Distributed Computation for Sparse Semidefinite Programming with Applications to Power Optimization Problems

Ramtin Madani, Abdulrahman Kalbat and Javad Lavaei

Abstract—Major power optimization problems, such as optimal power flow, state estimation and unit commitment, can be formulated or well approximated as semidefinite programs (SDPs). However, the inability to efficiently solve large-scale SDPs is an impediment to the deployment of such formulations in practice, which has made the industry to resort to simpler but less accurate formulations such as linear and quadratic programs. Motivated by the significant role of SDPs in revolutionizing the decision-making process for power systems, this paper designs a low-complexity algorithm for solving sparse SDPs, based on the alternating direction method of multipliers (ADMM). It is known that exploiting the sparsity of a large-scale SDP problem leads to a decomposed formulation with a lower computational cost. The algorithm proposed in this work solves the decomposed formulation of the SDP problem using an ADMM scheme, whose iterations consist of two subproblems. Both subproblems are highly parallelizable and enjoy closed-form solutions, which make the iterations computationally very cheap. The developed numerical method can be used to solve SDP relaxations or formulations of several power optimization problems. The proposed algorithm is tested on the SDP relaxation of the optimal power flow problem for real-world benchmark systems with more than 13000 nodes.

I. INTRODUCTION

Real-world power optimization problems are concerned with the efficiency, robustness, reliability, security and resiliency of power systems, whose decision variables consist of various real-time parameters across a time horizon and under several failure scenarios. These parameters include voltages, currents, phase angles, power productions, line flows, transformer settings, and the on/off statuses of generators and lines. Several factors contribute to a high computational complexity of power optimization problems, such as the nonlinearities induced by laws of physics and discrete variables, the scale of modern grids, the wide range of failure scenarios, and the level of uncertainty for demand and renewable energy sources. The above-mentioned factors give rise to non-convex mixed-integer optimization problems with tens of thousands of decision parameters. While the expected level of efficiency and reliability in recent years necessitates the use of accurate models of power systems that are inevitably highly nonconvex, current state-of-the-art solvers such as CPLEX, Gurobi and MOSEK are incapable of handling continuous non-convexity and a large number of discrete parameters arising in real-world power system optimization problems.

Recent approaches to tackle computationally-hard power optimization problems rely on convex algebraic and/or geometry methods, such as conic relaxation, sum of squares, and Lasserre and Sherali-Adams hierarchies [1]–[4]. These advanced techniques are based on solving semidefinite programs (SDPs) with a considerably large number of variables and constraints. Hence, it is imperative to solve semidefinite programs (SDPs) with a considerably large number of variables and constraints. Hence, it is imperative to design efficient and fast algorithms for large-scale SDPs, which are applicable to fundamental power optimization problems such as optimal power flow (OPF). The OPF problem is at the heart of the operation of power systems, which finds an optimal operating point of a power system by minimizing a certain objective function (e.g., transmission loss or generation cost) subject to power flow equations and operational constraints [5].

Several optimization techniques have been studied for the OPF problem in recent years [7]. Due to the non-convexity and NP-hardness of OPF, these algorithms are not robust, lack performance guarantees and may not find a global optimum. The paper [8] evaluates the potentials of SDP relaxations for OPF and shows that a global minimum of the problem can be found using an appropriate SDP formulation if the duality gap is zero. The proposed relaxation is shown to find global or near globally optimal solutions (with global optimality guarantees of at least 99%) for IEEE and Polish systems, and theoretically proven to work under different conditions [2], [9]–[14]. Due to the great success of SDP for the OPF problem, conic relaxations have been designed for other power optimization problems, including state estimation, unit commitment and charging of electric vehicles [15]–[18]. However, the high dimension of these conic formulations for real-world systems is an impediment to their implementation. The main objective of this work is to design a low-complexity algorithms for large-scale conic problems that can be used for a variety of power optimization problems.

A. Numerical Algorithms for Semidefinite Programs

Inspired by the seminal papers [19]–[21], there has been a growing interest in semidefinite programming, due in part to its applications in combinatorial optimization and a large set of real-world problems across engineering [8], [22]–[24]. Semidefinite programming offers a convex formulation or relaxation framework that is applicable to a wide range of non-convex optimization problems, and has been proven to achieve nontrivial bounds and approximation ratios that are beyond the reach of conventional methods [21], [25]–[27]. While small- to medium-sized SDPs are efficiently solvable by second-order-based interior point methods in polynomial time up to any arbitrary precision [28], these methods are mostly impractical for large-scale SDPs due to computation time and
memory issues. The primary obstacle is the requirement of calculating Schur complement matrices and their Cholesky factorizations. Several attempts have been made in order to parallelize this procedure, which have led to software packages such as SDPA and SMCP [29]–[31]. In presence of sparsity, a graph-theoretic analysis of SDP problems is proven to be effective in reducing the number variables and the order of conic constraints [32], [33]. This approach is remarkably helpful for structured SDP problems, particularly those arising in power system optimization [1].

A promising numerical technique for solving large-scale SDP problems is the alternating direction method of multipliers (ADMM), which is a first-order optimization algorithm proposed in the mid-1970s [34] and [35]. While second-order methods are capable of achieving a high accuracy via expensive iterations, a modest accuracy can be attained through tens of ADMM’s low-complexity iterations. In order to obtain a highly accurate solution in a reasonable number of iterations, great effort has been devoted to accelerating ADMM [36], [37]. Because of the sensitivity of the gradient methods to the condition number of the problem’s data, diagonal rescaling is proposed in [38] for a class of problems to improve the performance of ADMM. Moreover, several accelerated variants of ADMM as well as parameter tuning methods have been proposed in the literature to significantly improve the speed of convergence for specific application domains [39]. The \( O(\frac{1}{n}) \) worst-case convergence rate of ADMM is proven in [39] and [40] under certain assumptions, and a systematic framework is introduced in [41] for the convergence analysis of ADMM by means of control-theoretic methods.

In light of the scalability of ADMM, the main objective of this work is to design an ADMM-based parallel algorithm for solving sparse large-scale SDPs, with a guaranteed convergence under very mild assumptions. We start by defining a representative graph for the large-scale SDP problem, from which a decomposed SDP formulation is obtained using a tree/chordal/clique decomposition technique. This decomposition replaces the large-scale SDP matrix variable with certain submatrices of this matrix. In order to solve the decomposed SDP problem iteratively, a distributed ADMM-based algorithm is derived, whose iterations comprise entrywise matrix multiplication/division and eigendecomposition on certain submatrices of the SDP matrix. By finding the optimal solution for the distributed SDP, one could recover the solution to the original large-scale SDP formulation using an explicit formula.

This work is related to and improves upon some recent papers in this area. The paper [42] applies ADMM to the dual SDP formulation, leading to a centralized algorithm that is not parallelizable and is computationally expensive for large-scale SDPs. The work [33] decomposes a sparse SDP into smaller-sized SDPs through a tree decomposition, which are then solved by interior point methods. However, this approach is limited by the large number of consistency constraints. Using a first-order splitting method, [43] solves the decomposed SDP problem created by [33], but the algorithm needs to solve an optimization subproblem at every iteration. In contrast with the above papers, the algorithm proposed in this work is composed of low-complexity and parallelizable iterations, which run fast if the treewidth of the representative graph of the SDP problem is small. Since this treewidth is low for real-world power networks, our algorithm is well suited for power optimization problems, and indeed this is the main motivation behind this work. Our paper is also related to [44], which designs a distributed algorithm for second-order conic programs. In contrast to the existing methods, the algorithm to be proposed in this paper applies to higher-order conic problems, and does not require solving any optimization subproblem at any iteration. Since power optimization problems are mainly built upon the power flow equations, for the sake of space limitations, we will only focus on optimizing power flows to illustrate our results.

This paper is organized as follows. Some preliminaries and definitions are provided in Section II. An arbitrary sparse SDP is converted into a decomposed SDP in Section III for which a numerical algorithm is developed in Section IV. The application of this algorithm for OPF is investigated in Section V. Numerical examples are given in Section VI followed by concluding remarks in Section VII.

Notations: \( \mathbb{R}, \mathbb{C}, \text{ and } \mathbb{H}^n \) denote the sets of real numbers, complex numbers, and \( n \times n \) Hermitian matrices, respectively. The notation \( X_1 \odot X_2 \) refers to the Hadamard (entrywise) multiplication of matrices \( X_1 \) and \( X_2 \). The symbols \( \langle \cdot, \cdot \rangle \) and \( \| \cdot \|_F \) denote the Frobenius inner product and norm of matrices, respectively. The notation \( \| v \|_2 \) denotes the \( \ell_2 \)-norm of a vector \( v \). The \( m \times n \) rectangular identity matrix, whose \( (i, j) \) entry is equal to the Kronecker delta \( \delta_{ij} \), is denoted by \( I_{m \times n} \). The notations \( \text{Re}[W] \), \( \text{Im}[W] \), \( \text{rank}[W] \), and \( \text{diag}[W] \) denote the real part, imaginary part, rank, and diagonal of a Hermitian matrix \( W \), respectively. Given a vector \( v \), the notation \( \text{diag}(v) \) denotes a diagonal square matrix whose entries are given by \( v \). The notation \( W \succeq 0 \) means that \( W \) is Hermitian and positive semidefinite. The notation “i” is reserved for the imaginary unit. The superscripts \( (\cdot)^* \) and \( (\cdot)^T \) represent the conjugate transpose and transpose operators, respectively. Given a matrix \( W \), its \( (l, m) \) entry is denoted as \( W_{lm} \). The subscript \( (\cdot)_{opt} \) is used to show the optimal value of an optimization variable. Given a matrix \( W \), its Moore-Penrose pseudoinverse is denoted as \( \text{pinv}(W) \). Given a simple graph \( \mathcal{H} \), its vertex and edge sets are denoted by \( V_{\mathcal{H}} \) and \( E_{\mathcal{H}} \), respectively, and the graph \( \mathcal{H} \) is shown as \( \mathcal{H} = (V_{\mathcal{H}}, E_{\mathcal{H}}) \). Given two sets \( S_1 \) and \( S_2 \), the notation \( S_1 \lessdot S_2 \) denotes the set of all elements of \( S_1 \) that do not exist in \( S_2 \). Given a Hermitian matrix \( W \) and two sets of positive integer numbers \( S_1 \) and \( S_2 \), define \( W(S_1, S_2) \) as a submatrix of \( W \) obtained through two operations: (i) removing all rows of \( W \) whose indices do not belong to \( S_1 \), and (ii) removing all columns of \( W \) whose indices do not belong to \( S_2 \). For instance, \( W \{\{1, 2\}, \{2, 3\}\} \) is a \( 2 \times 2 \) matrix with the entries \( W_{12}, W_{13}, W_{22}, W_{23} \).
Consider the semidefinite program

$$\begin{align*}
\text{minimize} & \quad \langle X, M_0 \rangle \\
\text{subject to} & \quad l_s \leq \langle X, M_s \rangle \leq u_s, \quad s = 1, \ldots, p, \\
& \quad X \succeq 0.
\end{align*}$$

(1a) (1b) (1c)

where $M_0, M_1, \ldots, M_p \in \mathbb{H}^n$, and

$$\langle l_s, u_s \rangle \in \{(-\infty) \cup \mathbb{R} \} \times (\mathbb{R} \cup \{+\infty\})$$

for every $s = 1, \ldots, p$. Notice that the constraint (1b) reduces to an equality constraint if $l_s = u_s$.

Problem (1) is computationally expensive for a large number $n$ due to the presence of the positive semidefinite constraint (1c). However, if $M_0, M_1, \ldots, M_p$ are sparse, this expensive constraint can be decomposed and expressed in terms of some principal submatrices of $X$ with smaller dimensions. This will be explained next.

A. Representative Graph and Tree Decomposition

In order to leverage any possible sparsity of problem (1), a simple graph shall be defined to capture the zero-nonzero patterns of $M_0, M_1, \ldots, M_p$.

Definition 1. Define $G = (V_G, E_G)$ as the representative graph of the SDP problem (1), which is a simple graph with $n$ vertices whose edges are specified by the nonzero off-diagonal entries of $M_0, M_1, \ldots, M_p$. In other words, two arbitrary vertices $i$ and $j$ are connected if the $(i, j)$ entry of at least one of the matrices $M_0, M_1, \ldots, M_p$ is nonzero.

Using a tree decomposition algorithm (also known as chordal or clique decomposition), we can obtain a decomposed formulation for problem (1), in which the positive semidefinite requirement is imposed on certain principal submatrices of $X$ as opposed to $X$ itself.

Definition 2 (Tree decomposition). A tree graph $T$ is called a tree decomposition of $G$ if it satisfies the following properties:

1) Every node of $T$ corresponds to and is identified by a subset of $V_G$.
2) Every vertex of $G$ is a member of at least one node of $T$.
3) $T_k$ is a connected graph for every $k \in V_G$, where $T_k$ denotes the subgraph of $T$ induced by all nodes of $T$ containing the vertex $k$ of $G$.
4) The subgraphs $T_i$ and $T_j$ have at least one node in common, for every $(i, j) \in E_G$.

Each node of $T$ is a bag (collection) of vertices of $G$ and hence it is referred to as a bag.

Let $T = (V_T, E_T)$ be an arbitrary tree decomposition of $G$, with the set of bags $V_T = \{C_1, C_2, \ldots, C_q\}$. As will be discussed in the next section, it is possible to cast problem (1) in terms of those entries of $X$ that appear in at least one of the submatrices $X[C_1, C_1], X[C_2, C_2], \ldots, X[C_q, C_q]$. These entries of $X$ are referred to as important entries. Once the optimal values of the important entries of $X$ are found via an iterative algorithm, the remaining entries of $X$ can be obtained through an explicit formula to be stated later.

Among the factors that may contribute to the computational complexity of the decomposed problem are: the size of the largest bag, the number of bags, and the total number of important entries. Finding a tree decomposition that leads to the minimum number of important entries (minimum fill-in problem) or possesses the minimum size for its largest bag (treewidth problem) is known to be NP-hard. Nevertheless, there are many efficient algorithms in the literature that find near-optimal tree decompositions (especially for power networks due to their near planarity) [45], [46].

B. Sparsity Pattern of Matrices

Let $F^n$ denote the set of symmetric $n \times n$ matrices with entries belonging to the set $\{0, 1\}$. The distributed optimization scheme to be proposed in this work uses a group of sparse slack matrices. We identify the locations of nonzero entries of such matrix variables using descriptive matrices in $F^n$.

Definition 3. Given an arbitrary matrix $X \in \mathbb{H}^n$, define its sparsity pattern as a matrix $N \in F^n$ such that $N_{ij} = 1$ if and only if $X_{ij} \neq 0$ for every $i, j \in \{1, \ldots, n\}$. Let $|N|$ denote the number of nonzero entries of $N$. Define the set $S(N) \triangleq \{X \in \mathbb{H}^n \mid X \circ N = X\}$.

Due to the Hermitian property of $X$, if $d$ denotes the number of nonzero diagonal entries of $N$, then every $X \in S(N)$ can be specified by $(|N| + d)/2$ real-valued scalars corresponding to $\text{Re}(X)$ and $(|N| - d)/2$ real scalars corresponding to $\text{Im}(X)$. Therefore, $S(N)$ is $|N|$-dimensional over $\mathbb{R}$.

Definition 4. Suppose that $T = (V_T, E_T)$ is a tree decomposition of the representative graph $G$ with the bags $C_1, C_2, \ldots, C_q$.

- For $r = 1, \ldots, q$, define $C_r \in F^n$ as a sparsity pattern whose $(i, j)$ entry is equal to 1 if $\{i, j\} \subseteq C_r$ and is 0 otherwise for every $i, j \in \{1, \ldots, n\}$.
- Define $C \in F^n$ as an aggregate sparsity pattern whose $(i, j)$ entry is equal to 1 if and only if $\{i, j\} \subseteq C_r$ for at least one index $r \in \{1, \ldots, p\}$.
- For $s = 0, 1, \ldots, p$, define $N_s \in F^n$ as the sparsity pattern of $M_s$.

The sparsity pattern $C$, which can also be interpreted as the adjacency matrix of a chordal extension of $G$ induced by $T$, captures the locations of the important entries of $X$. The matrix $C$ will later be used to describe the domain of definition for the variable of the decomposed SDP problem.

C. Indicator Functions

To streamline the formulation, we will replace any positivity or positive semidefiniteness constraints in the decomposed SDP problem by the indicator functions introduced below.

Definition 5. For every $l \in (-\infty) \cup \mathbb{R}$ and $u \in \mathbb{R} \cup \{+\infty\}$, define the convex indicator function $I_{l,u} : \mathbb{R} \rightarrow \{0, +\infty\}$ as

$$I_{l,u}(x) \triangleq \begin{cases} 0 & \text{if } l \leq x \leq u \\ +\infty & \text{otherwise} \end{cases}$$
Definition 6. For every $r \in \{1, 2, \ldots, q\}$, define the convex indicator function $\mathcal{J}_r : \mathbb{R}^n \rightarrow \{0, +\infty\}$ as
\[
\mathcal{J}_r(X) \triangleq \begin{cases} 
0 & \text{if } X(C_r, C_r) \succeq 0 \\
+\infty & \text{otherwise}
\end{cases}
\]

III. Decomposed SDP

Consider the problem
\[
\begin{align*}
\text{minimize} & \quad \langle X, M_0 \rangle \\
\text{subject to} & \quad l_s \leq \langle X, M_s \rangle \leq u_s, \quad s = 1, \ldots, p, \\
& \quad X(C_r, C_r) \succeq 0, \quad r = 1, \ldots, q
\end{align*}
\]
which is referred to as decomposed SDP throughout this paper. Due to the chordal theorem \([47]\), problems (1) and (2) lead to the same optimal objective value. Furthermore, if $X_{\text{ref}} \in \mathcal{S}(C)$ denotes an arbitrary solution of the decomposed SDP problem (2), then there exists a solution $X_{\text{opt}}$ to the SDP problem (1) such that $X_{\text{opt}} \circ C = X_{\text{ref}}$.

To understand how $X_{\text{opt}}$ can be constructed from $X_{\text{ref}}$, observe that those entries of $X$ corresponding to the zeros of $C$ are $0$ due to the relation $X_{\text{ref}} \in \mathcal{S}(C)$. These entries of the matrix variable $X$ that are needed for $X_{\text{opt}}$ but have not been found by decomposed SDP are referred to as missing entries. Several completion approaches can be adopted in order to recover these missing entries. An algorithm is proposed in \([33]\) and \([32]\) that obtains a completion for $X_{\text{ref}}$ within the set $\{X \in \mathbb{H}^n \mid X \circ C = X_{\text{ref}}, X \succeq 0\}$ whose determinant is maximum. However, such a solution may not be favorable for applications that require a low-rank solution such as an SDP relaxation. It is also known that there exists a polynomial-time algorithm to fill a partially-known real-valued matrix in such a way that the rank of the resulting matrix becomes equal to the highest rank among all bags \([48], [49]\). In \([50]\), we extended this result to the complex domain by proposing a recursive algorithm that transforms $X_{\text{ref}} \in \mathcal{S}(C)$ into a solution $X_{\text{opt}}$ for the original SDP problem (1) whose rank is upper bounded by the maximum rank among the matrices $X_{\text{ref}}\{C_1, C_1\}, X_{\text{ref}}\{C_2, C_2\}, \ldots, X_{\text{ref}}\{C_q, C_q\}$. This algorithm is stated below for completeness.

Matrix completion algorithm:

1) Set $T' := T$ and $X := X_{\text{ref}}$.
2) If $T'$ has a single node, then consider $X_{\text{opt}}$ as $X$ and terminate; otherwise continue to the next step.
3) Choose two bags $C_x$ and $C_y$ of $T'$ such that $C_x$ is a leaf of $T'$ and $C_y$ is its unique neighbor.
4) Define
\[
\begin{align*}
K & \triangleq \text{pinv}\{X\{C_x \cap C_y, C_x \cap C_y\}\} \\
G_x & \triangleq X\{C_x \cap C_y, C_x \cap C_y\} \in \mathbb{C}^{d_x \times d_x} \\
G_y & \triangleq X\{C_y \cap C_x, C_y \cap C_x\} \in \mathbb{C}^{d_y \times d_y} \\
E_x & \triangleq X\{C_x \cap C_y, C_x \cap C_y\} \in \mathbb{C}^{d_x \times d_y} \\
E_y & \triangleq X\{C_y \cap C_x, C_y \cap C_x\} \in \mathbb{C}^{d_y \times d_x} \\
S_x & \triangleq E_x - G_xK_g^* = Q_xD_yQ_g^* \\
S_y & \triangleq E_y - G_yK_g^* = Q_yD_yQ_g^* \\
Q_x & \text{denote the eigenvalue decompositions of } S_x \text{ and } S_y \text{ with the diagonals of } D_x \text{ and } D_y \text{ arranged in descending order. Then, update a part of } X \text{ as follows:}
\end{align*}
\]
\[
X\{C_y \setminus C_x, C_y \setminus C_y\} := G_yK_g^* + Q_y\sqrt{D_y}I_{d_y \times d_x}\sqrt{D_x}Q_x^*
\]
and update $X\{C_x \setminus C_y, C_y \setminus C_y\}$ accordingly to preserve the Hermitian property of $X$.
5) Update $T'$ by merging $C_x$ into $C_y$, i.e., replace $C_y$ with $C_x \cup C_y$ and then remove $C_x$ from $T'$.
6) Go back to step 2.

Theorem 1. Consider an arbitrary solution $X_{\text{ref}}$ of the decomposed SDP problem (2). The output of the matrix completion algorithm, denoted as $X_{\text{opt}}$, is a solution of the original SDP problem (1). Moreover, the rank of $X_{\text{opt}}$ is smaller than or equal to:
\[
\max \left\{ \text{rank} \{X_{\text{ref}}\{C_r, C_r\} \mid r = 1, \ldots, q \} \right\}
\]

Proof. Please refer to \([50]\) or \([11]\) for the proof.

IV. Alternating Direction Method of Multipliers

Consider the optimization problem
\[
\begin{align*}
\text{minimize} & \quad f(x) + g(y) \\
\text{subject to} & \quad Ax + By = c
\end{align*}
\]
where $c \in \mathbb{R}^n$, $A \in \mathbb{R}^{n_x \times n_x}$ and $B \in \mathbb{R}^{n_y \times n_y}$ are constant matrices, and $f : \mathbb{R}^{n_x} \rightarrow \mathbb{R} \cup \{+\infty\}$ and $g : \mathbb{R}^{n_y} \rightarrow \mathbb{R} \cup \{+\infty\}$ are convex functions. Notice that the variables $x$ and $y$ are coupled through the linear constraint (4b) while the objective function is separable. The augmented Lagrangian function for problem (4) is equal to
\[
\mathcal{L}_\mu(x, y, \lambda) = f(x) + g(y) + \lambda^T(Ax + By - c) + (\mu/2)\|Ax + By - c\|_2^2, \quad \lambda \in \mathbb{R}^{n_c},
\]
where $\lambda \in \mathbb{R}^{n_c}$ is the Lagrange multiplier associated with the constraint (4b), and $\mu \in \mathbb{R}$ is a fixed parameter. ADMM is one approach for solving problem (4), which performs the following procedure at each iteration \([51]\):
\[
\begin{align*}
x^{k+1} & = \arg \min_{x \in \mathbb{R}^{n_x}} \mathcal{L}_\mu(x, y^k, \lambda^k), \quad k = 0, 1, 2, \ldots \\
y^{k+1} & = \arg \min_{y \in \mathbb{R}^{n_y}} \mathcal{L}_\mu(x^{k+1}, y, \lambda^k), \\
\lambda^{k+1} & = \lambda^k + \mu(Ax^{k+1} + By^{k+1} - c).
\end{align*}
\]
where $k = 0, 1, 2, \ldots$, for an arbitrary initialization $(x^0, y^0, \lambda^0)$. In these equations, “argmin” means an arbitrary minimizer of a convex function and does not need any uniqueness assumption. Notice that each of the updates (6a) and (6b) is an optimization sub-problem with respect to either $x$ and $y$, by freezing the other variable at its latest value.

Several acceleration techniques have been proposed in the literature, aiming to improve the convergence behavior of ADMM \([36], [41]\). One such approach, regarded as over-
relaxed ADMM, involves adopting a sequence of intermediate Lagrange multipliers \( \{ \lambda^k \}_{k=1}^\infty \) as follows:

\[
\begin{align*}
\mathbf{x}^{k+1} &= \arg \min_{\mathbf{x} \in \mathbb{R}^{n_x}} \mathcal{L}_\mu(\mathbf{x}, \mathbf{y}^k, \lambda^k), \\
\hat{\lambda}^{k+1} &= \lambda^k + \alpha (1 - \lambda^k) (A \mathbf{x}^{k+1} + B \mathbf{y}^k - \mathbf{c}), \\
\mathbf{y}^{k+1} &= \arg \min_{\mathbf{y} \in \mathbb{R}^{n_y}} \mathcal{L}_\mu(\mathbf{x}^{k+1}, \mathbf{y}, \hat{\lambda}^{k+1}), \\
\lambda^{k+1} &= \lambda^k + \mu (A \mathbf{x}^{k+1} + B \mathbf{y}^{k+1} - \mathbf{c}),
\end{align*}
\]

where \( \alpha \in [1, 2] \) is a fixed parameter. We employ the residue sequence \( \{ \varepsilon^k \}_{k=1}^\infty \) proposed in [36] as measure for convergence:

\[
\varepsilon^{k+1} = (1/\mu) \| \lambda^{k+1} - \lambda^k \|_2^2 + \mu \| B(y^{k+1} - y^k) \|_2^2
\]

ADMM is particularly interesting for the cases where subproblems can be solved efficiently through an explicit formula. Under such circumstances, it would be possible to execute a large number of iterations in a short amount of time. In this section, we first cast the decomposed SDP problem (2) in the form (4), and then regroup the variables into two blocks \( \mathcal{P}_1 \) and \( \mathcal{P}_2 \) playing the roles of \( \mathbf{x} \) and \( \mathbf{y} \) in the ADMM algorithm.

### A. Projection Onto Positive Semidefinite Cone

The algorithm to be proposed in this work requires the projection of \( q \) matrices belonging to \( \mathbb{H}^{[C_1]}, \mathbb{H}^{[C_2]}, \ldots, \mathbb{H}^{[C_q]} \) onto the positive semidefinite cone. This is probably the most computationally expensive part of each iteration.

**Definition 7.** For a given Hermitian matrix \( \hat{\mathbf{Z}} \), define the unique solution to the optimization problem

\[
\begin{align*}
\min_{\mathbf{Z} \in \mathbb{H}^{[Q]}} & \quad \| \mathbf{Z} - \hat{\mathbf{Z}} \|_F^2 \\
\text{subject to} & \quad \mathbf{Z} \succeq 0
\end{align*}
\]

as the projection of \( \hat{\mathbf{Z}} \) onto the cone of positive semidefinite matrices, and denote it as \( \mathbf{Z}^+ \).

The next Lemma reveals the interesting fact that problem (9) can be solved through an eigenvalue decomposition of \( \mathbf{Z} \).

**Lemma 1.** Let \( \hat{\mathbf{Z}} = \mathbf{Q} \times \text{diag}(\{\nu_1, \ldots, \nu_n\}) \times \mathbf{Q}^* \) denote the eigenvalue decomposition of \( \hat{\mathbf{Z}} \). The solution of the projection problem (9) is given by

\[
\hat{\mathbf{Z}}^+ = \mathbf{Q} \times \text{diag}(\{\max\{\nu_1, 0\}, \ldots, \max\{\nu_m, 0\}\}) \times \mathbf{Q}^*
\]

**Proof.** Please refer to [52] for the proof.

### B. ADMM for Decomposed SDP

We apply ADMM to the following reformulation of the decomposed SDP problem (2):

\[
\begin{align*}
\minimize_{\mathbf{X} \in \mathcal{S}(\mathbb{C})} & \quad z_0 + \sum_{s=1}^p \mathcal{I}_{l_s,u_s}(z_s) + \sum_{r=1}^q \mathcal{J}_r(\mathbf{X}_{C;r}) \\
\text{subject to} & \quad \mathbf{X} \circ \mathbf{C}_r = \mathbf{X}_{C;r}, \quad r = 1, 2, \ldots, q, \\
& \quad \mathbf{X} \circ \mathbf{N}_s = \mathbf{X}_{N;s}, \quad s = 0, 1, \ldots, p, \\
& \quad z_s = \langle \mathbf{M}_s, \mathbf{X}_{N;s} \rangle, \quad s = 0, 1, \ldots, p.
\end{align*}
\]

If \( \mathbf{X} \) is a feasible solution of (10) with a finite objective value, then

\[
\mathcal{J}_r(\mathbf{X}) = \mathcal{J}_r(\mathbf{X} \circ \mathbf{C}_r) \quad \text{(10a)}
\]

which concludes that \( \mathbf{X} \{ \mathbf{C}_r, \mathbf{C}_r \} \succeq 0 \). Moreover,

\[
\begin{align*}
\mathcal{I}_{l_s,u_s}(\langle \mathbf{X}, \mathbf{M}_s \rangle) &= \mathcal{I}_{l_s,u_s}(\mathbf{X} \circ \mathbf{N}_s, \mathbf{M}_s) \\
\mathcal{I}_{l_s,u_s}(\langle \mathbf{X}, \mathbf{M}_s \rangle) &= \mathcal{I}_{l_s,u_s}(\langle \mathbf{X}, \mathbf{M}_s \rangle) \\
\mathcal{I}_{l_s,u_s}(z_s) &= 0
\end{align*}
\]

which yields that \( l_s \leq \langle \mathbf{X}, \mathbf{M}_s \rangle \leq u_s \). Therefore, \( \mathbf{X} \) is a feasible point for problem (2) as well, with the same objective value. Define

1. \( \mathbf{A}_{C;r} \in \mathcal{S}(\mathbb{C}_r) \) as the Lagrange multiplier associated with the constraint (10a) for \( r = 1, 2, \ldots, q \),
2. \( \mathbf{A}_{N;s} \in \mathcal{S}(\mathbb{N}_s) \) as the Lagrange multiplier associated with the constraint (10b) for \( s = 0, 1, \ldots, p \),
3. \( \lambda_{z;s} \in \mathbb{R} \) as the Lagrange multiplier associated with the constraint (10c) for \( s = 0, 1, \ldots, p \).

We regroup the primal and dual variables as

\[
\begin{align*}
(\text{Block 1}) \quad & \mathcal{P}_1 = \{ \mathbf{X}, \{ z_s \}_{s=0}^p \} \\
(\text{Block 2}) \quad & \mathcal{P}_2 = \{ \langle \mathbf{X}, \mathbf{C}_r \rangle_{r=1}^q, \langle \mathbf{X}, \mathbf{N}_s \rangle_{s=0}^p \} \\
(\text{Dual}) \quad & \mathcal{D} = \{ \{ \mathbf{A}_{C;r} \}_{r=1}^q, \{ \mathbf{A}_{N;s} \}_{s=0}^p, \{ \lambda_{z;s} \}_{s=0}^p \}.
\end{align*}
\]

Note that “block 1”, “block 2” and “\( \mathcal{D} \)” play the roles of \( \mathbf{x}, \mathbf{y} \) and \( \lambda \) in the standard formulation of ADMM, respectively.

The augmented Lagrangian can be calculated as

\[
\begin{align*}
& (2/\mu) \mathcal{L}_\mu(\mathcal{P}_1, \mathcal{P}_2, \mathcal{D}) = \mathcal{L}_D(\mathcal{D})/\mu^2 \\
& + \| z_0 - \langle \mathbf{M}_0, \mathbf{X}_{N;0} \rangle \|_F^2 + (1 + \lambda_{z;0}) \| \| \|_F^2 \\
& + \sum_{s=1}^q (\| z_s - \langle \mathbf{M}_s, \mathbf{X}_{N;s} \rangle \|_F^2 + \mathcal{I}_{l_s,u_s}(z_s)) \\
& + \sum_{r=1}^q (\| \mathbf{X} \circ \mathbf{C}_r - \mathbf{X}_{C;r} \|_F^2 + (1/\mu) \mathbf{A}_{C;r} \|_F^2 + \mathcal{J}_r(\mathbf{X}_{C;r})) \\
& + \sum_{s=1}^p \| \mathbf{X} \circ \mathbf{N}_s - \mathbf{X}_{N;s} \|_F^2 + (1/\mu) \mathbf{A}_{N;s} \|_F^2
\end{align*}
\]

where

\[
\begin{align*}
& \mathcal{L}_D(\mathcal{D}) = - (1 + \lambda_{z;0})^2 \\
& - \sum_{s=1}^p \lambda_{z;s}^2 - \sum_{r=1}^q \| \mathbf{A}_{C;r} \|_F^2 - \sum_{s=1}^p \| \mathbf{A}_{N;s} \|_F^2
\end{align*}
\]
Using the blocks $\mathcal{P}_1$ and $\mathcal{P}_2$, the ADMM iterations for problem \(^{(10)}\) can be expressed as follows:

1) The subproblem \(^{(6a)}\) in terms of $\mathcal{P}_1$ consists of two parallel steps:
   
   (a) **Minimization in terms of $X$:** This step consists of $|C|$ scalar quadratic and unconstrained programs. It possesses an explicit formula that involves $|C|$ parallel multiplication operations.
   
   (b) **Minimization in terms of $\{z_s\}_{s=0}^p$:** This step consists of $p+1$ scalar quadratic programs each with a box constraint. It possesses an explicit formula that involves $p+1$ parallel multiplication operations.

2) The subproblem \(^{(6b)}\) in terms of $\mathcal{P}_2$ also consists of two parallel steps:
   
   (a) **Minimization in terms of $\{X_{C;r}\}_{r=1}^q$:** This step consists of $q$ projection problems of the form \(^{(9)}\). According to Lemma \(^{[1]}\) this reduces to $q$ parallel eigenvalue decomposition operations on matrices of sizes $|C_r| \times |C_r|$ for $r = 1, \ldots, q$.
   
   (b) **Minimization in terms of $\{X_{N;s}\}_{s=0}^p$:** This step consists of $p$ unconstrained quadratic programs of sizes $|N_s|$ for $s = 0, 1, \ldots, p$. The quadratic programs are parallel and each of them possesses an explicit formula that involves $2|N_s|$ multiplications.

3) Computation of the dual variables at each iteration, in equation \(^{(6c)}\), consists of three parallel steps:
   
   (a) **Updating $\{A_{C;r}\}_{r=1}^q$:** Computational costs for this step involve no multiplications and are negligible.
   
   (b) **Updating $\{A_{N;s}\}_{s=0}^p$:** Computational costs for this step involve no multiplications and are negligible.
   
   (c) **Updating $\{\lambda_{z;s}\}_{s=0}^p$:** This step is composed of $p+1$ parallel inner product computations, each involving $|N_s|$ multiplications for $s = 0, 1, \ldots, p$.

The fact that every step of the above algorithm has an explicit easy-to-compute formula makes the algorithm very appealing for large-scale SDPs.

**Notation 1.** For every $D, E \in \mathbb{H}^n$, the notation $D \odot E$ refers to the entrywise division of those entries of $D$ and $E$ that correspond to the ones of $C$, i.e.,

\[
(D \odot E)_{ij} \triangleq \begin{cases} 
D_{ij}/E_{ij} & \text{if } C_{ij} = 1 \\
0 & \text{if } C_{ij} = 0.
\end{cases}
\]

In what follows, we will elaborate on every step of the ADMM iterations:

**Block 1:** The first step of the algorithm that corresponds to \(^{(6a)}\) consists of the operation

\[
\mathcal{P}_1^{k+1} := \arg \min_{\mathcal{P}_1} \mathcal{L}_\mu(\mathcal{P}_1, \mathcal{P}_2^k, D^k).
\]

Notice that the minimization of $\mathcal{L}_\mu(\mathcal{P}_1, \mathcal{P}_2^k, D^k)$ with respect to $\mathcal{P}_1$ is decomposable in terms of the easy-to-compute formula makes the algorithm very appealing.

**Block 2:** The second step of the algorithm that corresponds to \(^{(6b)}\) consists of the operation

\[
\mathcal{P}_2^{k+1} = \arg \min \mathcal{L}_\mu(\mathcal{P}_1^k, \mathcal{P}_2, D^k)
\]

Notice that the minimization of $\mathcal{L}_\mu(\mathcal{P}_1, \mathcal{P}_2^k, D^k)$ with respect to $\mathcal{P}_1$ is decomposable in terms of the matrix variables $\{X_{C;r}\}_{r=1}^q$ and $\{X_{N;s}\}_{s=0}^p$. Hence, the update of $X_{C;r}$ reduces to the problem \(^{(9)}\) for $\bar{Z} = X_{C;r}(C_r, C_r)$. As shown in Lemma \(^{[1]}\) this can be performed via the eigenvalue decomposition of a $|C_r| \times |C_r|$ matrix. In addition, the updated value of $X_{N;s}$ is a minimizer of the function

\[
\mathcal{L}_{N;s}(Z) = \|z_s - (M_s, Z) + \lambda_{z;s}/\mu\|_F^2 + \|X \odot N_s - Z + (1/\mu)A_{N,s}\|_F^2
\]

By taking the derivatives of this function, it is possible to find an explicit formula for $Z_{\text{opt}}$. Define $\mathcal{L}'_{N;s}(Z) \in S(N_s)$ as the gradient of $\mathcal{L}_{N;s}(Z)$ with the following structure:

\[
\mathcal{L}'_{N;s}(Z) \triangleq \left[ \frac{\partial \mathcal{L}_{N;s}}{\partial \text{Re}\{Z_{ij}\}} + i \frac{\partial \mathcal{L}_{N;s}}{\partial \text{Im}\{Z_{ij}\}} \right]_{i,j=1,\ldots,n}
\]

Then, we have

\[
\mathcal{L}'_{N;s}(Z)/2 = Z - X \odot N_s - (1/\mu)A_{N,s} - (1/\mu)M_s,
\]

Therefore,

\[
Z_{\text{opt}} = X \odot N_s + (1/\mu)A_{N,s} + y_aM_s,
\]

where $y_a = z_s - (M_s, Z_{\text{opt}}) + \lambda_{z;s}/\mu$. Hence, it only remains to derive the scalar $y_s$, which can be done by inner multiplying $M_s$ to the both sides of the equation \(^{(18)}\).

Closed-form solutions for each step of the over-relaxed ADMM can be derived similarly, which leads to Algorithm \(^{[1]}\)

**Theorem 2.** Assume that Slater’s conditions hold for the decomposable SDP problem \(^{(1)}\). For $\alpha = 1$, the sequence $\{X^k\}_{k=0}^\infty$ generated by Algorithm \(^{[1]}\) converges to an optimal solution for \(^{(1)}\).

**Proof.** The convergence of both primal and dual variables is guaranteed for a standard ADMM problem if the matrix $B$ in \(^{(4)}\) has full column rank \(^{(3)}\). After realizing that $\mathcal{P}_1$ is obtained from a two-block ADMM procedure, the theorem can be concluded form the fact that the equivalent of $B$ for the algorithm \(^{(1)}\) is a mapping from the variables $\{X_{C;r}\}_{r=1}^q$ and $\{X_{N;s}\}_{s=0}^p$ to

\[
\{X_{C;r}\}_{r=1}^q, \{X_{N;s}\}_{s=0}^p \text{ and } \{M_s, X_{N;s}\}_{s=0}^p
\]

which is not singular, i.e., it has full column rank. The details are omitted for brevity.

**V. Optimal Power Flow**

Consider an $n$-bus electrical power network with the topology described by a simple graph $G = (\mathcal{V}, \mathcal{E})$, meaning that each vertex belonging to $\mathcal{V} = \{1, \ldots, n\}$ represents a node of the network and each edge belonging to $\mathcal{E}$ represents a transmission line. Let $Y \in \mathbb{C}^{n \times n}$ denote the admittance matrix of the network. Define $V \in \mathbb{C}^n$ as the voltage phasor vector,
Algorithm 1 Over-relaxed ADMM for decomposed SDP

1: Initialize $X^0, \{s^0_h\}_{h=1}^p, \{X^0_{C;r}\}_{r=1}^p, \{X^0_{N;s}\}_{s=0}^p, \{A^0_{C;r}\}_{r=1}^p, \{A^0_{N;s}\}_{s=0}^p, \{\lambda^0_{z;s}\}_{s=0}^p$
2: repeat
3: $X^{k+1} := \left[\sum_{r=1}^p C_r \circ (X_{C;r}^{k} - A_{C;r}^{k}/\mu) + \sum_{r=1}^p N_s \circ (X_{N;s}^{k} - A_{N;s}^{k}/\mu)\right] \circ c \left[\sum_{r=1}^p C_r + \sum_{s=1}^p N_s\right]$
4: $z^{k+1} := (M_0, X_{N;0}^k) - (\lambda_{z;0}^k + 1)/\mu$
5: $\lambda_{z;0}^{k+1} := \max \{\min \{\langle M_s, X_{N;s}^k \rangle - \lambda_{z;s}^k/\mu, u_s, l_s\} \text{ for } s = 1, 2, \ldots, p\}$
6: $\hat{\lambda}_{C;r}^{k+1} := \lambda_{C;r}^{k+1} + \mu(\alpha - 1)(X_{C;r}^{k+1} \circ C_r - X_{C;r}^k)$ for $r = 1, 2, \ldots, q$
7: $\hat{\lambda}_{N;s}^{k+1} := \lambda_{N;s}^{k+1} + \mu(\alpha - 1)(X_{N;s}^{k+1} \circ N_s - X_{N;s}^k)$ for $s = 0, 1, \ldots, p$
8: $\hat{\lambda}_{C;r}^{k+1} := \lambda_{C;r}^{k+1} + \mu(\alpha - 1)(z_{s;0}^{k+1} - \langle M_r, X_{N;s}^{k+1} \rangle)$ for $s = 0, 1, \ldots, p$
9: $X_{C;r}^{k+1} := (X_{C;r}^{k+1} \circ C_r + \hat{\lambda}_{C;r}^{k+1}/\mu)^+$ for $r = 1, 2, \ldots, q$
10: $y_{k+1} := z_{s;0}^{k+1} + \hat{\lambda}_{N;s}^{k+1}/\mu - \langle M_s, N_s \circ X_{N;s}^{k+1} \circ \hat{\lambda}_{N;s}^{k+1}/\mu \rangle$ for $s = 0, 1, \ldots, p$
11: $X_{N;s}^{k+1} := N_s \circ X_{N;s}^{k+1} + \hat{\lambda}_{N;s}^{k+1}/\mu + y_{k+1}^{k+1}/\mu$ for $s = 0, 1, \ldots, p$
12: $\hat{\lambda}_{C;r}^{k+1} := \hat{\lambda}_{C;r}^{k+1} + \mu(\alpha - 1)(X_{C;r}^{k+1} \circ C_r - X_{C;r}^{k+1})$ for $r = 1, 2, \ldots, q$
13: $\hat{\lambda}_{N;s}^{k+1} := \hat{\lambda}_{N;s}^{k+1} + \mu(\alpha - 1)(X_{N;s}^{k+1} \circ N_s - X_{N;s}^{k+1})$ for $s = 0, 1, \ldots, p$
14: $\hat{\lambda}_{C;r}^{k+1} := \hat{\lambda}_{C;r}^{k+1} + \mu(z_{s;0}^{k+1} - \langle M_r, X_{N;s}^{k+1} \rangle)$ for $s = 0, 1, \ldots, p$
15: until meet stopping criterion

where $V_{\text{in}}, V_{\text{max}}, P_{\text{in}}, P_{\text{max}}, Q_{\text{in}}$ and $Q_{\text{max}}$ are constant limits, and $f_k(P_k)$ is a convex function accounting for the power generation cost at node $k$. This problem often include additional constraints (such as thermal limits over the lines), which are ignored here to streamline the presentation. For the same reason, assume that the objective function is the total active power loss $\sum_{k \in V_H} P_k$. More details on a general formulation may be found in [6].

OPF is a highly non-convex problem, which is known to be difficult to solve in general. However, the constraints of problem (19) can all be expressed as linear functions of the entries of the quadratic matrix $VV^*$. This implies that the constraints of OPF are linear in terms of a matrix variable $W \triangleq VV^*$. One can reformulate OPF by replacing each $V_i V_i^*$ by $W_{ij}$ and represent the constraints in the form of problem (1) with a representative graph that is isomorphic to the network topology graph $\mathcal{H}$. In order to preserve the equivalence of the two formulations, two additional constraints must be added to the problem: (i) $W \succeq 0$, (ii) $\text{rank}(W) = 1$. If we drop the rank condition as the only non-convex constraint of the reformulated OPF problem, we attain the SDP relaxation of OPF that is convex:

\begin{align}
\text{minimize} & \quad \langle W, (Y + Y^*)/2 \rangle \quad (20a) \\
\text{subject to} & \quad \sum_{k \in V_H} |V_k| \leq V_{\text{max}}^k, \quad k \in \mathcal{N} \quad (20b) \\
& \quad Q_k \leq Q_k, \quad k \in \mathcal{N} \quad (20c) \\
& \quad P_k \leq P_k, \quad k \in \mathcal{N} \quad (20d) \\
& \quad P + Q = \text{diag}\{VV^*Y^*\} \quad (20e)
\end{align}

where $V_{\text{in}}, V_{\text{max}}, P_{\text{in}}, P_{\text{max}}, Q_{\text{in}}$ and $Q_{\text{max}}$ are constant limits, and $f_k(P_k)$ is a convex function accounting for the power generation cost at node $k$. This problem often include additional constraints (such as thermal limits over the lines), which are ignored here to streamline the presentation. For the same reason, assume that the objective function is the total active power loss $\sum_{k \in V_H} P_k$. More details on a general formulation may be found in [6].

As stated in the introduction, several papers in the literature have shown great promises for finding global or near-global solutions of OPF using the above relaxation or a penalized version of the SDP relaxation. The major drawback of relaxing the OPF problem to SDP is the requirement of defining a matrix variable, which makes the number of scalar variables of the problem quadratic with respect to the number of network buses. However, we have shown in [1] that real-world grids would have a low treewidth, e.g., at most 26 for the Polish test system with over 3000 buses. This makes our proposed numerical algorithm scalable and highly parallelizable for the
above SDP relaxation. As an example, the SDP relaxation of OPF for a large-scale European grid with 13659 buses amounts to simple operations over 1093 matrices of size 31 by 31 or smaller.

VI. SIMULATION RESULTS

In this section, we evaluate the performance of the proposed algorithm for solving the SDP relaxation of OPF over Pan European Grid Advanced Simulation and State Estimation (PEGASE) test systems [54], [55]. All simulations are run in MATLAB using a laptop with an Intel Core i7 quad-core 2.2 GHz CPU and 12 GB RAM. The time per 100 iteration is between 26 and 230 seconds in a MATLAB implementation without parallelization, which can be reduced significantly in C++ and through parallel computing.

The experiment results are summarized in Table I. The second column indicates the total number of upper and lower bounds of the form (25) (i.e., $2 \times p$). The number of positive-semidefinite submatrices of the form (25) and their maximum size are shown in the third and forth columns, respectively. The over-relaxation parameter $\alpha = 1.8$ is used for all cases. The quality of solutions obtained via 20,000 ADMM iterations is given in columns 6 to 10. As shown in Figure 1 for all simulated cases, the residue function $\varepsilon^k$ (as defined in (8)) is monotonically decreasing to the ultimate value in the sixth column of Table I. The convergence behavior of the ADMM coupling constraint (i.e., $\|Ax^{20000} + By^{20000} - c\|_2$) and the cost value for different cases are depicted in Figure 1 as well. In order to further assess the quality of solutions after 20,000 ADMM iterations, the maximum violation of inequality constraints in (26) and the largest absolute value among the negative eigenvalues of all submatrices $X^{20000}\{C_r, \bar{C}_r\}$ are given in the seventh and eighth columns.

To elaborate on the algorithm, note that every iteration amounts to a basic matrix operation or an eigendecomposition over matrices of size at most $31 \times 31$ for the PEGASE 13659-bus system. Further preconditioning efforts could dramatically reduce the number of iterations (as OPF is often very ill-conditioned due to high inductance-to-resistance ratios), and this is left for future work.

VII. CONCLUSIONS

Motivated by the application of sparse semidefinite programming (SDP) to power networks, the objective of this work is to design a fast and parallelizable algorithm for solving sparse SDPs. To this end, the underlying sparsity structure of a given SDP problem is captured using a tree decomposition technique, leading to a decomposed SDP problem. A highly distributed/parallelizable numerical algorithm is developed for solving the decomposed SDP, based on the alternating direction method of multipliers (ADMM). Each iteration of the designed algorithm has a closed-form solution, which involves multiplications and eigenvalue decompositions over certain submatrices induced by the tree decomposition of the sparsity graph. The proposed algorithm is applied to the classical optimal power flow problem, and also evaluated on large-scale PEGASE benchmark systems. The numerical technique developed in this paper enables solving complex models of various power optimization problems, such as optimal power flow, unit commitment and state estimation, through conic optimization.

<table>
<thead>
<tr>
<th>Test cases</th>
<th>Number of inequality constraints</th>
<th>Number of bags</th>
<th>Maximum size of bags</th>
<th>$\mu$</th>
<th>$\varepsilon^k$</th>
<th>Linear coupling violation</th>
<th>Maximum inequality violation</th>
<th>Maximum PSD violation</th>
<th>ADMM cost value</th>
<th>Running time of 100 iterations</th>
</tr>
</thead>
<tbody>
<tr>
<td>1354-bus system</td>
<td>8124</td>
<td>275</td>
<td>13</td>
<td>200</td>
<td>1.7e-3</td>
<td>2.7e-3</td>
<td>2.7e-3</td>
<td>3.4e-5</td>
<td>744.20</td>
<td>26 sec</td>
</tr>
<tr>
<td>2809-bus system</td>
<td>17214</td>
<td>641</td>
<td>13</td>
<td>1000</td>
<td>1.3e-2</td>
<td>1.8e-3</td>
<td>3.1e-4</td>
<td>1.4e-4</td>
<td>1352.74</td>
<td>60 sec</td>
</tr>
<tr>
<td>9241-bus system</td>
<td>55446</td>
<td>857</td>
<td>31</td>
<td>2000</td>
<td>1.0e-1</td>
<td>3.6e-3</td>
<td>6.4e-4</td>
<td>9.4e-5</td>
<td>3151.69</td>
<td>160 sec</td>
</tr>
<tr>
<td>13549-bus system</td>
<td>81954</td>
<td>1093</td>
<td>31</td>
<td>3000</td>
<td>1.8e-1</td>
<td>4.8e-3</td>
<td>7.9e-4</td>
<td>1.3e-4</td>
<td>3908.47</td>
<td>230 sec</td>
</tr>
</tbody>
</table>

REFERENCES

Fig. 1: These plots show the convergence behavior of the residue functions, indefeasibly sequences and cost values for PEGASE test systems. (a): 1354-bus system, (b): 2869-bus system, (c): 9241-bus system, (c): 13659-bus system. Solid red lines in (a) and (b) indicate the optimal values obtained by SDPT3, whereas the dotted red lines in (c) and (d) indicate the optimal values obtained by Mathpower (SDPT3 runs out of memory for the 9241-bus and 13659-bus systems, and therefore Mathpower is used).


