $O(N^3)$ to no higher than $O(N^{3/2})$, where $N$ is the number of RRHs in C-RAN.

Index Terms

Cloud-RAN; dynamic clustering; parallel processing
I. INTRODUCTION

The explosive growth in mobile data traffic threatens to outpace the infrastructure it relies on. To sustain the mobile data explosion with low bit-cost and high spectrum/energy efficiency, a revolutionary wireless cellular architecture, termed Cloud Radio Access Network (C-RAN), emerges as a promising solution [1]. In contrast to traditional base stations, the radio function units and baseband processing units (BBUs) in C-RANs are separated, and the latter are migrated to a centralized data center using an optical transport network with high bandwidth and low latency. This keeps the radio function units (also referred to as remote radio heads (RRHs)) light-weight, thereby allowing them to be deployed in a large number of small cells with low costs. Meanwhile, the high-bandwidth low-latency transport links and centralized processing allow RRHs to seamlessly cooperate with each other for flexible interference management, coordinated signal processing, etc. In this way, C-RAN opens up possibilities for significant system-capacity enhancement and cost reduction.

The exciting opportunities come hand in hand with new technical challenges. Existing studies on multiuser MIMO systems suggest that the highest system capacity is achieved when all RRHs cooperatively form a large-scale virtual antenna array that jointly detects the users’ signals. The full-scale cooperation, however, requires the processing of a very large channel matrix consisting of channel coefficients from all mobile users to all RRHs. For example, when the optimal linear receiver is adopted, the computational complexity typically grows cubically with the matrix size [2]. This implies that the average computational complexity per RRH or per mobile user grows quadratically with the network size, which fundamentally limits the scale of RRH cooperation. In addition, the full-scale joint RRH processing requires to estimate a large-scale channel matrix, causing significant channel estimation overhead. In [3], it is shown that the benefit of cooperation is fundamentally limited by the overhead of pilot-assisted channel estimation. As such, it is of critical importance to find signal processing algorithms, where the computational complexity and channel estimation overhead grows gracefully with the size of the network.

A. Contributions

The above-mentioned challenges lie in the estimation and processing of a very large channel matrix. One potential solution is to decompose the overall channel matrix into small submatrices. Then, each
submatrix can be estimated and processed separately. Indeed, this is the true essence of traditional clustering algorithms. However, decomposition of the channel matrix would inevitably cause performance loss. In this paper, we propose a unified theoretical framework for dynamic clustering, where trade-offs between system performance and other key parameters, such as computational complexity and channel estimation, are established.

In C-RAN, only a small fraction of the entries in the channel matrix have reasonably large gains, for that a user is only close to a small number of RRHs in its neighborhood, and vice versa. Thus, ignoring the small entries in the channel matrix can significantly sparsify the matrix, which potentially leads to a great reduction in processing complexity and channel estimation overhead. The key questions are (i) to what extent the channel matrix can be sparsified without substantially compromising the system performance, and (ii) how to reduce the computational complexity of signal processing based on the sparsified channel matrix.

In this paper, we endeavour to address the above two questions, based on which a dynamic nested clustering (DNC) algorithm for RRH clustering and parallel processing is derived. More specifically, to address the first question, we propose a threshold-based channel matrix sparsification method, where the channel entries are ignored if the corresponding link length is beyond a certain threshold. Through rigorous analysis, we derive a closed-form expression describing the relation between the threshold and the signal-to-interference-plus-noise ratio (SINR) loss due to matrix sparsification. The result shows that a vast majority of the channel coefficients can be ignored if we can tolerate a very small percentage of SINR loss. To address the second question, we show that with the sparsified channel matrix, the uplink signal detection is equivalent to a system of linear equations defined by a (nested) doubly bordered block diagonal (DBBD) matrix, if the RRHs are appropriately indexed. Interestingly, the following important insights for clustering and parallel signal processing can be derived from the structure of the DBBD matrix:

1) Each block on the main diagonal, except the last one, can be interpreted as a cluster of the network. Specifically, the RRHs belonging to one cluster are jointly processed, while different clusters can be processed in parallel.
2) The last block on the main diagonal, together with the borders of the DBBD matrix, captures the interaction among different clusters due to interference. A message passing algorithm can be correspondingly derived to facilitate cooperation among clusters.

3) A simple RRH labeling scheme enables us to easily adjust the size and number of the blocks, as well as the width of the border. This further allows us to flexibly balance the computation time, the total number of clusters, the number of parallel BBUs required, the allocation of computational power among BBUs, etc.

With the above insights, our work here serves as a unified theoretical framework for dynamic clustering of C-RAN networks.

B. Related Work

Existing solutions to reduce channel estimation overhead and computational complexity in distributed antenna or C-RAN systems include antenna selection [4], [5] and clustering algorithms [6]–[11]. By controlling the number of serving antennas or the cluster size, these two methods can limit the estimation overhead and computational complexity to a low level. However, such approaches inevitably reduce the system capacity. This is because by limiting the scale of cooperation to within a small antenna cluster, the centralized processing power of C-RAN is not fully exploited. To deal with this issue, inter-cluster coordination [7], [8] and overlapping-cluster algorithms [9], [10] have been proposed. For instance, [7] proposed a hierarchical interference mitigation scheme, in which the precoder is partitioned into an inner precoder and an outer precoder. The two precoders control the intra-cell and inter-cell interference respectively, with intra-cell channel station information (CSI) and inter-cell channel statistics required. In this way, the system capacity can be improved by mitigating the inter-cell/inter-cluster interference. However, most of the inter-cluster coordination and overlapping cluster algorithms only consider networks with small sizes due to the high computational complexity. In addition, perfect CSI is required to do clustering [9], [10], which is impractical for C-RAN. A recent work by Shi et al. [12] proposed a framework consisting of a compressive CSI acquisition and stochastically coordinated downlink beamforming. Similar to our work here, the channel matrix estimated in [12] is sparsified. However, [12] does not make use of the sparsity of the channel matrix to reduce the complexity of the
beamforming algorithm. Moreover, no theoretical analysis is provided to quantify the performance loss due to channel sparsification. According to their simulations, to guarantee a low performance loss, a very high percentage of CSI was required, say 60%, which means that the computational complexity and the channel estimation overhead can still be very high when the network size is large. In contrast, in our work, the amount of CSI needed per user is constant, implying the overhead per user does not scale with the network size.

The rest of this paper is organized as follows: in Section II, we describe the system model and outline the steps of the DNC algorithm. The first step in the DNC algorithm, threshold-based channel sparsification, is proposed and analysed in Section III. In Section IV, a single-layer DNC algorithm is proposed, and the detailed parallel implementation of this algorithm is discussed. The multi-layer DNC algorithm is introduced in Section V. Conclusions and discussions are given in Section VI.

II. System Model

A. System Setup

We consider the uplink transmission of a C-RAN with $N$ single-antenna RRHs and $K$ single-antenna mobile users uniformly located over the entire coverage area. The received signal vector $y \in \mathbb{C}^{N \times 1}$ at the RRHs is

$$y = HP^{\frac{1}{2}}x + n,$$

where $H \in \mathbb{C}^{N \times K}$ denotes the channel matrix, with the $(n, k)$th entry $H_{n,k}$ being the channel coefficient between the $k$th user and the $n$th RRH; $P \in \mathbb{R}^{N \times N}$ is a diagonal matrix with the $k$th diagonal entry $P_k$ being the transmitting power allocated to user $k$; $x \in \mathbb{C}^{K \times 1}$ is a vector of the transmitted signal from the $K$ users and $n \sim \mathcal{CN}(0, N_0 I)$ is a vector of noise received by RRHs. The transmit signals are assumed to follow an independent complex Gaussian distribution with unit variance, i.e. $E[xx^H] = I$. Further, the $(n, k)$th entry of $H$ is given by $H_{n,k} = \gamma_{n,k}d_{n,k}^{-\frac{\alpha}{2}}$, where $\gamma_{n,k}$ is the i.i.d Rayleigh fading coefficient with zero mean and variance 1, $d_{n,k}$ is the distance between the $n$th RRH and the $k$th user, and $\alpha$ is the path loss exponent. Then, $d_{n,k}^{-\alpha}$ is the path loss from the $k$th user to the $n$th RRH.
B. MMSE Detection for C-RAN

With centralized baseband processing, a C-RAN system can jointly detect all users’ signals through a full-scale RRH cooperation. Suppose that the optimal linear detection, i.e., MMSE detection, is employed. The receive beamforming matrix is

$$V = A^{-1} HP^\frac{1}{2},$$  

(2)

where $A = HPH^H + N_0 I$.

The decision statistics of the transmitted signal vector $x$ is a linear function of the received signal vector $y$, i.e.

$$\hat{x} = V^H y = V^H HP^\frac{1}{2} x + V^H n.$$  

(3)

Then, the decision statistics of $x_k$ for user $k$ is

$$\hat{x_k} = v_k^H h_k P_k^\frac{1}{2} x_k + v_k^H \sum_{j \neq k} h_j P_j^\frac{1}{2} x_j + v_k^H n,$$  

(4)

where $v_k \in \mathbb{C}^{N \times 1}$ is the $k$th column of the detection matrix $V$ and $h_k \in \mathbb{C}^{N \times 1}$ is the $k$th column of the channel matrix $H$. The SINR of user $k$ is

$$\text{SINR}_k = \frac{P_k |v_k^H h_k|^2}{\sum_{j \neq k} P_j |v_k^H h_j|^2 + N_0 v_k^H v_k}.$$  

(5)

Notice that to calculate the detection matrix $V$, the full channel matrix $H$ needs to be acquired and processed. That is, full channel state information (CSI) needs to be estimated at the RRHs’ side. In addition, it takes as much as $O(N^3)$ operations to calculate $V$, because the calculation involves the inverse of an $N \times N$ matrix $A$. As mentioned in Section I, a C-RAN generally covers a large area with a huge number of RRHs. For example, the preliminary C-RAN technology can already support around 10km separation between the BBU pool and RRHs, covering up to 1000 RRH sites [1]. As a result, the dimension of the channel matrix $H$ is extremely large, and the cost of acquiring and processing $H$ becomes prohibitively high. The key question is then how to obtain the best system performance by enabling a full-scale RRH cooperation without incurring high channel estimation overhead and computational complexity.
C. Sketch of the Proposed Approach

As described in the Introduction section, the work in this paper consists of the following steps:

1) Channel sparsification based on a link-distance threshold.

2) Transforming the MMSE detection into a system of linear equations defined by a (nested) DBBD matrix.

3) A parallel detection algorithm based on dynamic nested clustering.

We first introduce the channel-matrix sparsification approach in Section III. The transformation of the MMSE detection and parallel detection algorithms are discussed in both Section IV and V.

III. Threshold-based Channel Sparsification

In this section, we discuss the first step in the DNC algorithm, i.e., threshold-based channel sparsification. We first present the detailed sparsification approach in Subsection A. Then, a closed-form expression of the matrix sparsity as a function of the tolerable SINR loss is derived in Subsection B. Finally, verifications and discussions are given in Subsection C.

A. Sparsification Approach

Since the RRHs and users are distributed over a large area, an RRH can only receive reasonably strong signals from a small number of nearby users, and vice versa. Thus, ignoring the small entries in $H$ would significantly sparsify the matrix, hopefully with a negligible loss in system performance.

In this paper, we propose to ignore the entries of $H$ based on the distance between RRHs and users. In other words, the entry $H_{n,k}$ is set to 0 when the link distance $d_{n,k}$ is larger than a threshold $d_0$. The resulting sparsified channel matrix, denoted by $\hat{H}$, is given by

$$
\hat{H}_{n,k} = \begin{cases} 
H_{n,k}, & d_{n,k} < d_0 \\
0, & \text{otherwise}
\end{cases}
$$

(6)

Note that we propose to sparsify the channel matrix based on the link distance instead of the actual channel coefficients that are affected by both the link distances and fast channel fadings. In practice, link distances vary much more slowly than fast channel fading. With the distance-based channel sparsification, channel estimation is needed only on short links. Otherwise, if we sparsify $H$ based on the absolute values of the entries, then channel fading needs to be estimated on every link.
The received signal \( y \) can now be represented as
\[
y = \tilde{H}P^2 x + \tilde{H}P^2 x + n,
\]
where \( \tilde{H} = H - \hat{H} \) consists of the entries that have been ignored. Treating the first term in the RHS of (7) as signal, and the remaining two terms as interference plus noise, the detection matrix becomes
\[
\hat{V} = \hat{A}^{-1}\hat{H}P^2,
\]
where \( \hat{A} = \hat{H}P\hat{H}^H + \Gamma + N_0I \), and
\[
\Gamma = E \left[ \sum_{j=1}^{K} P_j (\hat{h}_j\hat{h}_j^H + \hat{h}_j\tilde{h}_j^H + \tilde{h}_j\tilde{h}_j^H) \right],
\]
with \( \hat{h}_k \) and \( \tilde{h}_k \) being the \( k \)th column of \( \hat{H} \) and \( \tilde{H} \) respectively.

With this, the SINR becomes
\[
\text{SINR}_k(d_0) = \frac{P_k|\hat{v}_k^Hh_k|^2}{\sum_{j \neq k} P_j|\hat{v}_k^Hh_j|^2 + N_0|\hat{v}_k^H\tilde{v}_k|},
\]
where \( \hat{v}_k \) is the \( k \)th column of \( \hat{V} \).

Notice that when the distance threshold \( d_0 \) is small, the matrix \( \tilde{H} \) can be very sparse, leading to a significant reduction in channel estimation overhead and processing complexity. The key question is: how small \( d_0 \) can be or how sparse the channel matrix can be without significantly affecting the system performance. In other words, how should we set \( d_0 \) so that \( \text{SINR}_k \) is not much lower than \( \text{SINR}_k \) in (5). This question will be answered in the next subsection.

### B. Distance Threshold Analysis

In this subsection, we show how to set the distance threshold \( d_0 \) if a high percentage of full SINR is to be achieved. Specifically, we wish to set \( d_0 \), such that the SINR ratio, defined as
\[
\rho(d_0) = \frac{E[\text{SINR}_k(d_0)]}{E[\text{SINR}_k]}
\]
is larger than a prescribed level \( \rho^* \), where the expectation \( E(\cdot) \) is taken over \( H \), with randomness including both path loss and Rayleigh fading.

In the following, we endeavor to derive a closed-form expression of \( d_0 \) as a function of the target SINR ratio \( \rho^* \). Let us first introduce two assumptions that make our analysis tractable.

**Assumption 1:** The distances \( d_{n,k} \), for all \( n, k \) are mutually independent.
As shown in Fig. 1, we plot the SINR ratio for systems with and without Assumption 1. The system area is assumed to be a circle with radius 2.5 km. The figure shows that the gap between the SINR ratio is very small, which validates the independence assumption.

**Assumption 2:** Conditioning on the distance threshold $d_0$, the matrices $\hat{H}$ and $\tilde{H}$ are mutually independent.

Note that $E[\hat{H}\tilde{H}^H] = E[\tilde{H}\hat{H}^H] = 0$, which means that $\hat{H}$ and $\tilde{H}$ are uncorrelated. With the independence assumption, the equality

$$E_{\hat{H}}[\text{SINR}_k(d_0)] = E_{\tilde{H}}[E_{\tilde{H}}[\text{SINR}_k(d_0)]]$$

(12)

holds. This assumption will be verified in our numerical results in Fig. 2, which shows that the gap between the simulated SINR ratio and the lower bound of $\rho(d_0)$ derived based on this assumption is small.

![Figure 1](image)

Fig. 1. Average SINR ratio vs distance threshold when $N = 1000$, $K = 600$.

Based on these two assumptions, we see that $\Gamma = N_1 I$, where $N_1 = E[\sum_{j=1}^{K} P_j |\tilde{h}_{n,j}|^2]$ for arbitrary RRH $n$. We can now derive a lower bound of $\rho(d_0)$ as shown in the following Theorem 1.

**Theorem 1.** Given a distance threshold $d_0$, a lower bound of SINR ratio $\rho(d_0)$ is given by

$$\rho(d_0) \geq \rho(d_0) \triangleq \frac{\tilde{\mu}N_0}{\mu \left( (\mu - \tilde{\mu}) \sum_{j \neq k} P_k + N_0 \right)},$$

(13)

where $\tilde{\mu} = \int_{x=0}^{d_0} x^{-\alpha} f(x)dx$ and $\mu = \int_{x=0}^{\infty} x^{-\alpha} f(x)dx$ respectively, and $f(x)$ is the probability density function of the distance between RRHs and users.
When each user transmits at the same amount of power $P$, the lower bound is simplified as

$$
\rho(d_0) = \frac{\hat{\mu} N_0}{\mu (P(\mu - \hat{\mu})(K - 1) + N_0)}.
$$

(14)

**Proof:** See Appendix.

We notice that $\rho(d_0)$ depends on the probability distribution of the distances between mobile users and RRHs. In [13], distance distributions are derived for different network area shapes, such as circle, square and rectangle. Take, for example, a circular network area with radius $r$. In this case, the distance distribution between two random points is [13]

$$
f(x, r) = \begin{cases} 
\int_0^{\alpha} \frac{2\pi}{2\pi} \left( \frac{2}{\pi} \arccos \left( \frac{y}{2\pi} \right) - \frac{y}{x} \sqrt{1 - \frac{y^2}{4\pi^2}} \right) dy, & x = r_0, \\
\frac{2\pi}{2\pi} \left( \frac{2}{\pi} \arccos \left( \frac{x}{2\pi} \right) - \frac{x}{x} \sqrt{1 - \frac{x^2}{4\pi^2}} \right), & r_0 < x < 2r,
\end{cases}
$$

(15)

where $r_0$ is the minimum distance between RRHs and users.

When the network radius $r$ becomes very large, (15) can be approximated as

$$
f(x, r) = \begin{cases} 
\frac{r^2}{2\pi}, & x = r_0, \\
\frac{2\pi}{x}, & r_0 < x < r.
\end{cases}
$$

(16)

Substituting (15) or (16) into (14), we obtain the relation between $d_0$ and the SINR requirement $\rho^*$:

**Theorem 2.** When $d_0$ is the solution of

$$
N_0 \int_{x=0}^{d_0} x^{-\alpha} f(x, r) dx = \rho^* \left( P(K - 1) \int_{x=d_0}^{\infty} x^{-\alpha} f(x, r) dx \right) \int_{x=0}^{\infty} x^{-\alpha} f(x, r) dx,
$$

(17)

where $f(x, r)$ is given in (15), an SINR ratio no smaller than $\rho^*$ can be achieved.

When the network size is very large (i.e., $r \gg r_0$), the solution to (17) can be approximated as

$$
d_0 = \left( r^{2-\alpha} + \frac{(\alpha r_0^{2-\alpha} - 2r^{2-\alpha})(1 - \rho^*) N_0}{2N_0 + \frac{2\rho^* (\alpha r_0^{2-\alpha} - 2r^{2-\alpha})(K - 1) P}{(\alpha - 2)r^2}} \right)^{-\frac{1}{2-\alpha}}.
$$

(18)

Particularly, when the network size goes to infinity (i.e., $r \to \infty$), $d_0$ can be further simplified as

$$
d_0 = \left( \frac{2N_0(\alpha - 2) + 2\alpha r_0^{2-\alpha} \rho^* \beta K P}{\alpha r_0^{2-\alpha} N_0(1 - \rho^*)(\alpha - 2)} \right)^{\frac{1}{2-\alpha}}.
$$

(19)

**C. Verifications and Discussions**

In this subsection, we first verify our analysis through numerical simulations. We then illustrate the effect of SINR ratio requirement on the choice of the distance threshold. Finally, we illustrate the
sparsity of the channel matrix and discuss the possibility of reducing estimation overhead based on the
sparsified matrix. Unless stated otherwise, we assume that the minimum distance between RRHs and
users is 1 meter, the path loss exponent is 3.7, and the average transmit SNR at the user side equals to
80dB. That is $\frac{P}{N_0} = 80$dB.

![Fig. 2. Average SINR ratio vs distance threshold when $K = 1000$, $r = 5$km.](image)

![Fig. 3. Distance threshold $d_0$ vs area radius $r$ when the user density $\beta_K = 8$/km$^2$.](image)

1) Verification of Theorem 1 and 2: Fig. 2 plots the SINR ratio against the distance threshold, when
$K = 1000$ and $r = 5$ km. The simulated SINR ratio with different numbers of RRHs, $N$, are plotted as
the blue curves and $\rho(d_0)$ derived based on the distributions in (15) and (16) are plotted as the red and
black curves, respectively. We can see that the gap between the lower bound based on (15) and that
based on (16) is negligible, which means that the approximation of distance distribution is reasonable.
Moreover, we notice that even though the simulated ratios vary with $N$, the lower bounds derived based
on Theorem 1 and (15), (16) remain unchanged for different $N$. 
In Fig. 3 we show that the distance threshold converges to a constant when the network radius $r$ becomes large, as predicted in (19). Here, the user density is $\beta_K = 8/\text{km}^2$, and the SINR ratio requirement is set to $\rho^* = 0.95$ and $\rho^* = 0.9$, respectively. As expected, the distance threshold converges quickly to a constant when the network radius increases. Indeed, the convergence is observed even when the network radius is as small as 5 km for both $\rho^* = 0.9$ and $\rho^* = 0.95$.

![Graph showing distance threshold vs SINR ratio](image)

Fig. 4. Distance threshold vs SINR ratio.

2) **SINR Loss versus Distance Threshold:** We then discuss the effect of the SINR requirement $\rho^*$ on the distance threshold. In Fig. 4, we plot the distance thresholds against $\rho^*$, when user density $\beta_K = 5$, 10, and 15/\text{km}^2, respectively. The network radius is assumed to be very large. We can see that the distance threshold remains very small for a wide range of $\rho^*$, i.e., when $\rho^*$ is smaller than 0.95. There is a sharp increase in $d_0$ when $\rho^*$ approaches 1. This implies an interesting tradeoff: if full SINR is to be achieved, we do need to process the full channel matrix $\mathbf{H}$ at the cost of high complexity when the network size is large. On the other hand, if a small percentage of SINR degradation can be tolerated, the channel matrix can be significantly sparsified, leading to low-complexity detection algorithms. We emphasize that the SINR degradation may not imply a loss in the system capacity. This is because the overhead of estimating the full channel matrix can easily outweigh the SINR gain. A little compromise in SINR (say, reducing from 100\% to 95\%) may yield a higher system capacity eventually.

3) **Sparsity of $\hat{\mathbf{H}}$:** As seen from (18) and (19), for a given $\rho^*$, the distance threshold $d_0$ converges to a constant when the network radius $r$ goes to infinity. Since the average number of non-zero channel coefficients each RRH is approximately $\pi d_0^2 \beta_K$, the convergence of $d_0$ implies that the number of non-
TABLE I
PERCENTAGE OF NON-ZERO ENTRIES IN THE CHANNEL MATRIX WITH $\beta_K = 10/\text{km}^2$, $P_{N_0} = 80\text{dB}$ AND $\rho^* = 0.95$

| $r$ (km) | 5   | 10  | 15  | 20  |
|----------|-----|-----|-----|-----|
| $d_0$ (meter) | 694 | 705 | 707 | 708 |
| Percentage of non-zero entries (%) | 1.93 | 0.50 | 0.20 | 0.13 |

TABLE II
PERCENTAGE OF NON-ZERO ENTRIES IN THE CHANNEL MATRIX WITH $\beta_K = 10/\text{km}^2$, $P_{N_0} = 80\text{dB}$ AND $r = 10\text{km}$

| $\rho^*$ | 0.90 | 0.93 | 0.96 | 0.99 |
|----------|-----|-----|-----|-----|
| $d_0$ (meter) | 456 | 572 | 807 | 1812 |
| Percentage of non-zero entries (%) | 0.21 | 0.33 | 0.65 | 3.28 |

zero entries per row or per column in $\hat{H}$ does not scale with the network radius $r$ in a large C-RAN. Moreover, the percentage of non-zero entries in $\hat{H}$ is approximately $\frac{d_0^2}{r^2}$, which can be very small when $r$ is large. In Table I, we list both $d_0$ and the corresponding percentage of non-zero entries in matrix $\hat{H}$ for various network sizes, with $\beta_K = 10/\text{km}^2$ and $\rho^* = 0.95$. It can be seen that, when $r$ is large, $d_0$ does not change much with the network radius $r$. Moreover, only a small percentage of entries (say 2% $\sim$ 0.13%) in $\hat{H}$ are non-zero for all values of $r$ considered in Table I. In other words, each RRH only needs to estimate the CSI of a small number of users closest to this RRH. The channel estimation overhead can be significantly reduced. If a larger SINR loss can be tolerated, the amount of CSI needed can be further reduced as shown in Table II which lists the percentages of non-zero entries in $\hat{H}$ for different $\rho^*$, with $\beta_K = 10/\text{km}^2$ and $r = 10\text{km}$. We see that the percentage of non-zero entries can be reduced from 3.28% to 0.21% by allowing a drop of the the SINR performance from 99% to 90%.

IV. SINGLE-LAYER DYNAMIC NESTED CLUSTERING

With the sparsified channel matrix, we now proceed to present the single-layer DNC algorithm in this section. As shown in (8), to estimation $x$ is to calculate $P_{1/2} \hat{H}^H \hat{A}^{-1} y$. Note that the computational complexity is $O(N^3)$ which is dominated by calculating $\hat{A}^{-1}$. This is because the sparse matrix $\hat{H}$ only contains a constant number of non-zeros per column, and so does $P_{1/2} \hat{H}^H$, when the network area goes to infinity. Suppose the average number of non-zero entries in each column of $\hat{H}$ is $c$. The computational complexity of multiplying $P_{1/2} \hat{H}^H$ and $\hat{A}^{-1} y \in C^{N \times 1}$ is only $O(cN)$ and is much smaller than $O(N^3)$, i.e., the computational complexity of inverting $\hat{A}$. Therefore, we focus on reducing the computational
complexity of calculating $\hat{A}^{-1}y$, which is equivalent to solving for $\omega$ in the equation:

$$\hat{A}\omega = y.$$  (20)

The physical meanings of the entries in $\hat{A}$ are explained as follows. According to the threshold-based channel matrix sparsification approach, the $(n, k)\text{th}$ entry of channel matrix $\hat{H}$ is non-zero only when the $k\text{th}$ user is in the service area of RRH $n$, i.e., a circular area with radius $d_0$ centered around RRH $n$. Consequently, from the definition of $\hat{A}$ in (8), the $(n_1, n_2)\text{th}$ entry in $\hat{A}$ is non-zero only when the service areas of RRH $n_1$ and $n_2$ overlap, and there is at least one user in the overlapping area.

Consider an ideal case where the whole set of RRHs can be divided into disjoint clusters. While the RRHs within one cluster have overlapping service areas, those from different clusters do not serve the same user(s). In this case, the matrix $\hat{A}$ becomes block diagonal with each block corresponding to one cluster. Then, the complexity of calculating $\hat{A}^{-1}$ reduces from $O(N^3)$ to $O(n_i^3)$, where $n_i$ is the number of RRHs in a cluster. Note that $n_i$ is typically much smaller than $N$, i.e., the total number of RRHs in a C-RAN.

In reality, however, adjacent clusters interact and interfere with each other. Particularly, the service areas of the RRHs in adjacent clusters are likely to overlap. Traditional clustering algorithms [6], [11] usually ignore such overlapping, resulting in a noticeable performance degradation. In what follows, we show that by properly labeling the RRHs, matrix $\hat{A}$ can be transformed to a DBBD matrix, where the borders capture the overlaps between clusters. Then, later in Subsection IV-B, the DNC algorithm that enables parallel computation is presented.

A. RRH Labelling Algorithm

To start with, we give the definition of a Hermitian DBBD matrix as follows:

**Definition 1.** A matrix $A$ is said to be a Hermitian DBBD matrix if it is in the following form

$$A = \begin{bmatrix}
A_{1,1} & A_{c1}^H \\
A_{2,2} & A_{c2}^H \\
\vdots & \vdots \\
A_{m,m} & A_{cm}^H \\
A_{c1} & A_{c2} & \cdots & A_{cm} & A_c
\end{bmatrix},$$  (21)

where the diagonal blocks $A_{ii}$ are $n_i \times n_i$ Hermitian matrices, the border blocks $A_{ci}$ are $n_c \times n_i$ matrices, and the cut-node block $A_c$ is an $n_c \times n_c$ Hermitian matrix.

![Geographical RRH grouping in C-RAN.](image)

We divide the entire C-RAN area into disjoint sub-areas as illustrated in Fig. 5. We then separate each sub-area into a width-$d_0$ boundary and a sub-area center which are colored by light-green and dark-green respectively. Here, $d_0$ is the distance threshold used in the channel sparification. We see that RRHs in a sub-area center do not have overlapping service region with RRHs in other sub-areas. Only RRHs in the width-$d_0$ boundary may have overlapping service region with the RRHs in adjacent sub-areas. This implies that matrix $\hat{A}$ can be transformed to a DBBD matrix with each diagonal block corresponding to RRHs in a sub-area center and the cut-node block corresponding to RRHs in the width-$d_0$ boundaries. The border blocks of $\hat{A}$ capture the interaction between different clusters due to interference.

Denote the coordinates of an arbitrary RRH $n$ as follows:

$$l_n = (l_{x_n}, l_{y_n}),$$

where $l_{x_n} \in [0, a_x], l_{y_n} \in [0, a_y]$, $a_x$ and $a_y$ are the side lengths of the whole network. The RRH labelling algorithm is given in Algorithm 1, where $b(n)$ is the label of RRH $n$. We first divide the overall network into disjoint squares with side length $r_1$, and group the RRHs into center clusters or the boundary cluster according to their locations in steps 2 to 9. Then, the RRHs are numbered based on
Algorithm 1 RRH Labelling Algorithm

Input: \( a_x, a_y, d_0, r_1, l_n, \forall n \)

Output: \( b(n), \forall n \)

1: Set \( m_x = \lceil \frac{a_x}{r_1} \rceil \), \( m_y = \lceil \frac{a_y}{r_1} \rceil \) and \( C_i = \Phi, \forall i \in \{1, 2, \cdots, m_x m_y + 1\} \)
2: for \( n = 1 \) to \( N \) do
3: Setting \( i = \lceil \frac{lx_n}{r_1} \rceil \)
4: \( j = \lceil \frac{ly_n}{r_1} \rceil \)
5: if \((i - 1)r_1 + d_0 \leq lx_n \leq ir_1 - d_0 \) AND \((j - 1)r_1 + d_0 \leq ly_n \leq jr_1 - d_0 \) then
6: \( C_{(i-1)m_x+j} \leftarrow \{C_{(i-1)m_x+j}, n\} \)
7: else
8: \( C_{m_x m_y + 1} \leftarrow \{C_{m_x m_y + 1}, n\} \)
9: end if
10: end for
11: Set \( j = 1 \)
12: for \( i = 1 \) to \( m_x m_y + 1 \) do
13: for \( n = 1 \) to \( N \) do
14: if \( n \in C_i \) then
15: Label RRH \( n \) by \( j \): \( b(n) \leftarrow j \)
16: \( j \leftarrow j + 1 \)
17: end if
18: end for
19: end for

Fig. 6. \( \hat{A} \) in a DBBD form, where \( N = 8100, r = 30 \text{km}, d_0 = 500 \text{meter}, r_1 = 15 \text{km} \).

the cluster they belong to, as shown in steps 10 to 18. After numbering all the RRHs, we organize the matrix \( \hat{A} \) in the ascending order of the RRHs’ numbers. For example, the first row of \( \hat{A} \) corresponds to the RRH with label \( b(n) = 1 \). The matrix \( \hat{A} \) now becomes a Hermitian DBBD matrix as shown in Fig. 6, where \( \hat{A} \) has 4 diagonal blocks and 8 border blocks, and one cut-node block.

We would like to emphasize that RRH labelling in Algorithm 1 is only based on the locations of RRHs, but not on those of the users. Therefore, RRH labelling only needs to be conducted once in
system initialization.

B. Single-Layer DNC with Parallel Computing

With Ā converted into a DBBD matrix, we are now ready to present the DNC algorithm. In particular, the diagonal blocks of Ā can be processed in parallel, leading to a significant reduction in computation time.

Suppose that the \( N \times N \) DBBD matrix Ā has \( m_1 \) diagonal blocks. Then, the linear equation (20) becomes

\[
\begin{bmatrix}
\hat{A}_{1,1} & \hat{A}_{c1}^H \\
\hat{A}_{2,2} & \hat{A}_{c2}^H \\
\vdots & \vdots \\
\hat{A}_{m_1,m_1} & \hat{A}_{cm_1}^H \\
\hat{A}_{c1} & \hat{A}_{c2} & \cdots & \hat{A}_{cm_1} & \hat{A}_c
\end{bmatrix}
\begin{bmatrix}
\omega_1 \\
\omega_2 \\
\vdots \\
\omega_{m_1} \\
\omega_c
\end{bmatrix}
= \begin{bmatrix}
y_1 \\
y_2 \\
\vdots \\
y_{m_1} \\
y_c
\end{bmatrix},
\]

(23)

where the \( n_i \times 1 \) vectors \( \omega_i \) and \( y_i \) are sub-vectors of \( \omega \) and \( y \), respectively. Likewise, \( \omega_c \) and \( y_c \) are \( n_c \times 1 \) sub-vectors.

The solution to the above equation is given by

\[
\omega_c = \left( \hat{A}_c - \sum_{i=1}^{i=m_1} \hat{A}_{ci} \hat{A}_{ci}^{-1} \hat{A}_{ci}^H \right)^{-1} \left( y_c - \sum_{i=1}^{i=m_1} \hat{A}_{ci} \hat{A}_{ci}^{-1} y_i \right),
\]

(24)

and

\[
\omega_i = \hat{A}_{ci}^{-1} \left( y_i - \hat{A}_{ci}^H \omega_c \right),
\]

(25)

for all \( i \in \{1, 2, \cdots, m_1\} \). From equations (24) and (25), we draw the following conclusions. First,
TABLE III
COMPUTATION TIME OF EACH STEP IN EQUATION (24) AND (25)

| step | operation | complexity/operation | total number of operations |
|------|-----------|----------------------|---------------------------|
| 1    | \( \hat{A}_{i,i}^{-1} \) | \( O(N_{d,1}^3) \) | \( m_1 \) |
| 2    | \( \hat{A}_{ci} \hat{A}_{i,i}^{-1} \hat{A}_{ci}^H \) \( \hat{A}_{ci} \hat{A}_{i,i}^{-1} y_i \) | \( O(L_1 m_1 N_{b,1} N_{d,1}) \) | \( m_1 \) |
| 3    | \( \hat{A}_{ci} - \sum_{i=m_1}^{i=m_1} \hat{A}_{ci} \hat{A}_{i,i}^{-1} \hat{A}_{ci}^H \) \( y_c - \sum_{i=m_1}^{i=m_1} \hat{A}_{ci} \hat{A}_{i,i}^{-1} y_i \) | \( O(m_1 N_{b,1}^2) \) | 1 |
| 4    | \( \left( \hat{A}_c - \sum_{i=1}^{i=m_1} \hat{A}_{ci} \hat{A}_{i,i}^{-1} \hat{A}_{ci}^H \right)^{-1} \left( y_c - \sum_{i=1}^{i=m_1} \hat{A}_{ci} \hat{A}_{i,i}^{-1} y_i \right) \) | \( O(N_{b,1}^3) \) | 1 |
| 5    | \( y_i - \hat{A}_{ci}^H \omega_c \) | \( O(L_1 m_1 N_{b,1}) \) | \( m_1 \) |
| 6    | \( \hat{A}_{i,i}^{-1} \left( y_i - \hat{A}_{ci}^H \omega_c \right) \) | \( O(N_{d,1}^2) \) | \( m_1 \) |

\( \omega_c \), the sub-vector corresponding to the cut-node block, can be calculated independently of the other sub-vector \( \omega_i \). Second, with \( \omega_c \) obtained from (24), we can calculate each \( \omega_i \) using (25) independently. The calculation of \( \omega_i \) only involves the \( i^{th} \) diagonal block of \( \hat{A} \) and the corresponding \( i^{th} \) border block \( \hat{A}_{ci} \). In other words, if we treat each diagonal block as a cluster, then the signals received by each cluster can be processed in parallel of each other, while the interactions between different clusters are captured by \( \omega_c \) and the border blocks. Based on the above discussions, Fig. 7 shows the architecture of the C-RAN BBU pool, where parallel signal processing is carried out. The arrows in Fig. 7 indicate the data flows between the processing units. As the figure shows, to expedite the calculation of \( \omega_c \), matrices \( \hat{A}_{ci} \hat{A}_{i,i}^{-1} \hat{A}_{ci}^H \) and vectors \( \hat{A}_{ci} \hat{A}_{i,i}^{-1} y_i \) are calculated at the same time by a number of parallel processing units and then fed into the central processing units. Then, \( \omega_c \) is calculated in a central processing unit. The result is fed back into the parallel processing units. Each parallel processing unit is responsible for processing one cluster and calculating \( \omega_i \). Specifically, we divide all the operations of signal processing in the proposed clustering algorithm into six steps as listed in Table II. Steps 1 and 2 are first carried out in the parallel processing units. After receiving the results of steps 1 and 2, the central processing unit performs steps 3 and 4. At last, steps 5 and 6 are carried out in the parallel processing units.
C. Optimizing the Computation Time

Table III also lists the detailed computational complexity of each step in the single-layer DNC algorithm, where $N_{d,1}$ and $N_{b,1}$ are the average size of the diagonal blocks and cut-node block respectively. $m_1$ is the average number of diagonal blocks. $L_1 \ll N$ is the average number of non-zero entries per row in $\hat{A}$, which is an increasing function of the distance threshold $d_0$. In practical, $L_1$ does not increase with $N$, and is much smaller than $N$. Then, the total computational complexity in the parallel processing units is $O(N_{d,1}^3)$. The complexity in the central one is $O(N_{b,1}^3)$.

Before optimizing the computation time, we make some assumptions on the C-RAN BBU pool. We notice that the size of diagonal block should be determined by the processing power of the corresponding parallel processing unit. This implies that the sub-areas should have different side lengths, say different $r_1$. To simplify later discussions, we assume that all the parallel processing units in the BBU pool have equal processing power. We also notice that the central processing unit should be more powerful than the parallel ones. Otherwise, the central processing unit is unnecessary. For example, the corresponding operations, i.e., steps 3 and 4, can be carried out at one of the parallel processing units instead of the central one, and the total computational complexity can be reduced. Then, we define an unbalanced processing power ratio $\varrho$ to represent the processing power of the central processing unit and parallel ones. That is, to perform a same operation, the computation time of the central processing unit is $\varrho$ times shorter than that of the parallel ones. Denote a log-N ratio as $s = \log N \varrho$ for notational brevity. Without loss of generality, the processing power of a parallel processing unit is normalized to be 1. Then, the processing power of the central one is $N^s$. As the operations in steps 1, 2, 5 and 6 can be performed in parallel, the total computation time is $O(N_{d,1}^3 + N_{b,1}^3 N^{-s})$.

The computation time is an increasing function of the block sizes, $N_{d,1}$ and $N_{b,1}$. To achieve a short computation time, $N_{d,1}$ and $N_{b,1}$ should be as small as possible. However, the block sizes cannot be adjusted arbitrarily. In fact, for a given $r_1$, there is a fixed ratio between $N_{d,1}$ and $N_{b,1}$. We denote this ratio as $N^{z_1} = \frac{N_{d,1}}{N_{b,1}}$. Specifically, since $N_{d,1}$ and $N_{b,1}$ equal to the average number of RRHs in the

\[1\]There are several matrix multiplication/inversion algorithms, which lead to various computational complexity. In this paper, we only take the complexity of some common algorithms as an example.
sub-area center and the boundaries respectively, the relationship between \( r_1 \) and the ratio \( N_{z_1} \) is

\[
(r_1 - 2d_0)^2 = 4(r_1 - d_0)d_0 \frac{r_2^2}{r_1^2} N_{z_1},
\]

where \( \beta_N \) is the RRH density. By adjusting \( r_1 \) from \( 2d_0 \) to \( r \), \( z_1 \) goes from \(-1\) to \( 1\). Based on (26), we obtain the following approximations of \( N_{d,1} \), \( N_{b,1} \) and \( m_1 \):

**Lemma 1.** In large C-RANs, given the block size ratio \( N_{z_1} \), the approximations of \( N_{d,1} \), \( N_{b,1} \) and \( m_1 \) are

\[
N_{d,1} \approx \left(4d_0 \beta_N \frac{1}{2} N_{1+z_1} \right)^{\frac{3}{2}},
\]

\[
N_{b,1} \approx \left(4d_0 \beta_N \frac{1}{2} N_{1-z_1} \right)^{\frac{3}{2}},
\]

\[
m_1 \approx \left(4d_0 \right)^{-\frac{3}{2}} \beta_N \frac{1}{2} N_{\frac{1}{2} - \frac{3}{2} z_1}.
\]

**Proof:** \( r_1 \) is the solution of (26), and we have

\[
(4d_0 \beta_N \frac{1}{2} N_{1+z_1} \right)^{\frac{3}{2}} \leq r_1 \leq (4d_0 \beta_N \frac{1}{2} N_{1-z_1} \right)^{\frac{3}{2}} + 2d_0,
\]

Then, when \( d_0 \) is much smaller than \( r_1 \),

\[
N_{d,1} = \beta_N (r_1 - 2d_0)^2 \approx \left(4d_0 \beta_N \frac{1}{2} N_{1+z_1} \right)^{\frac{3}{2}},
\]

\[
N_{b,1} = N_{d,1} N^{-z_1} \approx \left(4d_0 \beta_N \frac{1}{2} N_{1-z_1} \right)^{\frac{3}{2}},
\]

\[
m_1 = \frac{r_2^2}{r_1^2} \approx \left(4d_0 \right)^{-\frac{3}{2}} \beta_N \frac{1}{2} N_{\frac{1}{2} - \frac{3}{2} z_1}.
\]

![Fig. 8. Order of computation time vs log-N ratio s.](image)

After ignoring \( d_0, \beta_N \) and \( L_1 \), we obtain the optimal computation time with parallel computing below.
Lemma 2. When the log-N ratio $s \leq 3$, the minimum computation time with parallel computing is $O(N^{2-\frac{2}{3}s})$, with the optimal $z_1 = -\frac{s}{3}$. When $s > 3$, the minimum computation time is $O(N^{3-s})$, with the optimal $z_1 = -1$.

Remark 1. We notice that when the central processing unit is much more powerful than other units in the data center, performing all the operations in the central processor can achieve a shorter computation time than parallel computing. Based on Table III and Lemma 1, the computation time of serial computing at the central processing unit is $O(N^{\frac{15}{7}-s})$ with $z_1 = -\frac{1}{7}$.

Then, based on Lemma 2 and Remark 1, we show the minimum computation time in Proposition 1.

Proposition 1. In C-RAN with parallel processing units and a central processing unit, when the log-N ratio $s \leq \frac{3}{7}$, the optimal computation time, $O(N^{2-\frac{2}{7}s})$, is achieved by parallel computing. When $s > \frac{3}{7}$, the optimal computation time, $O(N^{\frac{15}{7}-s})$, is achieved by performing all the operations at the central processing unit in serial.

We also show the effect of $s$ on the total computation time in Fig. 8 where the order of computation time is the maximum exponent of the computation time function. For example, the order of $O(N^{2-\frac{2}{7}s})$ is $2 - \frac{2}{7}s$. We see that the order of the computation time is reduced with the increase of $s$. However, obviously, the price of the central processing unit is increased with the increase of $s$. Then, Fig. 8 illustrates a trade-off between the computation time and the economic cost. This trade-off can serve as a guideline during the deployment of BBU pool. For example, when the economic cost is a major concern in the C-RAN system or a long computation time can be tolerated, processing units with low processing power, which leads to a low price, should be deployed. When the computation time is more important than cost, more powerful processing units should be selected.

Remark 2. So far, we have assumed that there are always enough parallel processing units, regardless of $r_1$ or $z_1$. In this case, we only need to optimize the sizes of diagonal blocks and the cut-node block and ignore the number of blocks. Instead, when the number of parallel processing units is limited, the number of blocks, $m_1$, also has an effect on the total computation time. Based on Lemma 1, $m_1$ can also be adjusted by $r_1$ or $z_1$. In this way, we can balance the computation time with limited availability.
of processing units. More detailed analysis, however, is out of the scope of this paper.

In this section, we have shown that the simple RRH labelling algorithm allows us to easily optimize the size of diagonal blocks, which can be directly interpreted as the size of clusters in large C-RANs. The result further provides an important guideline as to the architecture design of the C-RAN BBU pool, including the number of processing units, the choice between parallel and serial processing, the allocation of processing power among BBUs, etc.

V. Multi-layer DNC Algorithm

In the preceding section, we propose a single-layer DNC algorithm, which reduces the total computation time from $O(N^3)$ to $O(N^2)$. In this section, we propose a multi-layer DNC algorithm to further reduce the computation time.

We notice that the computation time of the parallel processing units in the single-layer DNC algorithm is dominated by calculating $\hat{A}^{-1}_{i,i}$. Interestingly, a close study indicates that the diagonal blocks $\hat{A}_{i,i}$ are themselves sparse matrices. This is because the RRHs in the same cluster only have interactions with their neighboring RRHs instead of the whole cluster. This implies that $\hat{A}_{i,i}$ can be further permuted to a DBBD form, and the computation time of calculating $\hat{A}^{-1}_{i,i}$ can be reduced. As such, matrix $\hat{A}$ becomes a two-layer nested DBBD matrix. An example of such a matrix is shown in Fig. 9.

Fig. 9. $\hat{A}$ in a two-layer nested DBBD form after the second time RRH labelling, with $r_1 = 15$ km, $r_2 = 8$ km, where $N = 8100$, $r = 30$ km, $d_0 = 500$ meter.

Fig. 10 illustrates the RRH labelling strategy that turns $\hat{A}_{i,i}$ into a DBBD form. In particular, the RRHs in a cluster is grouped into sub-clusters. For example, for the top-left square, the RRHs at the dark green boundary area are clustered to the sub-border, and the RRHs at the center area (the white
area) are clustered to diagonal blocks. Intuitively, one can minimize the computation time by balancing the sizes of different blocks. This is the focus of our study in the remainder of this section.

Note that by repeating the process, $\hat{A}$ can be further permuted into a multi-layer nested DBBD matrix. For simplicity, we focus on the two-layer DNC algorithm in this section. The results, however, can be easily extended to the multi-layer case, as briefly discussed at the end of the section.

A. Multi-Layer DNC Algorithm with Parallel Computing

As discussed in the previous section, each parallel processor in Fig. needs to calculate $\hat{A}_{i,i}^{-1}$. In the following, we show how $\hat{A}_{i,i}^{-1}$ can be computed in parallel, if $\hat{A}$ is already a two-layer nested DBBD matrix with diagonal blocks $\hat{A}_{i,i}$ being DBBD as well. For notational brevity, we denote the diagonal block $\hat{A}_{i,i} \in \mathcal{C}^{L \times L}$ by $B$. Then, inverting $B$ is equivalent to solving the following system:

$$
\begin{bmatrix}
B_{1,1} & B_{c1}^H \\
B_{c1} & \vdots & \vdots \\
B_{c2} & \vdots & \vdots \\
\vdots & \ddots & \vdots \\
B_{cm} & B_{cm}^H & \vdots \\
B_{c1} & B_{c2} & \cdots & B_{cm} & B_c
\end{bmatrix}
\begin{bmatrix}
X_1 \\
X_2 \\
\vdots \\
X_m \\
X_c
\end{bmatrix}
= 
\begin{bmatrix}
I_1 \\
I_2 \\
\vdots \\
I_m \\
I_c
\end{bmatrix},
$$

(34)

where $X = [X_1^T, X_2^T, \cdots, X_m^T, X_c^T]^T$ and $I = [I_1^T, I_2^T, \cdots, I_m^T, I_c^T]^T$, with $X_i, I_i \in \mathcal{C}^{n_i \times L}$, and $X_c, I_c \in \mathcal{C}^{n_c \times L}$. 

Fig. 10. Geographical RRH grouping in C-RAN.
The columns in matrix $X$ is given below:

$$X_c = \left( B_c - \sum_{i=1}^{i=m} B_{c,i} B_{c,i}^{-1} B_{c,i}^H \right)^{-1} \left[ B_{c,1} B_{1,1}^{-1} , \cdots , B_{c,m} B_{m,m}^{-1} , I \right], \tag{35}$$

$$X_i = B_{i,i}^{-1} (I - B_{c,i}^H X_c). \tag{36}$$

Similar to equations (24) and (25), parallel computing can be adopted in calculating (35) and (36). Combining the parallel computing of calculating $\hat{A}_{i,i}^{-1}$ in the second layer and that of solving $\omega_i$ in the first layer, a nested parallel computing architecture is illustrated in Fig. 11. We first need to calculate $\hat{A}_{i,i}^{-1}$, i.e. $B_{i,i}^{-1}$. Since the diagonal blocks in $\hat{A}$ are in DBBD forms, the calculation of $B_{i,i}^{-1}$, can be split and allocated to a number of parallel processing units. As listed in Table IV, the calculation of $B_{i,i}^{-1}$ (i.e. step 1 in Table III) is divided into six steps. Steps 1.1 and 1.2 are first carried out in the level-3 processing units. The results are fed back into the level-2 processing units, which are responsible for performing steps 1.3 and 1.4. Then, steps 1.5 and 1.6 are carried out in the level-3 parallel processing units. Then, similar to the single-layer DNC algorithm, the level-2 processing units calculate matrices $\hat{A}_{c,i} \hat{A}_{i,i}^{-1} \hat{A}_{c,i}^H$ and vectors $\hat{A}_{c,i} \hat{A}_{i,i}^{-1} y_i$, and the results are fed into the level-1 processing unit. Then, $\omega_c$ is calculated by the level-1 processing unit, and $\omega_i$ is calculated by the level-2 processing unit.
TABLE IV
COMPUTATION TIME OF EACH STEP IN EQUATION (35) AND (36)

| step | operation | complexity/operation | total number of operations |
|------|-----------|----------------------|---------------------------|
| 1.1  | $B_{i,i}^{-1}$ | $O(N_d^3)$ | $m_2$ |
| 1.2  | $B_{ci}B_{i,i}^{-1}B_{ci}^H$ $B_{ci}B_{i,i}^{-1}I_i$ | $O(L_2 N_{b,t} N_{d,t})$ | $m_2$ |
| 1.3  | $B_{c} - \sum_{i=1}^{m_2} B_{ci}B_{i,i}^{-1}B_{ci}^H$ $I_c - \sum_{i=1}^{m_2} B_{ci}B_{i,i}^{-1}I_i$ | $O(m_2 N_b^2)$ | 1 |
| 1.4  | $\bigg(B_c - \sum_{i=1}^{m_2} B_{ci}B_{i,i}^{-1}B_{ci}^H\bigg)^{-1} \bigg(I_c - \sum_{i=1}^{m_2} B_{ci}B_{i,i}^{-1}I_i\bigg)$ | $O(N_b^2)$ | 1 |
| 1.5  | $I_i - B_{ci}^H X_c$ | $O(L_2 N_{b,t} N_{d,t})$ | $m_2$ |
| 1.6  | $B_{i,i}^{-1} \big(I_i - B_{ci}^H X_c\big)$ | $O(N_b^2)$ | $m_2$ |

B. Optimizing the Computation Time

The computation time of each step in calculating $B^{-1}$ is listed in Table IV, where $N_{d,t}$ and $N_{b,t}$ are the average diagonal block size and the cut-node block size in the $t$th layer, respectively, $m_2$ is the average number of diagonal blocks in $B$, and $L_2 \ll N_{d,1}$ is the average number of non-zero entries per row in $B$. Similarly to the single-layer DNC algorithm, we assume there exist enough processing units in each level, and the processing units in the same level have the same processing power. Moreover, the processing units in each level should be more powerful than those in higher levels. For example, processing units in Level 1 and 2 are more powerful than those in Level 3. Otherwise, we can remove Level 1 and 2 by shifting all the corresponding operations to Level 3. The reason is that the total number of processing units in Level 3 is always greater than that in Level 1 and 2. Then, we define an unbalanced processing power ratio $\varrho_t$ to represent the processing power ratio of processing units in the $t$th level and that in Level 3, where $t = 1$ or 2. Denote the corresponding log-N ratio as $s_t = \log_N \varrho_t$. Following the single-layer DNC algorithm, we can adjust the side length $r_t$ in the RRH labelling algorithm for different types of data centers as in the single-layer DNC algorithm. We first define the block-size ratio of a diagonal block and the cut-node block in the $t$th layer as $N^{z_t}$, i.e., $N^{z_t} = \frac{N_{d,t}}{N_{b,t}}$. By adjusting $r_t$, $N^{z_t}$ goes from $N^{-1}_{d,t-1}$ to $N_{d,t-1}$. Then, the upper bounds of the diagonal block size and the cut-node block size in each layer is given below.

Lemma 3. In large C-RANs, given the block size ratio $N^{z_t}$ in the $t$th layer, the side length $r_t$ is the
solution of equation \((r_t - 2d_0)^2 = 4(r_t - d_0)d_0 \frac{r_t^2}{r_t^2} N_{d,t}^{z_1}\). The approximations of \(N_{d,t}\), \(N_{b,t}\) and \(m_t\) are

\[
N_{d,t} \approx \left(4d_0 \beta N_{d,t-1}^N\right)^{\frac{1}{2}},
\]

(37)

\[
N_{b,t} \approx \left(4d_0 \beta N_{d,t-1}^N\right)^{\frac{3}{2}},
\]

(38)

\[
m_t \approx \left((4d_0)^{-2} \beta^{-1} N_{d,t-1}^N\right)^{-\frac{3}{2}}.
\]

(39)

Similar to the single-layer DNC algorithm, we find that when the processing units in Level 1 and Level 2 are much more powerful than those in Level 3, parallel computing is not the most efficient way. Here, we list three computing modes.

- **Mode 1**: Steps 1.1, 1.2, 1.5, and 1.6 are executed at the level-3 processing units, steps 1.3, 1.4, 2, 5 and 6 are executed at the level-2 processing units, and steps 3 and 4 are carried out at the level-1 processing unit.

- **Mode 2**: Steps 1, 2, 5 and 6 are executed at the level-2 processing units, and steps 3 and 4 are executed at the level-1 processing unit.

- **Mode 3**: All the steps are executed at the level-1 processing unit in serial.

Then, we conclude the computing strategy of the two-layer DNC algorithm for different \(s_1\) and \(s_2\) in Proposition 2.

**Proposition 2.** For a three-level BBU pool with log-N ratio \(s_1\) and \(s_2\),

- when \(s_1 + 7s_2 < 3\) and \(3s_1 - 2s_2 < \frac{4}{3}\), choose mode 1, and the minimum computation time is \(O(N^{\frac{12}{23} - \frac{14}{23}s_1 - \frac{16}{23}s_2})\) by setting \(z_1 = \frac{4}{23} - \frac{9}{23}s_1 + \frac{6}{23}s_2\) and \(z_2 = -\frac{1}{2}s_2\);

- when \(s_1 + 7s_2 \geq 3\) and \(s_1 - s_2 < \frac{1}{3}\), choose mode 2, and the minimum computation time is \(O(N^{\frac{15}{23} - \frac{7}{23}s_1 - \frac{3}{23}s_2})\) with \(z_1 = \frac{1}{8} - \frac{3}{8}s_1 + \frac{3}{8}s_2\) and \(z_2 = -\frac{3}{16} + \frac{1}{16}s_1 - \frac{1}{16}s_2\);

- when \(3s_1 - 2s_2 \geq \frac{4}{3}\) and \(s_1 - s_2 \geq \frac{1}{3}\), choose mode 3, and the minimum computation time is \(O(N^{2-s_1})\) with \(z_1 = 0\) and \(z_2 = -\frac{1}{6}\).

Repeating the parallel computing for the diagonal blocks, we can extend the DNC algorithm to multi-layer ones. We notice that the multi-layer DNC algorithm is more flexible for different data centers than the single-layer DNC algorithm. Multiple central processing units are allowed in the multi-layer DNC algorithm. Moreover, the total computation time is reduced in the multi-layer algorithm. Fig. 12 shows the order of the computation time versus \(s_1\) and \(s_2\) for two-layer DNC algorithm.
the total computation time of the single-layer DNC algorithm is $O(N^2)$ while the computation time of the two-layer one is only $O(N^{\frac{12}{3}})$. It is not difficult to see that the computation time can be further reduced when more layers are introduced.

VI. CONCLUSIONS AND DISCUSSIONS

In this paper, we proposed the DNC algorithm, based on which, both the channel estimation overhead and the computational complexity of the uplink signal processing can be significantly reduced. Moreover, our algorithm serves as a unified theoretical framework for dynamic clustering of C-RAN. By introducing a distance threshold, RRHs are grouped into disjoint center clusters or a boundary cluster based on their locations. The boundary cluster captures the interaction between different center clusters, which avoids the performance loss caused by conventional clustering. In addition, the operations in center clusters can be performed in parallel. We show that both the size and the number of the center clusters as well as the size of the boundary cluster can be easily adjusted. This further allows to flexibly balance the computation time, the number of parallel BBUs required, the allocation of computational power among BBUs, etc. Therefore, the DNC algorithm is adaptive to various architectures of the BBU pool.

The algorithm proposed in this paper can be improved in aspects, such as efficiency and flexibility. For example, according to our proposed algorithm, all the RRHs located at the boundary of the sub-areas are clustered to the boundary cluster. However, some of the boundary RRHs do not have overlapping users with RRHs located in other sub-areas. Thus, if we assign these boundary RRHs to the center
clusters, the size of boundary cluster be reduced. This implies that after balancing the cluster size, both
the center cluster sizes and the boundary cluster size can be reduced, with an increase of the number of
center clusters. Since the total computation time is limited by the block sizes, re-assigning the boundary
RRHs in general reduces the computation time. Furthermore, the proposed algorithm can be adjusted
to allow more flexibility in implementation. For example, center clusters with different sizes can be
made possible by dividing the whole network into sub-areas with different sizes. In this way, parallel
processing units with various processing powers can be used in implementing the proposed algorithm,
so as to meet the practical requirements of wireless networks. However, detailed analysis is out of the
scope of this paper, and will be an interesting topic for future research.

\section*{Appendix: Proof of Theorem 1}

\begin{align}
E \left[ \text{SINR}_k(d_0) \right] &= E \left[ \frac{P_k \left( |\hat{v}_k| \right)^2}{\hat{v}_k \left( \sum_{j \neq k} P_j \hat{h}_j \hat{h}_j^H + N_0 I \right) \hat{v}_k} \right] \quad (40a) \\
&= E \left[ \hat{v}_k \left( \sum_{j \neq k} P_j \hat{h}_j \hat{h}_j^H + N_0 I \right) \hat{v}_k \right] \quad (40b) \\
&\geq E_{\hat{h}_k} E_{\hat{h}_j, j \neq k} \left[ \frac{P_k |\hat{v}_k|^2}{\hat{v}_k \left( \sum_{j \neq k} P_j \hat{h}_j \hat{h}_j^H + N_0 I \right) \hat{v}_k} \right] \quad (40c) \\
&\geq E_{\hat{h}_k} \left[ \frac{P_k |\hat{v}_k|^2}{\hat{v}_k \left( \sum_{j \neq k} P_j \hat{h}_j \hat{h}_j^H + N_0 I \right) \hat{v}_k} \right] \quad (40d) \\
&= E_{\hat{h}_k} \left[ \frac{1}{1 - \hat{v}_k \hat{h}_k} - 1 \right] \quad (40e) \\
&= E_{\hat{h}_k} \left[ \frac{1}{1 - \hat{v}_k \hat{h}_k} - 1 \right] \quad (40f) \\
&= E_{\hat{h}_j, j \neq k} \left[ \frac{1}{1 - \hat{v}_k \hat{h}_k} - 1 \right] \quad (40g) \\
&= P_k \mu \left[ \frac{1}{1 - \hat{v}_k \hat{h}_k} - 1 \right] \quad (40h) \\
\end{align}

where (40d) follows the fact that $\hat{v}_k \left( \sum_{j \neq k} P_j \hat{h}_j \hat{h}_j^H + N_0 I \right) \hat{v}_k$ converges to
$\left(N_1 - P_k E \left[ |\tilde{h}_{n,k}|^2 \right] \right) \hat{v}_k \hat{v}_k$ as $N$ goes to infinity due to the law of large numbers.
Likewise, we obtain
\[
E[SINR_k] = P_k \mu E \left[ \text{tr} \left( \sum_{j \neq k} P_j h_j h_j^H + N_0 I \right)^{-1} \right].
\] (41)

Then, the SINR ratio becomes
\[
\rho(d_0) \geq \frac{P_k \tilde{\mu} E_{h_j, \forall j \neq k}}{P_k \mu E_{h_j, \forall j \neq k}} \left[ \text{tr} \left( \sum_{j \neq k} P_j \tilde{h}_j \tilde{h}_j^H + N_1 I + N_0 I \right)^{-1} \right]\]
\[
\geq \frac{P_k \tilde{\mu} E_{h_j, \forall j \neq k}}{P_k \mu E_{h_j, \forall j \neq k}} \left[ \text{tr} \left( \sum_{j \neq k} P_j h_j h_j^H + N_1 I + N_0 I \right)^{-1} \right]\]
\[
\geq \frac{P_k \tilde{\mu} E_{h_j, \forall j \neq k}}{P_k \mu E_{h_j, \forall j \neq k}} \left[ \text{tr} \left( \sum_{j \neq k} P_j h_j h_j^H + N_1 I + N_0 I \right)^{-1} \right]\]
\[
= \frac{\tilde{\mu} E}{\mu} \left[ \sum_{i=1}^{N} \frac{1}{\lambda_i + N_1 + N_0} \right]\]
\[
\geq \frac{\tilde{\mu} N_0}{\mu (N_1 + N_0)},
\]
(42d)
(42e)
where \(\lambda_1, \lambda_2, \ldots, \lambda_N\) are the eigenvalues of the positive semidefinite matrix \(\sum_{j \neq k} P_j h_j h_j^H\). (42e) holds since \(N_1 \geq 0, N_0 \geq 0\) and \(\lambda_i \geq 0, \forall i\).

Substituting \(N_1 = (\mu - \tilde{\mu}) \sum_{j \neq k} P_k\) into (42e), we have
\[
\rho(d_0) \geq \frac{\tilde{\mu} N_0}{\mu ((\mu - \tilde{\mu}) \sum_{j \neq k} P_k + N_0)}.
\] (43)

When the users transmit equal power, i.e., \(P_1 = P_2 = \cdots = P_K = P\), we have
\[
\rho(d_0) \geq \frac{\tilde{\mu} N_0}{\mu ((\mu - \tilde{\mu})(K - 1)P + N_0)}.
\] (44)

REFERENCES

[1] “C-RAN: The road towards green RAN,” China Mobile Res. Inst., Beijing, China, Oct. 2011, White Paper, ver. 2.5.

[2] M. Tuchler, A. Singer, and K. Ralf, “Minimum mean squared error equalization using a prior information,” IEEE Trans. Signal Process., vol. 50, no. 3, pp. 673-683, 2002.

[3] A. Lozano, R. Heath, and J. Andrews, “Fundamental limits of cooperation,” IEEE Trans. Inf. Theory, vol. 59, no. 9, pp. 5213-5226, Sep. 2013.

[4] A. Liu and V. Lau, “Joint power and antenna selection optimization in large cloud radio access networks,” IEEE Trans. Signal Process., vol. 62, no. 5, pp. 1319-1328, Mar. 2014.

[5] O. Mehanna, N. Sidiropoulos, and G. Giannakis, “Joint multicast beamforming and antenna selection”, IEEE Trans. Signal Process., vol. 61, no. 10, pp. 2660-2674, May. 2013.

[6] S. Akoum, and R. Heath, “Interference coordination: random clustering and adaptive limited feedback”, IEEE Trans. Signal Processing., vol. 61, no. 7, pp. 1822-1834, Apr. 2013.
[7] A. Liu, and V. Lau, “Hierarchical interference mitigation for large MIMO cellular networks”, IEEE Trans. Signal Processing., vol. PP, no. 99, pp. 1-1, 2014.

[8] J. Zhang, R. Chen, J. Andrews, A. Ghosh, and R. Heath, “Networked MIMO with clustered linear precoding,” IEEE Trans. Wireless Commun., vol. 8, no. 4, pp. 1910-1921, 2009.

[9] A. Papadogiannis, D. Gesbert, and E. Hardouin, “A dynamic clustering approach in wireless networks with multi-cell cooperative processing”, in Proc. IEEE ICC’08, May 2008, pp. 4033-4037.

[10] J. Gong, S. Zhou, Z. Niu, L. Geng, and M. Zheng, “Joint scheduling and dynamic clustering in downlink cellular networks”, in Proc. IEEE Global Telecom. Conf., 2011, pp. 1-5.

[11] N. Lee, D. Morales-Jimenez, A. Lozano, and R. Heath Jr., “Spectral efficiency of dynamic coordinated beamforming: a stochastic geometry approach”, arXiv.org, May 2014. Available: arxiv.org/abs/1405.1365

[12] Y. Shi, J. Zhang, and K. Letaief, “Optimal stochastic coordinate beamforming with compressive CSI acquisition for cloud-RAN”, 2013. [Online]. Available: [http://arxiv.org/abs/1312.0363](http://arxiv.org/abs/1312.0363)

[13] D. Moltchanov, “Distance distributions in random networks”, Ad Hoc Networks, vol. 10, no. 6, pp. 1146-1166, Mar 2012.