

Large-Scale Convex Optimization for Dense Wireless Cooperative Networks

Yuanming Shi, Jun Zhang, Brendan O'Donoghue, and Khaled B. Letaief, *Fellow, IEEE*

Abstract—Convex optimization is a powerful tool for resource allocation and signal processing in wireless networks. As the network density is expected to drastically increase in order to accommodate the exponentially growing mobile data traffic, performance optimization problems are entering a new era characterized by a high dimension and/or a large number of constraints, which poses significant design and computational challenges. In this paper, we present a novel two-stage approach to solve large-scale convex optimization problems for dense wireless cooperative networks, which can effectively detect infeasibility and enjoy modeling flexibility. In the proposed approach, the original large-scale convex problem is transformed into a standard cone programming form in the first stage via matrix stuffing, which only needs to copy the problem parameters such as channel state information (CSI) and quality-of-service (QoS) requirements to the pre-stored structure of the standard form. The capability of yielding infeasibility certificates and enabling parallel computing is achieved by solving the homogeneous self-dual embedding of the primal-dual pair of the standard form. In the solving stage, the operator splitting method, namely, the alternating direction method of multipliers (ADMM), is adopted to solve the large-scale homogeneous self-dual embedding. Compared with second-order methods, ADMM can solve large-scale problems in parallel with modest accuracy within a reasonable amount of time. Simulation results will demonstrate the speedup, scalability, and reliability of the proposed framework compared with the state-of-the-art modeling frameworks and solvers.

Index Terms—Dense wireless networking, large-scale optimization, matrix stuffing, operator splitting method, ADMM, homogeneous self-dual embedding.

I. INTRODUCTION

THE proliferation of smart mobile devices, coupled with new types of wireless applications, has led to an exponential growth of wireless and mobile data traffic. In order to provide high-volume and diversified data services, ultra-dense wireless cooperative network architectures have been proposed for next generation wireless networks [1], e.g., Cloud-RAN [2], [3], and distributed antenna systems [4]. To enable efficient interference management and resource allocation, large-scale multi-entity collaboration will play pivotal roles in dense wireless networks. For instance, in Cloud-RAN, all the baseband signal processing is shifted to a single cloud data center with

very powerful computational capability. Thus the centralized signal processing can be performed to support large-scale cooperative transmission/reception among the radio access units (RAUs).

Convex optimization serves as an indispensable tool for resource allocation and signal processing in wireless communication systems [5], [6], [7]. For instance, coordinated beamforming [8] often yields a direct convex optimization formulation, i.e., second-order cone programming (SOCP) [9]. The network max-min fairness rate optimization [10] can be solved through the bi-section method [9] in polynomial time, wherein a sequence of convex subproblems are solved. Furthermore, convex relaxation provides a principled way of developing polynomial-time algorithms for non-convex or NP-hard problems, e.g., group-sparsity penalty relaxation for the NP-hard mixed integer nonlinear programming problems [3], semidefinite relaxation [6] for NP-hard robust beamforming [11], [12] and multicast beamforming [13], and sequential convex approximation to the highly intractable stochastic coordinated beamforming [14].

Nevertheless, in dense wireless cooperative networks [1], which may possibly need to simultaneously handle hundreds of RAUs, resource allocation and signal processing problems will be dramatically scaled up. The underlying optimization problems will have high dimensions and/or large numbers of constraints (e.g., per-RAU transmit power constraints and per-MU (mobile user) QoS constraints). For instance, for a Cloud-RAN with 100 single-antenna RAUs and 100 single-antenna MUs, the dimension of the aggregative coordinated beamforming vector (i.e., the optimization variables) will be 10^4 . Most advanced off-the-shelf solvers (e.g., SeDuMi [15], SDPT3 [16] and MOSEK [17]) are based on the interior-point method. However, the computational burden of such second-order method makes it inapplicable for large-scale problems. For instance, solving convex quadratic programs has cubic complexity [18]. Furthermore, to use these solvers, the original problems need to be transformed to the standard forms supported by the solvers. Although the parser/solver modeling frameworks like CVX [19] and YALMIP [20] can automatically transform the original problem instances into standard forms, it may require substantial time to perform such transformation [21], especially for problems with a large number of constraints [22].

One may also develop custom algorithms to enable efficient computation by exploiting the structures of specific problems. For instance, the uplink-downlink duality [8] is exploited to extract the structures of the optimal beamformers [23] and enable efficient algorithms. However, such an approach still

Copyright (c) 2015 IEEE. Personal use of this material is permitted. However, permission to use this material for any other purposes must be obtained from the IEEE by sending a request to pubs-permissions@ieee.org. This work is supported by the Hong Kong Research Grant Council under Grant No. 16200214.

Y. Shi, J. Zhang and K. B. Letaief are with the Department of Electronic and Computer Engineering, Hong Kong University of Science and Technology (e-mail: {yshi, eejzhang, eekhaled}@ust.hk).

B. O'Donoghue is with the Electrical Engineering Department, Stanford University, Stanford, CA 94305 USA (e-mail: bodono@stanford.edu).

has the cubic complexity to perform matrix inversion at each iteration [24]. First-order methods, e.g., the ADMM algorithm [25], have recently attracted attention for their distributed and parallelizable implementation, as well as the capability of scaling to large problem sizes. However, most existing ADMM based algorithms cannot provide the certificates of infeasibility [11], [24], [26]. Furthermore, some of them may still fail to scale to large problem sizes, due to the SOCP subproblems [26] or semidefinite programming (SDP) subproblems [11] 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 RAUs, or resort to sub-optimal algorithms, e.g., zero-forcing based approaches [27], [28]. Meanwhile, from the above discussion, we see that the large-scale optimization algorithms to be developed should possess the following two features:

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

To address these two challenges in a unified way, in this paper, we shall propose a two-stage approach as shown in Fig. 1. The proposed framework is capable to solve large-scale convex optimization problems in parallel, as well as providing certificates of infeasibility. Specifically, the original problem \mathcal{P} will be first transformed into a standard cone programming form $\mathcal{P}_{\text{cone}}$ [18] based on the Smith form reformulation [29], via introducing a new variable for each subexpression in the disciplined convex programming form [30] of the original problem. This will eventually transform the coupled constraints in the original problem into the constraint only consisting 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 the infeasibility detection capability *simultaneously* via solving the homogeneous self-dual embedding [31] of the primal-dual pair of the standard form by the ADMM algorithm.

As the mapping between the standard cone program and the original problem only depends on the network size (i.e., the numbers of RAUs, MUs and antennas at each RAU), we can pre-generate and store the structures of the standard forms with different candidate network sizes. Then for each problem instance, that is, given the channel coefficients, QoS requirements, and maximum RAU transmit powers, we only need to copy the original problem parameters to the standard cone programming data. Thus, the transformation procedure can be very efficient and can avoid repeatedly parsing and re-generating problems [19], [20]. This technique is called *matrix stuffing* [21], [22], which is essential for the proposed framework to scale well to large problem sizes. It may also help rapid prototyping and testing for practical equipment development.

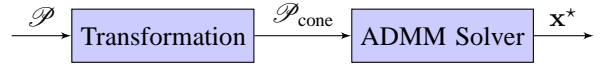


Fig. 1. The proposed two-stage approach for large-scale convex optimization. The optimal solution or the certificate of infeasibility can be extracted from \mathbf{x}^* by the ADMM solver.

A. Contributions

The major contributions of the paper are summarized as follows:

- 1) We formulate main performance optimization problems in dense wireless cooperative networks into a general framework. It is shown that all of them can essentially be solved through solving one or a sequence of large-scale convex optimization or convex feasibility problems.
- 2) To enable both the infeasibility detection capability and parallel computing capability, we propose to transform the original convex problem to an equivalent standard cone program. The transformation procedure scales very well to large problem sizes with the matrix stuffing technique. Simulation results will demonstrate the effectiveness of the proposed fast transformation approach over the state-of-art parser/solver modeling frameworks.
- 3) The operator splitting method is then adopted to solve the large-scale homogeneous self-dual embedding of the primal-dual pair of the transformed standard cone program in parallel. This first-order optimization algorithm makes the second stage scalable. Simulation results will show that it can speedup several orders of magnitude over the state-of-art interior-point solvers.
- 4) The proposed framework enables evaluating various cooperation strategies in dense wireless networks, and helps reveal new insights *numerically*. For instance, simulation results demonstrate a significant performance gain of optimal beamforming over sub-optimal schemes, which shows the importance of developing large-scale optimal beamforming algorithms.

This work will serve the purpose of providing practical and theoretical guidelines on designing algorithms for generic large-scale optimization problems in dense wireless networks.

B. Organization

The remainder of the paper is organized as follows. Section II presents the system model and problem formulations. In Section III, a systematic cone programming form transformation procedure is developed. The operator splitting method is presented in Section IV. The practical implementation issues are discussed in Section V. Numerical results will be demonstrated in Section VI. Finally, conclusions and discussions are presented in Section VII. To keep the main text clean and free of technical details, we divert most of the proofs, derivations to the appendix.

II. LARGE-SCALE OPTIMIZATION IN DENSE WIRELESS COOPERATIVE NETWORKS

In this section, we will first present two representative optimization problems in wireless cooperative networks, i.e., the network power minimization problem and the network

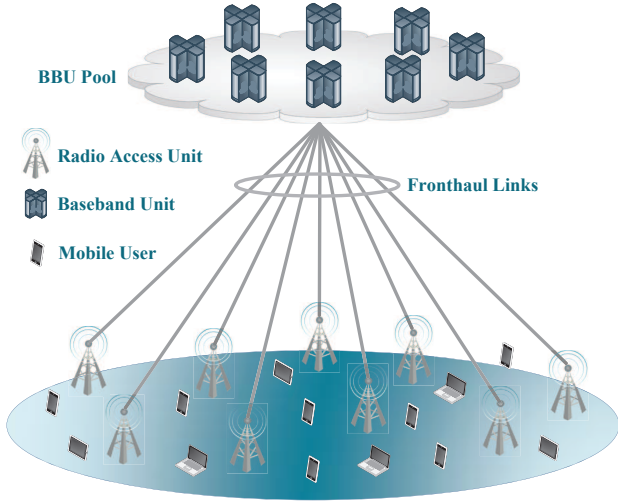


Fig. 2. The architecture of Cloud-RAN, in which, all the RAUs are connected to a BBU pool through high-capacity and low-latency optical fronthaul links. To enable full cooperation among RAUs, it is assumed that all the user data and CSI are available at the BBU pool.

utility maximization problem. We will then provide a unified formulation for large-scale optimization problems in dense wireless cooperative networks.

A. Signal Model

Consider a dense fully cooperative network¹ with L RAUs and K single-antenna MUs, where the l -th RAU is equipped with N_l antennas. The centralized signal processing is performed at a central processor, e.g., the baseband unit pool in Cloud-RAN [2], [3] as shown in Fig. 2. The propagation channel from the l -th RAU to the k -th MU is denoted as $\mathbf{h}_{kl} \in \mathbb{C}^{N_l}, \forall k, l$. In this paper, we focus on the downlink transmission for illustrative purpose. But our proposed approach can also be applied in the uplink transmission, as we only need to exploit the convexity of the resulting performance optimization problems. The received signal $y_k \in \mathbb{C}$ at MU k is given by

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

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 l -th RAU to the k -th MU, and $n_k \sim \mathcal{CN}(0, \sigma_k^2)$ is the additive Gaussian noise at MU k . We assume that s_k 's and n_k 's are mutually independent and all the users apply single user detection. Thus the signal-to-interference-plus-noise ratio (SINR) of MU k is given by

$$\Gamma_k(\mathbf{v}) = \frac{|\mathbf{h}_k^H \mathbf{v}_k|^2}{\sum_{i \neq k} |\mathbf{h}_k^H \mathbf{v}_i|^2 + \sigma_k^2}, \forall k, \quad (2)$$

¹The full cooperation among all the RAUs with global CSI and full user data sharing is used as an illustrative example. The proposed framework can be extended to more general cooperation scenarios, e.g., with partial user data sharing among RAUs as presented in [7, Section 1.3.1].

where $\mathbf{h}_k \triangleq [\mathbf{h}_{k1}^T, \dots, \mathbf{h}_{kL}^T]^T \in \mathbb{C}^N$ with $N = \sum_{l=1}^L N_l$, $\mathbf{v}_k \triangleq [\mathbf{v}_{1k}^T, \mathbf{v}_{2k}^T, \dots, \mathbf{v}_{Lk}^T]^T \in \mathbb{C}^N$ and $\mathbf{v} \triangleq [\mathbf{v}_1^T, \dots, \mathbf{v}_K^T]^T \in \mathbb{C}^{NK}$. We assume that each RAU has its own power constraint,

$$\sum_{k=1}^K \|\mathbf{v}_{lk}\|_2^2 \leq P_l, \forall l, \quad (3)$$

where $P_l > 0$ is the maximum transmit power of the l -th RAU. In this paper, we assume that the full and perfect CSI is available at the central processor and all RAUs only provide unicast/broadcast services.

B. Network Power Minimization

Network power consumption is an important performance metric for the energy efficiency design in wireless cooperative networks. Coordinated beamforming is an efficient way to design energy-efficient systems [8], in which, beamforming vectors \mathbf{v}_{lk} 's are designed to minimize the total transmit power among RAUs while satisfying the QoS requirements for all the MUs. Specifically, given the target SINRs $\boldsymbol{\gamma} = (\gamma_1, \dots, \gamma_K)$ for all the MUs with $\gamma_k > 0, \forall k$, we will solve the following total transmit power minimization problem:

$$\mathcal{P}_1(\boldsymbol{\gamma}) : \underset{\mathbf{v} \in \mathcal{V}}{\text{minimize}} \sum_{l=1}^L \sum_{k=1}^K \|\mathbf{v}_{lk}\|_2^2, \quad (4)$$

where \mathcal{V} is the intersection of the sets formed by transmit power constraints and QoS constraints, i.e.,

$$\mathcal{V} = \mathcal{P}_1 \cap \mathcal{P}_2 \cap \dots \cap \mathcal{P}_L \cap \mathcal{Q}_1 \cap \mathcal{Q}_2 \dots \cap \mathcal{Q}_K, \quad (5)$$

where \mathcal{P}_l 's are feasible sets of \mathbf{v} that satisfy the per-RAU transmit power constraints, i.e.,

$$\mathcal{P}_l = \left\{ \mathbf{v} \in \mathbb{C}^{NK} : \sum_{k=1}^K \|\mathbf{v}_{lk}\|_2^2 \leq P_l \right\}, \forall l, \quad (6)$$

and \mathcal{Q}_k 's are the feasible sets of \mathbf{v} that satisfy the per-MU QoS constraints, i.e.,

$$\mathcal{Q}_k = \{ \mathbf{v} \in \mathbb{C}^{NK} : \Gamma_k(\mathbf{v}) \geq \gamma_k \}, \forall k. \quad (7)$$

As all the sets \mathcal{Q}_k 's and \mathcal{P}_l 's can be reformulated into second-order cones as shown in [3], problem $\mathcal{P}_1(\boldsymbol{\gamma})$ can be reformulated as an SOCP problem.

However, in dense wireless cooperative networks, the mobile hauling network consumption can not be ignored. In [3], a two-stage group sparse beamforming (GSBF) framework is proposed to minimize the network power consumption for Cloud-RAN, including the power consumption of all optical fronthaul links and the transmit power consumption of all RAUs. Specially, in the first stage, the group-sparsity structure of the aggregated beamformer \mathbf{v} is induced by minimizing the weighted mixed ℓ_1/ℓ_2 -norm of \mathbf{v} , i.e.,

$$\mathcal{P}_2(\boldsymbol{\gamma}) : \underset{\mathbf{v} \in \mathcal{V}}{\text{minimize}} \sum_{l=1}^L \omega_l \|\tilde{\mathbf{v}}_l\|_2, \quad (8)$$

where $\tilde{\mathbf{v}}_l = [\mathbf{v}_{l1}^T, \dots, \mathbf{v}_{lK}^T]^T \in \mathbb{C}^{N_l K}$ is the aggregated beamforming vector at RAU l , and $\omega_l > 0$ is the corresponding weight for the beamformer coefficient group $\tilde{\mathbf{v}}_l$. Based on

the (approximated) group sparse beamformer \mathbf{v}^* , which is the optimal solution to $\mathcal{P}_2(\gamma)$, in the second stage, an RAU selection procedure is performed to switch off some RAUs so as to minimize the network power consumption. In this procedure, we need to check if the remaining RAUs can support the QoS requirements for all the MUs, i.e., check the feasibility of problem $\mathcal{P}_1(\gamma)$ given the active RAUs. Please refer to [3] for more details on the group sparse beamforming algorithm.

C. Network Utility Maximization

Network utility maximization is a general approach to optimize network performance. We consider maximizing an arbitrary network utility function $U(\Gamma_1(\mathbf{v}), \dots, \Gamma_K(\mathbf{v}))$ that is strictly increasing in the SINR of each MU [7], i.e.,

$$\mathcal{P}_3 : \underset{\mathbf{v} \in \mathcal{V}_1}{\text{maximize}} \quad U(\Gamma_1(\mathbf{v}), \dots, \Gamma_K(\mathbf{v})), \quad (9)$$

where $\mathcal{V}_1 = \cap_{l=1}^L \mathcal{P}_l$ is the intersection of the sets of the per-RAU transmit power constraints (6). It is generally very difficult to solve, though there are tremendous research efforts on this problem [7]. In particular, Liu *et al.* in [32] proved that \mathcal{P}_3 is NP-hard for many common utility functions, e.g., weighted sum-rate. Please refer to [7, Table 2.1] for details on classification of the convexity of utility optimization problems.

Assume that we have the prior knowledge of SINR values $\Gamma_1^*, \dots, \Gamma_K^*$ that can be achieved by the optimal solution to problem \mathcal{P}_3 . Then the optimal solution to problem $\mathcal{P}_1(\gamma)$ with target SINRs as $\gamma = (\Gamma_1^*, \dots, \Gamma_K^*)$ is an optimal solution to problem \mathcal{P}_3 as well [23]. The difference between problem $\mathcal{P}_1(\gamma)$ and problem \mathcal{P}_3 is that the SINRs in $\mathcal{P}_1(\gamma)$ are pre-defined, while the optimal SINRs in \mathcal{P}_3 need to be searched. For the max-min fairness maximization problem, optimal SINRs can be searched by the bi-section method [22], which can be accomplished in polynomial time. For the general increasing utility maximization problem \mathcal{P}_3 , the corresponding optimal SINRs can be searched as follows

$$\underset{\gamma \in \mathcal{R}}{\text{maximize}} \quad U(\gamma_1, \dots, \gamma_K), \quad (10)$$

where $\mathcal{R} \in \mathbb{R}_+^K$ is the achievable performance region

$$\mathcal{R} = \{(\Gamma_1(\mathbf{v}), \dots, \Gamma_K(\mathbf{v})) : \mathbf{v} \in \mathcal{V}_1\}. \quad (11)$$

Problem (10) is a monotonic optimization problem [33] and thus can be solved by the polyblock outer approximation algorithm [33] or the branch-reduce-and-bound algorithm [7]. The general idea of both algorithms is iteratively improving the lower-bound U_{\min} and upper-bound U_{\max} of the objective function of problem (10) such that

$$U_{\max} - U_{\min} \leq \epsilon, \quad (12)$$

for a given accuracy ϵ in finite iterations. In particular, at the m -iteration, we need to check the convex feasibility problem of $\mathcal{P}_1(\gamma^{[m]})$ given the target SINRs $\gamma^{[m]} = (\Gamma_1^{[m]}, \dots, \Gamma_K^{[m]})$. However, the number of iterations scales exponentially with the number of MUs [7]. Please refer to the tutorial [7, Section 2.3] for more details. Furthermore, the network achievable rate region [34] can also be characterized by the rate profile method [35] via solving a sequence of such convex feasibility problems $\mathcal{P}_1(\gamma)$.

D. A Unified Framework of Large-Scale Network Optimization

In dense wireless cooperative networks, the central processor can support hundreds of RAUs for simultaneously transmission/reception [2]. Therefore, all the above optimization problems are shifted into a new domain with a high problem dimension and a large number of constraints. As presented previously, to solve the performance optimization problems, we essentially need to solve a sequence of the following convex optimization problem with different problem instances (e.g., different channel realizations, network sizes and QoS targets)

$$\mathcal{P} : \underset{\mathbf{v} \in \mathcal{V}}{\text{minimize}} \quad f(\mathbf{v}), \quad (13)$$

where $f(\mathbf{v})$ is convex in \mathbf{v} as shown in $\mathcal{P}_1(\gamma)$ and $\mathcal{P}_2(\gamma)$. Solving problem \mathcal{P} means that the corresponding algorithm should return the optimal solution or the certificate of infeasibility.

For all the problems discussed above, problem \mathcal{P} can be reformulated as an SOCP problem, and thus it can be solved in polynomial time via the interior-point method, which is implemented in most advanced off-the-shelf solvers, e.g., public software packages like SeDuMi [15] and SDPT3 [16] and commercial software packages like MOSEK [17]. However, the computational cost of such second-order methods will be prohibitive for large-scale problems. On the other hand, most custom algorithms, e.g., the uplink-downlink approach [8] and the ADMM based algorithms [11], [24], [26], however, fail to either scale well to large problem sizes or detect the infeasibility effectively.

To overcome the limitations of the scalability of the state-of-art solvers and the capability of infeasibility detection of the custom algorithms, in this paper, we propose to solve the homogeneous self-dual embedding [31] (which aims at providing necessary certificates) of problem \mathcal{P} via a first-order optimization method [25] (i.e., the operator splitting method). This will be presented in Section IV. To arrive at the homogeneous self-dual embedding and enable parallel computing, the original problem will be first transformed into a standard cone programming form as will be presented in Section III. This forms the main idea of the two-stage based large-scale optimization framework as shown in Fig. 1.

III. MATRIX STUFFING FOR FAST STANDARD CONE PROGRAMMING TRANSFORMATION

Although the parser/solver modeling language framework, like CVX [19] and YALMIP [20], can automatically transform the original problem instance into a standard form, it requires substantial time to accomplish this procedure [21], [22]. In particular, for each problem instance, the parser/solver modeling frameworks need to repeatedly parse and canonicalize it. To avoid such modeling overhead of reading problem data and repeatedly parsing and canonicalizing, we propose to use the matrix stuffing technique [21], [22] to perform fast transformation by exploiting the problem structures. Specifically, we will first generate the mapping from the original problem to the cone program, and then the structure of the standard form

will be stored. This can be accomplished offline. Therefore, for each problem instance, we only need to stuff its parameters to data of the corresponding pre-stored structure of the standard cone program. Similar ideas were presented in the emerging parse/generator modeling frameworks like CVXGEN [36] and QCML [21], which aim at embedded applications for some specific problem families. In this paper, we will demonstrate in Section VI that matrix stuffing is essential to scale to large problem sizes for fast transformation at the first stage of the proposed framework.

A. Conic Formulation of Convex Programs

In this section, we describe a systematic way to transform the original problem \mathcal{P} to the standard cone program. To enable parallel computing, a common way is to replicate some variables through either exploiting problem structures [11], [24] or using the consensus formulation [25], [26]. However, directly working on these reformulations is difficult to provide computable mathematical certificates of infeasibility. Therefore, heuristic criteria are often adopted to detect the infeasibility, e.g., the underlying problem instance is reported to be infeasible when the algorithm exceeds the pre-defined maximum iterations without convergence [24]. To unify the requirements of parallel and scalable computing and to provide computable mathematical certificates of infeasibility, in this paper, we propose to transform the original problem \mathcal{P} to the following equivalent cone program $\mathcal{P}_{\text{cone}}$:

$$\begin{aligned} \mathcal{P}_{\text{cone}} : \underset{\nu, \mu}{\text{minimize}} \quad & \mathbf{c}^T \nu \\ \text{subject to} \quad & \mathbf{A}\nu + \mu = \mathbf{b} \\ & (\nu, \mu) \in \mathbb{R}^n \times \mathcal{K}, \end{aligned} \quad (14)$$

where $\nu \in \mathbb{R}^n$ and $\mu \in \mathbb{R}^m$ are the optimization variables, $\mathcal{K} = \{0\}^r \times \mathcal{S}^{m_1} \times \dots \times \mathcal{S}^{m_q}$ with \mathcal{S}^p as the standard second-order cone of dimension p

$$\mathcal{S}^p = \{(y, \mathbf{x}) \in \mathbb{R} \times \mathbb{R}^{p-1} \mid \|\mathbf{x}\| \leq y\}, \quad (16)$$

and \mathcal{S}^1 is defined as the cone of nonnegative reals, i.e., \mathbb{R}_+ . Here, each \mathcal{S}^i has dimension m_i such that $(r + \sum_{i=1}^q m_i) = m$, $\mathbf{A} \in \mathbb{R}^{m \times n}$, $\mathbf{b} \in \mathbb{R}^m$, $\mathbf{c} \in \mathbb{R}^n$. The equivalence means that the optimal solution or the certificate of infeasibility of the original problem \mathcal{P} can be extracted from the solution to the equivalent cone program $\mathcal{P}_{\text{cone}}$. To reduce the storage and memory overhead, we store the matrix \mathbf{A} , vectors \mathbf{b} and \mathbf{c} in the sparse form [37] by only storing the non-zero entries.

The general idea of such transformation is to rewrite the original problem \mathcal{P} into a Smith form by introducing a new variable for each subexpression in disciplined convex programming form [30] of problem \mathcal{P} . The details are presented in the Appendix. Working with this transformed standard cone program $\mathcal{P}_{\text{cone}}$ has the following two advantages:

- The homogeneous self-dual embedding of the primal-dual pair of the standard cone program can be induced, thereby providing certificates of infeasibility. This will be presented in Section IV-A.
- The feasible set \mathcal{V} (5) formed by the intersection of a finite number of constraint sets \mathcal{P}_l 's and \mathcal{Q}_k 's in the

original problem \mathcal{P} can be transformed into two sets in $\mathcal{P}_{\text{cone}}$: a subspace (14) and a convex cone \mathcal{K} , which is formed by the Cartesian product of second-order cones. This salient feature will be exploited to enable parallel and scalable computing, as will be presented in Section IV-B.

B. Matrix Stuffing for Fast Transformation

Inspired by the work [21] on fast optimization code deployment for embedding second-order cone program, we propose to use the matrix stuffing technique [21], [22] to transform the original problem into the standard cone program quickly. Specifically, for any given network size, we first generate and store the structure that maps the original problem \mathcal{P} to the standard form $\mathcal{P}_{\text{cone}}$. 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 $[r, m_1, m_2, \dots, m_q]$), and the symbolic problem parameters \mathbf{A} , \mathbf{b} and \mathbf{c} . This procedure can be done offline.

Based on the pre-stored structure, for a given problem instance \mathcal{P} , we only need to copy its parameters (i.e., the channel coefficients \mathbf{h}_K 's, maximum transmit powers P_l 's, SINR targets γ_k 's) to the corresponding data in the standard form $\mathcal{P}_{\text{cone}}$ (i.e., \mathbf{A} and \mathbf{b}). Details of the exact description of copying data for transformation are presented in the Appendix. As the procedure for transformation only needs 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.

Remark 1: As shown in the Appendix, the dimension of the transformed standard cone program $\mathcal{P}_{\text{cone}}$ becomes $m = (L + K) + (2NK + 1) + \sum_{l=1}^L (2KN_l + 1) + K(2K + 2)$, which is much larger than the dimension of the original problem, i.e., $2NK$ in the equivalent real-field. But as discussed above, there are unique advantages of working with this standard form, which compensate for the increase in the size, as will be explicitly presented in later sections.

IV. THE OPERATOR SPLITTING METHOD FOR LARGE-SCALE HOMOGENEOUS SELF-DUAL EMBEDDING

Although the standard cone program $\mathcal{P}_{\text{cone}}$ itself is suitable for parallel computing via the operator splitting method [38], directly working on this problem may fail to provide certificates of infeasibility. To address this limitation, based on the recent work by O'Donoghue *et al.* [39], we propose to solve the homogeneous self-dual embedding [31] of the primal-dual pair of the cone program $\mathcal{P}_{\text{cone}}$. The resultant homogeneous self-dual embedding is further solved via the operator splitting method, a.k.a. the ADMM algorithm [25].

A. Homogeneous Self-Dual Embedding of Cone Programming

The basic idea of the homogeneous self-dual embedding is to embed the primal and dual problems of the cone program $\mathcal{P}_{\text{cone}}$ 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 $\mathcal{P}_{\text{cone}}$ can be extracted from the solution of the embedded problem.

The dual problem of $\mathcal{P}_{\text{cone}}$ is given by [39]

$$\begin{aligned} \mathcal{D}_{\text{cone}} : \text{maximize}_{\boldsymbol{\eta}, \boldsymbol{\lambda}} \quad & -\mathbf{b}^T \boldsymbol{\eta} \\ \text{subject to} \quad & -\mathbf{A}^T \boldsymbol{\eta} + \boldsymbol{\lambda} = \mathbf{c} \\ & (\boldsymbol{\lambda}, \boldsymbol{\eta}) \in \{0\}^n \times \mathcal{K}^*, \end{aligned} \quad (17)$$

where $\boldsymbol{\lambda} \in \mathbb{R}^n$ and $\boldsymbol{\eta} \in \mathbb{R}^m$ are the dual variables, \mathcal{K}^* is the dual cone of the convex cone \mathcal{K} . Note that $\mathcal{K} = \mathcal{K}^*$, i.e., \mathcal{K} is self dual. Define the optimal values of the primal program $\mathcal{P}_{\text{cone}}$ and dual program $\mathcal{D}_{\text{cone}}$ are p^* and d^* , respectively. Let $p^* = +\infty$ and $p^* = -\infty$ indicate primal infeasibility and unboundedness, respectively. Similarly, let $d^* = -\infty$ and $d^* = +\infty$ indicate the dual infeasibility and unboundedness, respectively. We assume strong duality for the convex cone program $\mathcal{P}_{\text{cone}}$, i.e., $p^* = d^*$, including cases when they are infinite. This is a standard assumption for practically designing solvers for conic programs, e.g., it is assumed in [15], [16], [17], [31], [39]. Besides, we do not make any regularity assumption on the feasibility and boundedness assumptions on the primal and dual problems.

1) *Certificates of Infeasibility*: Given the cone program $\mathcal{P}_{\text{cone}}$, a main task is to detect feasibility. In [40, Theorem 1], a sufficient condition for the existence of strict feasible solution was provided for the transmit power minimization problem without power constraints. However, for the general problem \mathcal{P} with per-MU QoS constraints and per-RAU transmit power constraints, it is difficult to obtain such a feasibility condition analytically. Therefore, most existing works either assume that the underlying problem is feasible [8] or provide heuristic ways to handle infeasibility [24].

Nevertheless, the only way to detect infeasibility effectively is to provide a certificate or proof of infeasibility as presented in the following proposition.

Proposition 1: [Certificates of Infeasibility] The following system

$$\mathbf{A}\boldsymbol{\nu} + \boldsymbol{\mu} = \mathbf{b}, \boldsymbol{\mu} \in \mathcal{K}, \quad (18)$$

is infeasible if and only if the following system is feasible

$$\mathbf{A}^T \boldsymbol{\eta} = \mathbf{0}, \boldsymbol{\eta} \in \mathcal{K}^*, \mathbf{b}^T \boldsymbol{\eta} < 0. \quad (19)$$

Therefore, any dual variable $\boldsymbol{\eta}$ satisfying the system (19) provides a certificate or proof that the primal program $\mathcal{P}_{\text{cone}}$ (equivalently the original problem \mathcal{P}) is infeasible.

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

$$-\mathbf{A}\boldsymbol{\nu} \in \mathcal{K}, \mathbf{c}^T \boldsymbol{\nu} < 0, \quad (20)$$

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

Proof: This result directly follows the theorem of strong alternatives [9, Section 5.8.2]. ■

2) *Optimality Conditions*: If the transformed standard cone program $\mathcal{P}_{\text{cone}}$ is feasible, then $(\boldsymbol{\nu}^*, \boldsymbol{\mu}^*, \boldsymbol{\lambda}^*, \boldsymbol{\eta}^*)$ are optimal if and only if they satisfy the following Karush-Kuhn-Tucker

(KKT) conditions

$$\mathbf{A}\boldsymbol{\nu}^* + \boldsymbol{\mu}^* - \mathbf{b} = \mathbf{0} \quad (21)$$

$$\mathbf{A}^T \boldsymbol{\eta}^* - \boldsymbol{\lambda}^* + \mathbf{c} = \mathbf{0} \quad (22)$$

$$(\boldsymbol{\eta}^*)^T \boldsymbol{\mu}^* = 0 \quad (23)$$

$$(\boldsymbol{\nu}^*, \boldsymbol{\mu}^*, \boldsymbol{\lambda}^*, \boldsymbol{\eta}^*) \in \mathbb{R}^n \times \mathcal{K} \times \{0\}^n \times \mathcal{K}^*. \quad (24)$$

In particular, the complementary slackness condition (23) can be rewritten as

$$\mathbf{c}^T \boldsymbol{\nu}^* + \mathbf{b}^T \boldsymbol{\eta}^* = 0, \quad (25)$$

which explicitly forces the duality gap to be zero.

3) *Homogeneous Self-Dual Embedding*: We can first detect feasibility by Proposition 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 [31]. To avoid such inefficiency, we propose to solve the following homogeneous self-dual embedding [31]:

$$\mathbf{A}\boldsymbol{\nu} + \boldsymbol{\mu} - \mathbf{b}\tau = \mathbf{0} \quad (26)$$

$$\mathbf{A}^T \boldsymbol{\eta} - \boldsymbol{\lambda} + \mathbf{c}\tau = \mathbf{0} \quad (27)$$

$$\mathbf{c}^T \boldsymbol{\nu} + \mathbf{b}^T \boldsymbol{\eta} + \kappa = 0 \quad (28)$$

$$(\boldsymbol{\nu}, \boldsymbol{\mu}, \boldsymbol{\lambda}, \boldsymbol{\eta}, \tau, \kappa) \in \mathbb{R}^n \times \mathcal{K} \times \{0\}^n \times \mathcal{K}^* \times \mathbb{R}_+ \times \mathbb{R}_+, \quad (29)$$

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

$$\begin{aligned} \mathcal{F}_{\text{HSD}} : \text{find } & (\mathbf{x}, \mathbf{y}) \\ \text{subject to } & \mathbf{y} = \mathbf{Q}\mathbf{x} \\ & \mathbf{x} \in \mathcal{C}, \mathbf{y} \in \mathcal{C}^*, \end{aligned} \quad (30)$$

where

$$\underbrace{\begin{bmatrix} \boldsymbol{\lambda} \\ \boldsymbol{\mu} \\ \kappa \end{bmatrix}}_{\mathbf{y}} = \underbrace{\begin{bmatrix} \mathbf{0} & \mathbf{A}^T & \mathbf{c} \\ -\mathbf{A} & \mathbf{0} & \mathbf{b} \\ -\mathbf{c}^T & -\mathbf{b}^T & \mathbf{0} \end{bmatrix}}_{\mathbf{Q}} \underbrace{\begin{bmatrix} \boldsymbol{\nu} \\ \boldsymbol{\eta} \\ \tau \end{bmatrix}}_{\mathbf{x}}, \quad (31)$$

$\mathbf{x} \in \mathbb{R}^{m+n+1}$, $\mathbf{y} \in \mathbb{R}^{m+n+1}$, $\mathbf{Q} \in \mathbb{R}^{(m+n+1) \times (m+n+1)}$, $\mathcal{C} = \mathbb{R}^n \times \mathcal{K}^* \times \mathbb{R}_+$ 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 \mathcal{F}_{HSD} is thus a feasibility problem finding a nonzero 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 [31]:

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

$$\hat{\boldsymbol{\nu}} = \boldsymbol{\nu}/\tau, \hat{\boldsymbol{\eta}} = \boldsymbol{\eta}/\tau, \hat{\boldsymbol{\mu}} = \boldsymbol{\mu}/\tau \quad (32)$$

are the primal and dual solutions to the cone program $\mathcal{P}_{\text{cone}}$.

- **Case 2**: $\tau = 0$, $\kappa > 0$; this implies $\mathbf{c}^T \boldsymbol{\nu} + \mathbf{b}^T \boldsymbol{\eta} < 0$, then

- 1) If $\mathbf{b}^T \boldsymbol{\eta} < 0$, then $\hat{\boldsymbol{\eta}} = \boldsymbol{\eta}/(-\mathbf{b}^T \boldsymbol{\eta})$ is a certificate of the primal infeasibility as

$$\mathbf{A}^T \hat{\boldsymbol{\eta}} = \mathbf{0}, \hat{\boldsymbol{\eta}} \in \mathcal{V}^*, \mathbf{b}^T \hat{\boldsymbol{\eta}} = -1. \quad (33)$$

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

$$-\mathbf{A} \hat{\boldsymbol{\nu}} \in \mathcal{V}, \mathbf{c}^T \hat{\boldsymbol{\nu}} = -1. \quad (34)$$

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

Therefore, from the solution to the homogeneous self-dual embedding, we can extract either the optimal solution (based on (60)) or the certificate of infeasibility for the original problem. Furthermore, as the set (29) is a Cartesian product of a finite number of sets, this will enable parallelizable algorithm design. With the distinct advantages of the homogeneous self-dual embedding, in the sequel, we focus on developing efficient algorithms to solve the large-scale feasibility problem \mathcal{F}_{HSD} via the operator splitting method.

B. The Operator Splitting Method

Conventionally, the convex homogeneous self-dual embedding \mathcal{F}_{HSD} can be solved via the interior-point method, e.g., [15], [16], [17], [31]. However, such second-order method has cubic computational complexity for the second-order cone programs [18], and thus the computational cost will be prohibitive for large-scale problems. Instead, O'Donoghue *et al.* [39] develop a first-order optimization algorithm based on the operator splitting method, i.e., the ADMM algorithm [25], to solve the large-scale homogeneous self-dual embedding. The key observation is that the convex cone constraint in \mathcal{F}_{HSD} is the Cartesian product of standard convex cones (i.e., second-order cones, nonnegative reals and free variables), which enables parallelizable computing. Furthermore, we will show that the computation of each iteration in the operator splitting method is very cheap and efficient.

Specifically, the homogeneous self-dual embedding \mathcal{F}_{HSD} can be rewritten as

$$\text{minimize } I_{\mathcal{C} \times \mathcal{C}^*}(\mathbf{x}, \mathbf{y}) + I_{\mathbf{Q}\mathbf{x}=\mathbf{y}}(\mathbf{x}, \mathbf{y}), \quad (35)$$

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 it is $+\infty$ otherwise. By replicating variables \mathbf{x} and \mathbf{y} , problem (35) can be transformed into the following consensus form [25, Section 7.1]

$$\begin{aligned} \mathcal{P}_{\text{ADMM}} : \text{minimize } & I_{\mathcal{C} \times \mathcal{C}^*}(\mathbf{x}, \mathbf{y}) + I_{\mathbf{Q}\tilde{\mathbf{x}}=\tilde{\mathbf{y}}}(\tilde{\mathbf{x}}, \tilde{\mathbf{y}}) \\ \text{subject to } & (\mathbf{x}, \mathbf{y}) = (\tilde{\mathbf{x}}, \tilde{\mathbf{y}}), \end{aligned} \quad (36)$$

which is readily to be solved by the operator splitting method.

Applying the ADMM algorithm [25, Section 3.1] to problem $\mathcal{P}_{\text{ADMM}}$ and eliminating the dual variables by exploiting the self-dual property of the problem \mathcal{F}_{HSD} (Please refer to [39, Section 3] on how to simplify the ADMM algorithm), the final algorithm is shown as follows:

$$\mathcal{OS}_{\text{ADMM}} : \begin{cases} \tilde{\mathbf{x}}^{[i+1]} = (\mathbf{I} + \mathbf{Q})^{-1}(\mathbf{x}^{[i]} + \mathbf{y}^{[i]}) \\ \mathbf{x}^{[i+1]} = \Pi_{\mathcal{C}}(\tilde{\mathbf{x}}^{[i+1]} - \mathbf{y}^{[i]}) \\ \mathbf{y}^{[i+1]} = \mathbf{y}^{[i]} - \tilde{\mathbf{x}}^{[i+1]} + \mathbf{x}^{[i+1]}, \end{cases} \quad (37)$$

where $\Pi_{\mathcal{C}}(\mathbf{x})$ denotes the Euclidean projection of \mathbf{x} onto the set \mathcal{C} . This algorithm has the $\mathcal{O}(1/k)$ convergence rate [41] with k as the iteration counter (i.e., the ϵ accuracy can be achieved in $\mathcal{O}(1/\epsilon)$ iterations) and will not converge to zero if a nonzero solution exists [39, 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.

1) *Subspace Projection via Factorization Caching:* The first step in the algorithm $\mathcal{OS}_{\text{ADMM}}$ is a subspace projection. After simplification [39, Section 4], we essentially need to solve the following linear equation at each iteration, i.e.,

$$\underbrace{\begin{bmatrix} \mathbf{I} & -\mathbf{A}^T \\ -\mathbf{A} & -\mathbf{I} \end{bmatrix}}_{\mathbf{S}} \underbrace{\begin{bmatrix} \boldsymbol{\nu} \\ -\boldsymbol{\eta} \end{bmatrix}}_{\mathbf{x}} = \underbrace{\begin{bmatrix} \boldsymbol{\nu}^{[i]} \\ \boldsymbol{\eta}^{[i]} \end{bmatrix}}_{\mathbf{b}}, \quad (38)$$

for the given $\boldsymbol{\nu}^{[i]}$ 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 [42]. To enable quicker inversions and reduce memory overhead via exploiting the sparsity of the matrix \mathbf{S} , the sparse permuted LDL^T factorization [37] method can be adopted. Specifically, such factor-solve method can be carried out by first computing the sparse permuted LDL^T factorization as follows

$$\mathbf{S} = \mathbf{P}\mathbf{L}\mathbf{D}\mathbf{L}^T\mathbf{P}^T, \quad (39)$$

where \mathbf{L} is a lower triangular matrix, \mathbf{D} is a diagonal matrix [38] and \mathbf{P} with $\mathbf{P}^{-1} = \mathbf{P}^T$ is a permutation matrix to fill-in of the factorization [37], i.e., the number of nonzero entries in \mathbf{L} . Such factorization exists for any permutation \mathbf{P} , as the matrix \mathbf{S} is symmetric quasidefinite [42, 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 such factorization only needs to be computed once in the first iteration and can be cached for re-using in the sequent iterations for subspace projections. This is called the *factorization caching* technique [39].

Given the cached factorization (39), solving subsequent projections $\mathbf{x} = \mathbf{S}^{-1}\mathbf{b}$ (38) can be carried out by solving the following much easier equations:

$$\mathbf{P}\mathbf{x}_1 = \mathbf{b}, \mathbf{L}\mathbf{x}_2 = \mathbf{x}_1, \mathbf{D}\mathbf{x}_3 = \mathbf{x}_2, \mathbf{L}^T\mathbf{x}_4 = \mathbf{x}_3, \mathbf{P}^T\mathbf{x} = \mathbf{x}_4, \quad (40)$$

which cost zero flops, $\mathcal{O}(sd)$ flops by forward substitution with s as the number of nonzero entries in \mathbf{L} , $\mathcal{O}(d)$ flops, $\mathcal{O}(sd)$ flops by backward substitution, and zero flops, respectively [9, Appendix C].

2) *Cone Projection via Proximal Operator Evaluation:* 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 the finite number of 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. Specifically, for nonnegative real $\mathcal{C}_i = \mathbb{R}_+$, we have that [43, Section 6.3.1]

$$\Pi_{\mathcal{C}_i}(\boldsymbol{\omega}) = \boldsymbol{\omega}_+, \quad (41)$$

where the nonnegative part operator $(\cdot)_+$ is taken elementwise. For the second-order cone $\mathcal{C}_i = \{(y, \mathbf{x}) \in \mathbb{R} \times \mathbb{R}^{p-1} \mid \|\mathbf{x}\| \leq y\}$, we have that [43, Section 6.3.2]

$$\Pi_{\mathcal{C}_i}(\boldsymbol{\omega}, \tau) = \begin{cases} 0, \|\boldsymbol{\omega}\|_2 \leq -\tau \\ (\boldsymbol{\omega}, \tau), \|\boldsymbol{\omega}\|_2 \leq \tau \\ (1/2)(1 + \tau/\|\boldsymbol{\omega}\|_2)(\boldsymbol{\omega}, \|\boldsymbol{\omega}\|_2), \|\boldsymbol{\omega}\|_2 \geq |\tau|. \end{cases} \quad (42)$$

In summary, we have presented that each step in the algorithm $\mathcal{OS}_{\text{ADMM}}$ can be computed efficiently. In particular, from both (41) and (42), we see that the cone projection can be carried out very efficiently with closed-forms, leading to parallelizable algorithms.

V. PRACTICAL IMPLEMENTATION ISSUES

In previous sections, we have presented the unified two-stage framework for large-scale convex optimization in dense wireless cooperative networks. In this section, we will focus on the implementation issues of the proposed framework.

A. Automatic Code Generation for Fast Transformation

In the Appendix, we describe a systematic way to transform the original problem to the standard cone programming form. The resultant structure that maps the original problem to the standard form can be stored and re-used for fast transforming via matrix stuffing. This can significantly reduce the modeling overhead compared with the parse/solver modeling frameworks like CVX. However, it requires tedious manual works to find the mapping and may not be easy to verify the correctness of the generated mapping. Chu *et al.* [21] gave such an attempt intending to automatically generate the code for matrix stuffing. However, the corresponding software package QCML [21], so far, is far from complete and may not be suitable for our applications. Extending the numerical-based transformation modeling frameworks like CVX to the symbolic-based transformation modeling frameworks like QCML is not trivial and requires tremendous mathematical and technical efforts. In this paper, we derive the mapping in the Appendix manually and verify the correctness by comparing with CVX through extensive simulations.

B. Implementation of the Operator Splitting Algorithm

Theoretically, the presented operator splitting algorithm $\mathcal{OS}_{\text{ADMM}}$ is compact, parameter-free, with parallelizable computing and linear convergence. Practically, there are typically several ways to improve the efficiency of the algorithm. In particular, there are various tricks that can be employed to improve the convergence rate, e.g., over-relaxation, warm-starting and problem data scaling as described in [39]. In the dense wireless cooperative networks with multi-entity collaborative architecture, we are interested in two particular ways to speed up the subspace projection of the algorithm $\mathcal{OS}_{\text{ADMM}}$, which is the main computational bottleneck. Specifically, one way is to use the parallel algorithms for the factorization (39) by utilizing the distributed computing and memory resources [44]. For instance, in the cloud computing environments in Cloud-RAN, all the baseband units share the computing, memory and storage resources in a single baseband unit pool [2],

[3]. Another way is to leverage *symbolic* factorization (39) to speed up the numerical factorization for each problem instance, which is a general idea for the code generation system CVXGEN [36] for realtime convex quadratic optimization [45] and the interior-point method based SOCP solver [46] for embedded systems. Eventually, the ADMM solver in Fig. 1 can be symbolic based so as to provide numerical solutions for each problem instance extremely fast and in a realtime way. However, this requires further investigation.

VI. NUMERICAL RESULTS

In this section, we simulate the proposed two-stage based large-scale convex optimization framework for performance optimization in dense wireless cooperative networks. *The corresponding MATLAB code that can reproduce all the simulation results using the proposed large-scale convex optimization algorithm is available online².*

We consider the following channel model for the link between the k -th MU and the l -th RAU:

$$\mathbf{h}_{kl} = 10^{-L(d_{kl})/20} \sqrt{\varphi_{kl} s_{kl}} \mathbf{f}_{kl}, \forall k, l, \quad (43)$$

where $L(d_{kl})$ is the path-loss in dB at distance d_{kl} as shown in [3, 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 use the standard cellular network parameters as showed in [3, Table I]. All the simulations are carried out on a personal computer with 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 online³, which is a general software package for solving large-scale convex cone problems based on [39] and can be called by the modeling frameworks CVX and CVXPY [47]. The settings (e.g., the stopping criteria) of SCS can be found in [39].

The proposed two-stage approach framework, termed “**Matrix Stuffing+SCS**”, is compared with the following state-of-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 \mathcal{P} written in the disciplined convex programming form) into the standard cone programming form and then call an interior-point solver, e.g., SeDuMi [15], SDPT3 [16] or MOSEK [17].
- **CVX+SCS**: In this first-order method based framework, 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 for the second stage, and the “**total time**” as the time of the two stages for solving one problem instance. As the large-scale convex optimization algorithm 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.

²<https://github.com/SHIYUANMING/large-scale-convex-optimization>

³<https://github.com/cvxgrp/scs>

Given the network size, we first generate and store the problem structure of the standard form $\mathcal{P}_{\text{cone}}$, i.e., the structure of \mathbf{A} , \mathbf{b} , \mathbf{c} and the descriptions of \mathcal{V} . As this procedure can be done offline for all the candidate network sizes, we thus ignore this step for time comparison. We repeat the following procedures to solve the large-scale convex optimization problem \mathcal{P} with different parameters and sizes using the proposed framework “Matrix Stuffing+SCS”:

- 1) Copy the parameters in the problem instance \mathcal{P} to the data in the pre-stored structure of the standard cone program $\mathcal{P}_{\text{cone}}$.
- 2) Solve the resultant standard cone programming instance $\mathcal{P}_{\text{cone}}$ using the solver SCS.
- 3) Extract the optimal solutions of \mathcal{P} from the solutions to $\mathcal{P}_{\text{cone}}$ 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 aims at scaling well to large problem sizes and thus provides solutions to modest accuracy within reasonable time, while the interior-point method intends to provide highly accurate solutions. Furthermore, the modeling framework CVX aims at rapid prototyping and providing a user-friendly tool for automatically transformations for general problems, while the matrix-stuffing technique targets at scaling to large-scale problems for the specific problem family \mathcal{P} . Therefore, these frameworks and solvers are not really comparable with different purposes and application capabilities. We mainly use them to verify the effectiveness and reliability of our proposed framework in terms of the solution time and the solution quality.

A. Effectiveness and Reliability of the Proposed Large-Scale Convex Optimization Framework

Consider a network with L 2-antenna RAUs and K single-antenna MUs uniformly and independently distributed⁴ in the square region $[-3000, 3000] \times [-3000, 3000]$ meters with $L = K$. We consider the total transmit power minimization problem $\mathcal{P}_1(\gamma)$ with the QoS requirements for each MU as $\gamma_k = 5$ dB, $\forall k$. Table I demonstrates the comparison of 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 scaling fading realization for each large-scale fading realization).

For the modeling time comparisons, this table shows that the value of the proposed matrix stuffing technique ranges between 0.01 and 30 seconds⁵ for different network sizes and can speedup about 15x to 60x compared to the parser/solver modeling framework CVX. In particular, for large-scale problems,

⁴Consider the CSI acquisition overhead, our proposed approach is mainly suitable in the low user mobility scenarios.

⁵This value can be significantly reduced in practical implementations, e.g., at the BBU pool in Cloud-RAN, which, however, requires substantial further investigation. Meanwhile, the results effectively confirm that the proposed matrix stuffing technique scales well to large-scale problems.

the transformation using CVX is time consuming and becomes the bottleneck, as the “**modeling time**” is comparable and even larger than the “**solving time**”. For example, when $L = 150$, the “**modeling time**” using CVX is about 3 minutes, while the matrix stuffing only requires about 10 seconds. Therefore, the 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 speedup by several orders of magnitude over the interior-point solvers. For example, for $L = 50$, it can speedup about 20x and 130x over MOSEK⁶ and SDPT3, respectively, while SeDuMi is inapplicable for this problem size as the running time exceeds the pre-defined maximum value, i.e., one hour. In particular, all the interior-point solvers fail to solve large-scale problems (i.e., $L = 100, 150, 200$), denoted as “N/A”, while the operator splitting solver SCS can scale well to large problem sizes. For the largest problems with $L = 200$, the operator splitting solver can solve them in about 5 minutes.

For the quality of the solutions, this table shows that the propose 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 method based 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 based large-scale convex optimization framework scales well to large-scale problem modeling and solving simultaneously. Therefore, it provides 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 need further investigation. In particular, this set of results indicate that the scale of cooperation in dense wireless networks may be fundamentally constrained by the computation complexity/time.

B. Infeasibility Detection Capability

A unique property of the proposed framework is its infeasibility detection capability, which will be verified in this part. Consider a network with $L = 50$ single-antenna RAUs and $K = 50$ single-antenna MUs uniformly and independently distributed in the square region $[-2000, 2000] \times [-2000, 2000]$ meters. The empirical probabilities of feasibility in Fig. 3 show that the proposed framework can detect the infeasibility accurately compared with the second-order method framework “CVX+SDPT3” and the first-order method framework “CVX+SCS”. Each point of the simulation results is averaged over 200 randomly generated network realizations. The average (“**total time**”, “**solving time**”) for obtaining a single point with “CVX+SDPT3”, “CVX+SCS” and “Matrix

⁶Although SeDuMi, SDPT3 and MOSEK (commercial software) are all based on the interior-point method, the implementation efficiency of the corresponding software packages varies substantially. In the following simulations, we mainly compare with the state-of-art public solver SDPT3.

TABLE I
TIME AND SOLUTION RESULTS FOR DIFFERENT CONVEX OPTIMIZATION FRAMEWORKS

Network Size ($L = K$)		20	50	100	150	200
CVX+SeDuMi	Total Time [sec]	8.1164	N/A	N/A	N/A	N/A
	Objective [W]	12.2488	N/A	N/A	N/A	N/A
CVX+SDPT3	Total Time [sec]	5.0398	330.6814	N/A	N/A	N/A
	Objective [W]	12.2488	6.5216	N/A	N/A	N/A
CVX+MOSEK	Total Time [sec]	1.2072	51.6351	N/A	N/A	N/A
	Objective [W]	12.2488	6.5216	N/A	N/A	N/A
CVX+SCS	Total Time [sec]	0.8501	5.6432	51.0472	227.9894	725.6173
	Modeling Time [sec]	0.7563	4.4301	38.6921	178.6794	534.7723
	Objective [W]	12.2505	6.5215	3.1303	2.0693	1.5404
Matrix Stuffing+SCS	Total Time [sec]	0.1137	2.7222	26.2242	90.4190	328.2037
	Modeling Time [sec]	0.0128	0.2401	2.4154	9.4167	29.5813
	Objective [W]	12.2523	6.5193	3.1296	2.0689	1.5403

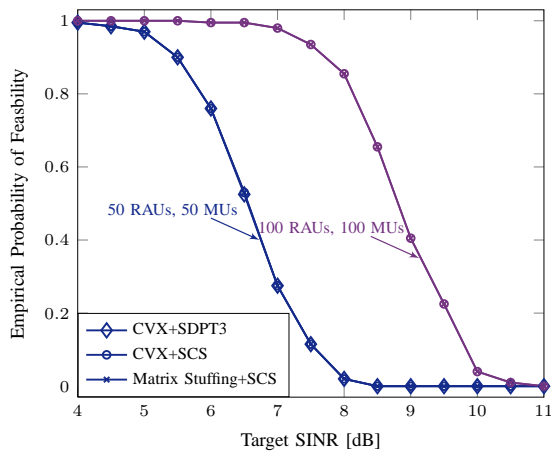


Fig. 3. The empirical probability of feasibility versus target SINR with different network sizes.

Stuffing+SCS” are (101.7635, 99.1140) seconds, (5.0754, 2.3617) seconds and (1.8549, 1.7959) seconds, respectively. This shows that the operator splitting solver can speedup about 50x over the interior-point solver.

We further consider a larger-sized network with $L = 100$ single-antenna RAUs and $K = 100$ single-antenna MUs uniformly and independently distributed in the square region $[-2000, 2000] \times [-2000, 2000]$ meters. As the second-order method framework fails to scale to this size, we only compare with the first-order method framework. Fig. 3 demonstrates that the proposed framework has the same infeasibility detection capability as the first-order method framework. This verifies the correctness and the reliability of the proposed fast transformation via matrix stuffing. Each point of the simulation results is averaged over 200 randomly generated network realizations. The average (“**solving time**”, “**modeling time**”) for obtaining a single point with “CVX+SCS” and “Matrix Stuffing+SCS” are (41.9273, 18.6079) seconds and (31.3660, 0.5028) seconds, respectively. This shows that the matrix stuffing technique can speedup about 40x over the numerical based parser/solver modeling framework CVX. We also note that the solving time of the proposed framework is smaller than the framework “CVX+SCS”, the speedup is due to the warm-starting [39, Section 4.2].

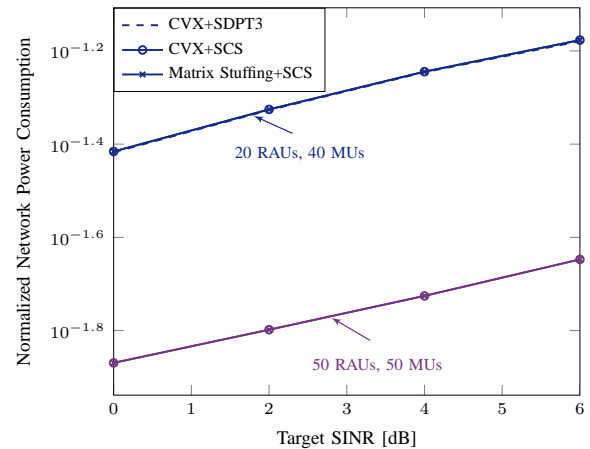


Fig. 4. Average normalized network power consumption (i.e., the obtained optimal total network power consumption over the maximum network power consumption with all the RAUs active and full power transmission) versus target SINR with different network sizes.

C. Group Sparse Beamforming for Network Power Minimization

In this part, we simulate the network power minimization problem using the group sparse beamforming algorithm [3, Algorithm 2]. We set each fronthaul link power consumption as $5.6W$ and set the power amplifier efficiency coefficient for each RAU as 25%. In this algorithm, a sequence of convex feasibility problems need to be solved to determine the active RAUs and one convex optimization problem needs to be solved to determine the transmit beamformers. This relies on the infeasibility detection capability of the proposed framework.

Consider a network with $L = 20$ 2-antenna RAUs and $K = 40$ single-antenna MUs uniformly and independently distributed in the square region $[-1000, 1000] \times [-1000, 1000]$ meters. Each point of the simulation results is averaged over 50 randomly generated network realizations. Fig. 4 demonstrates the accuracy of the solutions in the network power consumption obtained by the proposed framework compared with the second-order method framework “CVX+SDPT3” and the first-order method framework “CVX+SCS”. The average (“**total time**”, “**solving time**”) for obtaining a single point with

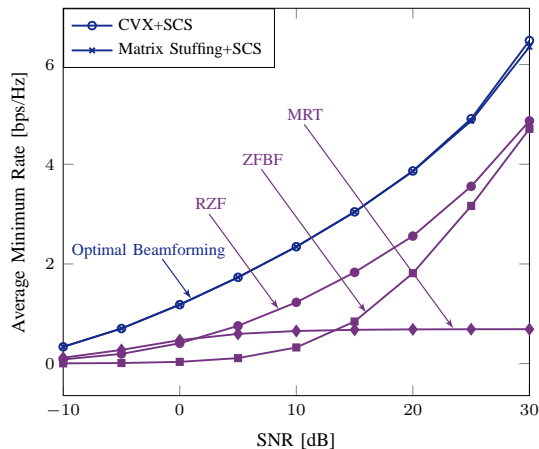


Fig. 5. The minimum network-wide achievable versus transmit SNR with 55 single-antenna RAUs and 50 single-antenna MUs.

“CVX+SDPT3”, “CVX+SCS” and “Matrix Stuffing+SCS” are (48.6916, 41.0316) seconds, (9.4619, 1.7433) seconds and (2.4673, 2.3061) seconds, respectively. This shows that the operator splitting solver can speedup about 20x over the interior-point solver.

We further consider a larger-sized network with $L = 50$ 2-antenna RAUs and $K = 50$ single-antenna MUs uniformly and independently distributed in the square region $[-3000, 3000] \times [-3000, 3000]$ meters. As the second-order method framework is not applicable to this problem size, we only compare with the first-order method framework. Each point of the simulation results is averaged over 50 randomly generated network realizations. Fig. 4 shows that the proposed framework can achieve the same solutions in network power consumption as the first-order method framework “CVX+SCS”. The average (“**solving time**”, “**modeling time**”) for obtaining a single point with “CVX+SCS” and “Matrix Stuffing+SCS” are (11.9643, 69.0520) seconds and (14.6559, 2.1567) seconds, respectively. This shows that the matrix stuffing technique can speedup about 30x over the numerical based parser/solver modeling framework CVX.

In summary, Fig. 4 demonstrates the capability of infeasibility detection (as a sequence of convex feasibility problems need to be solved in the RAU selection procedure), the accuracy of the solutions, and speedups provided by the proposed framework over the existing frameworks.

D. Max-min Rate Optimization

We will simulate the minimum network-wide achievable rate maximization problem using the max-min fairness optimization algorithm in [22, Algorithm 1] via the bi-section method, which requires to solve 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 the low-complexity and heuristic transmission strategies, i.e., zero-forcing beamforming (ZFBF) [48], [28], regularized zero-forcing beamforming (RZF) [49] and maximum ratio transmission (MRT) [50].

Consider a network with $L = 55$ single-antenna RAUs and $K = 50$ single-antenna MUs uniformly and independently

distributed in the square region $[-5000, 5000] \times [-5000, 5000]$ meters. Fig. 5 demonstrates the minimum network-wide achievable rate versus different SNRs (which is defined as the transmit power at all the RAUs over the receive noise power at all the MUs) using different algorithms. Each point of the simulation results is averaged over 50 randomly generated network realizations. For the 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**”, “**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 modelling time via warm-starting and matrix stuffing, respectively.

Furthermore, this figure also shows that the optimal beamforming can achieve quite an improvement for the per-user rate compared to suboptimal transmission strategies RZF, ZFBF and MRT, which clearly shows the importance of developing optimal beamforming algorithms for such networks. The average (“**solving time**”, “**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, which is 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 bi-section search procedure. The main time consuming part is from transformation using CVX.

VII. CONCLUSIONS AND FURTHER WORKS

In this paper, we proposed a unified two-stage framework for large-scale optimization in dense wireless cooperative networks. We showed that various performance optimization problems can be essentially solved by solving one or a sequence of convex optimization or feasibility problems. The proposed framework only requires the convexity of the underlying problems (or subproblems) without any other structural assumptions, e.g., smooth or separable functions. This is achieved by first transforming the original convex problem to the 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 demonstrated the infeasibility detection capability, the modeling flexibility and computing scalability, and the reliability of the proposed framework.

In principle, one may apply the proposed framework to any large-scale convex optimization problems and only needs to focus on the standard form reformulation as shown in Appendix, as well as to compute the proximal operators for different cone projections in (42). However, in practice, we need to address the following issues to provide a user-friendly framework and to assist practical implementation:

- Although the parse/solver frameworks like CVX can automatically transform an original convex problem into

the standard form *numerically* based on the graph implementation, extending such an idea to the *automatic and symbolic* transformation, thereby enabling matrix stuffing, is desirable but challenging in terms of reliability and correctness verification.

- Efficient projection algorithms are highly desirable. For the subspace projection, as discussed in Section V-B, parallel factorization and symbolic factorization are especially suitable for the cloud computing environments as in Cloud-RAN [2], [3]. For the cone projection, although the projection on the second-order cone is very efficient, as shown in (42), projecting on the semidefinite cone (which is required to solve the semidefinite programming problems) is computationally expensive, as it requires to perform eigenvalue decomposition [13]. The structure of the cone projection should be exploited to make speedups.
- It is interesting to apply the proposed framework to various non-convex optimization problems. For instance, the well-known majorization-minimization optimization provides a principled way to solve the general non-convex problems, whereas a sequence of convex subproblems need to be solved at each iteration. Enabling scalable computation at each iteration will hopefully lead to scalability of the overall algorithm.

APPENDIX

CONIC FORMULATION FOR CONVEX PROGRAMS

We shall present a systematic way to transform the original problem to the standard convex cone programming form. We first take the real-field problem \mathcal{P} with the objective function $f(\mathbf{x}) = \|\mathbf{v}\|_2$ as an example. At the end of this subsection, we will show how to extend it to the complex-field.

According to the principle of the disciplined convex programming [30], the original problem \mathcal{P} can be rewritten as the following disciplined convex programming form [30]

$$\begin{aligned} \mathcal{P}_{\text{cvx}} : \text{minimize } & \|\mathbf{v}\|_2 \\ \text{subject to } & \|\mathbf{D}_l \mathbf{v}\|_2 \leq \sqrt{P_l}, l = 1, \dots, L \quad (44) \\ & \|\mathbf{C}_k \mathbf{v} + \mathbf{g}_k\|_2 \leq \beta_k \mathbf{r}_k^T \mathbf{v}, k = 1, \dots, K, (45) \end{aligned}$$

where $\mathbf{D}_l = \text{blkdiag}\{\mathbf{D}_l^1, \dots, \mathbf{D}_l^K\} \in \mathbb{R}^{N_l K \times N K}$ with $\mathbf{D}_l^k = \begin{bmatrix} \mathbf{0}_{N_l \times \sum_{i=1}^{l-1} N_i} & \mathbf{I}_{N_l \times N_l} & \mathbf{0}_{N_l \times \sum_{i=l+1}^L N_i} \end{bmatrix} \in \mathbb{R}^{N_l \times N}$, $\beta_k = \sqrt{1 + 1/\gamma_k}$, $\mathbf{r}_k = \begin{bmatrix} \mathbf{0}_{(k-1)N}^T & \mathbf{h}_k^T & \mathbf{0}_{(K-k)N}^T \end{bmatrix}^T \in \mathbb{R}^{N K}$, $\mathbf{g}_k = [\mathbf{0}_{K}^T, \sigma_k]^T \in \mathbb{R}^{K+1}$, and $\mathbf{C}_k = [\tilde{\mathbf{C}}_k, \mathbf{0}_{N K}]^T \in \mathbb{R}^{(K+1) \times N K}$ with $\tilde{\mathbf{C}}_k = \text{blkdiag}\{\mathbf{h}_k, \dots, \mathbf{h}_k\} \in \mathbb{R}^{N K \times K}$. It is thus easy to check the convexity of problem \mathcal{P}_{cvx} , following the disciplined convex programming ruleset [30].

A. Smith Form Reformulation

To arrive at the standard convex cone program $\mathcal{P}_{\text{cone}}$, we rewrite problem \mathcal{P}_{cvx} as the following Smith form [29] by introducing a new variable for each subexpression in \mathcal{P}_{cvx} ,

$$\begin{aligned} \text{minimize } & x_0 \\ \text{subject to } & \|\mathbf{x}_1\| = x_0, \mathbf{x}_1 = \mathbf{v} \\ & \mathcal{G}_1(l), \mathcal{G}_2(k), \forall k, l, \quad (46) \end{aligned}$$

where $\mathcal{G}_1(l)$ is the Smith form reformulation for the transmit power constraint for RAU l (44) as follows

$$\mathcal{G}_1(l) : \begin{cases} (y_0^l, \mathbf{y}_1^l) \in \mathcal{Q}^{K N_l + 1} \\ y_0^l = \sqrt{P_l} \in \mathbb{R} \\ \mathbf{y}_1^l = \mathbf{D}_l \mathbf{v} \in \mathbb{R}^{K N_l}, \end{cases} \quad (47)$$

and $\mathcal{G}_2(k)$ is the Smith form reformulation for the QoS constraint for MU k (45) as follows

$$\mathcal{G}_2(k) : \begin{cases} (t_0^k, \mathbf{t}_1^k) \in \mathcal{Q}^{K+1} \\ t_0^k = \beta_k \mathbf{r}_k^T \mathbf{v} \in \mathbb{R} \\ \mathbf{t}_1^k = \mathbf{t}_2^k + \mathbf{t}_3^k \in \mathbb{R}^{K+1} \\ \mathbf{t}_2^k = \mathbf{C}_k \mathbf{v} \in \mathbb{R}^{K+1} \\ \mathbf{t}_3^k = \mathbf{g}_k \in \mathbb{R}^{K+1}. \end{cases} \quad (48)$$

Nevertheless, the Smith form reformulation (46) is not convex due to the non-convex constraint $\|\mathbf{x}_1\| = x_0$. We thus relax the non-convex constraint as $\|\mathbf{x}_1\| \leq x_0$, yielding the following relaxed Smith form

$$\begin{aligned} \text{minimize } & x_0 \\ \text{subject to } & \mathcal{G}_0, \mathcal{G}_1(l), \mathcal{G}_2(k), \forall k, l, \quad (49) \end{aligned}$$

where

$$\mathcal{G}_0 : \begin{cases} (x_0, \mathbf{x}_1) \in \mathcal{Q}^{N K + 1} \\ \mathbf{x}_1 = \mathbf{v} \in \mathbb{R}^{N K}. \end{cases} \quad (50)$$

It can be easily proved that the constraint $\|\mathbf{x}_1\| \leq x_0$ has to be active at the optimal solution; otherwise, we can always scale down x_0 such that the cost function can be further minimized while still satisfying the constraints. Therefore, we conclude that the relaxed Smith form (49) is equivalent to the original problem \mathcal{P}_{cvx} .

B. Conic Reformulation

Now, the relaxed Smith form reformulation (49) is readily to be reformulated as the standard cone programming form $\mathcal{P}_{\text{cone}}$. Specifically, define the optimization variables $[x_0; \mathbf{v}]$ with the same order of equations as in \mathcal{G}_0 , then \mathcal{G}_0 can be rewritten as

$$\mathbf{M}[x_0; \mathbf{v}] + \boldsymbol{\mu}_0 = \mathbf{m}, \quad (51)$$

where the slack variables belong to the following convex set

$$\boldsymbol{\mu}_0 \in \mathcal{Q}^{N K + 1}, \quad (52)$$

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

$$\mathbf{M} = \begin{bmatrix} -1 & \\ & -\mathbf{I}_{N K} \end{bmatrix}, \mathbf{m} = \begin{bmatrix} 0 \\ \mathbf{0}_{N K} \end{bmatrix}, \quad (53)$$

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

$$\mathbf{P}^l [y_0^l; \mathbf{v}] + \boldsymbol{\mu}_1^l = \mathbf{p}^l, \quad (54)$$

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

$$\boldsymbol{\mu}_1^l \in \mathcal{Q}^1 \times \mathcal{Q}^{K N_l + 1}, \quad (55)$$

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

$$\mathbf{P}^l = \left[\begin{array}{c|c} 1 & \\ \hline -1 & \\ \hline & -\mathbf{D}_l \end{array} \right], \mathbf{p}^l = \begin{bmatrix} \sqrt{P_l} \\ 0 \\ \mathbf{0}_{KN_l} \end{bmatrix}, \quad (56)$$

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

$$\mathbf{Q}^k [t_0^k; \mathbf{v}] + \boldsymbol{\mu}_2^k = \mathbf{q}^k, \quad (57)$$

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

$$\boldsymbol{\mu}_2^k \in \mathcal{Q}^1 \times \mathcal{Q}^{K+2}, \quad (58)$$

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

$$\mathbf{Q}^k = \left[\begin{array}{c|c} 1 & -\beta_k \mathbf{r}_k^T \\ \hline -1 & \\ \hline & -\mathbf{C}_k \end{array} \right], \mathbf{q}^k = \begin{bmatrix} 0 \\ 0 \\ \mathbf{g}_k \end{bmatrix}, \quad (59)$$

respectively.

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

$$\boldsymbol{\nu} = [x_0; y_0^1; \dots; y_0^L; t_0^1; \dots; t_0^K; \mathbf{v}], \quad (60)$$

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

$$n = 1 + L + K + NK, \quad (61)$$

$$m = (L + K) + (NK + 1) + \sum_{l=1}^L (KN_l + 1) + K(K + 2), \quad (62)$$

$$\mathcal{K} = \underbrace{\mathcal{Q}^1 \times \dots \times \mathcal{Q}^1}_{L+K} \times \mathcal{Q}^{NK+1} \times \underbrace{\mathcal{Q}^{KN_1+1} \times \dots \times \mathcal{Q}^{KN_L+1}}_L \times \underbrace{\mathcal{Q}^{K+2} \times \dots \times \mathcal{Q}^{K+2}}_K, \quad (63)$$

where \mathcal{K} is the Cartesian product of $2(L+K)+1$ second-order

ones, and \mathbf{A} and \mathbf{b} are given as follows:

$$\mathbf{A} = \begin{bmatrix} | & 1 & | & & | \\ & \ddots & & & \\ & & 1 & & \\ \hline & & & 1 & -\beta_1 \mathbf{r}_1^T \\ & & & \ddots & \vdots \\ & & & & 1 & -\beta_K \mathbf{r}_K^T \\ \hline -1 & & & & & -\mathbf{I}_{NK} \\ \hline & -1 & & & & -\mathbf{D}_1 \\ & & \vdots & & & \vdots \\ & & & -1 & & -\mathbf{D}_L \\ & & & & -1 & -\mathbf{C}_1 \\ & & & & \vdots & \vdots \\ \hline & & & & -1 & -\mathbf{C}_K \end{bmatrix}, \mathbf{b} = \begin{bmatrix} \sqrt{P_1} \\ \vdots \\ \sqrt{P_L} \\ 0 \\ \vdots \\ 0 \\ 0_{NK} \\ 0 \\ 0_{KN_1} \\ \vdots \\ 0 \\ 0_{KN_L} \\ 0 \\ \mathbf{g}_1 \\ \vdots \\ 0 \\ \mathbf{g}_K \end{bmatrix}, \quad (64)$$

respectively.

C. Matrix Stuffing

Given a problem instance \mathcal{P} , to arrive at the standard cone program form, we only need to copy the parameters of the maximum transmit power P_l 's to the data of the standard form, i.e., $\sqrt{P_l}$'s in \mathbf{b} , copy the parameters of the SINR thresholds γ to the data of the standard form, i.e., β_k 's in \mathbf{A} , and copy the parameters of the channel realizations \mathbf{h}_k 's to the data of the standard form, i.e., \mathbf{r}_k 's and \mathbf{C}_k 's in \mathbf{A} . As we only need to perform copying the memory for the transformation, this procedure can be very efficient compared to the state-of-the-art numerical based modeling frameworks like CVX.

D. Extension to the Complex Case

For $\mathbf{h}_k \in \mathbb{C}^N$, $\mathbf{v}_i \in \mathbb{C}^N$, we have

$$\mathbf{h}_k^H \mathbf{v}_i \Rightarrow \underbrace{\begin{bmatrix} \Re(\mathbf{h}_k) & -\Im(\mathbf{h}_k) \\ \Im(\mathbf{h}_k) & \Re(\mathbf{h}_k) \end{bmatrix}^T}_{\tilde{\mathbf{h}}_k} \underbrace{\begin{bmatrix} \Re(\mathbf{v}_i) \\ \Im(\mathbf{v}_i) \end{bmatrix}}_{\tilde{\mathbf{v}}_i}, \quad (65)$$

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 the real-field problem by the transformations: $\mathbf{h}_k \Rightarrow \tilde{\mathbf{h}}_k$ and $\mathbf{v}_i \Rightarrow \tilde{\mathbf{v}}_i$.

ACKNOWLEDGMENT

The authors would like to thank Dr. Eric Chu for his insightful discussions and constructive comments related to this work.

REFERENCES

- [1] Y. Shi, J. Zhang, K. B. Letaief, B. Bai, and W. Chen, "Large-scale convex optimization for ultra-dense Cloud-RAN," *IEEE Wireless Commun. Mag.*, to appear, 2015.
- [2] China Mobile, "C-RAN: the road towards green RAN," *White Paper, ver. 3.0*, Dec. 2013.
- [3] Y. Shi, J. Zhang, and K. B. Letaief, "Group sparse beamforming for green Cloud-RAN," *IEEE Trans. Wireless Commun.*, vol. 13, pp. 2809–2823, May 2014.
- [4] S. Zhou, M. Zhao, X. Xu, J. Wang, and Y. Yao, "Distributed wireless communication system: a new architecture for future public wireless access," *IEEE Commun. Mag.*, vol. 41, no. 3, pp. 108–113, 2003.
- [5] 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.
- [6] 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, pp. 20–34, May 2010.
- [7] E. Björnson and E. Jorswieck, "Optimal resource allocation in coordinated multi-cell systems," *Found. Trends Commun. Inf. Theory*, vol. 9, pp. 113–381, Jan. 2013.
- [8] H. Dahrouj and W. Yu, "Coordinated beamforming for the multicell multi-antenna wireless system," *IEEE Trans. Wireless Commun.*, vol. 9, pp. 1748–1759, Sep. 2010.
- [9] S. P. Boyd and L. Vandenberghe, *Convex optimization*. Cambridge university press, 2004.
- [10] R. Zakhour and S. V. Hanly, "Base station cooperation on the downlink: Large system analysis," *IEEE Trans. Inf. Theory*, vol. 58, pp. 2079–2106, Apr. 2012.
- [11] 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, pp. 2988–3003, Jun. 2012.
- [12] Y. Shi, J. Zhang, and K. Letaief, "Robust group sparse beamforming for multicast green Cloud-RAN with imperfect CSI," *IEEE Trans. Signal Process.*, to appear, 2015.
- [13] 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. on Commun. (ICC), London, UK*, 2015.
- [14] Y. Shi, J. Zhang, and K. Letaief, "Optimal stochastic coordinated beamforming for wireless cooperative networks with CSI uncertainty," *IEEE Trans. Signal Process.*, vol. 63, pp. 960–973, Feb. 2015.
- [15] 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.
- [16] 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, no. 1-4, pp. 545–581, 1999.
- [17] 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*, pp. 197–232, Springer, 2000.
- [18] Y. Nesterov, A. Nemirovskii, and Y. Ye, *Interior-point polynomial algorithms in convex programming*, vol. 13. SIAM, 1994.
- [19] CVX Research, Inc., "CVX: Matlab software for disciplined convex programming, version 2.0 (beta)," 2013.
- [20] J. Lofberg, "YALMIP: A toolbox for modeling and optimization in MATLAB," in *IEEE Int. Symp. Computer-Aided Control Systems Design*, (Taipei, Taiwan, R.O.C.), pp. 284–289, Sep. 2004.
- [21] E. Chu, N. Parikh, A. Domahidi, and S. Boyd, "Code generation for embedded second-order cone programming," in *Control Conference (ECC), 2013 European*, pp. 1547–1552, Jul. 2013.
- [22] Y. Shi, J. Zhang, and K. B. Letaief, "Scalable coordinated beamforming for dense wireless cooperative networks," in *Proc. IEEE Global Communications Conf. (GLOBECOM)*, Austin, TX, 2014.
- [23] 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, pp. 142–148, Jul. 2014.
- [24] 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, pp. 3939–3952, Aug. 2014.
- [25] 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 in Mach. Learn.*, vol. 3, pp. 1–122, Jul. 2011.
- [26] S. K. Joshi, M. Codreanu, and M. Latva-aho, "Distributed resource allocation for MISO downlink systems via the alternating direction method of multipliers," *EURASIP Journal on Wireless Communications and Networking*, vol. 2014, pp. 1–19, Jan. 2014.
- [27] 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, pp. 1910–1921, Apr. 2009.
- [28] T. Yoo and A. Goldsmith, "On the optimality of multi-antenna broadcast scheduling using zero-forcing beamforming," *IEEE J. Sel. Areas Commun.*, vol. 24, pp. 528–541, Mar. 2006.
- [29] E. Smith, *On the optimal design of continuous processes*. PhD thesis, Imperial College London (University of London), 1996.
- [30] M. C. Grant and S. P. Boyd, "Graph implementations for nonsmooth convex programs," in *Recent advances in learning and control*, pp. 95–110, Springer, 2008.
- [31] Y. Ye, M. J. Todd, and S. Mizuno, "An $O(\sqrt{nL})$ -iteration homogeneous and self-dual linear programming algorithm," *Math. Oper. Res.*, vol. 19, no. 1, pp. 53–67, 1994.
- [32] Y.-F. Liu, Y.-H. Dai, and Z.-Q. Luo, "Coordinated beamforming for MISO interference channel: Complexity analysis and efficient algorithms," *IEEE Trans. Signal Process.*, vol. 59, pp. 1142–1157, Mar. 2011.
- [33] H. Tuy, "Monotonic optimization: Problems and solution approaches," *SIAM Journal on Optimization*, vol. 11, no. 2, pp. 464–494, 2000.
- [34] E. A. Jorswieck, E. G. Larsson, and D. Danev, "Complete characterization of the pareto boundary for the MISO interference channel," *IEEE Trans. Signal Process.*, vol. 56, pp. 5292–5296, Oct. 2008.
- [35] R. Zhang and S. Cui, "Cooperative interference management with MISO beamforming," *IEEE Trans. Signal Process.*, vol. 58, pp. 5450–5458, Oct. 2010.
- [36] J. Mattingley and S. Boyd, "CVXGEN: a code generator for embedded convex optimization," *Optimization and Engineering*, vol. 13, no. 1, pp. 1–27, 2012.
- [37] T. A. Davis, *Direct Methods for Sparse Linear Systems*. Society for Industrial and Applied Mathematics, 2006.
- [38] E. Chu, B. O'Donoghue, N. Parikh, and S. Boyd, "A primal-dual operator splitting method for conic optimization," 2013, [Online]. Available: <http://web.stanford.edu/~boyd/papers/pdos.html>
- [39] B. O'Donoghue, E. Chu, N. Parikh, and S. Boyd, "Conic optimization via operator splitting and homogeneous self-dual embedding," *arXiv preprint arXiv:1312.3039*, 2013, [Online]. Available: <http://arxiv.org/abs/1312.3039>
- [40] A. Wiesel, Y. Eldar, and S. Shamai, "Linear precoding via conic optimization for fixed MIMO receivers," *IEEE Trans. Signal Process.*, vol. 54, pp. 161–176, Jan. 2006.
- [41] T. Goldstein, B. O'Donoghue, S. Setzer, and R. Baraniuk, "Fast alternating direction optimization methods," *SIAM Journal on Imaging Sciences*, vol. 7, no. 3, pp. 1588–1623, 2014.
- [42] R. Vanderbei, "Symmetric quasidfinite matrices," *SIAM J. Optim.*, vol. 5, no. 1, pp. 100–113, 1995.
- [43] N. Parikh and S. Boyd, "Proximal algorithms," *Foundations and trends in optimization*, vol. 1, Jan. 2014.
- [44] 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 Transactions on Mathematical Software (TOMS)*, vol. 39, p. 13, Feb. 2013.
- [45] J. Mattingley and S. Boyd, "Real-time convex optimization in signal processing," *IEEE Signal Process. Mag.*, vol. 27, no. 3, pp. 50–61, 2010.
- [46] A. Domahidi, E. Chu, and S. Boyd, "ECOS: An SOCP solver for embedded systems," in *Control Conference (ECC), 2013 European*, pp. 3071–3076, IEEE, 2013.
- [47] S. Diamond, E. Chu, and S. Boyd, "CVXPY: A Python-embedded modeling language for convex optimization, version 0.2," May 2014.
- [48] J. Zhang and J. G. Andrews, "Adaptive spatial intercell interference cancellation in multicell wireless networks," *IEEE J. Sel. Areas Commun.*, vol. 28, pp. 1455–1468, Dec. 2010.
- [49] C. B. Peel, B. M. Hochwald, and A. L. Swindlehurst, "A vector-perturbation technique for near-capacity multi-antenna multiuser communication-part I: channel inversion and regularization," *IEEE Trans. Commun.*, vol. 53, pp. 195–202, Jan. 2005.
- [50] F. Rusek, D. Persson, B. K. Lau, E. Larsson, T. Marzetta, O. Edfors, and F. Tufvesson, "Scaling up MIMO: Opportunities and challenges with

very large arrays," *IEEE Signal Process. Mag.*, vol. 30, no. 1, pp. 40–60, 2013.