7 Large-Scale Convex Optimization for C-RANs

Yuanming Shi, Jun Zhang, Khaled B. Letaief, Bo Bai, and Wei Chen

7.1 Introduction

7.1.1 C-RANs

The proliferation of "smart" mobile devices, coupled with new types of wireless applications, has led to an exponential growth in wireless and mobile data traffic. In order to provide high-volume and diversified data services, C-RAN [1, 2] has been proposed; it enables efficient interference management and resource allocation by shifting all the baseband units (BBUs) to a single cloud data center, i.e., by forming a BBU pool with powerful shared computing resources. Therefore, with efficient hardware utilization at the BBU pool, a substantial reduction can be obtained in both the CAPEX (e.g., via low-cost site construction) and the OPEX (e.g., via centralized cooling). Furthermore, the powerful conventional base stations are replaced by light and low-cost remote radio heads (RRHs), with the basic functionalities of signal transmission and reception, which are then connected to the BBU pool by high-capacity and low-latency optical fronthaul links. The capacity of C-RANs can thus be significantly improved through network densification and large-scale centralized signal processing at the BBU pool. By further pushing a substantial amount of data, storage, and computing resources (e.g., the radio access units and end-user devices) to the edge of the network, using the principle of mobile edge computing (i.e., fog computing) [3], heterogeneous C-RANs [4], as well as Fog-RANs and MENG-RANs [5] can be formed. These evolved architectures will further improve user experience by offering on-demand and personalized services and location-aware and content-aware applications. In this chapter we investigate the computation aspects of this new network paradigm, and in particular focus on the large-scale convex optimization for signal processing and resource allocation in C-RANs.

7.1.2 Large-Scale Convex Optimization: Challenges and Previous Work

Convex optimization serves as an indispensable tool for resource allocation and signal processing in wireless networks [6–9]. For instance, coordinated beamforming [10] often yields a convex optimization formulation, i.e., second-order cone programming (SOCP) [11]. The network max–min fairness-rate optimization [12] can be solved through the bisection method [11], in polynomial time; in this method a sequence of convex subproblems needs to be solved. Furthermore, convex relaxation provides

a principled way to develop polynomial-time algorithms for non-convex or NP-hard problems, e.g., group-sparsity penalty relaxation for NP-hard mixed-integer non-linear programming problems [2], semidefinite relaxation [7] for NP-hard robust beamforming [13, 14] and multicast beamforming [15], and a sequential convex approximation to the highly intractable stochastic coordinated beamforming problem [16].

Nevertheless, in dense C-RANs [5], which may possibly need to handle hundreds of RRHs simultaneously, resource allocation and signal processing problems will be dramatically scaled up. The underlying optimization problems will have a high dimension and/or a large number of constraints, e.g., per-RRH transmit power constraints and per-MU (mobile user) QoS constraints. For instance, for a C-RAN with 100 single-antenna RRHs and 100 single-antenna MUs, the dimension of the aggregative coordinated beamforming vector of the optimization variables will be 10^4. Most advanced off-the-shelf solvers (e.g., SeDuMi [17], SDPT3 [18], and MOSEK [19]) are based on the interior-point method. However, the computational burden of such a second-order method makes it inapplicable for large-scale problems. For instance, solving convex quadratic programs has cubic complexity [20]. Furthermore, to use these solvers the original problems need to be transformed to the standard forms supported by the solvers. Although parser/solver modeling frameworks such as CVX [21] and YALMIP [22] can automatically transform original problem instances into standard forms, they may require substantial time to perform such a transformation [23], especially for problems with a large number of constraints [24].

One may also develop custom algorithms to enable efficient computation by exploiting the structures of specific problems. For instance, the uplink–downlink duality [10] can be exploited to extract the structures of optimal beamformers [25] and enable efficient algorithms. However, such an approach still has cubic complexity since it performs matrix inversion at each iteration [26]. First-order methods, e.g., the alternating-direction method of multipliers (ADMM) algorithm [27], have recently attracted attention for their distributed and parallelizable implementation as well as for their capability of scaling to large problem sizes. However, most existing ADMM-based algorithms cannot provide the certificates of infeasibility [13, 26, 28] which are needed for such problems as max–min rate maximization [24] and group sparse beamforming [2]. Furthermore, some of these algorithms may still fail to scale to large problem sizes, owing to SOCP subproblems [28] or semidefinite programming (SDP) subproblems [13] needed to be solved at each iteration.

Without efficient and scalable algorithms, previous studies of wireless cooperative networks either only demonstrate performance in small-size networks, typically with less than 10 RRHs, or resort to suboptimal algorithms, e.g., zero-forcing-based approaches [29, 30]. Meanwhile, from the above discussion, we see that the large-scale optimization algorithms to be developed should possess the following two features:

- they should scale well to large problem sizes with parallel computing capability;
- they should detect problem infeasibility effectively, i.e., provide certificates of infeasibility.

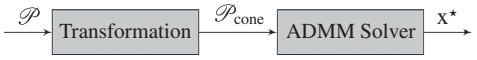

Figure 7.1 The proposed two-stage approach for large-scale convex optimization. The optimal solution or the certificate of infeasibility can be extracted from $\mathbf{x}^\star$ by the ADMM solver.

7.1.3 A Two-Stage Approach for Large-Scale Convex Optimization

To address the above two requirements in a unified way, in this chapter we shall present a two-stage approach, as shown in Fig. 7.1. The proposed framework [31] is capable of solving large-scale convex optimization problems in parallel, as well as providing certificates of infeasibility. Specifically, the original problem $\mathscr{P}$ is first transformed into a standard cone programming form $\mathscr{P}_{\text{cone}}$ [20] based on the Smith-form reformulation [32], which involves introducing a new variable for each subexpression in the disciplined convex programming form [33] of the original problem. This will eventually transform the coupled constraints in the original problem into a structured constraint consisting only of two convex sets: a subspace, and a convex set formed by a Cartesian product of a finite number of standard convex cones. Such a structure helps to develop efficient parallelizable algorithms and enable infeasibility detection capability *simultaneously* via solving the homogeneous self-dual embedding [34] of the primal–dual pair of standard form by the ADMM algorithm.

As the mapping between the standard cone program and the original problem depends only on the network size (i.e., the numbers of RRHs, MUs and antennas at each RRH), we can pre-generate and store the structures of the standard forms with different candidate network sizes. Then for each problem instance, i.e., given the channel coefficients, QoS requirements, and maximum RRH transmit powers, we only need to copy the original problem parameters to the standard cone-programming data. Thus, the transformation procedure will be very efficient and can avoid repeatedly parsing and re-generating problems [21, 22]. This technique is called *matrix stuffing* [23, 24] and is essential for the proposed framework to scale well to large problem sizes. It may also help rapid prototyping and testing in practical equipment development.

7.1.4 Outline

In Section 7.2 we demonstrate that typical signal processing and resource allocation problems in C-RANs can be solved essentially through addressing one or a sequence of large-scale convex optimization or convex feasibility problems. In Section 7.3, a systematic cone-programming form-transformation procedure is developed. The operator splitting method with detailed discussions is presented in Section 7.4. Numerical results will be reported in Section 7.5. We give a summary and conclusions in Section 7.6.

7.2 Large-Scale Convex Optimization in Dense C-RANs

Consider the following parametric family $\mathscr{P}$ of convex optimization problems:

$$\begin{aligned} \mathscr{P}: \underset{\mathbf{x}}{\text{minimize}} \quad & f_0(\mathbf{x};\boldsymbol{\alpha}) \\ \text{s. t.} \quad & f_i(\mathbf{x};\boldsymbol{\alpha}) \le g_i(\mathbf{x};\boldsymbol{\alpha}), \quad i=1,\ldots,m, \qquad (7.1) \\ & u_i(\mathbf{x};\boldsymbol{\alpha}) = v_i(\mathbf{x};\boldsymbol{\alpha}), \quad i=1,\ldots,p, \qquad (7.2) \end{aligned}$$

where $\mathbf{x} \in \mathbb{R}^n$ is the vector of optimization variables and $\boldsymbol{\alpha} \in \boldsymbol{\mathcal{A}}$ is the problem parameter vector; $\mathcal{A}$ denotes the parameter space. For each fixed $\boldsymbol{\alpha} \in \boldsymbol{\mathcal{A}}$, the problem instance $\mathscr{P}(\boldsymbol{\alpha})$ is convex if the functions f_i and g_i are convex and concave, respectively, and the functions u_i and v_i are affine. The reader should refer to [11] for an introduction to the basics of convex optimization. In this section we will first illustrate that typical optimization problems in C-RANs can be formulated in this parametric form of convex programming, and then the proposed framework for large-scale convex optimization will be introduced.

7.2.1 Convex Optimization Examples in C-RANs

To illustrate the wide-ranging applications of convex optimization in C-RANs, we mainly focus on the generic scenario for downlink transmission with full cooperation among the RRHs. The proposed methodology in this chapter can be easily applied to uplink transmission and more general cooperation scenarios in heterogeneous C-RANs [4, 8], Fog-RANs, and MENG-RANs [5] as we need only exploit the convexity of the resulting optimization problems.

Signal Model

Consider a downlink fully cooperative C-RAN with L multi-antenna RRHs and K single-antenna MUs, where the lth RRH is equipped with N_l antennas. The wireless propagation channel from the lth RRH to the kth MU is denoted as $\mathbf{h}_{kl} \in \mathbb{C}^{N_l}, \forall k, l$. The received signal $y_k \in \mathbb{C}$ at MU k is given by

$$y_k = \sum_{l=1}^{L} \mathbf{h}_{kl}^H \mathbf{v}_{lk} s_k + \sum_{i \ne k} \sum_{l=1}^{L} \mathbf{h}_{kl}^H \mathbf{v}_{li} s_i + n_k, \quad \forall k, \qquad (7.3)$$

where s_k is the encoded information symbol for MU k with $\mathbb{E}\{|s_k|^2\} = 1$, $\mathbf{v}_{lk} \in \mathbb{C}^{N_l}$ is the transmit beamforming vector from the lth RRH to the kth MU, and $n_k \sim \mathcal{CN}(0, \sigma_k^2)$ is the additive Gaussian noise at MU k.

We assume that the s_k and n_k are mutually independent and that all the users apply single-user detection. Therefore, the signal-to-interference-plus-noise ratio (SINR) of MU k is given by

$$\text{SINR}_k(\mathbf{v}_1, \ldots, \mathbf{v}_K) = \frac{|\mathbf{h}_k^H \mathbf{v}_k|^2}{\sum_{i \ne k} |\mathbf{h}_k^H \mathbf{v}_i|^2 + \sigma_k^2}, \quad \forall k, \qquad (7.4)$$

where $\mathbf{h}_k \triangleq [\mathbf{h}_{k1}^T \cdots \mathbf{h}_{kL}^T]^T \in \mathbb{C}^N$, with $N = \sum_{l=1}^L N_l$, is the channel vector consisting of the channel coefficients from all the RRHs to the kth MU and $\mathbf{v}_k \triangleq [\mathbf{v}_{1k}^T \mathbf{v}_{2k}^T \cdots \mathbf{v}_{Lk}^T]^T \in \mathbb{C}^N$ is the beamforming vector consisting of the beamforming coefficients from all the RRHs to MU k.

Coordinated beamforming is an efficient way to design energy-efficient and spectrally efficient systems [10] in which the beamforming vectors $\mathbf{v}_{lk}$ are designed to minimize the network power consumption and maximize the network utility, respectively. Three representative examples are given below to illustrate the power of convex optimization to design efficient transmit strategies to optimize the system performance of C-RANs, for which coordinated beamforming is the basic building block.

Example 7.1 Coordinated beamforming via second-order cone programming Consider the following coordinated beamforming problem to, that of minimizing the total transmit power while satisfying the QoS requirements and the transmit power constraints for RRHs [14]:

$$\begin{aligned} \underset{\mathbf{v}_1,\ldots,\mathbf{v}_K}{\text{minimize}} \quad & \sum_{l=1}^{L}\sum_{k=1}^{K} \|\mathbf{v}_{lk}\|_2^2 \\ \text{s. t.} \quad & \frac{|\mathbf{h}_k^H \mathbf{v}_k|^2}{\sum_{i\neq k} |\mathbf{h}_k^H \mathbf{v}_i|^2 + \sigma_k^2} \geq \gamma_k, \quad k = 1,\ldots,K, \qquad (7.5a) \\ & \sum_{k=1}^{K} \|\mathbf{v}_{lk}\|_2^2 \leq P_l, \quad l = 1,\ldots,L, \qquad (7.5b) \end{aligned}$$

where $\gamma_k > 0$ is the target SINR for MU k and $P_l > 0$ is the maximum transmit power of the lth RRH.

Since the phases of $\mathbf{v}_k$ will not change the objective function or constraints of problem (7.5) [35], the SINR constraints (7.5a) are equivalent to the following second-order cone constraints:

$$\sqrt{\sum_{i\neq k} |\mathbf{h}_k^H \mathbf{v}_i|^2 + \sigma_k^2} \leq \frac{1}{\sqrt{\gamma_k}} \Re(\mathbf{h}_k^H \mathbf{v}_k), \quad k = 1,\ldots,K, \qquad (7.6)$$

where $\Re(\cdot)$ denotes the real part. Therefore, problem (7.5) can be reformulated as the following (SOCP) problem:

$$\begin{aligned} \underset{\mathbf{v}_1,\ldots,\mathbf{v}_K}{\text{minimize}} \quad & \sum_{l=1}^{L}\sum_{k=1}^{K} \|\mathbf{v}_{lk}\|_2^2 \\ \text{s. t.} \quad & \sqrt{\sum_{i\neq k} |\mathbf{h}_k^H \mathbf{v}_i|^2 + \sigma_k^2} \leq \frac{1}{\sqrt{\gamma_k}} \Re(\mathbf{h}_k^H \mathbf{v}_k), \quad k = 1,\ldots,K, \qquad (7.7a) \\ & \sqrt{\sum_{k=1}^{K} \|\mathbf{v}_{lk}\|_2^2} \leq P_l, \quad l = 1,\ldots,L. \qquad (7.7b) \end{aligned}$$

Example 7.2 Network power minimization via group sparse beamforming To design a green C-RAN the network power consumption, including the power consumption of each RRH and of each associated fronthaul link, needs to be minimized while satisfying the QoS requirements for all the MUs. Mathematically, we need to solve the following network-power minimization problem [2]:

$$\begin{aligned} \underset{\mathbf{v}_1,\ldots,\mathbf{v}_K}{\text{minimize}} \quad & \sum_{l=1}^{L}\sum_{k=1}^{K}\frac{1}{\eta_l}\|\mathbf{v}_{lk}\|_2^2 + \sum_{l=1}^{L} P_l^c I(\text{Supp}(\mathbf{v})\cap\mathcal{V}_l \neq \emptyset) \\ \text{s. t.} \quad & (7.7a), (7.7b), \end{aligned} \tag{7.8}$$

where $\eta_l > 0$ is the drain-inefficiency coefficient of the radio frequency power amplifier and $P_l^c \geq 0$ is the relative fronthaul-link power consumption [2], representing the static power saving when both the RRH and the corresponding fronthaul link are switched off. Here, $I(\text{Supp}(\mathbf{v})\cap\mathcal{V}_l \neq \emptyset)$ is an indicator function that takes the value 0 if $\text{Supp}(\mathbf{v})\cap\mathcal{V}_l = \emptyset$ (i.e., all the beamforming coefficients at the lth RRH are zeros, indicating that the corresponding RRH is switched off) and 1 otherwise, where $\mathcal{V}_l$ is defined as $\mathcal{V}_l := \{K\sum_{l=1}^{L-1} N_l + 1, \ldots, K\sum_{l=1}^{L} N_l\}$; $\text{Supp}(\mathbf{v})$ is the support of the beamforming vector $\mathbf{v} = [\tilde{\mathbf{v}}_l] \in \mathbb{C}^{KN}$ with $\tilde{\mathbf{v}}_l = [\mathbf{v}_{l1}^T, \ldots, \mathbf{v}_{lK}^T]^T \in \mathbb{C}^{N_l K}$ as the aggregated beamforming vector at RRH l. Problem (7.8) is a mixed combinatorial optimization problem and is NP-hard in general.

Observing that all the beamforming coefficients in the vector $\tilde{\mathbf{v}}_l$ will be set to zero simultaneously when the lth RRH is switched off, the aggregate beamforming vector $\mathbf{v}$ has the group-sparsity structure if multiple RRHs need to be switched off to reduce the network power consumption [36]. A three-stage group sparse beamforming algorithm with polynomial time complexity was thus proposed to minimize the network power consumption by adaptively selecting active RRHs via controlling the group-sparsity structure of the aggregative beamforming vector $\mathbf{v}$. Specifically, in the first stage, the group-sparsity structure of the aggregated beamformer $\mathbf{v}$ is induced by minimizing the following weighted mixed ℓ_1/ℓ_2-norm of $\mathbf{v}$:

$$\begin{aligned} \mathscr{P}_{\text{SOCP}}: \underset{\mathbf{v}_1,\ldots,\mathbf{v}_K}{\text{minimize}} \quad & \sum_{l=1}^{L}\omega_l\|\tilde{\mathbf{v}}_l\|_2 \\ \text{s. t.} \quad & (7.7a), (7.7b), \end{aligned} \tag{7.9}$$

where $\omega_l > 0$ is the corresponding weight for the beamformer coefficient group $\tilde{\mathbf{v}}_l$. On the basis of the (approximate) group sparse beamformer $\mathbf{v}^\star$, which is the optimal solution to the convex SOCP problem $\mathscr{P}_{\text{SOCP}}$, (7.9), in the second stage an RRH selection procedure is performed to switch off some RRHs so as to minimize the network power consumption. In this procedure we need to check whether the remaining RRHs can support the QoS requirements for all the MUs, i.e., to check the feasibility of problem (7.9) given the active RRHs. In the third stage, we need to further minimize the total transmit power with those RRHs determined as active while satisfying the QoS requirements for all the MUs, which is amounts to solving a coordinated beamforming problem, as in Example 7.1. More details on the group sparse beamforming algorithm can be found

in [2]. Extensions on semidefinite programming for network power minimization with imperfect channel state information in multicast C-RANs can be found in [14].

Example 7.3 Stochastic coordinated beamforming via sequential convex approximations In practice, inevitably there will be uncertainty in the available channel state information (CSI). Such uncertainty may originate from various sources, e.g., training-based channel estimation [37], limited feedback [38], delays [39, 40], hardware deficiencies [41], or partial CSI acquisition [16]. The uncertainty in the available CSI brings a new technical challenge for system design. To guarantee performance, we impose a probabilistic QoS system constraint by assuming that the distribution information of the channel knowledge is available. Considering a unicast transmission scenario, the stochastic coordinated beamforming problem is formulated to minimize the total transmit power while satisfying a probabilistic QoS system constraint, as follows [16]:

$$\begin{aligned} \underset{\mathbf{v}_1,\ldots,\mathbf{v}_K}{\text{minimize}} \quad & \sum_{l=1}^{L}\sum_{k=1}^{K}\|\mathbf{v}_{lk}\|^2 \\ \text{s. t.} \quad & \Pr\left\{\frac{|\mathbf{h}_k^H\mathbf{v}_k|^2}{\sum_{i\neq k}|\mathbf{h}_k^H\mathbf{v}_i|^2+\sigma_k^2}\geq\gamma_k,\quad k=1,\ldots,K\right\}\geq 1-\epsilon, \end{aligned} \tag{7.10}$$

where the distribution information for the $\mathbf{h}_k$ is known and $0<\epsilon<1$ indicates that the QoS requirements should be guaranteed for all the MUs simultaneously with probability at least $1-\epsilon$. However, problem (7.10) is a joint chance-constrained program [42] and is known to be intractable in general.

In [16] a stochastic difference-of-convex (DC) programming algorithm was proposed for finding a KKT point by solving the following stochastic convex programming problems iteratively:

$$\begin{aligned} \underset{\mathbf{v}_1,\ldots,\mathbf{v}_K,\kappa>0}{\text{minimize}} \quad & \sum_{l=1}^{L}\sum_{k=1}^{K}\|\mathbf{v}_{lk}\|^2 \\ \text{subject to} \quad & u(\mathbf{v},\kappa)-u(\mathbf{v}^{[j]},0)-2\langle\nabla_{\mathbf{v}^*}u(\mathbf{v}^{[j]},0),\mathbf{v}-\mathbf{v}^{[j]}\rangle\leq\kappa\epsilon, \end{aligned} \tag{7.11}$$

where $u(\mathbf{v},\nu)=\mathbb{E}\left\{\max_{1\leq k\leq K+1}s_k(\mathbf{v},\mathbf{h},\nu)\right\}$ is a convex function with $s_k(\mathbf{v},\mathbf{h},\nu)=\sum_{i\neq k}\mathbf{v}_i^H\mathbf{h}_k\mathbf{h}_k^H\mathbf{v}_i+\sigma_k^2+\sum_{i\neq k}\gamma_i^{-1}\mathbf{v}_i^H\mathbf{h}_i\mathbf{h}_i^H\mathbf{v}_i, k=1,\ldots,K$, and $s_{K+1}(\mathbf{v},\mathbf{h},\nu)=\sum_{i=1}^{K}\gamma_i^{-1}\mathbf{v}_i^H\mathbf{h}_i\mathbf{h}_i^H\mathbf{v}_i$ are convex quadratic functions in $\mathbf{v}$. Here, the complex gradient of $u(\mathbf{v},0)$ with respect to $\mathbf{v}^*$ (the complex conjugate of $\mathbf{v}$) is given by

$$\nabla_{\mathbf{v}^*}u(\mathbf{v},0)=\mathbb{E}\{\nabla_{\mathbf{v}^*}s_{k^\star}(\mathbf{v},\mathbf{h},0)\}, \tag{7.12}$$

where $k^\star=\arg\max_{1\leq k\leq K+1}s_k(\mathbf{v},\mathbf{h},0)$, and $\nabla_{\mathbf{v}^*}s_k(\mathbf{v},\mathbf{h},0)=[\boldsymbol{\nu}_{k,i}]_{1\leq i\leq K}$ $(1\leq k\leq K)$, with $\boldsymbol{\nu}_{k,i}\in\mathbb{C}^N$ given by

$$\boldsymbol{\nu}_{k,i}=\begin{cases}\left(\mathbf{h}_k\mathbf{h}_k^H+\gamma_i^{-1}\mathbf{h}_i\mathbf{h}_i^H\right)\mathbf{v}_i & \text{if } i\neq k, 1\leq k\leq K,\\ 0, & \text{otherwise,}\end{cases} \tag{7.13}$$

and $\nabla_{\mathbf{v}^*} s_{K+1}(\mathbf{v}, \mathbf{h}, \kappa) = [\boldsymbol{\nu}_{K+1,i}]_{1 \le i \le K}$ with $\boldsymbol{\nu}_{K+1,i} = \gamma_i^{-1} \mathbf{h}_i \mathbf{h}_i^H \mathbf{v}_i, \forall i$. Furthermore, the gradient of $u(\mathbf{v}, 0)$ with respect to κ is zero, as $\kappa = 0$ is a constant in the function $u(\mathbf{v}, 0)$.

To solve the stochastic convex programming problem (7.11) efficiently at each iteration, the sample-average approximation method is further adopted; this involves solving the following convex quadratically constrained quadratic program (QCQP):

$$\begin{aligned} \mathscr{P}_{\text{QCQP}} : \underset{\mathbf{v}_1,\ldots,\mathbf{v}_K,\kappa,\mathbf{x}}{\text{minimize}} \quad & \sum_{l=1}^{L} \sum_{k=1}^{K} \|\mathbf{v}_{lk}\|^2 \\ \text{s. t.} \quad & \frac{1}{M} \sum_{m=1}^{M} x_m - \bar{u}(\mathbf{v}^{[j]}, 0) - 2\langle \bar{\nabla}_{\mathbf{v}^*} u(\mathbf{v}^{[j]}, 0), \mathbf{v} - \mathbf{v}^{[j]} \rangle \le \kappa \epsilon, \\ & s_k(\mathbf{v}, \mathbf{h}^m, \kappa) \le x_m, \quad x_m \ge 0, \quad \kappa > 0, \quad \forall k, m. \end{aligned} \tag{7.14}$$

Here, $\mathbf{x} = [x_m]_{1 \le m \le M} \in \mathbb{R}^M$ is the collection of slack variables with M independent realizations of the random vector $\mathbf{h} \in \mathbb{C}^{NK}$.

More examples on applying convex optimization for resource allocation and signal processing in wireless cooperative networks can be found in [8, 9, 31]. In summary, the above examples illustrate that the new architecture of C-RANs brings up new design challenges, while convex optimization can serve as a powerful tool to formulate and solve these problems. Meanwhile, as the problem sizes scale up in C-RANs, it becomes critical to solve the resulting convex optimization problems efficiently.

7.2.2 A Unified Framework for Large-Scale Convex Optimization

As presented previously, a sequence of convex optimization problems $\mathscr{P}$ needs to be solved for typical signal processing and resource allocation problems in C-RANs. In dense C-RANs the BBU pool can support hundreds of RRHs for simultaneous transmission and reception [1]. Therefore, all the resulting convex optimization problems $\mathscr{P}$ are shifted into a new domain with a high problem dimension and a large number of constraints. Although the convex programs $\mathscr{P}$ can be solved in polynomial time using the interior-point method, which is implemented in most advanced off-the-shelf solvers (e.g., public software packages like SeDuMi [17] and SDPT3 [18] and commercial software packages like MOSEK [19]), the computational cost of such second-order methods will be prohibitive for large-scale problems. However, most custom algorithms, e.g., the uplink–downlink approach [10] and the ADMM-based algorithms [13, 26, 28], fail either to scale well to large problem sizes or to detect infeasibility effectively.

To overcome the limitations of the scalability of state-of-art solvers and the capability of infeasibility detection of custom algorithms, in this chapter we propose a two-stage large-scale optimization framework as shown in Fig. 7.1. Specifically, in the first stage the original problem will be transformed into a standard cone programming, thereby providing the capability of infeasibility detection and parallel computing. This will be presented in Section 7.3. In the second stage, the first-order alternating-direction

method of multipliers (ADMM) algorithm [27], i.e., the operator splitting method, will be adopted to solve the large-scale homogeneous self-dual embedding (HSD) system. This will be presented in Section 7.4.

7.3 Matrix Stuffing for Fast Cone-Programming Transformation

Consider the following parametric family $\mathscr{P}_{\text{cone}}$ of primal conic optimization problems:

$$\begin{aligned} \mathscr{P}_{\text{cone}} : \underset{\boldsymbol{v},\boldsymbol{\mu}}{\text{minimize}} \quad & \mathbf{c}^T \boldsymbol{v} \\ \text{s. t.} \quad & \mathbf{A}\boldsymbol{v} + \boldsymbol{\mu} = \mathbf{b}, && \text{(7.15a)} \\ & (\boldsymbol{v}, \boldsymbol{\mu}) \in \mathbb{R}^n \times \mathcal{K}, && \text{(7.15b)} \end{aligned}$$

where $\boldsymbol{v} \in \mathbb{R}^n$ and $\boldsymbol{\mu} \in \mathbb{R}^m$ (with $n \leq m$) are the optimization variables, $\mathbf{A} \in \mathbb{R}^{m \times n}$, $\mathbf{b} \in \mathbb{R}^m$, $\mathbf{c} \in \mathbb{R}^n$, and $\mathcal{K} = \mathcal{K}_1 \times \cdots \times \mathcal{K}_q \in \mathbb{R}^m$ is a Cartesian product of q closed convex cones. Here, each $\mathcal{K}_i$ has dimension m_i such that $\sum_{i=1}^q m_i = m$. Let $\boldsymbol{\beta} = \{\mathbf{A}, \mathbf{b}, \mathbf{c}\} \in \mathcal{D}$ be the problem data with $\mathcal{D}$ as the data space.

Although $\mathcal{K}_i$ is allowed to be any closed convex cone, we are primarily interested in the following symmetric cones:

- the nonnegative reals, $\mathbb{R}_+ = \{x \in \mathbb{R} | x \geq 0\}$;
- a second-order cone, $\mathcal{Q}^d = \{(y, \mathbf{x}) \in \mathbb{R} \times \mathbb{R}^{d-1} | \|\mathbf{x}\| \leq y\}$;
- a positive semidefinite cone, $\mathbf{S}_+^n = \{\mathbf{X} \in \mathbb{R}^{n \times n} | \mathbf{X} = \mathbf{X}^T, \mathbf{X} \succeq \mathbf{0}\}$.

In particular, a problem instance $\mathscr{P}_{\text{cone}}(\boldsymbol{\beta})$ is known as a linear program (LP), SOCP, or SDP if all the cones $\mathcal{K}_i$ are restricted to $\mathbb{R}_+$, $\mathcal{Q}^d$, or $\mathbf{S}_+^n$, respectively. Therefore, almost all the original convex optimization problem family $\mathscr{P}$ can be equivalently transformed into the standard conic optimization problem family $\mathscr{P}_{\text{cone}}$ on the basis of the principle of disciplined convex programming [33]. This equivalence means that the optimal solution or the certificate of infeasibility of the original problem instance $\mathscr{P}(\boldsymbol{\alpha})$ can be extracted from the solution to the corresponding equivalent cone-program instance $\mathscr{P}_{\text{cone}}(\boldsymbol{\beta})$.

To develop a generic large-scale convex optimization framework, instead of exploiting the special structures in the original specific problems $\mathscr{P}$, we propose to work with the transformed equivalent standard conic optimization problems $\mathscr{P}_{\text{cone}}$. This approach bears the following advantages.

1. The convex programs $\mathscr{P}$ can be equivalently transformed into the standard cone programs $\mathscr{P}_{\text{cone}}$. This will be presented in Section 7.3.1.
2. The homogeneous self-dual embedding of the primal–dual pair of the standard cone program $\mathscr{P}_{\text{cone}}$ can be induced, thereby providing certificates of infeasibility. This will be presented in Section 7.4.1.
3. The feasible set in $\mathscr{P}_{\text{cone}}$ is formed by two sets: a subspace constraint (7.15a) and a convex cone $\mathcal{K}$, which is formed by the Cartesian product of smaller convex cones.

This salient feature will be exploited to enable parallel and scalable computing in Section 7.4.2.

7.3.1 Standard Conic Optimization Transformation

Our goal is to map the parameters $\boldsymbol{\alpha}$ in the original convex optimization problem family $\mathscr{P}$ to the problem data $\boldsymbol{\beta}$ in the equivalent standard conic optimization problem family $\mathscr{P}_{\text{cone}}$ with the description of the convex cone $\mathcal{K}$. The general idea of such a transformation is to rewrite the original problem family $\mathscr{P}$ in a Smith form by introducing a new variable for each subexpression in the disciplined convex programming form [33] of the problem family $\mathscr{P}$. In the following we will take the coordinated beamforming problem as an example to illustrate such a transformation.

Example 7.4 Standard cone program for coordinated beamforming Consider the coordinated beamforming problem (7.7) with the objective function as $\|\mathbf{v}\|_2$, which can be rewritten as the following disciplined convex programming form [33]:

$$\begin{aligned} \text{minimize} \quad & \|\mathbf{v}\|_2 \\ \text{s. t.} \quad & \|\mathbf{D}_l\mathbf{v}\|_2 \le \sqrt{P_l}, \quad l = 1, \ldots, L, && \text{(7.16a)} \\ & \|\mathbf{C}_k\mathbf{v} + \mathbf{g}_k\|_2 \le \beta_k \mathbf{r}_k^T \mathbf{v}, \quad k = 1, \ldots, K, && \text{(7.16b)} \end{aligned}$$

where $\beta_k = \sqrt{1 + 1/\gamma_k}$, $\mathbf{D}_l = \text{blk diag}\{\mathbf{D}_l^1, \ldots, \mathbf{D}_l^K\} \in \mathbb{R}^{N_l K \times NK}$ with $\mathbf{D}_l^k = \left[\mathbf{0}_{N_l \times \sum_{i=1}^{l-1} N_i} \; \mathbf{I}_{N_l \times N_l} \; \mathbf{0}_{N_l \times \sum_{i=l+1}^{L} N_i}\right] \in \mathbb{R}^{N_l \times N}$, $\mathbf{g}_k = [\mathbf{0}_K^T \; \sigma_k]^T \in \mathbb{R}^{K+1}$, $\mathbf{r}_k = \left[\mathbf{0}_{(k-1)N}^T, \; \mathbf{h}_k^T \; \mathbf{0}_{(K-k)N}^T\right]^T \in \mathbb{R}^{NK}$, and $\mathbf{C}_k = [\tilde{\mathbf{C}}_k \; \mathbf{0}_{NK}]^T \in \mathbb{R}^{(K+1) \times NK}$ with $\tilde{\mathbf{C}}_k = \text{blk diag}\{\mathbf{h}_k, \ldots, \mathbf{h}_k\} \in \mathbb{R}^{NK \times K}$.

Smith form reformulation To arrive at the standard cone program $\mathscr{P}_{\text{cone}}$, we rewrite problem (7.16) in the following Smith form [32] by introducing a new variable for each subexpression in (7.16):

$$\begin{aligned} \text{minimize} \quad & x_0 \\ \text{s. t.} \quad & \|\mathbf{x}_1\| = x_0, \quad \mathbf{x}_1 = \mathbf{v}, \\ & \mathcal{G}_1(l), \mathcal{G}_2(k), \quad \forall k, l, && \text{(7.17)} \end{aligned}$$

where $\mathcal{G}_1(l)$ is the Smith-form reformulation for the transmit power constraint for RRH l (7.16a), given as follows:

$$\mathcal{G}_1(l): \begin{cases} (y_0^l, \mathbf{y}_1^l) \in \mathcal{Q}^{KN_l+1} \\ y_0^l = \sqrt{P_l} \in \mathbb{R}, \\ \mathbf{y}_1^l = \mathbf{D}_l\mathbf{v} \in \mathbb{R}^{KN_l}, \end{cases} \qquad \text{(7.18)}$$

and $\mathcal{G}_2(k)$ is the Smith-form reformulation for the QoS constraint for MU k (7.16b), given as follows:

$$\mathcal{G}_2(k): \begin{cases} (t_0^k, \mathbf{t}_1^k) \in \mathcal{Q}^{K+1}, \\ t_0^k = \beta_k \mathbf{r}_k^T \mathbf{v} \in \mathbb{R}, \\ \mathbf{t}_1^k = \mathbf{t}_2^k + \mathbf{t}_3^k \in \mathbb{R}^{K+1}, \\ \mathbf{t}_2^k = \mathbf{C}_k \mathbf{v} \in \mathbb{R}^{K+1}, \\ \mathbf{t}_3^k = \mathbf{g}_k \in \mathbb{R}^{K+1}. \end{cases} \tag{7.19}$$

Nevertheless, the Smith form reformulation (7.17) is not convex owing to the non-convex constraint $\|\mathbf{x}_1\| = x_0$. We thus relax this non-convex constraint to $\|\mathbf{x}_1\| \leq x_0$, yielding the following relaxed Smith form:

$$\begin{aligned} &\text{minimize} \quad x_0 \\ &\text{s. t.} \qquad \mathcal{G}_0, \mathcal{G}_1(l), \mathcal{G}_2(k), \quad \forall k, l, \end{aligned} \tag{7.20}$$

where

$$\mathcal{G}_0: \begin{cases} (x_0, \mathbf{x}_1) \in \mathcal{Q}^{NK+1}, \\ \mathbf{x}_1 = \mathbf{v} \in \mathbb{R}^{NK}. \end{cases} \tag{7.21}$$

It can be easily proved that the constraint $\|\mathbf{x}_1\| \leq x_0$ has to be active at the optimal solution, otherwise we could always scale down x_0 in such a way that the cost function is further minimized while still satisfying the constraints. Therefore, we conclude that the relaxed Smith form (7.20) is equivalent to the original problem (7.16).

Conic reformulation Now the relaxed Smith-form reformulation (7.20) is readily reformulated as the standard cone-programming form $\mathscr{P}_{\text{cone}}$. Specifically, define optimization variables $[x_0; \mathbf{v}]$ with the same type of equations as in $\mathcal{G}_0$; then $\mathcal{G}_0$ can be rewritten as

$$\mathbf{M}[x_0; \mathbf{v}] + \boldsymbol{\mu}_0 = \mathbf{m}, \tag{7.22}$$

where the slack variables $\boldsymbol{\mu}_0$ belong to the following convex set,

$$\boldsymbol{\mu}_0 \in \mathcal{Q}^{NK+1}, \tag{7.23}$$

and $\mathbf{M} \in \mathbb{R}^{(NK+1)\times(NK+1)}$ and $\mathbf{m} \in \mathbb{R}^{NK+1}$ are given as follows:

$$\mathbf{M} = \left[\begin{array}{c|c} -1 & \\ \hline & -\mathbf{I}_{NK} \end{array}\right], \quad \mathbf{m} = \left[\begin{array}{c} 0 \\ \hline \mathbf{0}_{NK} \end{array}\right], \tag{7.24}$$

respectively. Now define optimization variables $[y_0^l; \mathbf{v}]$ with the same type of equations as in $\mathcal{G}_1(l)$; then $\mathcal{G}_1(l)$ can be rewritten as

$$\mathbf{P}^l[y_0^l; \mathbf{v}] + \boldsymbol{\mu}_1^l = \mathbf{p}^l, \tag{7.25}$$

where the slack variables $\boldsymbol{\mu}_1^l \in \mathbb{R}^{KN_l+2}$ belongs to the following convex set formed by the Cartesian product of two convex sets,

$$\boldsymbol{\mu}_1^l \in \mathcal{Q}^1 \times \mathcal{Q}^{KN_l+1}, \tag{7.26}$$

and $\mathbf{P}^l \in \mathbb{R}^{(KN_l+2)\times(NK+1)}$ and $\mathbf{p}^l \in \mathbb{R}^{KN_l+2}$ are given as follows:

$$\mathbf{P}^l = \left[\begin{array}{c|c} 1 & \\ \hline -1 & \\ & -\mathbf{D}_l \end{array}\right], \quad \mathbf{p}^l = \left[\begin{array}{c} \sqrt{P_l} \\ \hline 0 \\ \hline \mathbf{0}_{KN_l} \end{array}\right], \tag{7.27}$$

respectively. Next, define optimization variables $[t_0^k; \mathbf{v}]$ with the same type of equations as in $\mathcal{G}_2(k)$; then $\mathcal{G}_2(k)$ can be rewritten as

$$\mathbf{Q}^k[t_0^k; \mathbf{v}] + \boldsymbol{\mu}_2^k = \mathbf{q}^k, \tag{7.28}$$

where the slack variables $\boldsymbol{\mu}_2^k \in \mathbb{R}^{K+3}$ belong to the following convex set formed by the Cartesian product of two convex sets,

$$\boldsymbol{\mu}_2^k \in \mathcal{Q}^1 \times \mathcal{Q}^{K+2}, \tag{7.29}$$

and $\mathbf{Q}^k \in \mathbb{R}^{(K+3)\times(NK+1)}$ and $\mathbf{q}^k \in \mathbb{R}^{K+3}$ are given as follows:

$$\mathbf{Q}^k = \left[\begin{array}{c|c} 1 & -\beta_k \mathbf{r}_k^T \\ \hline -1 & \\ & -\mathbf{C}_k \end{array}\right], \quad \mathbf{q}^k = \left[\begin{array}{c} 0 \\ \hline 0 \\ \hline \mathbf{g}_k \end{array}\right], \tag{7.30}$$

respectively.

Therefore, we arrive at the standard form $\mathscr{P}_{\text{cone}}$ by writing the optimization variables $\boldsymbol{v} \in \mathbb{R}^n$ as follows:

$$\boldsymbol{v} = [x_0; y_0^1; \ldots; y_0^L; t_0^1; \ldots; t_0^K; \mathbf{v}], \tag{7.31}$$

and $\mathbf{c} = [1; \mathbf{0}_{n-1}]$. The structure of the standard cone-programming $\mathscr{P}_{\text{cone}}$ is characterized by the following data:

$$n = 1 + L + K + NK, \tag{7.32}$$

$$m = (L + K) + (NK + 1) + \sum_{l=1}^{L}(KN_l + 1) + K(K + 2), \tag{7.33}$$

$$\mathcal{K} = \underbrace{\mathcal{Q}^1 \times \cdots \times \mathcal{Q}^1}_{L+K \text{ times}} \times \mathcal{Q}^{NK+1} \times \underbrace{\mathcal{Q}^{KN_1+1} \times \cdots \times \mathcal{Q}^{KN_L+1}}_{L \text{ times}} \times \underbrace{\mathcal{Q}^{K+2} \times \cdots \times \mathcal{Q}^{K+2}}_{K \text{ times}}, \tag{7.34}$$

where $\mathcal{K}$ is the Cartesian product of $2(L+K)+1$ second-order 1s, and $\mathbf{A}$ and $\mathbf{b}$ are given as follows:

$$\mathbf{A} = \left[\begin{array}{c|c|c|c} & \begin{matrix}1 & & \\ & \ddots & \\ & & 1\end{matrix} & & \\ \hline & & \begin{matrix}1 & \\ & \ddots \\ & 1\end{matrix} & \begin{matrix}-\beta_1\mathbf{r}_1^T \\ \vdots \\ -\beta_K\mathbf{r}_K^T\end{matrix} \\ \hline \begin{matrix}-1\end{matrix} & & & -\mathbf{I}_{NK} \\ \hline & \begin{matrix}-1\end{matrix} & & -\mathbf{D}_1 \\ \hline & \begin{matrix}\vdots\end{matrix} & \begin{matrix}\vdots\end{matrix} & \begin{matrix}\vdots\end{matrix} \\ \hline & \begin{matrix}-1\end{matrix} & & -\mathbf{D}_L \\ \hline & & \begin{matrix}-1\end{matrix} & -\mathbf{C}_1 \\ \hline & & \begin{matrix}\vdots\end{matrix} & \begin{matrix}\vdots\end{matrix} \\ \hline & & \begin{matrix}-1\end{matrix} & -\mathbf{C}_K \end{array}\right], \quad \mathbf{b} = \left[\begin{array}{c} \sqrt{P_1} \\ \vdots \\ \sqrt{P_L} \\ \hline 0 \\ \vdots \\ 0 \\ \hline 0 \\ \mathbf{0}_{NK} \\ \hline 0 \\ \mathbf{0}_{KN_1} \\ \hline \vdots \\ 0 \\ \mathbf{0}_{KN_1} \\ \hline 0 \\ \mathbf{g}_1 \\ \hline \vdots \\ 0 \\ \mathbf{g}_K \end{array}\right]. \tag{7.35}$$

Extension to the complex case For $\mathbf{h}_k \in \mathbb{C}^N, \mathbf{v}_i \in \mathbb{C}^N$, we have

$$\mathbf{h}_k^H \mathbf{v}_i \Longrightarrow \underbrace{\begin{bmatrix} \Re(\mathbf{h}_k) & -\Im(\mathbf{h}_k) \\ \Im(\mathbf{h}_k) & \Re(\mathbf{h}_k) \end{bmatrix}^T}_{\tilde{\mathbf{h}}_k} \underbrace{\begin{bmatrix} \Re(\mathbf{v}_i) \\ \Im(\mathbf{v}_i) \end{bmatrix}}_{\tilde{\mathbf{v}}_i}, \tag{7.36}$$

where $\tilde{\mathbf{h}}_k \in \mathbb{R}^{2N\times 2}$ and $\tilde{\mathbf{v}}_i \in \mathbb{R}^{2N}$. Therefore, the complex-field problem can be changed into a real-field problem by the transformations $\mathbf{h}_k \Rightarrow \tilde{\mathbf{h}}_k$ and $\mathbf{v}_i \Rightarrow \tilde{\mathbf{v}}_i$.

7.3.2 Matrix Stuffing for Fast Transformation

Inspired by the recent work [23] on fast-optimization code deployment for embedding a second-order cone program, we propose to use the matrix stuffing technique [23, 24] to transform the original problem instance $\mathscr{P}(\boldsymbol{\alpha})$ into the standard cone-program instance $\mathscr{P}_{\text{cone}}(\boldsymbol{\beta})$ quickly. Specifically, for any given network size we first generate and store the structure that maps the original problem family $\mathscr{P}$ to the standard conic optimization problem family $\mathscr{P}_{\text{cone}}$. Thus, the pre-stored standard-form structure includes the problem dimensions (i.e., m and n), the description of $\mathcal{V}$ (i.e., the array of the cone sizes $[m_1, m_2, \ldots, m_q]$), and the symbolic problem parameters $\boldsymbol{\beta} = \{\mathbf{A}, \mathbf{b}, \mathbf{c}\}$. This procedure

can be done offline. Furthermore, to reduce the storage and memory overhead we store the problem data $\boldsymbol{\beta}$ in a sparse form [43] by storing only the non-zero entries.

Using the pre-stored structure, for a given problem instance $\mathscr{P}(\boldsymbol{\alpha})$ we need only copy its parameters $\boldsymbol{\alpha}$ to the corresponding problem data $\boldsymbol{\beta}$ in the standard conic-optimization problem family $\mathscr{P}_{\text{cone}}$. As the procedure for transformation needs only to copy memory, it thus is suitable for fast transformation and can avoid repeated parsing and generating as in parser/solver modeling frameworks like CVX [21] and YALMIP [22].

Example 7.5 Matrix stuffing for the coordinated beamforming problem For the convex coordinated beamforming problem (7.16), to arrive at the standard cone program form $\mathscr{P}_{\text{cone}}(\boldsymbol{\beta})$, we need only copy the parameters of the transmit power constraints P_l to the data of the standard form, i.e., for the $\sqrt{P_l}$ in $\mathbf{b}$, copy the parameters of the SINR thresholds $\boldsymbol{\gamma}$ to the data of the standard form, i.e., the β_k in $\mathbf{A}$, and copy the parameters of the channel realizations $\mathbf{h}_k$ to the data of the standard form, i.e., the $\mathbf{r}_k$ and $\mathbf{C}_k$ in $\mathbf{A}$. As we need to copy the memory only for the transformation, this procedure can be very efficient compared with state-of-the-art numerical-based modeling frameworks such as CVX.

7.3.3 Practical Implementation Issues

We have presented a systematic way to equivalently transform the original problems $\mathscr{P}$ to standard conic optimization problems $\mathscr{P}_{\text{cone}}$. The resultant structure that maps the original problem to the standard form can be stored and reused for fast transforming via matrix stuffing. This can significantly reduce the modeling overhead compared with the parser/solver modeling frameworks such as CVX. However, it requires tedious manual work to find the mapping, and it may not be easy to verify its correctness. Chu *et al.* [23] made an attempt that was intended to automatically generate the code for matrix stuffing. However, so far the corresponding software package QCML [23] is far from complete and may not be suitable for our applications. Extending numerically based transformation modeling frameworks like CVX to symbolically based transformation modeling frameworks like QCML is non-trivial and requires tremendous mathematical and technical effort.

7.4 Operator Splitting for Large-Scale Homogeneous Self-Dual Embedding

Although the standard cone program $\mathscr{P}_{\text{cone}}$ itself is suitable for parallel computing via the operator splitting method [44], directly working on this problem may fail to provide certificates of infeasibility. To address this limitation, on the basis of the recent work by O'Donoghue *et al.* [45] we propose to solve the homogeneous self-dual embedding [34] of the primal–dual pair of the cone program $\mathscr{P}_{\text{cone}}$. The resultant homogeneous

self-dual embedding is further solved via the operator splitting method, also known as the ADMM algorithm [27].

7.4.1 Homogeneous Self-Dual Embedding of Cone Programming

The basic idea of homogeneous self-dual embedding is to embed the primal and dual problems of the cone program $\mathscr{P}_{\text{cone}}$ into a single feasibility problem (i.e., finding a feasible point of the intersection of a subspace and a convex set) such that either the optimal solution or the certificate of infeasibility of the original cone program $\mathscr{P}_{\text{cone}}$ can be extracted from the solution of the embedded problem.

The dual problem of $\mathscr{P}_{\text{cone}}$ is given by [45]

$$\begin{aligned} \mathscr{D}_{\text{cone}}: \underset{\boldsymbol{\eta},\boldsymbol{\lambda}}{\text{maximize}} \quad & -\mathbf{b}^T\boldsymbol{\eta} \\ \text{s. t.} \quad & -\mathbf{A}^T\boldsymbol{\eta} + \boldsymbol{\lambda} = \mathbf{c}, \\ & (\boldsymbol{\lambda}, \boldsymbol{\eta}) \in \{0\}^n \times \mathcal{K}^*, \end{aligned} \tag{7.37}$$

where $\boldsymbol{\lambda} \in \mathbb{R}^n$ and $\boldsymbol{\eta} \in \mathbb{R}^m$ are the dual variables and $\mathcal{K}^*$ is the dual cone of the convex cone $\mathcal{K}$. Define the optimal values of the primal program $\mathscr{P}_{\text{cone}}$ and dual program $\mathscr{D}_{\text{cone}}$ as $p^\star$ and $d^\star$, respectively. Let $p^\star = +\infty$ and $p^\star = -\infty$ indicate primal infeasibility and unboundedness, respectively. Similarly, let $d^\star = -\infty$ and $d^\star = +\infty$ indicate dual infeasibility and unboundedness, respectively. We assume strong duality for the convex cone program $\mathscr{P}_{\text{cone}}$, i.e., $p^\star = d^\star$, including the cases when they are infinite. This is a standard assumption made when practically designing solvers for conic programs, e.g., it is assumed in [17–19, 34, 45]. Besides this, we do not make any regularity assumption on the feasibility and boundedness assumptions on the primal and dual problems.

Certificates of Infeasibility

Given the cone program $\mathscr{P}_{\text{cone}}$, a main task is to detect feasibility. However, most existing custom algorithms assume that the original problem $\mathscr{P}$ is feasible [10] or provide heuristic ways to handle infeasibility [26]. Nevertheless, the only way to detect infeasibility effectively is to provide a certificate or proof of infeasibility, as presented in the following proposition.

PROPOSITION 7.1 (Certificates of infeasibility) The following system,

$$\mathbf{A}\boldsymbol{v} + \boldsymbol{\mu} = \mathbf{b}, \quad \boldsymbol{\mu} \in \mathcal{K}, \tag{7.38}$$

is infeasible if and only if the following system is feasible

$$\mathbf{A}^T\boldsymbol{\eta} = \mathbf{0}, \quad \boldsymbol{\eta} \in \mathcal{K}^\star, \quad \mathbf{b}^T\boldsymbol{\eta} < 0. \tag{7.39}$$

Therefore, any dual variable $\boldsymbol{\eta}$ satisfying the system (7.39) provides a certificate or proof that the primal program $\mathscr{P}_{\text{cone}}$ (equivalently, the original problem $\mathscr{P}$) is infeasible.

Similarly, any primal variable $\boldsymbol{v}$ satisfying the system

$$-\mathbf{A}\boldsymbol{v} \in \mathcal{K}, \quad \mathbf{c}^T\boldsymbol{v} < 0, \tag{7.40}$$

is a certificate of the infeasibility of the dual program $\mathscr{D}_{\text{cone}}$.

Proof This result follows directly from the theorem of strong alternatives [11, Section 5.8.2]. □

Optimality Conditions

If the transformed standard cone program $\mathscr{P}_{\text{cone}}$ is feasible then $(\boldsymbol{\nu}^\star, \boldsymbol{\mu}^\star, \boldsymbol{\lambda}^\star, \boldsymbol{\eta}^\star)$ are optimal if and only if they satisfy the following Karush–Kuhn–Tucker (KKT) conditions:

$$\mathbf{A}\boldsymbol{\nu}^\star + \boldsymbol{\mu}^\star - \mathbf{b} = \mathbf{0}, \tag{7.41}$$

$$\mathbf{A}^T\boldsymbol{\eta}^\star - \boldsymbol{\lambda}^\star + \mathbf{c} = \mathbf{0}, \tag{7.42}$$

$$(\boldsymbol{\eta}^\star)^T\boldsymbol{\mu}^\star = 0, \tag{7.43}$$

$$(\boldsymbol{\nu}^\star, \boldsymbol{\mu}^\star, \boldsymbol{\lambda}^\star, \boldsymbol{\eta}^\star) \in \mathbb{R}^n \times \mathcal{K} \times \{0\}^n \times \mathcal{K}^*. \tag{7.44}$$

In particular, the complementary slackness condition (7.43) can be rewritten as

$$\mathbf{c}^T\boldsymbol{\nu}^\star + \mathbf{b}^T\boldsymbol{\eta}^\star = 0, \tag{7.45}$$

which explicitly forces the duality gap to be zero.

Homogeneous Self-Dual Embedding

We can first detect feasibility by Proposition 7.1 and then solve the KKT system if the problem is feasible and bounded. However, the disadvantage of such a two-phase method is that two related problems (i.e., checking feasibility and solving KKT conditions) need to be solved sequentially [34]. To avoid such inefficiency, we propose to solve the following homogeneous self-dual embedding [34]:

$$\mathbf{A}\boldsymbol{\nu} + \boldsymbol{\mu} - \mathbf{b}\tau = \mathbf{0}, \tag{7.46}$$

$$\mathbf{A}^T\boldsymbol{\eta} - \boldsymbol{\lambda} + \mathbf{c}\tau = \mathbf{0}, \tag{7.47}$$

$$\mathbf{c}^T\boldsymbol{\nu} + \mathbf{b}^T\boldsymbol{\eta} + \kappa = \mathbf{0}, \tag{7.48}$$

$$(\boldsymbol{\nu}, \boldsymbol{\mu}, \boldsymbol{\lambda}, \boldsymbol{\eta}, \tau, \kappa) \in \mathbb{R}^n \times \mathcal{K} \times \{0\}^n \times \mathcal{K}^* \times \mathbb{R}_+ \times \mathbb{R}_+, \tag{7.49}$$

which embeds all the information on infeasibility and optimality into a single system by introducing two new nonnegative variables τ and κ, which encode different outcomes. Thus the homogeneous self-dual embedding can be rewritten in the following compact form:

$$\begin{aligned} \mathscr{F}_{\text{HSD}}: \text{find} \quad & (\mathbf{x}, \mathbf{y}) \\ \text{s. t.} \quad & \mathbf{y} = \mathbf{Q}\mathbf{x}, \\ & \mathbf{x} \in \mathcal{C}, \quad \mathbf{y} \in \mathcal{C}^*, \end{aligned} \tag{7.50}$$

where

$$\underbrace{\begin{bmatrix} \boldsymbol{\lambda} \\ \boldsymbol{\mu} \\ \boldsymbol{\kappa} \end{bmatrix}}_{\mathbf{y}} = \underbrace{\begin{bmatrix} \mathbf{0} & \mathbf{A}^T & \mathbf{c} \\ -\mathbf{A} & \mathbf{0} & \mathbf{b} \\ -\mathbf{c}^T & -\mathbf{b}^T & \mathbf{0} \end{bmatrix}}_{\mathbf{Q}} \underbrace{\begin{bmatrix} \boldsymbol{\nu} \\ \boldsymbol{\eta} \\ \tau \end{bmatrix}}_{\mathbf{x}}. \tag{7.51}$$

In (7.51), $\mathbf{x} \in \mathbb{R}^{m+n+1}$, $\mathbf{y} \in \mathbb{R}^{m+n+1}$, $\mathbf{Q} \in \mathbb{R}^{(m+n+1)\times(m+n+1)}$, $\mathcal{C} = \mathbb{R}^n \times \mathcal{K}^* \times \mathbb{R}_+$, and $\mathcal{C}^* = \{0\}^n \times \mathcal{K} \times \mathbb{R}_+$. This system has a trivial solution with all variables as zeros.

The homogeneous self-dual embedding problem $\mathscr{F}_{\text{HSD}}$ is thus a feasibility problem finding a non-zero solution in the intersection of a subspace and a convex cone. Let $(\boldsymbol{\nu}, \boldsymbol{\mu}, \boldsymbol{\lambda}, \boldsymbol{\eta}, \tau, \kappa)$ be a non-zero solution of the homogeneous self-dual embedding. We then have the following remarkable trichotomy, derived in [34]:

- **Case 1**, $\tau > 0, \kappa = 0$, then

$$\hat{\boldsymbol{\nu}} = \boldsymbol{\nu}/\tau, \quad \hat{\boldsymbol{\eta}} = \boldsymbol{\eta}/\tau, \quad \hat{\boldsymbol{\mu}} = \boldsymbol{\mu}/\tau \tag{7.52}$$

  are the primal and dual solutions to the cone program $\mathscr{P}_{\text{cone}}$.
- **Case 2**, $\tau = 0, \kappa > 0$; this implies that $\mathbf{c}^T\boldsymbol{\nu} + \mathbf{b}^T\boldsymbol{\eta} < 0$. Then
  1. If $\mathbf{b}^T\boldsymbol{\eta} < 0$ then $\hat{\boldsymbol{\eta}} = \boldsymbol{\eta}/(-\mathbf{b}^T\boldsymbol{\eta})$ is a certificate of primal infeasibility, as

$$\mathbf{A}^T\hat{\boldsymbol{\eta}} = \mathbf{0}, \quad \hat{\boldsymbol{\eta}} \in \mathcal{V}^\star, \quad \mathbf{b}^T\hat{\boldsymbol{\eta}} = -1. \tag{7.53}$$

  2. If $\mathbf{c}^T\boldsymbol{\nu} < 0$ then $\hat{\boldsymbol{\nu}} = \boldsymbol{\nu}/(-\mathbf{c}^T\hat{\boldsymbol{\nu}})$ is a certificate of dual infeasibility, as

$$-\mathbf{A}\hat{\boldsymbol{\nu}} \in \mathcal{V}, \quad \mathbf{c}^T\hat{\boldsymbol{\nu}} = -1. \tag{7.54}$$

- **Case 3**, $\tau = \kappa = 0$; no conclusion can be made about the cone problem $\mathscr{P}_{\text{cone}}$.

Therefore, from the solution to the homogeneous self-dual embedding, we can extract either the optimal solution (based on (7.31)) or the certificate of infeasibility for the original problem. Furthermore, as the set (7.49) is a Cartesian product of a finite number of sets, this will enable parallelizable algorithm design. With these distinct advantages of homogeneous self-dual embedding, in the sequel we focus on developing efficient algorithms to solve the large-scale feasibility problem $\mathscr{F}_{\text{HSD}}$ via the operator splitting method.

7.4.2 The Operator Splitting Method

Conventionally, the convex homogeneous self-dual embedding $\mathscr{F}_{\text{HSD}}$ can be solved via the interior-point method, e.g. [17–19, 34]. However, the computational cost of such a second-order method can still be prohibitive for large-scale problems. Instead, O'Donoghue *et al.* [45] developed a first-order optimization algorithm based on the operator splitting method, i.e., the ADMM algorithm [27], to solve a large-scale homogeneous self-dual embedding. The key observation is that the convex cone constraint in $\mathscr{F}_{\text{HSD}}$ is the Cartesian product of smaller standard convex cones (i.e., second-order cones, semidefinite cones, and nonnegative reals), which enables parallelizable computing.

Specifically, the homogeneous self-dual embedding problem $\mathscr{F}_{\text{HSD}}$ can be rewritten as

$$\text{minimize} \quad I_{\mathcal{C}\times\mathcal{C}^*}(\mathbf{x}, \mathbf{y}) + I_{\mathbf{Qx}=\mathbf{y}}(\mathbf{x}, \mathbf{y}), \tag{7.55}$$

where $I_{\mathcal{S}}$ is the indicator function of the set $\mathcal{S}$, i.e., $I_{\mathcal{S}}(z)$ is zero for $z \in \mathcal{S}$ and $+\infty$ otherwise. By replicating the variables $\mathbf{x}$ and $\mathbf{y}$, problem (7.55) can be transformed into the following consensus form [27, Section 7.1]:

$$\begin{aligned} \mathscr{P}_{\text{ADMM}}: \text{minimize} \quad & I_{\mathcal{C}\times\mathcal{C}^*}(\mathbf{x},\mathbf{y}) + I_{\mathbf{Q}\tilde{\mathbf{x}}=\tilde{\mathbf{y}}}(\tilde{\mathbf{x}},\tilde{\mathbf{y}}) \\ \text{s. t.} \quad & (\mathbf{x},\mathbf{y}) = (\tilde{\mathbf{x}},\tilde{\mathbf{y}}), \end{aligned} \tag{7.56}$$

which is readily solved by the operator splitting method.

Applying the ADMM algorithm [27, Section 3.1] to the problem $\mathscr{P}_{\text{ADMM}}$ and eliminating the dual variables by exploiting the self-dual property of the problem $\mathscr{F}_{\text{HSD}}$ (refer to [45, Section 3] on how to simplify the ADMM algorithm), the final algorithm is obtained as follows:

$$\mathcal{OS}_{\text{ADMM}}: \begin{cases} \tilde{\mathbf{x}}^{[i+1]} = (\mathbf{I}+\mathbf{Q})^{-1}(\mathbf{x}^{[i]} + \mathbf{y}^{[i]}), \\ \mathbf{x}^{[i+1]} = \Pi_{\mathcal{C}}(\tilde{\mathbf{x}}^{[i+1]} - \mathbf{y}^{[i]}), \\ \mathbf{y}^{[i+1]} = \mathbf{y}^{[i]} - \tilde{\mathbf{x}}^{[i+1]} + \mathbf{x}^{[i+1]}, \end{cases} \tag{7.57}$$

where $\Pi_{\mathcal{C}}(\mathbf{x})$ denotes the Euclidean projection of $\mathbf{x}$ onto the set $\mathcal{C}$. This algorithm has an $O(1/k)$ convergence rate [46] with k as the iteration counter (i.e., ϵ-accuracy can be achieved in $O(1/\epsilon)$ iterations) and will not converge to zero if a non-zero solution exists [45, Section 3.4]. Empirically, this algorithm can converge to modest accuracy within a reasonable amount of time. As the last step is computationally trivial, in the sequel we will focus on how to solve the first two steps efficiently.

7.4.3 Subspace Projection Algorithms

The first step in the algorithm $\mathcal{OS}_{\text{ADMM}}$ is a subspace projection. After simplification [45, Section 4], we essentially need to solve the following linear equation at each iteration, i.e.,

$$\underbrace{\begin{bmatrix} \mathbf{I} & -\mathbf{A}^T \\ -\mathbf{A} & -\mathbf{I} \end{bmatrix}}_{\mathbf{S}} \underbrace{\begin{bmatrix} \boldsymbol{v} \\ -\boldsymbol{\eta} \end{bmatrix}}_{\mathbf{x}} = \underbrace{\begin{bmatrix} \boldsymbol{v}^{[i]} \\ \boldsymbol{\eta}^{[i]} \end{bmatrix}}_{\mathbf{b}}, \tag{7.58}$$

for the given $\boldsymbol{v}^{[i]}$ and $\boldsymbol{\eta}^{[i]}$ at iteration i, where $\mathbf{S} \in \mathbb{R}^{d\times d}$ with $d = m + n$ is a *symmetric quasidefinite matrix* [47]. Several approaches will be presented to solve the large-scale linear system (7.58) efficiently, i.e., so as to trade off the solving time and accuracy.

Factorization Caching Approach

To enable quicker inversions and reduce memory overhead via exploiting the sparsity of the matrix $\mathbf{S}$, the method of sparse permuted LDL^T factorization [43] can be adopted. Specifically, such a factor-solve method can be carried out by first computing the sparse permuted LDL^T factorization as follows:

$$\mathbf{S} = \mathbf{P}\mathbf{L}\mathbf{D}\mathbf{L}^T\mathbf{P}^T, \tag{7.59}$$

where $\mathbf{L}$ is a lower-triangular matrix, $\mathbf{D}$ is a diagonal matrix [44], and $\mathbf{P}$ with $\mathbf{P}^{-1} = \mathbf{P}^T$ is a permutation matrix to fill in the factorization [43], i.e., the non-zero entries in $\mathbf{L}$.

Such a factorization exists for any permutation $\mathbf{P}$, as the matrix $\mathbf{S}$ is symmetric quasidefinite [47, Theorem 2.1]. Computing the factorization costs much less than $\mathcal{O}(1/3d^3)$ flops, while the exact value depends on d and the sparsity pattern of $\mathbf{S}$ in a complicated way. Note that this factorization needs to be computed only once, in the first iteration, and can be cached for reusing in the sequent iterations for subspace projections. This is called the *factorization caching* technique [45].

Given the cached factorization (7.59), solving subsequent projections $\mathbf{x} = \mathbf{S}^{-1}\mathbf{b}$ (7.58) can be carried out by solving the following much easier equations,

$$\mathbf{P}\mathbf{x}_1 = \mathbf{b}, \quad \mathbf{L}\mathbf{x}_2 = \mathbf{x}_1, \quad \mathbf{D}\mathbf{x}_3 = \mathbf{x}_2, \quad \mathbf{L}^T\mathbf{x}_4 = \mathbf{x}_3, \quad \mathbf{P}^T\mathbf{x} = \mathbf{x}_4, \tag{7.60}$$

which cost respectively zero flops, $\mathcal{O}(sd)$ flops by forward substitution with s as the number of non-zero entries in $\mathbf{L}$, $\mathcal{O}(d)$ flops, $\mathcal{O}(sd)$ flops by backward substitution, and zero flops, respectively [11, Appendix C].

Approximate Approaches

To scale the linear system (7.58) to large problem sizes for, approximate algorithms can be adopted to trade off the accuracy of the solution and the solving time. We first rewrite (7.58) as follows:

$$\boldsymbol{\nu} = (\mathbf{I} + \mathbf{A}^T\mathbf{A})^{-1}(\boldsymbol{\nu}^{[i]} - \mathbf{A}^T\boldsymbol{\eta}^{[i]}), \tag{7.61}$$

$$\boldsymbol{\eta} = \boldsymbol{\eta}^{[i]} + \mathbf{A}\boldsymbol{\nu}. \tag{7.62}$$

The conjugate gradient method [48] is then applied to find an approximation to the above linear system. Specifically, let $\mathbf{G} = \mathbf{I} + \mathbf{A}^T\mathbf{A}$. The conjugate gradient algorithm to find $\mathbf{x}$ such that $\mathbf{G}\mathbf{x} = \mathbf{b}$ is given by [48, Section 10.2]

$$\boldsymbol{\beta}_k = \mathbf{r}_{k-1}^T\mathbf{r}_{k-1} / \left(\mathbf{r}_{k-2}^T\mathbf{r}_{k-2}\right), \tag{7.63}$$

$$\mathbf{p}_k = \mathbf{r}_{k-1} + \boldsymbol{\beta}_k\mathbf{p}_{k-1}, \tag{7.64}$$

$$\boldsymbol{\alpha}_k = \mathbf{r}_{k-1}^T\mathbf{r}_{k-1} / \left(\mathbf{p}_k^T\mathbf{G}\mathbf{p}_k\right), \tag{7.65}$$

$$\mathbf{x}_k = \mathbf{x}_{k-1} + \boldsymbol{\alpha}_k\mathbf{p}_k, \tag{7.66}$$

$$\mathbf{r}_k = \mathbf{r}_{k-1} - \boldsymbol{\alpha}_k\mathbf{G}\mathbf{p}_k. \tag{7.67}$$

This is used until the norm of the residual $\|\mathbf{r}_k\|_2$ is sufficiently small. As the iterations only require a matrix–vector multiplication operation, the conjugate gradient method can be very efficient. Other approximate approaches to solving the linear system (7.61), (7.62) can be found in [48]. The applicability of the approximate algorithms method is based on the fact that if the subspace projection error is bounded by a summable sequence then the ADMM algorithm $\mathcal{OS}_{\text{ADMM}}$ will converge [45, 49].

7.4.4 Cone Projection

Proximal Algorithm

The second step in the algorithm $\mathcal{OS}_{\text{ADMM}}$ is to project a point $\boldsymbol{\omega}$ onto the cone $\mathcal{C}$. As $\mathcal{C}$ is the Cartesian product of a finite number of smaller convex cones $\mathcal{C}_i$, we can perform projection onto $\mathcal{C}$ by projecting onto $\mathcal{C}_i$ separately and in parallel. Furthermore,

the projection onto each convex cone can be done with closed forms. For example, for nonnegative real $\mathcal{C}_i = \mathbb{R}_+$, we have that [50, Section 6.3.1]

$$\Pi_{\mathcal{C}_i}(\boldsymbol{\omega}) = \boldsymbol{\omega}_+, \tag{7.68}$$

where the nonnegative-part operator $(\cdot)_+$ is taken elementwise. For the second-order cone $\mathcal{C}_i = \{(y, \mathbf{x}) \in \mathbb{R} \times \mathbb{R}^{p-1} | \|\mathbf{x}\| \leq y\}$, we have that [50, Section 6.3.2]

$$\Pi_{\mathcal{C}_i}(\boldsymbol{\omega}, \tau) = \begin{cases} 0, & \|\boldsymbol{\omega}\|_2 \leq -\tau, \\ (\boldsymbol{\omega}, \tau), & \|\boldsymbol{\omega}\|_2 \leq \tau \\ (1/2)(1 + \tau/\|\boldsymbol{\omega}\|_2)(\boldsymbol{\omega}, \|\boldsymbol{\omega}\|_2), & \|\boldsymbol{\omega}\|_2 \geq |\tau|. \end{cases} \tag{7.69}$$

For the semidefinite cone $\mathcal{C}_i = \{\mathbf{X} \in \mathbb{R}^{n \times n} | \mathbf{X} = \mathbf{X}^T, \mathbf{X} \succeq \mathbf{0}\}$, we have that [50, Section 6.3.3]

$$\Pi_{\mathcal{C}_i}(\boldsymbol{\Omega}) = \sum_{i=1}^{n} (\lambda_i)_+ \mathbf{u}_i \mathbf{u}_i^T, \tag{7.70}$$

where $\sum_{i=1}^{n} \lambda_i \mathbf{u}_i \mathbf{u}_i^T$ is the eigenvalue decomposition of $\boldsymbol{\Omega}$. More examples on the cone projection (e.g., the exponential cone projection) can be found in [50].

Randomized Algorithms

Although the cone projection can be performed in parallel with closed forms, the scaling may be prohibitive on the semidefinite cone projection via eigenvalue decomposition. Therefore, to scale well to large problem sizes for SDP problems, it is of great interest to develop efficient algorithms to solve approximately the semidefinite cone projection problem with rigorous performance and convergence guarantees for the resulting ADMM algorithm $\mathcal{OS}_{\text{ADMM}}$.

Randomized sketching provides powerful randomized and sampling techniques for large-scale matrices by compressing them into much smaller matrices, thereby saving solving time and memory by reducing the problem dimensions. For semidefinite cone projection, to project a symmetric matrix $\mathbf{A} \in \mathbb{R}^{n \times n}$ onto the positive semidefinite cone, we need to first perform its eigenvalue expansion and then drop the terms associated with negative eigenvalues. The randomized algorithms for the eigenvalue decomposition of the symmetric matrix $\mathbf{A} \in \mathbb{R}^{n \times n}$ generally consist of the following simple steps [51]:

1. generate the orthonormal matrix $\mathbf{Q} \in \mathbb{R}^{m \times n}$ ($m < n$) such that $\|\mathbf{A} - \mathbf{Q}\mathbf{Q}^T\mathbf{A}\| \leq \epsilon$ with ϵ as the computational tolerance;
2. form the smaller matrix $\mathbf{B} = \mathbf{Q}\mathbf{A}\mathbf{Q}^T$;
3. compute an eigenvalue decomposition $\mathbf{B} = \mathbf{V}\boldsymbol{\Lambda}\mathbf{V}^T$;
4. form the orthonormal matrix $\mathbf{U} = \mathbf{Q}\mathbf{V}$ such that $\mathbf{A} \approx \mathbf{U}\boldsymbol{\Lambda}\mathbf{U}^T$.

For step 1, several efficient randomized schemes were discussed in [51, Section 4] to minimize the sampling size and computational cost for producing the matrix $\mathbf{Q}$. A simple scheme is based on the Gaussian random matrix. In step 3, once we have $\mathbf{B}$ we can adopt any of the standard deterministic factorization techniques in [51, Section 3.3] to produce eigenvalue decomposition. More efficient algorithms for factorization can

be found in [51, Section 5.2] by exploiting the information in $\mathbf{Q}$. Typically, randomized algorithms only require $\mathcal{O}(n^2 \log(k))$ flops while classic algorithms require $\mathcal{O}(n^2 k)$ flops to do eigenvalue decomposition for a rank-k matrix $\mathbf{A}$. More recent progress on randomized sketching methods for numerical linear algebra can be found in [52, 53].

Although randomized algorithms can exploit randomness as a source for speedup, it is non-trivial to apply these algorithms directly for approximate cone projections with performance and convergence guarantees for the resulting operator splitting algorithm $\mathcal{OS}_{\text{ADMM}}$. It is thus of great interest to establish theoretical guarantees for the randomized cone projection methods in the algorithm $\mathcal{OS}_{\text{ADMM}}$.

7.4.5 Practical Implementation Issues

We have thus far presented the two-stage parallel computing framework for large-scale convex optimization in dense C-RANs. Here, we will discuss implementation issues of the proposed framework in C-RANs, thereby exploiting the computational architectures to obtain further speed gains.

Parallel and Distributed Implementation

The operator splitting algorithm $\mathcal{OS}_{\text{ADMM}}$ that we have presented is compact and parameter-free, with parallelizable computing and linear convergence. In particular, each iteration of the algorithm is simple and easy for parallel and distributed computing. This allows the algorithm $\mathcal{OS}_{\text{ADMM}}$ to utilize the cloud computing environments in C-RANs with shared computing and memory resources in a single BBU pool. Specifically, the parallel algorithms can be leveraged in the subspace projection for LDL^T factorization and sparse matrix–vector multiplication [54]. The cone projection can be parallelized easily by projecting onto $\mathcal{K}_i$ separately and in parallel. However, it is challenging to accommodate the operator splitting algorithm to the distributed environments in heterogeneous C-RANs [4, 8], Fog-RAN, and MENG-RAN [5]. In particular, message updates across the network (e.g., backhaul network) and synchronization among heterogenous computation units will significantly increase the communication complexity in the distributed algorithms, which may result in delay and loss of performance. Therefore, it is critical to design large-scale distributed optimization algorithms with the minimal requirements of synchronization and communication.

Real-Time Implementation

In dense C-RANs, to satisfy the strict low-latency demands, e.g., in Tactile Internet the end-to-end latency is constrained to one millisecond [55], we need to solve large-scale optimization problems in a millisecond. This brings significant challenges compared with large-scale optimization problems in machine learning and big data, where latency is not a big issue but the problem dimension is often in the order of millions.

To solve a large-scale optimization problem in a real-time way, one promising approach is to leverage the *symbolic* subspace and cone projections. The general idea is to generate and store all the structures and descriptions of the algorithm for the specific problem family $\mathscr{P}_{\text{cone}}$. Eventually, the ADMM solver can be symbolically based so as

to provide numerical solutions for each problem instance $\mathscr{P}_{\text{cone}}(\boldsymbol{\beta})$ extremely quickly within a hard real-time deadline. This idea has already been successfully applied in the code generation system CVXGEN [56] for real-time convex quadratic optimization [57] and in the interior-point based SOCP solver [58] for embedded systems. It is of great interest to implement this idea for the operator splitting algorithm $\mathcal{OS}_{\text{ADMM}}$ for general real-time conic optimization.

7.5 Numerical Results

In this section we simulate the proposed two-stage large-scale convex optimization framework for performance optimization in dense C-RANs. *The corresponding MATLAB code that can reproduce all the simulation results using the proposed large-scale convex optimization algorithm is available online.*[1]

We considered the following channel model for the link between the kth MU and the lth RRH:

$$\mathbf{h}_{kl} = 10^{-L(d_{kl})/20}\sqrt{\varphi_{kl}s_{kl}}\mathbf{f}_{kl}, \quad \forall k, l, \tag{7.71}$$

where $L(d_{kl})$ is the path loss in dB at distance d_{kl}, as in [2, Table I], s_{kl} is the shadowing coefficient, φ_{kl} is the antenna gain, and $\mathbf{f}_{kl}$ is the small-scale fading coefficient. We used the standard cellular network parameters as in [2, Table I]. All the simulations were carried out on a personal computer with a 3.2 GHz quad-core Intel Core i5 processor and 8 GB of RAM running Linux. The reference implementation of the operator splitting algorithm SCS is available online;[2] it is a general software package for solving large-scale convex cone problems based on [45] and can be called by the modeling frameworks CVX and CVXPY [59]. The settings (e.g., the stopping criteria) of SCS can be found in [45].

The proposed two-stage approach framework, termed "Matrix Stuffing+SCS", is compared with the following state-of-the-art frameworks:

- CVX+SeDuMi/SDPT3/MOSEK This category adopts second-order methods. The modeling framework CVX will first automatically transform the original problem instance (e.g., the problem $\mathscr{P}$ written in disciplined convex programming form) into the standard cone-programming form and then call an interior-point solver, e.g., SeDuMi [17], SDPT3 [18], or MOSEK [19].
- CVX+SCS In this framework based on first-order methods, CVX first transforms the original problem instance into the standard form and then calls the operator splitting solver SCS.

We define the *modeling time* as the transformation time for the first stage, the *solving time* as the time spent on the second stage, and the *total time* as the time for the two stages to solve one problem instance. As the large-scale convex optimization algorithm

[1] https://github.com/ShiYuanming/large-scale-convex-optimization
[2] https://github.com/cvxgrp/scs

should scale well to both the modeling part and the solving part simultaneously, the time comparison of each individual stage will demonstrate the effectiveness of the proposed two-stage approach.

Given the network size, we first generate and store the problem structure of the standard conic optimization problem family $\mathscr{P}_{\text{cone}}$, i.e., the structure of $\mathbf{A}$, $\mathbf{b}$, $\mathbf{c}$, and the descriptions of $\mathcal{K}$. As this procedure can be done offline for all the candidate network sizes, we thus ignore this step for time comparison. The following procedures to solve the large-scale convex optimization problem instances $\mathscr{P}(\boldsymbol{\alpha})$ are repeated with different parameters $\boldsymbol{\alpha}$ and sizes using the proposed framework Matrix Stuffing+SCS:

1. Copy the parameters in the problem instance $\mathscr{P}(\boldsymbol{\alpha})$ to the data in the pre-stored structure of the standard cone program $\mathscr{P}_{\text{cone}}$.
2. Solve the resultant standard conic optimization problem instance $\mathscr{P}_{\text{cone}}(\boldsymbol{\beta})$ using the solver SCS.
3. Extract the optimal solutions of $\mathscr{P}(\boldsymbol{\alpha})$ from the solutions to $\mathscr{P}_{\text{cone}}(\boldsymbol{\beta})$ produced by the solver SCS.

Finally, note that all the interior-point solvers are multiple threaded (i.e., they can utilize multiple threads to gain extra speedups), while the operator splitting algorithm solver SCS is single threaded. Nevertheless, we will show that SCS performs much faster than the interior-point solvers. We also emphasize that the operator splitting method is aimed to scale well to large problem sizes and thus to provide solutions to modest accuracy within a reasonable time, while the interior-point method's intended to provide highly accurate solutions. Furthermore, the modeling framework CVX provides rapid prototyping and a user-friendly tool for automatic transformations for general problems, while the matrix-stuffing technique targets large-scale problems for the specific problem family $\mathscr{P}$. Therefore, these frameworks and solvers are not really comparable in view of their different purposes and application capabilities. We mainly use them to verify the effectiveness and reliability of our proposed framework in terms of solution time and solution quality.

7.5.1 Effectiveness and Reliability of the Large-Scale Optimization Framework

Consider a network with L two-antenna RRHs, K single-antenna MUs, and $L = K$, where all the RRHs and MUs are uniformly and independently distributed in the square region $[-3000, 3000] \times [-3000, 3000]$ meters. We consider the total transmit-power minimization problem $\mathscr{P}_{\text{SOCP}}$ with the objective function as $\|\mathbf{v}\|_2^2$ and the QoS requirements for each MU as $\gamma_k = 5$ dB, $\forall k$. Table 7.1 demonstrates for comparison the running time and solutions using different convex optimization frameworks. Each point of the simulation results is averaged over 100 randomly generated network realizations (i.e., one small-scale fading realization for each large-scale fading realization).

For the modeling time comparisons, this table shows that the time value of the proposed matrix-stuffing technique ranges between 0.01 and 30 seconds for different network sizes and can bring a speedup of about 15 to 60 times compared with the parser/solver modeling framework CVX. In particular, for large-scale problems, the

Table 7.1 Time and solution results for different convex optimization frameworks

Network Size ($L = K$)		20	50	100	200
CVX+SeDuMi	**Total time** (s)	**8.1164**	N/A	N/A	N/A
	Objective (W)	12.2488	N/A	N/A	N/A
CVX+SDPT3	**Total time** (s)	**5.0398**	**330.6814**	N/A	N/A
	Objective (W)	12.2488	6.5216	N/A	N/A
CVX+MOSEK	**Total time** (s)	**1.2072**	**51.6351**	N/A	N/A
	Objective (W)	12.2488	6.5216	N/A	N/A
CVX+SCS	**Total time** (s)	**0.8501**	**5.6432**	**51.0472**	**725.6173**
	Modeling time (s)	0.7563	4.4301	38.6921	534.7723
	Objective [W]	12.2505	6.5215	3.1303	1.5404
Matrix Stuffing+SCS	**Total time** (s)	**0.1137**	**2.7222**	**26.2242**	**328.2037**
	Modeling time (s)	0.0128	0.2401	2.4154	29.5813
	Objective (W)	12.2523	6.5193	3.1296	1.5403

transformation using CVX is time consuming and becomes the bottleneck, as the modeling time is comparable with and even larger than the solving time. For example, when $L = 150$, the modeling time using CVX is about 3 minutes, while matrix stuffing only requires about 10 seconds. Therefore, matrix stuffing for fast transformation is essential for solving large-scale convex optimization problems quickly.

For the solving time (which can be easily calculated by subtracting the modeling time from the total time) using different solvers, this table shows that the operator splitting solver can provide a speedup of several orders of magnitude over the interior-point solvers. For example, for $L = 50$, the speedup is about 20 and 130 times over MOSEK and SDPT3, respectively, while SeDuMi is inapplicable for this problem size as the running time exceeds the predefined maximum value, i.e., one hour. In particular, all the interior-point solvers fail to solve large-scale problems (i.e., $L = 100, 150, 200$), which is indicated as N/A, while the operator splitting solver SCS can scale well to large problem sizes. Regarding the largest problems, with $L = 200$, the operator splitting solver can solve them in about 5 minutes.

Regarding the quality of the solutions, Table 7.1 shows that the proposed framework can provide a solution to modest accuracy within much less time. For the two problem sizes, i.e., $L = 20$ and $L = 50$, which can be solved by the interior-point frameworks, the optimal values attained by the proposed framework are within 0.03% of that obtained via the second-order-method frameworks.

In summary, the proposed two-stage large-scale convex optimization framework scales well to simultaneous large-scale problem modeling and solving. Therefore it could provide an effective way to evaluate the system performance via large-scale optimization in dense wireless networks. However, its implementation and performance in practical systems still needs further investigation. In particular, this set of results indicates that the scale of cooperation in dense wireless networks may be fundamentally constrained by the computation complexity on time.

7.5.2 Max–Min Fairness Rate Optimization

We will simulate the minimum network-wide achievable rate maximization problem using the max–min fairness optimization algorithm in [24, Algorithm 1] via the bisection method, which requires solving a sequence of convex feasibility problems. We will not only show the quality of the solutions and speedups provided by the proposed framework but also demonstrate that the optimal coordinated beamformers significantly outperform low-complexity and heuristic transmission strategies, i.e., zero-forcing beamforming (ZFBF) [30, 60], regularized zero-forcing beamforming (RZF) [61], and maximum ratio transmission (MRT) [62].

Consider a network with $L = 55$ single-antenna RRHs and $K = 50$ single-antenna MUs uniformly and independently distributed in the square region $[-5000, 5000] \times [-5000, 5000]$ meters. Figure 7.2 demonstrates the minimum network-wide achievable rate (which is defined as the transmit power at all the RRHs divided by the receive noise power at all the MUs) versus SNR using different algorithms. Each point of the simulation results is averaged over 50 randomly generated network realizations. For optimal beamforming, this figure shows the accuracy of the solutions obtained by the proposed framework compared with the first-order method framework CVX+SCS. The average solving time and modeling time for obtaining a single point for the optimal beamforming with CVX+SCS and Matrix Stuffing+SCS are (176.3410, 55.1542) seconds and (82.0180, 1.2012) seconds, respectively. This shows that the proposed framework can reduce both the solving time and modeling time via warm-starting and matrix stuffing, respectively.

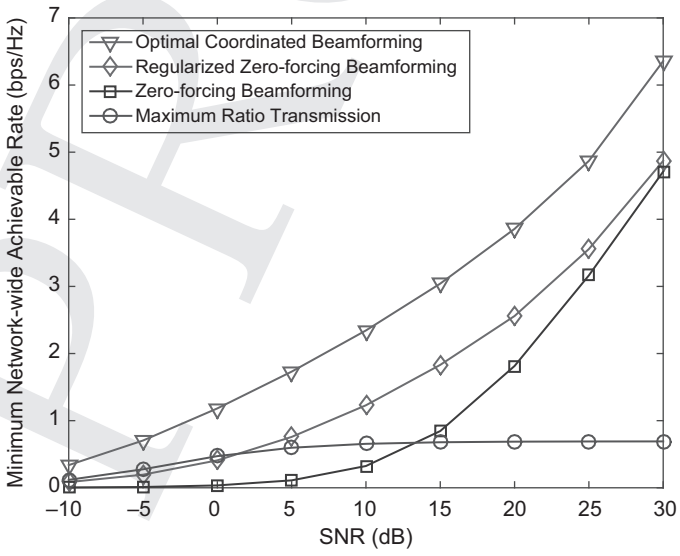

Figure 7.2 The minimum network-wide achievable rate versus transmit SNR with 55 single-antenna RRHs and 50 single-antenna MUs.

Furthermore, this figure also shows that optimal beamforming can achieve quite an improvement for the per-user rate compared with the suboptimal transmission strategies RZF, ZFBF, and MRT; this clearly shows the importance of developing optimal beamforming algorithms for such networks. The average solving time and modeling time for a single point using CVX+SDPT3 for the RZF, ZFBF, and MRT are (2.6210, 30.2053) seconds, (2.4592, 30.2098) seconds, and (2.5966, 30.2161) seconds, respectively. Note that the solving time is very small, because we only need to solve a sequence of linear programming problems for power control when the directions of the beamformers are fixed during the bisection search procedure. The main time-consuming part is the transformation using CVX.

7.6 Summary and Discussion

In this chapter we have presented a unified two-stage framework for large-scale optimization in dense C-RANs. We showed that various performance optimization problems in C-RANs can be essentially solved by solving one, or a sequence of, convex optimization or convex feasibility problems. The proposed framework requires only the convexity of the underlying problems without any other structural assumptions, e.g., smooth or separable functions. This is achieved by first transforming the original convex problem to a standard form via matrix stuffing and then using the ADMM algorithm to solve the homogeneous self-dual embedding of the primal–dual pair of the transformed standard cone program. Simulation results demonstrate the infeasibility detection capability, the modeling flexibility and computing scalability, as well as the reliability of the proposed framework.

In principle one may apply the proposed framework to any large-scale convex optimization problem; one needs only to focus on the standard conic optimization form reformulation as well as to compute the proximal operators for different cone projections. However, in practice the following issues need to be addressed in order to provide a user-friendly framework and to assist practical implementation.

1. Developing a software package automatically generating the code for matrix stuffing is desirable but challenging in terms of reliability and correctness verification.
2. Efficient subspace and cone projection algorithms are highly desirable. In particular, the randomized algorithms may provide a powerful method to scale up the projections at each iteration, thereby trading off the solving time and the accuracy of solutions.
3. It is of great interest to implement the proposed large-scale convex optimization framework in C-RANs by exploiting parallel and distributed computation architectures, thereby further investigating the feasibility of this approach for real-time applications with strict low-latency requirements in wireless networks.
4. It is also of interest apply the proposed framework to various non-convex optimization problems, e.g., optimization on manifolds [63, 64].

References

[1] China Mobile, "C-RAN: the road towards green RAN," White Paper, ver. 3.0, December 2013.

[2] Y. Shi, J. Zhang, and K. B. Letaief, "Group sparse beamforming for green Cloud-RAN," *IEEE Trans. Wireless Commun.*, vol. 13, no. 5, pp. 2809–2823, May 2014.

[3] F. Bonomi, R. Milito, P. Natarajan, and J. Zhu, "Fog computing: a platform for internet of things and analytics," in *Big Data and Internet of Things: A Roadmap for Smart Environments*, Springer International Publishing, 2014, pp. 169–186.

[4] M. Peng, Y. Li, J. Jiang, J. Li, and C. Wang, "Heterogeneous cloud radio access networks: a new perspective for enhancing spectral and energy efficiencies," *IEEE Wireless Commun. Mag.*, vol. 21, no. 6, pp. 126–135, December 2014.

[5] Y. Shi, J. Zhang, K. Letaief, B. Bai, and W. Chen, "Large-scale convex optimization for ultra-dense cloud-ran," *IEEE Wireless Commun. Mag.*, vol. 22, no. 3, pp. 84–91, June 2015.

[6] A. B. Gershman, N. D. Sidiropoulos, S. Shahbazpanahi, M. Bengtsson, and B. Ottersten, "Convex optimization-based beamforming: from receive to transmit and network designs," *IEEE Signal Process. Mag*, vol. 27, no. 3, pp. 62–75, 2010.

[7] Z.-Q. Luo, W.-K. Ma, A.-C. So, Y. Ye, and S. Zhang, "Semidefinite relaxation of quadratic optimization problems," *IEEE Signal Process. Mag.*, vol. 27, no. 3, pp. 20–34, May 2010.

[8] E. Björnson and E. Jorswieck, "Optimal resource allocation in coordinated multi-cell systems," *Found. Trends Commun. Inf. Theory*, vol. 9, nos. 2–3, pp. 113–381, January 2013.

[9] D. P. Palomar and Y. C. Eldar, *Convex Optimization in Signal Processing and Communications*. Cambridge University Press, 2010.

[10] H. Dahrouj and W. Yu, "Coordinated beamforming for the multicell multi-antenna wireless system," *IEEE Trans. Wireless Commun.*, vol. 9, no. 5, pp. 1748–1759, September 2010.

[11] S. P. Boyd and L. Vandenberghe, *Convex Optimization*. Cambridge University Press, 2004.

[12] R. Zakhour and S. V. Hanly, "Base station cooperation on the downlink: large system analysis," *IEEE Trans. Inf. Theory*, vol. 58, no. 4, pp. 2079–2106, April 2012.

[13] C. Shen, T.-H. Chang, K.-Y. Wang, Z. Qiu, and C.-Y. Chi, "Distributed robust multicell coordinated beamforming with imperfect CSI: an ADMM approach," *IEEE Trans. Signal Process.*, vol. 60, no. 6, pp. 2988–3003, June 2012.

[14] Y. Shi, J. Zhang, and K. Letaief, "Robust group sparse beamforming for multicast green cloud-ran with imperfect csi," *IEEE Trans. Signal Process.*, vol. 63, no. 17, pp. 4647–4659, September 2015.

[15] J. Cheng, Y. Shi, B. Bai, W. Chen, J. Zhang, and K. Letaief, "Group sparse beamforming for multicast green Cloud-RAN via parallel semidefinite programming," *IEEE Int. Conf. Communications*, London, 2015.

[16] Y. Shi, J. Zhang, and K. Letaief, "Optimal stochastic coordinated beamforming for wireless cooperative networks with CSI uncertainty," *IEEE Trans. Signal Process.*, vol. 63, no. 4, pp. 960–973, February 2015.

[17] J. F. Sturm, "Using SeDuMi 1.02, a MATLAB toolbox for optimization over symmetric cones," *Optim. Methods Softw.*, vol. 11, no. 1–4, pp. 625–653, 1999.

[18] K.-C. Toh, M. J. Todd, and R. H. Tütüncü, "SDPT3 – a MATLAB software package for semidefinite programming, version 1.3," *Optim. Methods Softw.*, vol. 11, nos. 1–4, pp. 545–581, 1999.

[19] E. D. Andersen and K. D. Andersen, "The mosek interior point optimizer for linear programming: an implementation of the homogeneous algorithm," in *High Performance Optimization*, Springer, 2000, pp. 197–232.

[20] Y. Nesterov, A. Nemirovskii, and Y. Ye, *Interior-Point Polynomial Algorithms in Convex Programming*. SIAM, 1994, vol. 13.

[21] CVX Research, Inc., "CVX: Matlab software for disciplined convex programming, version 2.0 (beta)," 2013. Online. Available at http://cvxr.com/cvx/

[22] J. Lofberg, "YALMIP: a toolbox for modeling and optimization in MATLAB," in *Proc. IEEE Int. Symp. Computer-Aided Control Systems Design*, Taipei, September 2004, pp. 284–289.

[23] E. Chu, N. Parikh, A. Domahidi, and S. Boyd, "Code generation for embedded second-order cone programming," in *Proc. 2013 European Control Conf.*, July 2013, pp. 1547–1552.

[24] Y. Shi, J. Zhang, and K. Letaief, "Scalable coordinated beamforming for dense wireless cooperative networks," in *Proc. IEEE Global Communications Conf.*, Austin, TX, December 2014, pp. 3603–3608.

[25] E. Bjornson, M. Bengtsson, and B. Ottersten, "Optimal multiuser transmit beamforming: a difficult problem with a simple solution structure [lecture notes]," *IEEE Signal Process. Mag.*, vol. 31, no. 4, pp. 142–148, July 2014.

[26] W.-C. Liao, M. Hong, Y.-F. Liu, and Z.-Q. Luo, "Base station activation and linear transceiver design for optimal resource management in heterogeneous networks," *IEEE Trans. Signal Process.*, vol. 62, no. 15, pp. 3939–3952, August 2014. Online. Available at http://arxiv.org/abs/1309.4138

[27] S. Boyd, N. Parikh, E. Chu, B. Peleato, and J. Eckstein, "Distributed optimization and statistical learning via the alternating direction method of multipliers," *Found. Trends Mach. Learn.*, vol. 3, no. 1, pp. 1–122, July 2011.

[28] S. K. Joshi, M. Codreanu, and M. Latva-aho, "Distributed resource allocation for MISO downlink systems via the alternating direction method of multipliers," *EURASIP J. Wireless Commun. Netw.*, vol. 2014, no. 1, pp. 1–19, January 2014.

[29] J. Zhang, R. Chen, J. G. Andrews, A. Ghosh, and R. W. Heath, "Networked MIMO with clustered linear precoding," *IEEE Trans. Wireless Commun.*, vol. 8, no. 4, pp. 1910–1921, April 2009.

[30] T. Yoo and A. Goldsmith, "On the optimality of multiantenna broadcast scheduling using zero-forcing beamforming," *IEEE J. Sel. Areas Commun.*, vol. 24, no. 3, pp. 528–541, March 2006.

[31] Y. Shi, J. Zhang, B. O'Donoghue, and K. Letaief, "Large-scale convex optimization for dense wireless cooperative networks," *IEEE Trans. Signal Process.*, vol. 63, no. 18, pp. 4729–4743, September 2015.

[32] E. Smith, "On the optimal design of continuous processes," Ph.D. thesis, Imperial College London (University of London), 1996.

[33] M. C. Grant and S. P. Boyd, "Graph implementations for nonsmooth convex programs," in *Recent Advances in Learning and Control*, Springer, 2008, pp. 95–110.

[34] Y. Ye, M. J. Todd, and S. Mizuno, "An $\mathcal{O}(\sqrt{n}L)$-iteration homogeneous and self-dual linear programming algorithm," *Math. Oper. Res.*, vol. 19, no. 1, pp. 53–67, 1994.

[35] A. Wiesel, Y. Eldar, and S. Shamai, "Linear precoding via conic optimization for fixed MIMO receivers," *IEEE Trans. Signal Process.*, vol. 54, no. 1, pp. 161–176, January 2006.

[36] Y. Shi, J. Zhang, and K. Letaief, "Group sparse beamforming for green cloud radio access networks," in *Proc. IEEE Global Communications Conf.*, Atlanta, GA, December 2013, pp. 4635–4640.

[37] N. Jindal and A. Lozano, "A unified treatment of optimum pilot overhead in multipath fading channels," *IEEE Trans. Commun.*, vol. 58, no. 10, pp. 2939–2948, October 2010.

[38] D. J. Love, R. W. Heath, V. K. Lau, D. Gesbert, B. D. Rao, and M. Andrews, "An overview of limited feedback in wireless communication systems," *IEEE J. Sel. Areas Commun.*, vol. 26, no. 8, pp. 1341–1365, October 2008.

[39] M. A. Maddah-Ali and D. Tse, "Completely stale transmitter channel state information is still very useful," *IEEE Trans. Inf. Theory*, vol. 58, no. 7, pp. 4418–4431, July 2012.

[40] J. Zhang, R. W. Heath, M. Kountouris, and J. G. Andrews, "Mode switching for the multi-antenna broadcast channel based on delay and channel quantization," *Proc. EURASIP Conf., J. Adv. Signal Process, Special Issue on Multiuser Lim. Feedback*, vol. 2009, Article ID 802548, 15 pp., 2009.

[41] E. Björnson, G. Zheng, M. Bengtsson, and B. Ottersten, "Robust monotonic optimization framework for multicell MISO systems," *IEEE Trans. Signal Process.*, vol. 60, no. 5, pp. 2508–2523, May 2012.

[42] L. J. Hong, Y. Yang, and L. Zhang, "Sequential convex approximations to joint chance constrained programs: a Monte Carlo approach," *Oper. Res.*, vol. 59, no. 3, pp. 617–630, May–June 2011.

[43] T. A. Davis, *Direct Methods for Sparse Linear Systems*. Society for Industrial and Applied Mathematics, 2006. Online. Available at http://epubs.siam.org/doi/abs/10.1137/1.9780898718881.

[44] E. Chu, B. O'Donoghue, N. Parikh, and S. Boyd, "A primal–dual operator splitting method for conic optimization," 2013. Online. Available at www.stanford.edu/ boyd/papers/pdos.html.

[45] B. O'Donoghue, E. Chu, N. Parikh, and S. Boyd, "Conic optimization via operator splitting and homogeneous self-dual embedding," 2013. Online. Available at arxiv.org/abs/1312.3039.

[46] T. Goldstein, B. O'Donoghue, S. Setzer, and R. Baraniuk, "Fast alternating direction optimization methods," *SIAM J. Imaging Sci.*, vol. 7, no. 3, pp. 1588–1623, 2014. Online. Available at dx.doi.org/10.1137/120896219.

[47] R. Vanderbei, "Symmetric quasidefinite matrices," *SIAM J. Optim.*, vol. 5, no. 1, pp. 100–113, 1995. Online. Available at dx.doi.org/10.1137/0805005.

[48] G. H. Golub and C. F. Van Loan, *Matrix Computations*. Johns Hopkins University Press, 2012, vol. 3.

[49] J. Eckstein and D. P. Bertsekas, "On the douglas–rachford splitting method and the proximal point algorithm for maximal monotone operators," *Math. Progr.*, vol. 55, no. 1-3, pp. 293–318, 1992.

[50] N. Parikh and S. Boyd, "Proximal algorithms," *Found. Trends Optim.*, vol. 1, no. 3, January 2014. Online. Available at www.stanford.edu/ boyd/papers.

[51] N. Halko, P.-G. Martinsson, and J. A. Tropp, "Finding structure with randomness: probabilistic algorithms for constructing approximate matrix decompositions," *SIAM Review*, vol. 53, no. 2, pp. 217–288, 2011.

[52] M. W. Mahoney, "Randomized algorithms for matrices and data," *Found. Trends Mach. Learn.*, vol. 3, no. 2, pp. 123–224, 2011. Online. Available at dx.doi.org/10.1561/2200000035.

[53] D. P. Woodruff, "Sketching as a tool for numerical linear algebra," *Found. Trends Theoret. Computer Sci.*, vol. 10, no. 1-2, pp. 1–157, 2014. Online. Available at dx.doi.org/10.1561/0400000060.

[54] J. Poulson, B. Marker, R. A. van de Geijn, J. R. Hammond, and N. A. Romero, "Elemental: a new framework for distributed memory dense matrix computations," *ACM Trans. Math. Softw.*, vol. 39, no. 2, p. 13, February 2013.

[55] G. Fettweis, "The tactile internet: applications and challenges," *IEEE Veh. Technol. Mag.*, vol. 9, no. 1, pp. 64–70, March 2014.

[56] J. Mattingley and S. Boyd, "CVXGEN: a code generator for embedded convex optimization," *Optim. Engineering*, vol. 13, no. 1, pp. 1–27, 2012.

[57] J. Mattingley and S. Boyd, "Real-time convex optimization in signal processing," *IEEE Signal Process. Mag*, vol. 27, no. 3, pp. 50–61, May 2010.

[58] A. Domahidi, E. Chu, and S. Boyd, "ECOS: an SOCP solver for embedded systems," in *Proc. 2013 European Control Conf.*, IEEE, 2013, pp. 3071–3076.

[59] S. Diamond, E. Chu, and S. Boyd, "CVXPY: A Python-embedded modeling language for convex optimization, version 0.2," May 2014.

[60] J. Zhang and J. G. Andrews, "Adaptive spatial intercell interference cancellation in multicell wireless networks," *IEEE J. Sel. Areas Commun.*, vol. 28, no. 9, pp. 1455–1468, December 2010.

[61] C. B. Peel, B. M. Hochwald, and A. L. Swindlehurst, "A vector-perturbation technique for near-capacity multiantenna multiuser communication – Part I: channel inversion and regularization," *IEEE Trans. Commun.*, vol. 53, no. 1, pp. 195–202, January 2005.

[62] F. Rusek, D. Persson, B. K. Lau, E. Larsson, T. Marzetta, O. Edfors *et al.*, "Scaling up MIMO: opportunities and challenges with very large arrays," *IEEE Signal Process. Mag.*, vol. 30, no. 1, pp. 40–60, January 2013.

[63] P.-A. Absil, R. Mahony, and R. Sepulchre, *Optimization Algorithms on Matrix Manifolds*. Princeton University Press, 2009.

[64] Y. Shi, J. Zhang, and K. B. Letaief, "Low-rank matrix completion via Riemannian pursuit for topological interference management," in *Proc. IEEE Int. Symp. on Information Theory*, Hong Kong, 2015.