Sparse Semidefinite Programs with Near-Linear Time Complexity

Richard Y. Zhang and Javad Lavaei

***Abstract*—Some of the strongest polynomial-time relaxations to NP-hard combinatorial optimization problems are semidefinite programs (SDPs), but their solution complexity of up to $O(n^{6.5}L)$ time and $O(n^4)$ memory for L accurate digits limits their use in all but the smallest problems. Given that combinatorial SDP relaxations are often sparse, a technique known as chordal conversion can sometimes reduce complexity substantially. In this paper, we describe a modification of chordal conversion that allows any general-purpose interior-point method to solve a certain class of sparse SDPs with a guaranteed complexity of $O(n^{1.5}L)$ time and $O(n)$ memory. To illustrate the use of this technique, we solve the MAX k-CUT relaxation and the Lovasz Theta problem on power system models with up to $n = 13659$ nodes in 5 minutes, using SeDuMi v1.32 on a 1.7 GHz CPU with 16 GB of RAM. The empirical time complexity for attaining L decimal digits of accuracy is $\approx 0.001n^{1.1}L$ seconds.**

I. Introduction

Consider large-and-sparse instance of the semidefinite program

$$\begin{aligned} &\text{minimize} && C \bullet X && \text{(SDP)}\\ &\text{subject to} && A_i \bullet X = b_i \quad \forall i \in \{1,\dots,m\}\\ &&& X \succeq 0 \end{aligned}$$

where the problem data are the sparse $n \times n$ real symmetric matrices $C, A_1, \dots, A_m$ sharing a common sparsity pattern and the m-dimensional vector b. The notation $A_i \bullet X \equiv \operatorname{tr} A_i X$ refers to the usual matrix inner product, and the constraint $X \succeq 0$ restricts the matrix X to be symmetric positive semidefinite.

Interior-point methods are the most reliable approach for solving small- and medium-scale SDPs, but become prohibitively time- and memory-intensive for large-scale problems. A fundamental difficulty is the positive semidefinite constraint $X \succeq 0$, which usually couples every element in the matrix decision variable X with everyone else, causing all of them to take on nonzero values. The solution $X^\star$ of (SDP) is usually fully dense, despite any sparsity in the problem data. Interior-point methods solve highly sparse SDPs in approximately the same time as dense SDPs of the same size: to L digits of accuracy in $O(n^{6.5}L)$ time and $O(n^4)$ memory.

Semidefinite programs arise as some of the best convex relaxations to a number of important nonconvex optimization problems, including integer programming, combinatorial optimization [1], [2], but their high complexity severely limit their use in practice. Semidefinite relaxations are generally considered intractible for problems that arise in real-life applications.

A. Sparse semidefinite programming

Much larger instances of (SDP) can be solved when the data matrices $C, A_1, \dots, A_m$ are sparse, with a nonzero pattern that has a *sparse chordal completion*. If we identify the problem sparsity with an undirected graph on n vertices, then this is to say that the graph has a *bounded treewidth*. Algebraically, this means that every matrix of the form $S = C - \sum_i y_i A_i \succeq 0$ factors into a *sparse* Cholesky factor L satisfying $LL^T = S$, possibly after reordering the rows and columns. Note that this condition is satisfied by many real-life problems, with applications ranging from electric power systems [3], [4], [5] to machine learning [6].

Problems with the chordal sparsity structure may be solved at reduced complexity using a reformulation procedure introduced by Fukuda and Nakata et al. [7], [8]. The essential idea is to split the large conic constraint $X \succeq 0$ into $\ell \le n$ smaller conic constraints, enforced over its principal submatrices (we refer to the principal submatrix of X indexed by $\mathcal{I}_j$ as $X[\mathcal{I}_j, \mathcal{I}_j]$)

$$\begin{aligned} &\text{minimize} && C \bullet X && \text{(CSDP)}\\ &\text{subject to} && A_i \bullet X = b_i \quad \forall i \in \{1,\dots,m\}\\ &&& X[\mathcal{I}_j, \mathcal{I}_j] \succeq 0 \quad \forall j \in \{1,\dots,\ell\} \end{aligned}$$

where the index sets $\mathcal{I}_1, \dots, \mathcal{I}_\ell \subset \{1,\dots,n\}$ are chosen to make (SDP) and (CSDP) equivalent. To induce sparsity in an interior-point solution of (CSDP), the ℓ principal submatrices $X[\mathcal{I}_1, \mathcal{I}_1], \dots, X[\mathcal{I}_\ell, \mathcal{I}_\ell]$ are split into distinct matrix variables $X_1, \dots, X_\ell$, constrained by the need for their overlapping elements to agree

$$\begin{aligned} &\text{minimize} && \sum_{j=1}^{\ell} C_j \bullet X_j && \text{(CTC)}\\ &\text{subject to} && \sum_{j=1}^{\ell} A_{i,j} \bullet X_j = b_i \quad \forall i \in \{1,\dots,m\},\\ &&& X_j \succeq 0 \quad \forall j \in \{1,\dots,\ell\},\\ &&& N_{u,v}(X_v) = N_{v,u}(X_u) \quad \forall \{u,v\} \in \mathcal{E}. && (1) \end{aligned}$$

Here, the linear operator $N_{i,j}(\cdot)$ outputs the overlapping elements of two principal submatrices given the latter as the argument

$$N_{u,v}(X[\mathcal{I}_v, \mathcal{I}_v]) = X[\mathcal{I}_u \cap \mathcal{I}_v, \mathcal{I}_u \cap \mathcal{I}_v],$$

This work was supported in part by a DARPA YFA Award, an ONR YIP Award, and an AFOSR YIP Award.

This work was supported by the ONR YIP Award, DARPA YFA Award, AFOSR YIP Award, NSF CAREER Award, and NSF EPCN Award.

R.Y. Zhang and J. Lavaei are with the Department of Industrial Engineering and Operations Research, University of California, Berkeley, CA 94720, USA `ryz@berkeley.edu` and `lavaei@berkeley.edu`

and $\mathcal{E}$ are the edges to a certain tree graph. Once a solution $X_1^\star, \dots, X_\ell^\star$ to (CTC) is found, a corresponding solution for (SDP) is recovered by solving

$$\begin{aligned} &\text{find} && X \succeq 0 && \text{(PSDMC)} \\ &\text{satisfying} && X[\mathcal{I}_j, \mathcal{I}_j] = X_j^\star \quad \forall j \in \{1, \dots, \ell\}, \end{aligned}$$

in *closed-form* using a modification of the sparse Cholesky factorization algorithm (see e.g. [9]). Section II reviews this reformulation procedure in detail, including implementation considerations.

The complexity of solving the reformulated problem (CTC) is closely associated with the following constant

$$\text{(Clique size)} \qquad \omega = \max\{|\mathcal{I}_1|, |\mathcal{I}_2|, \dots, |\mathcal{I}_\ell|\},$$

whose value has the interpretation as the size of the largest clique in on the chordal graph completion. Finding the *best* choice of $\mathcal{I}_1, \dots, \mathcal{I}_m$ to minimize ω is NP-complete [10], but provably good choices within a $\log(n)$ factor of the minimum are readily available, using heuristics originally developed for the fill-reduction problem in numerical linear algebra [11], [12], [13].

Our assumption of sparsity with a sparse chordal completion guarantees the existence of $\mathcal{I}_1, \dots, \mathcal{I}_m$ with bounded $\omega = O(1)$. In practice, such a choice is readily found using standard fill-reducing heuristics in a negligible amount of time. Our numerical results in Section V used MATLAB's inbuilt approximate minimum degree heuristic to find choices of $\mathcal{I}_1, \dots, \mathcal{I}_m$ with $\omega \le 35$ in less than 0.01 seconds on a standard desktop for problems as large as $n = 13659$.

To give a rigorous complexity bound for (CTC), we state some nondegeneracy assumptions, which are standard for interior-point methods.

Assumption 1 (Nondegeneracy). *We assume in (CTC):*

1) *(Linear independence) The matrix* $\mathbf{A} = [\operatorname{vec} A_1, \dots, \operatorname{vec} A_m]$ *has full column-rank, meaning that* $\mathbf{A}^T\mathbf{A}$ *is invertible.*
2) *(Slater's condition) There exist* $X_1, \dots, X_\ell \succ 0$, y, *and* $S_1, \dots, S_\ell \succ 0$, *such that* $\sum_j A_{i,j} \bullet X_j = b_i$ *and* $\sum_i y_i A_{i,j} + S_j = C_j$ *for all* $i \in \{1, \dots, m\}$ *and* $j \in \{1, \dots, m\}$.

Note that linear independence sets $m = O(n)$, since we have already assumed each column of $\mathbf{A}$ to contain no more than $O(n)$ nonzeros. Also, Slater's condition is satisfied in solvers like SeDuMi [14] and MOSEK [15] using the homogenous self-dual embedding technique [16].

Theorem 1. *Under Assumption 1, there exists an interior-point method that computes an approximate solution for (CTC) that is accurate to L digits in* $O(\omega^{6.5} n^{3.5} L)$ *time and* $O(\omega^4 n^2)$ *memory.*

Proof: It is a seminal result by Nesterov and Nemirovski [17] that an interior-point method solves every order-θ linear conic program to L accurate digits in $k = O(\sqrt{\theta} L)$ iterations assuming Slater's condition. Problem (CTC) optimizes over a convex cone of order $\theta \le \omega n$. We show in Section IV that each interior-point iteration takes $O(\omega^6 n^3)$ time and $O(\omega^4 n^2)$ memory. ■

B. Main result

Empirical results [7], [8], [18], [4] find that many instances of (CTC) can be actually solved in near-linear time using an interior-point method. Unfortunately, other problem instances attain the worst-case cubic complexity quoted in Theorem 1. The key issue is the large number of the *overlap* constraints

$$N_{u,v}(X_v) = N_{v,u}(X_u) \quad \forall \{u, v\} \in \mathcal{E},$$

which are imposed in addition to the constraints already present in the original problem. These overlap constraints can significantly increase the size of the linear system solved at each interior-point interation, thereby offsetting the benefits of increased sparsity [19, Section 14.2]. In [20], omitting some of the constraints made (CTC) easier to solve, but at the cost of also making the reformulation from (SDP) inexact.

The main result in this paper is a set of extra conditions on (SDP) that allow (CTC) to solved in near-linear time, by applying a generic interior-point method to the *dualized* version of the problem.

Assumption 2 (Decoupled submatrices). *Let* $\mathcal{A} = \{A_1, \dots, A_m\}$ *be the data, and* $\mathcal{I} = \{\mathcal{I}_1, \dots, \mathcal{I}_j\}$ *be the index sets in (CSDP). Then for every* $A_i \in \mathcal{A}$, *there is a* $\mathcal{I}_j \in \mathcal{I}$ *and some* $A_{i,j} \in \mathbb{S}^n$ *such that* $A_i \bullet X = A_{i,j} \bullet X[\mathcal{I}_j, \mathcal{I}_j]$.

The condition is similar to the *correlative sparsity* property studied in [21]. However, note that previous efforts have only been able to exploit this property inside a first-order method [22], [21], [23], [24]. These have time complexity that is exponential $O(\exp L)$ in the number of accurate digits L, are sensitive to the conditioning of the data, and require considerable fine-tuning of the parameters in practice.

Theorem 2. *Under Assumptions 1 and 2, there exists an interior-point method that computes an approximate solution for (CTC) that is accurate to L digits in* $O(\omega^{6.5} n^{1.5} L)$ *time and* $O(\omega^4 n)$ *memory.*

Proof: We repeat the proof of Theorem 1, but show in Section IV that, under Assumption 2, each interior-point iteration of the *dualized* version of (CTC) solves in just $O(\omega^6 n)$ time and $O(\omega^4 n)$ memory, due to a chordal block-sparsity structure discussed in Section III. ■

C. Applications

Assumption 2 is widely applicable to many important SDP relaxations of combination problems. In this paper, we consider two examples that trivially satisfy the assumption by construction. Accordingly, the techniques in this paper are able to solve these problems on graphs with up to $n = 13659$ vertices to 5-6 digits of accuracy in less than 5 minutes on a modest work station; see our numerical results in Section V.

MAXCUT and MAX k-CUT relaxations. Let C be the (weighted) Laplacian matrix for a graph $G = \{V, \mathcal{E}\}$ with n vertices. Frieze and Jerrum [2] proposed a randomized algorithm to solve MAX k-CUT with an approximation ratio of $1 - 1/k$ based on solving

$$\begin{aligned}\text{maximize}\quad & \frac{k-1}{2k} C \bullet X && \text{(MkC)}\\ \text{subject to}\quad & X_{i,i} = 1 \quad \forall i \in \{1, \dots, n\}\\ & X_{u,v} \geq \frac{-1}{k-1} \quad \forall \{u, v\} \in \mathcal{E}\\ & X \succeq 0\end{aligned}$$

for all $i \in \{1, \dots, n\}$, and all $\{u, v\} \in \mathcal{E}$, based on the Goemans and Williamson 0.878 algorithm [1], which is recovered by setting $k = 2$ and removing the redundant constraint $X_{u,v} \geq -1$. In both cases, observe that each constraint in (MkC) affects just a single matrix element in X, so Assumption 2 is trivially satisfied.

Lovasz Theta. Let $G = \{V, \mathcal{E}\}$ be a graph with n vertices. The Lovasz number $\vartheta(G)$ [25] is the solution to the following dual semidefinite program (here, e_j is the j-th column of the size-n identity matrix)

$$\begin{aligned}\text{minimize}\quad & \lambda && \text{(LT)}\\ \text{subject to}\quad & \mathbf{1}\mathbf{1}^T - \sum_{\{i,j\} \in \mathcal{E}} y_{i,j}(e_i e_j^T + e_j e_i^T) \preceq \lambda I\end{aligned}$$

and serves as a bound for a number of graph theoretical quantities that are NP-hard to compute. Problem (LT) does not satisfy Assumption 2. However, given that $\lambda > 0$ for all interesting applications, we may divide the linear matrix inequality through by λ, redefine $y \leftarrow y/\lambda$, apply the Schur complement lemma, and take the Lagrangian dual to yield a sparse formulation

$$\begin{aligned}\text{minimize}\quad & \begin{bmatrix} I & \mathbf{1} \\ \mathbf{1}^T & 0 \end{bmatrix} \bullet X && \text{(LT}'\text{)}\\ \text{subject to}\quad & X_{i,j} = 0 \quad \forall \{i, j\} \in \mathcal{E}\\ & X_{(n+1),(n+1)} = 1\\ & X \succeq 0\end{aligned}$$

for all $\{i, j\} \in \mathcal{E}$. Like in (MkC), each constraint just a single matrix element in X, so Assumption 2 is trivially satisfied.

Even problems that do not strictly satisfy Assumption 2 tend to solve much faster after the "dualizing" step. An example with important real-life impliciations is the SDP relaxation of the OPF problem. In Section V we solve instances of the OPF SDP with as many as $n = 13659$ buses to 5-6 digits of accuracy in less than 5 minutes on the same modest work station, with an empirical linear-time complexity.

Optimal power flow (OPF). The OPF problem is a large-scale optimization problem that plays a vital role in the operation of an electric power system. Variants of this problem is solved by the system operator from every few minutes to every several months, in order to ensure a reliable and inexpensive supply of electricity. The OPF problem is well-known to be nonlinear, nonconvex, and NP-hard in general, but can be relaxed into an SDP problem using standard techniques:

$$\begin{aligned}\text{minimize}\quad & C \bullet X && \text{(OPF)}\\ \text{subject to}\quad & b_i^\ell \leq A_i \bullet X \leq b_i^u \quad \forall i \in \{1, \dots, m\}\\ & X \succeq 0.\end{aligned}$$

Here, the X is a Hermitian positive semidefinite matrix, the constraint matrices $A_1, \dots, A_m$ measure physical quantities within the system (e.g. voltage magnitudes, power flows, and power injections), and $b^\ell, b^u \in \mathbb{R}^m$ provide a corresponding list of upper- and lower-bounds. If the solution matrix $\hat{X}$ is rank-one, then the relaxation (OPF) is exact, and a globally-optimal solution to the original NP-hard problem can be extracted by factoring $\hat{X} = \hat{v}\hat{v}^T$.

A surprising feature of the OPF problem is that the SDP relaxation (OPF) is often exact [26]. Even when this is not the case, the problem can be slightly perturbed to guarantee an exact relaxation; a solution of the perturbed problem can then serve as a near-globally-optimal solution to the original problem. This perturbation technique, known as penalized SDP, was tested on IEEE benchmarks, Polish, Ontario and New York Grids, over 7000 settings, where it always obtained a solution with global optimality guarantee exceeding 99% [27], [4].

It is therefore safe to say that the practical cost of solving a size-n OPF is about the same as that of solving a size-n SDP. Structurally, the OPF problem *almost* satisfies Assumption 2. More specifically, the voltage magnitude, and line flow measures satisfy the assumption, but the nodal power constraints do not. Nevertheless, we would expect its solution time to closely match the figure quoted in Theorem 2. Our numerical results in Section V find this to indeed be the case.

D. Related work

Conversion methods, which split a size-n semidefinite cone into numerous smaller cones, were first proposed in [7], [8], and have been implemented as the SparseCoLO packaged in [18]. The same idea has been explored extensively in power systems applications to solve SDPs with n on the order of several thousands, largely motivated by the OPF problem described above [4], [3]. Our work builds upon the existing literature by studying the sparsity patterns of the converted problems in detail. Our goal is to offer an explanation for the rapid solution times, as well as guarantees in special cases where Assumption 2 is satisfied.

Another property of chordal sparsity is that it allows the barrier function $-\mu \log\det S$ and its gradient and Hessian to have recursive close-form solutions, with about the same cost as computing a sparse Cholesky factorization. Andersen, Dahl, and Vandenberghe [28], [9] used this insight to develop an efficient interior-point method on the nonsymmetric cone of sparse positive semidefinite matrices, available as the CVXOPT package [29]. It is worth noting that the algorithm explicitly forms and factorizes the $m \times m$ fully-dense Hessian matrix, so its worst-case per-iteration complexity is cubic $O(n^3)$ time and quadratic $O(n^2)$ memory.

The per-iteration cost of an optimization algorithm can be significantly reduced by avoiding second-order information. This idea appears in first-order methods like bundle methods [30] and ADMM [24], [23], as well as nonconvex algorithms based on factoring the primal variable $X = RR^T$ [31]. Unfortunately, convergence guarantees are significantly weakened. Of the methods with global convergence bounds, only sublinear error rates of $O(1/k^{\alpha})$ can be established; an approximation with L accurate digits require exponential iterations $k = O(\exp(L))$ to compute.

By comparison, our results guarantee the Fukuda and Nakata et al. reformulation procedure [7], [8] to solve SDPs satisfying Assumption 2 in $O(n^{1.5}L)$ time and $O(n)$ memory—linear in the number of accurate digits L and near-linear in the problem size n. In Section V, we solve the MAX 3-CUT, Lovasz Theta, and OPF problems for a suite of 40 power system graphs, the largest of which contains $n = 13659$ nodes, but with $\omega \leq 35$. Our results have a time complexity of $\approx 0.001n^{1.1}L$ seconds for L decimal digits. In particular, we solve an example with $n = 13659$ to $L = 4.5$ digits in 5 minutes, on a relatively modest CPU with just 16 GB of RAM.

Sparse matrix notations

The sets $\mathbb{R}^{m\times n}, \mathbb{S}^n, \mathbb{S}^n_+, \mathbb{S}^n_{++}$ refer to the $m \times n$ real matrices, $n\times n$ real symmetric matrices, positive semidefinite matrices, and positive definite matrices, respectively. A **sparsity pattern** is a binary matrix $E \in \{0,1\}^{m\times n}$. The sum of two sparsity patterns is their logical "or". A matrix X is said **to have sparsity pattern** E if $X_{i,j} = 0$ whenever $E_{i,j} = 0$; the set $\mathbb{R}^{m\times n}_E$ (resp. $\mathbb{S}^n_E$) refers to the $m \times n$ matrices (resp. $n \times n$ symmetric matrices) with sparsity pattern E. The **Euclidean projection onto sparsity pattern** is denoted as $\Pi_E(\cdot)$: the (i,j)-th element of $\Pi_E(M)$ is zero if $E_{i,j} = 0$, and $M_{i,j}$ if $E_{i,j} = 1$.

II. Preliminaries

In this section, we review the steps taken to convert (SDP) into (CTC) under the assumption that the data matrices $C, A_1, \ldots, A_m$ share a sparsity pattern E with a sparse chordal completion. The original derivations in [7], [8] made extensive use of graph theory. This section gives an alternative presentation of the conversion method using linear algebra. As a result, the conversion can be easily implemented with high-level function calls to sparse linear algebra software libraries.

A. Sparse Chordal Completion

We identify the symmetric sparsity pattern E with an undirected graph on n vertices, containing an edge between the i-th and j-th vertices whenever $E_{i,j} \neq 0$. The pattern E is said to be *chordal* if its graph does not contain an induced cycle with length greater than three. If E is not chordal, then we may add nonzeros to it until it becomes chordal; the resulting pattern $E' \geq 0$ is called a *chordal completion* (or *triangulation*) of E. Any chordal completion E' with at most $O(n)$ nonzeros is a *sparse* chordal completion of E.

Matrices with chordal sparsity are precisely those that factor with no fill. To explain, consider solving the linear equations $Sx = b$ with S being positive definite by factoring S into LL^T. If S is sparse with sparsity pattern E, then its Cholesky factor $L = \mathbf{chol}(S)$ has sparsity pattern $F = \mathbf{symbchol}(E)$ determined by symbolic analysis. Now, F contains all nonzero elements in the lower-triangular portion of E, but usually also additional nonzeros in positions where E is zero, known as *fill-in*, which adversely affect complexity. Fill-in can often be reduced by reordering the rows and columns. i.e. by choosing a permutation Q and factoring the permuted pattern QEQ^T. It is a classic result that E can be reordered to factor with zero fill-in if and only if E is a chordal [32, p.851]; see also [19, Sec.4.2].

Matrices with sparse chordal completions are precisely those with sparse Cholesky factorizations; see the discussion in [19, Sec.6.6]. More specifically, if the Cholesky factorization $F = \mathbf{symbchol}(QEQ^T)$ is sparse, then $E' = Q^T(F + F^T)Q$ is a sparse chordal completion. Finding the optimal permutation Q that minimizes fill-in is NP-hard, but provably good orderings can be computed in polynomial time, and heuristics like minimum degree and nested dissection are usually sufficient in practice [13].

B. Chordal decomposition

Let the data matrices $C, A_1, \ldots, A_m$ share a sparsity pattern E. Since the data can always be permuted ahead of time, we assume without loss of generality that $F = \mathbf{symbchol}(E)$ is sparse. Then, it is natural to describe F in terms of the index sets $\mathcal{I}_1, \ldots, \mathcal{I}_n \subseteq \{1, \ldots, n\}$, where[1]:

$$\text{(Cliques)} \qquad \mathcal{I}_j = \{i \in \{1, \ldots, n\} : F_{i,j} \neq 0\}. \qquad (2)$$

The following result is well-known in the study of sparse positive semidefinite matrices [33], and can be proved using only linear algebra arguments; see [19, Thm.10.1].

Theorem 3 (Primal cone separability). *Given E and $F = \mathbf{symbchol}(E)$, define $E' = F + F^T$ as the chordal completion, and define $\{\mathcal{I}_j\}$ as in (2). Then we have*

$$X \in \Pi_{E'}(\mathbb{S}^n_+) \iff X[\mathcal{I}_j, \mathcal{I}_j] \succeq 0 \quad \forall \in \{1, \ldots, n\}.$$

The objective and constraints in (SDP) affect the decision variable X only where E' is nonzero; the remaining elements have their values determined solely by the conic constraint $X \succeq 0$. Replacing the constraint $X \succeq 0$ with $X = \Pi_{E'}(\mathbb{S}^n_+)$ and applying Theorem 3 to the latter problem yields (CSDP). Indeed, it is always possible to recover a solution $X^\star \succeq 0$ given $\Pi_{E'}(X^\star)$; see [7] and also [19, Sec.10] and [9].

For certain problems like (MkC), we also need to factor $X^\star = V^T V$ and compute matrix-vectors of the form $s = V^T a$, where a is a given vector. Knowing only $\Pi_{E'}(X^\star)$, an ϵ-approximation to s may be computed in linear time and memory, by using a maximum determinant completion algorithm [19, Alg.10.1] to find a sparse triangular Cholesky factor $L \in \mathbb{R}^{n\times n}_F$ satisfying $\Pi_{E'}[(LL^T)^{-1}] = \Pi_{E'}(X + \epsilon I)$ and computing $s \approx L^{-T}a$ via back-substitution.

[1]The index sets $\mathcal{I}_1, \ldots, \mathcal{I}_n$ are the *cliques* of the chordal completion for the problem sparsity graph.

C. Clique tree conversion

Next, we split the principal submatrices of X into distinct matrix variables, in order to put the problem in a canonical form, and to induce sparsity in an interior-point solution. Defining $\{C_j\}, \{A_{i,j}\} \subset \mathbb{S}^n_{E'}$ to satisfy

$$\sum_{j=1}^{n} C_j \bullet X[\mathcal{I}_j, \mathcal{I}_j] = C \bullet X \tag{3}$$
$$\sum_{j=1}^{n} A_{i,j} \bullet X[\mathcal{I}_j, \mathcal{I}_j] = A_i \bullet X$$

for all $i \in \{1, \dots, m\}$ allows us to rewrite (CSDP) in terms of $X_j = X[\mathcal{I}_j, \mathcal{I}_j]$, each constrained to lie within its own cone $X_j \succeq 0$. These new variables are coupled by the need for their overlapping elements to remain equal. Naively enforcing this results in $O(n^2)$ pairwise comparisons, but the figure may be reduced to $O(n)$ by restricting the comparisons to the edges of the corresponding elimination tree, as in

$$N_{u,v}(X_v) = N_{v,u}(X_u) \quad \forall \{u, v\} \in \mathcal{E}. \tag{4}$$

Given the index sets $\mathcal{I}_1, \dots, \mathcal{I}_n$ from (2), the associated *elimination tree* $\mathcal{T} = \{V, \mathcal{E}\}$ is a rooted tree (or forest) on n vertices, with edges $\mathcal{E} = \{\{1, p_1\}, \dots, \{n, p_n\}\}$ defined to connect each j-th vertex to its parent at the p_j-th vertex (except root nodes, which have "0" as their parent) [34]:

$$\text{(E-tree)} \qquad p_j = \begin{cases} 0 & |\mathcal{I}_j| = 1 \\ \min\{i > j : i \in \mathcal{I}_j\} & |\mathcal{I}_j| > 1 \end{cases} \tag{5}$$

The index sets $\mathcal{I}_1, \dots, \mathcal{I}_n$ and the elimination tree $\mathcal{T}$ combine to form a *clique tree* (also known as a *tree decomposition*, a *join tree* or a *junction tree*) for the graph associated with our original sparsity pattern E [34, Sec.4.2]; see also [19, Sec.4.3]. It follows from properties of the clique tree that just $O(n)$ comparisons along the edges $\mathcal{E}$ in (4) are needed to guarantee that all submatrices agree in their overlapping elements [7], [18].

D. Redundant index sets

Some of the index sets $\mathcal{I}_1, \dots, \mathcal{I}_n$ are redundant, and may be removed to yield $\ell \le n$ remaining index sets. More specifically, if $\mathcal{I}_k \subseteq \mathcal{I}_j$, then Theorem 3 is unaffected by deleting $\mathcal{I}_k$, while X_k may be dropped from (CTC), after modifying the splittings (3) to set all $A_{i,k} = C_k = 0$, and substituting $\mathcal{I}_j$ in lieu of $\mathcal{I}_k$ for all overlap constraints. Redundant index sets can be naively identified in $O(n^2)$ pairwise comparisons, or more efficiently in $O(n)$ comparisons using the elimination tree; see [19, Sec.4.4].

III. Block sparsity in the overlap constraints

Problem (CTC) can be rewritten in vectorized form as

$$\begin{aligned} \text{minimize} \quad & c^T x \\ \text{subject to} \quad & \begin{bmatrix} \mathbf{A}^T \\ \mathbf{N}^T \end{bmatrix} x = \begin{bmatrix} b \\ 0 \end{bmatrix} \\ & x \in \mathcal{K} \end{aligned} \tag{6}$$

in which $\mathbf{A}$, c, and x are vectorized version of the matrix variables,

$$\mathbf{A} = \begin{bmatrix} \operatorname{vec} A_{1,1} & \cdots & \operatorname{vec} A_{m,1} \\ \vdots & \ddots & \vdots \\ \operatorname{vec} A_{1,\ell} & \cdots & \operatorname{vec} A_{m,\ell} \end{bmatrix} = \begin{bmatrix} \mathbf{A}_1 \\ \vdots \\ \mathbf{A}_\ell \end{bmatrix},$$
$$c = \begin{bmatrix} \operatorname{vec} C_1 \\ \vdots \\ \operatorname{vec} C_\ell \end{bmatrix}, \quad x = \begin{bmatrix} \operatorname{vec} X_1 \\ \vdots \\ \operatorname{vec} X_\ell \end{bmatrix},$$

the cone $\mathcal{K} = \mathbb{S}_+^{|\mathcal{I}_1|} \times \cdots \times \mathbb{S}_+^{|\mathcal{I}_\ell|}$ is the product of ℓ small semidefinite cones, and the matrix $\mathbf{N}$ enforces the *overlap constraints* in (4)

$$\mathbf{N} = \begin{bmatrix} \mathbf{N}_1 & \cdots & \mathbf{N}_\ell \end{bmatrix}, \tag{7}$$
$$\mathbf{N}_j^T x = \operatorname{vec}[N_{p_j,j}(X_j) - N_{j,p_j}(X_{p_j})].$$

Observe that the matrix $\mathbf{N}$ comprises (up to) ℓ block-columns $\mathbf{N}_1, \dots, \mathbf{N}_\ell$, and that the j-th block-column contains exactly two nonzero blocks, with one at the j-th block-row, and the other at the p_j-th block-row. In other words, the block sparsity pattern of $\mathbf{N}$ matches that of the incidence matrix for the elimination tree $\mathcal{T} = \{V, \mathcal{E}\}$. The following is an immediate consequence.

Proposition 4. *The matrix $\mathbf{N}\mathbf{N}^T$ has a block-sparsity pattern that coincides with the adjacency matrix for the elimination tree $\mathcal{T}$: its (i, j)-th block is nonzero if and only if the i-th and j-th vertices of $\mathcal{T}$ are adjacent.*

Trees are the simplest chordal graphs, and $\mathbf{N}\mathbf{N}^T$ is guaranteed to factor with zero block-fill (i.e. without introducing additional blocks in positions where $\mathbf{N}\mathbf{N}^T$ is zero) after a minimum degree block-ordering, although each existing block may become fully-dense. Each block in $\mathbf{N}$ has up to $\omega(\omega+1)/2$ rows and columns, where $\omega = \max_j |\mathcal{I}_j|$. Therefore, the Cholesky factor of $\mathbf{N}\mathbf{N}^T$ (as well as any matrix with the same block sparsity pattern) requires at most $O(\omega^6 n)$ time and $O(\omega^4 n)$ memory to compute. The following is also immediate consequence.

Proposition 5. *The matrix $\mathbf{N}^T\mathbf{N}$ has a block-sparsity pattern that coincides with the adjacency matrix for the line graph of the elimination tree $\mathcal{L}(\mathcal{T})$: its (i, j)-th block is nonzero if and only if the i-th and j-th edges of $\mathcal{T}$ share a common vertex.*

The line graph of a tree is not necessarily sparse. Suppose $\mathcal{T} = \{V, \mathcal{E}\}$ were the star graph, with vertices $V = \{v_1, \dots, v_n\}$ and edges $\mathcal{E} = \{\{v_1, v_n\}, \{v_2, v_n\}, \dots, \{v_{n-1}, v_n\}\}$. Then all $n-1$ edges share the common vertex v_n, so the line graph $\mathcal{L}(\mathcal{T})$ is the complete graph, and $\mathbf{N}^T\mathbf{N}$ would be fully dense despite any sparsity in $\mathbf{N}$. Hence, in the worst case, the Cholesky factor for $\mathbf{N}^T\mathbf{N}$ requires $O(\omega^6 n^3)$ time and $O(\omega^4 n^2)$ memory to compute.

IV. Complexity of an interior-point iteration

The cost of an interior-point iteration is entirely dominated by the cost of forming and solving the so-called *normal* *equation*, which is used to compute the Newton search direction. In this section, we show that the normal equation associated with the *dualized version* of (6) can be solved in linear time and memory under Assumption 2.

A. The standard iteration

Let us begin by examining the cost of a single interior-point iteration for (CTC), following the general approach outlined in the authoritative survey [14]. Each iteration begins by forming the positive definite scaling matrix $\mathbf{D}$. Given that $\mathcal{K} = \mathbb{S}_+^{|\mathcal{I}_1|} \times \cdots \times \mathbb{S}_+^{|\mathcal{I}_\ell|}$, the matrix $\mathbf{D} = \mathrm{diag}(\mathbf{D}_1^s, \ldots, \mathbf{D}_\ell^s)$ is block-diagonal with $\ell \le n$ fully-dense blocks, and each block $\mathbf{D}_j^s$ is of size $\frac{1}{2}|\mathcal{I}_j|(|\mathcal{I}_j|+1)$. Forming $\mathbf{D}$ takes $O(\omega^3 n)$ time and $O(\omega^2 n)$ memory, where $\omega = \max_j |\mathcal{I}_j|$.

Next, the normal equation is formulated, with Hessian matrix

$$\begin{bmatrix}\mathbf{A} & \mathbf{N}\end{bmatrix}^T \mathbf{D} \begin{bmatrix}\mathbf{A} & \mathbf{N}\end{bmatrix} = \sum_{j=1}^{\ell} \begin{bmatrix} \mathbf{A}_j^T \mathbf{D}_j^s \mathbf{A}_j & \mathbf{A}_j^T \mathbf{D}_j^s \mathbf{N}_j \\ \mathbf{N}_j^T \mathbf{D}_j^s \mathbf{A}_j & \mathbf{N}_j^T \mathbf{D}_j^s \mathbf{N}_j \end{bmatrix} \quad (8)$$

no larger than $\frac{1}{2}\omega(\omega+1)n$, if we take (CTC) to be primal-dual strictly feasible. Forming and factoring the matrix in (8) is the most expensive part of the iteration. It was shown in [21] that the $(1,1)$, $(1,2)$ and $(2,1)$ blocks are often be sparse (indeed, they are diagonal under Assumption 2), and in the best case, the Hessian matrix can be formed and factored in near-linear time and memory. However, observe that the matrix $\mathbf{N}^T\mathbf{D}\mathbf{N}$ in the $(2,2)$ block shares its block-sparsity pattern with $\mathbf{N}^T\mathbf{N}$, which may be fully-dense according to Proposition 5. In the worst case, the cost of forming and factoring the Hessian matrix is $O(\omega^6 n^3)$ time and $O(\omega^4 n^2)$ memory.

Once the Hessian matrix (8) is factored, the normal equation is solved with a right-hand side, the solution is used to compute a Newton search direction, and the method takes a step along this direction, with the step-size determined using a line search. Although we will not go into the details, we note that all of these operations have the same cost as setting up the scaling matrix $\mathbf{D}$. After taking the step, the method either terminates if the desired accuracy has been achieved, or proceeds with a new iteration. We arrive at a per-iteration cost of $O(\omega^6 n^3)$ time and $O(\omega^4 n^2)$ memory.

B. The dualized iteration

Instead, applying the dualizing technique of [35] reformulates (6) into

$$\begin{aligned} \text{maximize} \quad & -c^T y \\ \text{subject to} \quad & \begin{bmatrix}\mathbf{A}^T \\ \mathbf{N}^T\end{bmatrix} y + s_1 = \begin{bmatrix} b \\ 0 \end{bmatrix} \\ & -y + s_2 = 0 \\ & s_1 \in \{0\}^f,\ s_2 \in \mathcal{K} \end{aligned} \quad (9)$$

with $f \le \frac{1}{2}\omega(\omega+1)n$ equality constraints, if we take (CTC) to be primal-dual strictly feasible. One way of enforcing these equality constraints is to embed them into a size-$2f$ linear cone, i.e. by splitting the equality constraint $s_1 = 0$ into linear inequalities $s_1 \ge 0$ and $s_1 \le 0$. The resulting cone has scaling matrix $\mathbf{D} = \mathrm{diag}(\mathbf{D}^l, \mathbf{D}^s)$, where $\mathbf{D}^s$ is the scaling matrix for $\mathcal{K}$ from above, and $\mathbf{D}^l = \mathrm{diag}(\mathbf{D}_1^l, \mathbf{D}_2^l, \mathbf{D}_3^l, \mathbf{D}_4^l)$ is the *diagonal* scaling matrix for the linear cone $\mathbb{R}_+^{2f}$. The normal equation has Hessian matrix

$$\mathbf{D}^s + \mathbf{N}(\mathbf{D}_1^l + \mathbf{D}_3^l)\mathbf{N}^T + \mathbf{A}(\mathbf{D}_3^l + \mathbf{D}_4^l)\mathbf{A}^T, \quad (10)$$

Under Assumption 2, $\mathbf{A}\mathbf{D}^\ell\mathbf{A}^T$ is block-diagonal, so (10) has the same block sparsity pattern as $\mathbf{N}\mathbf{N}^T$. After a minimum degree block-ordering, the cost of forming and factoring (10) is guaranteed to be linear $O(\omega^6 n)$ time and $O(\omega^4 n)$ memory, by virtue of Proposition 4. Combining this with the $O(\omega^3\ell + f)$ time and $O(\omega^2\ell + f)$ memory needed to form $\mathbf{D}$, and to compute and perform line searches along the Newton search direction, we arrive at a per-iteration cost of $O(\omega^6 n)$ time and $O(\omega^4 n)$ memory.

Alternatively, we may also embed the f equality constraints in (9) into a size-$(f+1)$ second-order cone, i.e. by replacing $s_1 = 0$ with the conic constraint $\|s_1\| \le s_0$ and the linear constraint $0 \cdot y + s_0 = 0$. The resulting cone has scaling matrix $\mathbf{D} = \mathrm{diag}(\mathbf{D}^q, \mathbf{D}^s)$, where $\mathbf{D}^s$ is the same scaling matrix for $\mathcal{K}$, and $\mathbf{D}^q = rr^T - \mu J$ is the *diagonal-plus-rank-1* scaling matrix for the quadratic cone, with $r = [r_0; r_1; r_2] \in \mathbb{R}^{f+1}$, $\mu > 0$, and $J = \mathrm{diag}(1, -I_f)$. The resulting normal equation has Hessian matrix

$$\mathbf{D}^s + \mu\mathbf{N}(I + r_1 r_1^T)\mathbf{N}^T + \mu\mathbf{A}(I + r_2 r_2^T)\mathbf{A}^T. \quad (11)$$

This matrix is a dense rank-2 update of a sparse matrix with the same sparsity pattern as (10); the standard solution technique is to compute the Cholesky factor of the sparse portion, and then to make a rank-2 update to the Cholesky factor in product form [36]. Assuming that $\mathbf{A}\mathbf{A}^T$ is block-diagonal as above, the cost of forming and factoring the sparse portion, and the making a rank-2 update is $O(\omega^6 n)$ time and $O(\omega^4 n)$ memory, again by virtue of Proposition 4. Combining this with the $O(\omega^3\ell + f)$ time and $O(\omega^2\ell + f)$ memory needed to form $\mathbf{D}$, and to compute and perform line searches along the Newton search direction, we again arrive at a per-iteration cost of $O(\omega^6 n)$ time and $O(\omega^4 n)$ memory.

V. Numerical examples

Using the techniques described above, we solve sparse SDPs posed on the 40 power system test cases in the MATPOWER suite [37]. The largest two cases have $n = 9241$ and $n = 13659$; the associated SDPs are so large that simply forming the $n \times n$ matrix decision variable X would deplete all memory. Fortunately, power systems are graphs with bounded treewidths [4], and their corresponding sparsity patterns are guaranteed to have sparse chordal completions [12].

In every problem listed below, we formulate a semidefinite program of the exact form (SDP). Before we apply the steps in Section II to reformulate (SDP) into (CTC), we first reorder the rows and columns of the data matrices $C, A_1, \ldots, A_m$ using a minimum degree ordering, in order to reduce the value of the clique number ω. As we mentioned before, finding the optimal fill-reducing ordering is NP-hard, but finding a "good-enough" ordering is easy. For these 40

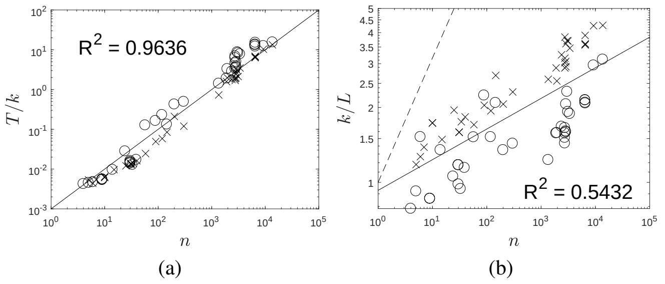

Figure 1: MAX 3-CUT ($\circ$) and Lovasz Theta ($\times$) problems: (a) Time per iteration, with regression $T/k = 10^{-3}n$; (b) Iterations per decimal digits of accuracy, with (solid) regression $k/L = 0.929n^{0.123}$ and (dashed) bound $k/L = \sqrt{n}$.

test cases, the minimum degree ordering was able to reduce $\omega \leq 35$, which is low enough to be considered $O(1)$.

We each dualized version of (CTC) using the standard version of SeDuMi v1.32 [38], on a Xeon 3.3 GHz quad-core CPU with 16 GB of RAM. The solver is based on a wide-region neighborhood, and has a guarantee iteration bound of $k = O(\sqrt{\theta}\log\epsilon^{-1})$ for an order-θ convex cone [14].

A. MAXCUT and Lovasz Theta

We begin by considering the MAXCUT and Lovasz Theta problems, which satisfy Assumption 2 by default, and hence have solution complexities of $O(n^{1.5})$ time and $O(n)$ memory. For each of the 40 test cases, we use the MATPOWER function `makeYbus` to generate the bus admittance matrix $Y_{bus} = [Y_{i,j}]_{i,j=1}^{n}$, and symmetricize to yield $Y_{abs} = \frac{1}{2}[|Y_{i,j}| + |Y_{j,i}|]_{i,j=1}^{n}$. We view this matrix as the weighted adjacency matrix for the system graph. For MAX k-CUT, we define the weighted Laplacian matrix $C = \mathrm{diag}(Y_{abs}\mathbf{1}) - Y_{abs}$, and set up problem (MkC) with $k = 3$. For Lovasz Theta, we extract the location of the graph edges from Y_{abs} and set up (LT$'$).

We then apply the techniques decribed in the paper to solve the 80 SDPs. At each k-th iteration, we compute the associated number of accurate decimal digits $L = \log_{10}(1/\max\{\mathrm{pinf}, \mathrm{dinf}, \mathrm{gap}\})$ using the DIMACS feasibility and duality gap metrics $\mathrm{pinf} = \|\mathcal{A}(X) - b\|_2/(1 + \|b\|_2)$, $\mathrm{dinf} = \lambda_{\max}(\mathcal{A}^T(y) - C)/(1 + \|C\|_2)$, and $\mathrm{gap} = (C \bullet X - b^T y)/(1 + |C \bullet X| + |b^T y|)$. Of the 80 SDPs considered, 79 solved to $L \geq 5$ digits in $k \leq 23$ iterations and $T \leq 306$ seconds; the largest instance solved to $L = 4.48$. After chordal decomposition (Section II-B), Figure 1b plots T/k, the mean time taken per-iteration. Unsurprisingly, the per-iteration time is linear with respect to n. A log-log regression yields $T/k = 10^{-3}n$, with $R^2 = 0.9636$. Figure 1b plots k/L, the number of iterations to a factor-of-ten error reduction. We see that SeDuMi's guaranteed iteration complexity $k = O(\sqrt{n}\log\epsilon^{-1}) = O(\sqrt{n}L)$ is a significant over-estimate; a log-log regression yields $k/L = 0.929n^{0.123} \approx n^{1/8}$, with $R^2 = 0.5432$. Combined, the data suggests an actual time complexity of $T \approx 10^{-3}n^{1.1}L$.

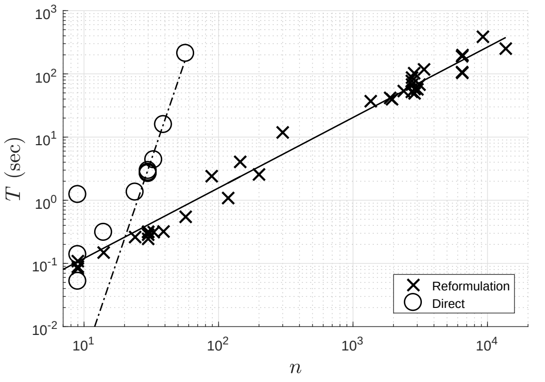

Figure 2: Comparison of solving OPF reformulated into (CTC) using SeDuMi vs a direct solution of (SDP) via SeDuMi. Regression lines show $O(n^{1.12})$ and $O(n^{6.27})$ respectively.

Table I: The five largest Optimal Power Flow SDP problems.

Problem Description				Time (sec)		
#	n	m	ω	Prep	Sol	Total
35	6468	9000	30	6.62	99.85	106.47
36	6470	9005	30	6.62	183.17	189.78
37	6495	9019	31	6.78	97.02	103.80
38	6515	9037	31	6.66	191.60	198.26
39	9241	16049	35	10.56	375.76	386.32
40	13659	20467	35	17.48	233.32	250.81

B. Optimal power flow

We now solve instances of the OPF posed on the same 40 power systems as mentioned above. Here, we use the MATPOWER function `makeYbus` to generate the bus admittance matrix Y_{bus}, and then manually generate each constraint matrix A_i from Y_{bus} using the recipes described in [26]. For each example, we formulate the OPF problem to minimize the cost of generation, which is set to equal the real-power injection of each generator a fixed price per MW, which is set to \$1 without loss of generality. Every system bus is limited to a voltage deviation of 0.05 per-unit, meaning that we constrain each voltage magnitude to be between 0.95 and 1.05 per-unit. Every generator is constrained by its power curve, comprising minimum and maximum real and reactive power limits. We fixed the load profile according to the original power flow solution.

Figure 2 compares the running times between our embedding–dualization approach and a direct solution with SeDuMi, for a relatively coarse solution with $L \approx 3$. Table I gives the associated details for the 5 largest examples: here n is the number of buses (i.e. vertices), m is the number of branches (i.e. edges), ω is the clique number, "Prep" is the time to convert (SDP) into (CTC), "Sol" is the time used to solve the dualized version, and "Total" is the total time. Our approach manages to solve every problem within seven minutes, with an empirical complexity of $T \approx 8 \cdot 10^{-3} \cdot n^{1.12}$. In comparison, the original problem posed directly over the size-n semidefinite cone becomes too time-consuming to solve past $n \approx 100$. It is worth emphasizing that both figures were obtained by applying the same solver to the same problem formulated in two different ways.

Despite failing to satisfying Assumption 2, the empirical time complexity of our method is nevertheless near-linear. This result suggests that Assumption 2 is too strong, and can probably be weakened without significantly affecting complexity figure quoted in Theorem 2.

VI. Conclusion

Chordal decomposition and clique tree conversion split a large semidefinite variable $X \succeq 0$ into many smaller semidefinite variables $X_j \succeq 0$. These smaller variables are coupled by "overlapping constraints", whose block-sparsity pattern coincides with the incidence matrix of a tree. In this paper, we describe a reformulation whose normal equations have a block-sparsity pattern that matches the adjacency matrix of the same tree (under Assumption 2). Such matrices factor with zero block-fill, so any interior-point method solves the reformulation with a per-iteration complexity that is linear in time and memory. We use this insight to solve the MAX 3-CUT relaxation, the Lovasz Theta problem, and the optimal power flow SDP relaxation on power system test cases—the largest of which has $n = 13659$—in 7 minutes or less using an off-the-shelf solver on a modest computer.

References

[1] M. X. Goemans and D. P. Williamson, "Improved approximation algorithms for maximum cut and satisfiability problems using semidefinite programming," *J. ACM*, vol. 42, no. 6, pp. 1115–1145, 1995.

[2] A. Frieze and M. Jerrum, "Improved approximation algorithms for maxk-cut and max bisection," *Algorithmica*, vol. 18, no. 1, pp. 67–81, 1997.

[3] D. K. Molzahn, J. T. Holzer, B. C. Lesieutre, and C. L. DeMarco, "Implementation of a large-scale optimal power flow solver based on semidefinite programming," *IEEE Transactions on Power Systems*, vol. 28, no. 4, pp. 3987–3998, 2013.

[4] R. Madani, M. Ashraphijuo, and J. Lavaei, "Promises of conic relaxation for contingency-constrained optimal power flow problem," *IEEE Trans. Power Syst.*, vol. 31, no. 2, pp. 1297–1307, 2016.

[5] R. Madani, S. Sojoudi, G. Fazelnia, and J. Lavaei, "Finding low-rank solutions of sparse linear matrix inequalities using convex optimization," *SIAM Journal on Optimization*, vol. 27, no. 2, pp. 725–758, 2017.

[6] R. G. Cowell, P. Dawid, S. L. Lauritzen, and D. J. Spiegelhalter, *Probabilistic networks and expert systems: Exact computational methods for Bayesian networks*. Springer, 1999.

[7] M. Fukuda, M. Kojima, K. Murota, and K. Nakata, "Exploiting sparsity in semidefinite programming via matrix completion I: General framework," *SIAM J. Optim.*, vol. 11, no. 3, pp. 647–674, 2001.

[8] K. Nakata, K. Fujisawa, M. Fukuda, M. Kojima, and K. Murota, "Exploiting sparsity in semidefinite programming via matrix completion II: Implementation and numerical results," *Math. Program.*, vol. 95, no. 2, pp. 303–327, 2003.

[9] M. S. Andersen, J. Dahl, and L. Vandenberghe, "Logarithmic barriers for sparse matrix cones," *Optimization Methods and Software*, vol. 28, no. 3, pp. 396–423, 2013.

[10] S. Arnborg, D. G. Corneil, and A. Proskurowski, "Complexity of finding embeddings in ak-tree," *SIAM Journal on Algebraic Discrete Methods*, vol. 8, no. 2, pp. 277–284, 1987.

[11] J. R. Gilbert, "Some nested dissection order is nearly optimal," *Information Processing Letters*, vol. 26, no. 6, pp. 325–328, 1988.

[12] H. L. Bodlaender, J. R. Gilbert, H. Hafsteinsson, and T. Kloks, "Approximating treewidth, pathwidth, frontsize, and shortest elimination tree," *Journal of Algorithms*, vol. 18, no. 2, pp. 238–255, 1995.

[13] A. Agrawal, P. Klein, and R. Ravi, "Cutting down on fill using nested dissection: Provably good elimination orderings," in *Graph Theory and Sparse Matrix Computation*. Springer, 1993, pp. 31–55.

[14] J. F. Sturm, "Implementation of interior point methods for mixed semidefinite and second order cone optimization problems," *Optim. Method. Softw.*, vol. 17, no. 6, pp. 1105–1154, 2002.

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

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

[17] Y. Nesterov and A. Nemirovskii, *Interior-point polynomial algorithms in convex programming*. SIAM, 1994.

[18] S. Kim, M. Kojima, M. Mevissen, and M. Yamashita, "Exploiting sparsity in linear and nonlinear matrix inequalities via positive semidefinite matrix completion," *Math. Program.*, vol. 129, no. 1, pp. 33–68, 2011.

[19] L. Vandenberghe, M. S. Andersen *et al.*, "Chordal graphs and semidefinite optimization," *Foundations and Trends in Optimization*, vol. 1, no. 4, pp. 241–433, 2015.

[20] M. S. Andersen, A. Hansson, and L. Vandenberghe, "Reduced-complexity semidefinite relaxations of optimal power flow problems," *IEEE Transactions on Power Systems*, vol. 29, no. 4, pp. 1855–1863, 2014.

[21] Y. Sun, M. S. Andersen, and L. Vandenberghe, "Decomposition in conic optimization with partially separable structure," *SIAM J. Optim.*, vol. 24, no. 2, pp. 873–897, 2014.

[22] Z. Lu, A. Nemirovski, and R. D. Monteiro, "Large-scale semidefinite programming via a saddle point mirror-prox algorithm," *Mathematical programming*, vol. 109, no. 2-3, pp. 211–237, 2007.

[23] R. Madani, A. Kalbat, and J. Lavaei, "Admm for sparse semidefinite programming with applications to optimal power flow problem," in *Decision and Control (CDC), 2015 IEEE 54th Annual Conference on*. IEEE, 2015, pp. 5932–5939.

[24] A. Kalbat and J. Lavaei, "A fast distributed algorithm for decomposable semidefinite programs," in *Decision and Control (CDC), 2015 IEEE 54th Annual Conference on*. IEEE, 2015, pp. 1742–1749.

[25] L. Lovász, "On the shannon capacity of a graph," *IEEE Trans. Inf. Theory*, vol. 25, no. 1, pp. 1–7, 1979.

[26] J. Lavaei and S. H. Low, "Zero duality gap in optimal power flow problem," *IEEE Transactions on Power Systems*, vol. 27, no. 1, pp. 92–107, 2012.

[27] R. Madani, S. Sojoudi, and J. Lavaei, "Convex relaxation for optimal power flow problem: Mesh networks," *IEEE Transactions on Power Systems*, vol. 30, no. 1, pp. 199–211, 2015.

[28] M. S. Andersen, J. Dahl, and L. Vandenberghe, "Implementation of nonsymmetric interior-point methods for linear optimization over sparse matrix cones," *Mathematical Programming Computation*, vol. 2, no. 3, pp. 167–201, 2010.

[29] ——, "Cvxopt: A python package for convex optimization, version 1.1. 6," *Available at cvxopt. org*, vol. 54, 2013.

[30] C. Helmberg and F. Rendl, "A spectral bundle method for semidefinite programming," *SIAM J. Optim.*, vol. 10, no. 3, pp. 673–696, 2000.

[31] S. Burer and R. D. Monteiro, "A nonlinear programming algorithm for solving semidefinite programs via low-rank factorization," *Math. Program.*, vol. 95, no. 2, pp. 329–357, 2003.

[32] D. Fulkerson and O. Gross, "Incidence matrices and interval graphs," *Pacific journal of mathematics*, vol. 15, no. 3, pp. 835–855, 1965.

[33] R. Grone, C. R. Johnson, E. M. Sá, and H. Wolkowicz, "Positive definite completions of partial hermitian matrices," *Linear Algebra Appl.*, vol. 58, pp. 109–124, 1984.

[34] J. W. Liu, "The role of elimination trees in sparse factorization," *SIAM J. Matrix Anal. Appl.*, vol. 11, no. 1, pp. 134–172, 1990.

[35] J. Löfberg, "Dualize it: software for automatic primal and dual conversions of conic programs," *Optimization Methods & Software*, vol. 24, no. 3, pp. 313–325, 2009.

[36] D. Goldfarb and K. Scheinberg, "Product-form cholesky factorization in interior point methods for second-order cone programming," *Math. Program.*, vol. 103, no. 1, pp. 153–179, 2005.

[37] R. D. Zimmerman, C. E. Murillo-Sánchez, and R. J. Thomas, "Matpower: Steady-state operations, planning, and analysis tools for power systems research and education," *IEEE Trans. Power Syst.*, vol. 26, no. 1, pp. 12–19, 2011.

[38] J. F. Sturm, "Using sedumi 1.02, a matlab toolbox for optimization over symmetric cones," *Optim. Method. Softw.*, vol. 11, no. 1-4, pp. 625–653, 1999.