Mathematical Programming
https://doi.org/10.1007/s10107-024-02137-5

FULL LENGTH PAPER

Series A

Complexity of chordal conversion for sparse semidefinite programs with small treewidth

Richard Y. Zhang[1]

Received: 12 June 2023 / Accepted: 10 August 2024

Abstract

If a sparse semidefinite program (SDP), specified over $n \times n$ matrices and subject to m linear constraints, has an aggregate sparsity graph G with small treewidth, then chordal conversion will sometimes allow an interior-point method to solve the SDP in just $O(m+n)$ time per-iteration, which is a significant speedup over the $\Omega(n^3)$ time per-iteration for a direct application of the interior-point method. Unfortunately, the speedup is not guaranteed by an $O(1)$ treewidth in G that is independent of m and n, as a diagonal SDP would have treewidth zero but can still necessitate up to $\Omega(n^3)$ time per-iteration. Instead, we construct an extended aggregate sparsity graph $\overline{G} \supseteq G$ by forcing each constraint matrix A_i to be its own clique in G. We prove that a small treewidth in $\overline{G}$ does indeed guarantee that chordal conversion will solve the SDP in $O(m+n)$ time per-iteration, to ϵ-accuracy in at most $O(\sqrt{m+n}\log(1/\epsilon))$ iterations. This sufficient condition covers many successful applications of chordal conversion, including the MAX-k-CUT relaxation, the Lovász theta problem, sensor network localization, polynomial optimization, and the AC optimal power flow relaxation, thus allowing theory to match practical experience.

1 Introduction

Consider directly applying a general-purpose interior-point method solver, like SeDuMi [37], SDPT3 [42], and MOSEK [32], to solve the standard-form semidefinite program to high accuracy:

$$\min_{X \in \mathbb{S}^n} \langle C, X \rangle \quad \text{s.t. } X \succeq 0, \quad \langle A_i, X \rangle \leq b_i \text{ for all } i \in \{1, 2, \ldots, m\}. \tag{SDP}$$

Financial support for this work was provided in part by the NSF CAREER Award ECCS-2,047,462 and in part by C3.ai Inc. and the Microsoft Corporation via the C3.ai Digital Transformation Institute.

✉ Richard Y. Zhang
ryz@illinois.edu

[1] Dept. of Electrical and Computer Engineering, University of Illinois at Urbana-Champaign, 306 N Wright St, Urbana, IL 61801, USA

Published online: 17 September 2024

Here, $\mathbb{S}^n$ denotes the set of $n \times n$ real symmetric matrices with inner product $\langle A_i, X\rangle = \text{tr}(A_i X)$, and $X \succeq 0$ means that X is positive semidefinite. At each iteration, the $n \times n$ matrix variable X is generally fully dense, even when the problem data $C, A_1, \ldots, A_m \in \mathbb{S}^n$ and $b_1, \ldots, b_m \in \mathbb{R}$ are sparse. The per-iteration cost of the solver is usually at least $\Omega(n^3)$ time, which in practice limits the value of n to no more than a few thousand.

Instead, to handle n as large as a hundred thousand, researchers have found empirical success by first performing a simple preprocessing step called *chordal conversion* (CC), which was first introduced in by Fukuda et al. [16]. Suppose that every $S = C - \sum_i y_i A_i \succ 0$ factors as $S = LL^T$ into a lower-triangular Cholesky factor L that is sparse. It turns out, by defining $J_j = \{i : L[i, j] \neq 0\}$ as the possible locations of nonzeros in the j-th column of L, that (SDP) is exactly equivalent,[1] to the following

$$\min_{X \in \mathbb{S}^n} \quad \langle C, X\rangle \quad \text{s.t.} \quad \begin{array}{l} \langle A_i, X\rangle \leq b_i \quad \text{for all } i \in \{1, 2, \ldots, m\}, \\ X[J_j, J_j] \succeq 0 \quad \text{for all } j \in \{1, 2, \ldots, n\}. \end{array} \tag{CC}$$

While not immediately obvious, (CC) is actually an optimization over a *sparse* matrix variable X, because the matrix elements that are not indexed by a constraint $\langle A_i, X\rangle$ or $X[J_j, J_j]$ can be set to zero without affecting the optimization. Hence, the point of the reformulation is to reduce the number of optimization variables, from $\frac{1}{2}n(n+1)$ in (SDP) to at most $\omega \cdot n$ in (CC), where $\omega = \max_j |J_j|$ is defined as the maximum number of nonzeros per column of the Cholesky factor L.

Clearly, chordal conversion needs $\omega \ll n$ in order to be efficient. Where this condition holds, the interior-point method solver is consistently able to solve (CC) in just $O(m+n)$ time per-iteration, thereby solving some of the largest instances of (SDP) ever considered. Unfortunately, a small ω does not actually guarantee the empirically observed $O(m+n)$ time figure. Consider the following counterexample, which has $\omega = 1$, but would nevertheless incur a cost of at least $\Omega(n^3)$ time per-iteration.

Example 1.1 *(Diagonal SDP)* Given vectors $c, a_1, \ldots, a_m \in \mathbb{R}^n$, consider the following instance of (SDP):

$$\min_{X \in \mathbb{S}^n} \ \langle \text{diag}(c), X\rangle \quad \text{s.t. } X \succeq 0, \quad \langle \text{diag}(a_i), X\rangle \leq b_i \text{ for all } i \in \{1, 2, \ldots, m\}.$$

One can verify that $J_j = \{i : L[i, j] \neq 0\} = \{j\}$, so the corresponding (CC) is a linear program over n variables and m constraints:

$$\min_{x \in \mathbb{R}^n} \ \langle c, x\rangle \quad \text{s.t. } x \geq 0, \quad \langle a_i, x\rangle \leq b_i \text{ for all } i \in \{1, 2, \ldots, m\}.$$

Despite $\omega = \max_j |J_j| = 1$, if the number of linear constraints is at least $m = \Omega(n)$, then it would take *any* method at least $\Omega(n^3)$ time to take a single iteration. □

While a small $\omega \ll n$ is clearly necessary for chordal conversion to be fast, Example 1.1 shows that it is not sufficient. In this paper, we fill this gap, by providing

[1] The equivalence is due to Grone, Johnson, Sá and Wolkowicz [20, Theorem 7]; see also [16, Theorem 2.3] and [44, Theorem 10.1].

a sufficient condition for a general-purpose interior-point method to solve (CC) in $O(m+n)$ time per-iteration. The sufficient condition covers many successful applications of chordal conversion, including the MAX-k-CUT relaxation [16, 24, 33], the Lovász theta problem [16, 24, 33], sensor network localization [23, 24], polynomial optimization [26, 35, 45], and the AC optimal power flow relaxation in electric grid optimization [13, 22, 29, 31], thus allowing theory to match practical experience.

1.1 Our results: complexity of chordal conversion

In order for chordal conversion to be fast, a well-known *necessary* condition is for the underlying aggregate sparsity graph, which is defined $G = (V, E)$ with

$$V = \{1, 2, \ldots, n\}, \quad E = \mathrm{spar}(C) \cup \mathrm{spar}(A_1) \cup \cdots \cup \mathrm{spar}(A_m), \qquad (1)$$
$$\text{where } \mathrm{spar}(C) \equiv \{(i, j) : C[i, j] \neq 0 \text{ for } i > j\},$$

to be "tree-like" with a small *treewidth* $\mathrm{tw}(G) \ll n$. We defer a formal definition of the treewidth to Definition 2.4, and only note that the value of ω is lower-bounded as $\omega \geq 1 + \mathrm{tw}(G)$. In other words, while ω can be decreased by symmetrically reordering the columns and rows of the data matrices, as in $C \leftarrow \Pi C \Pi^T$ and $A_i \leftarrow \Pi A_i \Pi^T$ for some permutation matrix Π, actually achieving $\omega \ll n$ is possible only if $\mathrm{tw}(G) \ll n$. However, as illustrated by Example 1.1, even on a graph with $\mathrm{tw}(G) = 0$, and even when using the optimal $\omega = 1$, chordal conversion may still not be fast.

In this paper, we show that a *sufficient* condition for chordal conversion to be fast is for a supergraph $\overline{G} \supseteq G$, that additionally captures the correlation between constraints, to also[2] be "tree-like" with a small treewidth $\mathrm{tw}(\overline{G}) \ll n$. Concretely, we construct the *extended* aggregate sparsity graph $\overline{G} = (V, \overline{E})$ by forcing each constraint matrix A_i to be its own clique in G:

$$V = \{1, 2, \ldots, n\}, \quad \overline{E} = \mathrm{spar}(C) \cup \mathrm{clique}(A_1) \cup \cdots \cup \mathrm{clique}(A_m) \qquad (2)$$
$$\text{where } \mathrm{clique}(A) = \{(i, j) : A[i, k] \neq 0 \text{ or } A[k, j] \neq 0 \text{ for some } k\}.$$

This is the union between G and the constraint intersection graph [17] (or the dual graph [12] or the correlative sparsity [25, 45]) for the rank-1 instance of (SDP). In other words, we add a new edge (i, j) to E whenever x_i and x_j appear together in a common constraint $x^T A_k x \leq b_k$ for some k. The fact that this contributes $\mathrm{clique}(A_k) \subseteq \overline{E}$ reflects the reality that each $x^T A_k x \leq b_k$ densely couples all affected elements of x together, forcing them to be optimized simultaneously. In contrast, the cost $x^T C x$ can be optimized sequentially over the elements of x, which is why $\mathrm{clique}(C)$ is absent.

Our main result Theorem 3.3 says that if the extended graph $\overline{G}$ has small treewidth $\mathrm{tw}(\overline{G}) = O(1)$ with respect to m and n, then one can find a fill-reducing permutation Π such that, after reordering the data as $C \leftarrow \Pi C \Pi^T$ and $A_i \leftarrow \Pi A_i \Pi^T$, the resulting instance of (CC) is solved by a general-purpose interior-point method in guaranteed $O(m+n)$ time per-iteration, over at most $O(\sqrt{m+n}\log(1/\epsilon))$ iterations.

[2] Note that $\mathrm{tw}(\overline{G}) \geq \mathrm{tw}(G)$ holds by virtue of $\overline{G} \supseteq G$.

In practice, a "good enough" permutation Π is readily found by applying an efficient fill-reducing heuristic to $\overline{G}$, and a primal-dual interior-point method often converges to ϵ accuracy in dimension-free $O(\log(1/\epsilon))$ iterations (without the square-root factor). If we take these two empirical observations[3] as formal assumptions, then a small treewidth $\text{tw}(\overline{G}) = O(1)$ in the extended graph $\overline{G}$ is indeed sufficient for chordal conversion to solve the instance of (SDP) in $O((m+n)\log(1/\epsilon))$ empirical time.

In the case that G and $\overline{G}$ coincide, our analysis becomes exact; a small treewidth in $\overline{G}$ is both necessary and sufficient for chordal conversion to achieve $O((m+n)\log(1/\epsilon))$ empirical time. This is the case for the MAX-k-CUT relaxation [15, 19] and the Lovász theta problem [28], two classic SDPs that constitute a majority of test problems in the SDPLIB [7] and the DIMACS [36] datasets. Here, $G = \overline{G}$ because each constraint matrix A_i indexes just a single matrix element, as in $\langle A_i, X\rangle = \alpha_i \cdot X[j_i, k_i]$. Below, we write e_j as the j-th column of the identity matrix, and $\mathbf{1} = [1, 1, \ldots, 1]^T$.

Example 1.2 *(MAX-k-CUT)* Let C be the (weighted) Laplacian matrix for a graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ with $\mathcal{V} = \{1, 2, \ldots, d\}$. Frieze and Jerrum [15] proposed a randomized algorithm to solve MAX k-CUT with an approximation ratio of $1 - 1/k$ based on solving

$$\max_{X \succeq 0} \quad \frac{k-1}{2k} \langle C, X\rangle \quad \text{s.t.} \quad \begin{array}{c} X[i,i] = 1 \quad \text{for all } i \in \mathcal{V} \\ X[i,j] \geq \frac{-1}{k-1} \quad \text{for all } (i,j) \in \mathcal{E} \end{array}$$

The classic Goemans–Williamson 0.878 algorithm [19] for MAXCUT is recovered by setting $k = 2$ and removing the redundant constraint $X[i, j] \geq -1$. □

Example 1.3 (Lovász theta) The Lovász number $\vartheta(\mathcal{G})$ of a graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ with $\mathcal{V} = \{1, 2, \ldots, d\}$ is the optimal value to the following [28]

$$\min_{\lambda, y_{i,j} \in \mathbb{R}} \quad \lambda \quad \text{s.t.} \quad \mathbf{1}\mathbf{1}^T - \sum_{(i,j)\in\mathcal{E}} y_{i,j}(e_i e_j^T + e_j e_i^T) \preceq \lambda I.$$

Given that $\vartheta(\mathcal{G}) \geq 1$ holds for all graphs G, dividing through by λ and applying the Schur complement lemma yields a sparse reformulation

$$\min_{X \succeq 0} \quad \left\langle \begin{bmatrix} I & \mathbf{1} \\ \mathbf{1}^T & 0 \end{bmatrix}, X \right\rangle \quad \text{s.t.} \quad \begin{array}{c} X[d+1, d+1] = 1, \\ X[i,j] = 0 \quad \text{for all } (i,j) \in \mathcal{E}. \end{array}$$

□

We also have $G = \overline{G}$ in the sensor network localization problem, one of the first successful applications of chordal conversion to a real-world problem [23], because $\text{spar}(A_i) = \text{clique}(A_i)$ holds for all i. (Assume without loss of generality that each a_k below contains only nonzero elements.)

Example 1.4 (Sensor network localization) We seek to find unknown sensor points $x_1, \ldots, x_n \in \mathbb{R}^d$ such that

$$\|x_i - x_j\| = r_{i,j} \quad \text{for all } (i,j) \in N_x, \qquad \|x_i - a_k\| = \rho_{i,k} \quad \text{for all } (i,k) \in N_a$$

[3] In Sect. 5, we provide detailed numerical experiments to validate these empirical observations on real-world datasets.

given a subset N_x of distances $r_{i,j}$ between the i-th and j-th sensors, and a subset N_a of distances $\rho_{i,k}$ between the i-th sensor and the k-th known anchor point $a_k \in \mathbb{R}^d$. Biswas and Ye [6] proposed the following SDP relaxation

$$\min_{X \succeq 0} \quad \langle 0, X \rangle \quad \text{s.t.} \quad \begin{array}{r} \left\langle \begin{bmatrix} 1 & -1 \\ -1 & 1 \end{bmatrix}, \begin{bmatrix} X[i,i] & X[i,j] \\ X[i,j] & X[j,j] \end{bmatrix} \right\rangle = r_{i,j}^2 \quad \text{for all } (i,j) \in N_x, \\ \left\langle \begin{bmatrix} 1 & -a_k^T \\ -a_k & a_k a_k^T \end{bmatrix}, \begin{bmatrix} X[i,i] & X[i,K] \\ X[K,i] & X[K,K] \end{bmatrix} \right\rangle = \rho_{i,k}^2 \quad \text{for all } (i,k) \in N_a, \\ X[K,K] = I_d, \end{array}$$

where $K = (n+1, \ldots, n+d)$. □

Our result can also be applied to the chordal conversion of SDPs that arise in polynomial optimization. The following class of polynomial optimization covers many of the unconstrained test problems in the original paper [45] that first introduced chordal conversion to this setting. Below, a matrix $U \in \mathbb{R}^{p \times p}$ is said to be Hankel if its skew-diagonals are constant, i.e. $U[i,j] = U[i+1, j-1]$ for all $1 \le i, j \le p$. We see that $G = \overline{G}$ holds because the Hankel constraint is dense over its support.

Example 1.5 (Unconstrained polynomial optimization) Given $C_{i,j} \in \mathbb{R}^{p \times p}$ for $i, j \in \{1, 2, \ldots, n\}$, consider the following

$$\min_{x_1, \ldots, x_n} \sum_{j=1}^{n} u_i^T C_{i,j} u_j \text{ where } u_j = [1, x_j, x_j^2, \ldots, x_j^{p-1}]^T.$$

The basic Lasserre–Parrilo SDP relaxation [26, 35] for this problem (without cross terms) reads

$$\min_{[U_{i,j}]_{i,j=1}^n \succeq 0} \quad \sum_{j=1}^{n} \langle C_{i,j}, U_{i,j} \rangle \quad \text{s.t.} \quad U_{i,i} \text{ is Hankel}, \quad U_{i,i}[1,1] = 1 \quad \text{for all } i,$$

where $[U_{i,j}]_{i,j=1}^n$ denotes an $np \times np$ matrix, comprising of $n \times n$ blocks of $p \times p$, with $U_{i,j} \in \mathbb{R}^{p \times p}$ as its (i,j)-th block. □

But the real strength of our result is its ability to handle cases for which $G \subset \overline{G}$ holds strictly. An important real-world example is the SDP relaxation of the AC optimal power flow problem [4, 27], which plays a crucial role in the planning and operations of electricity grids.

Example 1.6 (AC optimal power flow relaxation) Given a graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ on d vertices $\mathcal{V} = \{1, \ldots, d\}$, we say that $A_i \in \mathbb{S}^{2d}$ implements a power flow constraint at vertex $k \in \mathcal{V}$ if it can be written in terms of $\alpha_{i,k} \in \mathbb{S}^2$ and $\alpha_{i,j} \in \mathbb{R}^{2 \times 2}$ as:

$$A_i = e_k e_k^T \otimes \alpha_{i,k} + \frac{1}{2} \sum_{(j,k) \in E} \left[e_j e_k^T \otimes \alpha_{i,j} + e_k e_j^T \otimes \alpha_{i,j}^T \right].$$

An instance of the AC optimal power flow relaxation is written

$$\min_{X \succeq 0} \quad \sum_j \langle C_j, X \rangle \quad \text{s.t.} \quad b_i^{\text{lb}} \le \langle A_i, X \rangle \le b_i^{\text{ub}}$$

in which every A_i and C_j implements a power flow constraint at some vertex $v \in \mathcal{V}$. □

It can be verified that $\text{tw}(G) = 2 \cdot \text{tw}(\mathcal{G})$ and $\text{tw}(\overline{G}) = 2 \cdot \text{tw}(\mathcal{G}^2)$, where the square graph $\mathcal{G}^2$ is defined so that $(i, j) \in \mathcal{G}^2$ if and only if i and j are at most a distance of 2 away in $\mathcal{G}$. In fact, knowing that an electric grid $\mathcal{G}$ is "tree-like" does not in itself guarantee chordal conversion to be fast, because it does not imply that $\mathcal{G}^2$ would also be "tree-like". While chordal conversion is already widely used to solve the AC optimal power flow relaxation [13, 22, 29, 31], our finding in Sect. 5 that $\text{tw}(\mathcal{G}^2) \ll n$ holds for real-world power systems (see Table 2) provides the first definitive explanation for its observed $O((m+n)\log(1/\epsilon))$ empirical time complexity.

It remains future work to understand the cases where $\text{tw}(G)$ and $\text{tw}(\overline{G})$ are very different. In the case of the AC optimal power flow relaxation, it is not difficult to construct a counterexample where $\text{tw}(G) = 2$ and $\text{tw}(\overline{G}) = n - 2$ (set $\mathcal{G}$ to be the star graph, so that $\mathcal{G}^2$ is the complete graph) and observe $\Omega(n^3)$ per-iteration cost after chordal conversion. This counterexample, along with the trivial Example 1.1, both hint at the possibility that a small treewidth in $\overline{G}$ is both necessary and sufficient for $O(m+n)$ time per-iteration, but more work is needed establish this rigorously.

1.2 Prior results: complexity of clique-tree conversion

Our result is related to a prior work of Zhang and Lavaei [46] that studied a different conversion method called *clique-tree conversion*, also due to Fukuda et al. [16]. This can best be understood as a second step of conversion added on top of chordal conversion. Recall that chordal conversion converts (SDP) into (CC), and then solves the latter using an interior-point method. Clique-tree conversion further converts (CC) into the following problem by splitting each submatrix $X_j \equiv X[J_j, J_j]$ into its own variable:

$$\min_{X_1,\ldots,X_n} \sum_{j=1}^{n} \langle C_j, X_j \rangle \text{ s.t. } \begin{array}{c} \sum_{j=1}^n \langle A_{i,j}, X_j \rangle \le b_i \quad \text{for all } i \in \{1,2,\ldots,m\}, \\ X_j \succeq 0 \quad \text{for all } j \in \{1,2,\ldots,n\}, \\ \mathcal{N}_{u,v}(X_v) = \mathcal{N}_{v,u}(X_u) \quad \text{for all } (u,v) \in \mathcal{T}. \end{array} \tag{CTC}$$

The constraint $\mathcal{N}_{u,v}(X_v) = \mathcal{N}_{v,u}(X_u)$ is added to enforce agreement between overlapping submatrices, over the edges of the eponymous clique tree $\mathcal{T}$.

The point of converting (CC) to (CTC) is to force a favorable sparsity pattern in the Schur complement equations solved at each iteration of the interior-point method, which is known as the *Schur complement sparsity* [44, Section 13.1] or the *correlative sparsity* [24, 25]. In fact, Zhang and Lavaei [46] pointed out that the Schur complement sparsity of (CTC) is particularly simple to analyze. Under small treewidth assumptions, they proved that an interior-point method solves (CTC) in guaranteed $O(m+n)$ time

per-iteration, over at most $O(\sqrt{m+n}\log(1/\epsilon))$ iterations; see also Gu and Song [21]. But a major weakness of this result is that it critically hinges on the second step of conversion, from (CC) to (CTC). On a basic level, it does not explain the plethora of numerical experiments showing that interior-point methods are able to solve (CC) directly in $O(m+n)$ time per-iteration *without* a second conversion step [22–24, 45].

Indeed, the numerical experiments of Kim et al. [24] strongly suggest, for instances of (CTC) with favorable Schur complement sparsity, that the Schur complement sparsity of (CC) had already been favorable in the first place, so the second conversion step was unnecessary, other than for the sake of a proof. Unfortunately, the Schur complement sparsity of (CC) is much more complicated than that of (CTC). Prior to this work, Kobayashi et al. [25] provided a characterization for when the Schur complement sparsity of (CC) is favorable. However, their characterization can only be checked numerically, in a similar amount of work as performing a single iteration of the interior-point method, and so gives no deeper insight on what classes of SDPs can be efficiently solved. Our main technical contribution in this paper is Theorem 3.1, which characterizes the complicated Schur complement sparsity in terms of the much-simpler extended sparsity $\overline{E}$. It is this simplicity that allowed us to analyze many successful applications of chordal conversion, as detailed in the previous section.

In practice, the second conversion step from (CC) to (CTC) results in a massive performance penalty, both in preprocessing time and in the solution time. In our large-scale experiments in Sect. 5, the second step of converting from (CC) into (CTC) can sometimes take more than 100 times longer than the first step of converting (SDP) into (CC). Also, we find that the state-of-the-art solver MOSEK [32] takes a factor of 2 to 100 times more time to solve (CTC) than the original instance of (CC). Previously, clique-tree conversion was used to solve an instance of AC optimal power flow relaxation with $n = 8.2 \times 10^4$ and $m \approx 2.5 \times 10^5$ on a high-performance computing (HPC) node with 24 cores and 240 GB memory in 8 h [13]. In this paper, we solve this same problem using chordal conversion on a modest workstation with 4 cores and 32 GB of RAM in just 4 h.

1.3 Other approaches

In this paper, we focus on chordal conversion in the context of high-accuracy interior-point methods. We mention that chordal conversion has also been used to reduce the per-iteration cost of first-order methods to $O(m+n)$ time [40, 47], but these can require many iterations to converge to high accuracy. Also, nonconvex approaches [8, 9] have recently become popular, but it remains unclear how these could be made to benefit from chordal conversion.

The recent preprint of Gu and Song [21] combined the fast interior-point method of Dong et al. [12, Theorem 1.3] and the clique-tree conversion formulation of Zhang and Lavaei [46] to prove that, if the extended graph $\overline{G}$ has small treewidth, then there exists an algorithm to solve (SDP) to ϵ accuracy in $\tilde{O}((m+n)\log(1/\epsilon))$ worst-case time. This improves over our $O(m+n)$ time per-iteration figure, which must be spread across $O(\sqrt{m+n}\log(1/\epsilon))$ worst-case iterations, for a total of $O((m+$

$n)^{1.5}\log(1/\epsilon))$ worst-case time. However, it is important to point out that these "fast" interior-point methods [12, 21] are purely theoretical; their analysis hides numerous leading constants, and it is unclear whether a real-world implementation could be made competitive against state-of-the-art solvers. On the other hand, primal-dual solvers like MOSEK [32] typically converge to ϵ accuracy in dimension-free $O(\log(1/\epsilon))$ iterations (see our experiments in Sect. 5), so in practice, our algorithm is already able to solve (SDP) to ϵ accuracy in $O((m+n)\log(1/\epsilon))$ empirical time.

2 Preliminaries

2.1 Notations and basic definitions

Write $\mathbb{R}^{m\times n}$ as the set of $m\times n$ matrices with real coefficients, with associated matrix inner product $\langle A, B\rangle = \text{tr}\, A^T B$ and norm $\|A\|_F = \sqrt{\langle A, A\rangle}$. Write $\mathbb{S}^n \subseteq \mathbb{R}^{n\times n}$ as the set of $n\times n$ real symmetric matrices, meaning that $X = X^T$ holds for all $X \in \mathbb{S}^n$, and write $\mathbb{S}^n_+ \subseteq \mathbb{S}^n$ as the set of symmetric positive semidefinite matrices. Write $\mathbb{R}^n_+ \subseteq \mathbb{R}^n$ as the usual positive orthant.

The set of $n\times n$ symmetric matrices with sparsity pattern E can be defined as

$$\mathbb{S}^n_E \equiv \left\{X \in \mathbb{S}^n : X[i,j] = X[j,i] = 0 \quad \text{for all } i \neq j, \quad (i,j) \notin E\right\}.$$

Conversely, the minimum sparsity pattern of a *symmetric* matrix $X \in \mathbb{S}^n$ is denoted

$$\text{spar}(X) \equiv \{(i,j) : X[i,j] \neq 0, \quad i \geq j\}.$$

We also write $\text{spar}(M) \equiv \text{spar}(M + M^T)$ for a nonsymmetric matrix M where there is no confusion. We write $\text{proj}_E(M) \equiv \arg\min_{X\in\mathbb{S}^n_E} \|M - X\|_F$ as the projection of $M \in \mathbb{S}^n$ onto the sparsity pattern E.

We define the dense sparsity pattern induced by $J \subseteq \{1, 2, \ldots, n\}$ as follows

$$\text{clique}(J) = \{(i,j) : i, j \in J, \quad i \geq j\}.$$

We also define the vertex support of a possibly nonsymmetric matrix M as the following

$$\text{supp}(M) = \{i : M[i,j] \neq 0 \quad \text{for some } j\}.$$

We write $\text{clique}(A) \equiv \text{clique}(\text{supp}(A))$ where there is no confusion. This notation is motivated by the fact that $\text{spar}(A) \subseteq \text{clique}(A)$ for $A \in \mathbb{S}^n$, and $\text{spar}(PDP^T) \subseteq \text{clique}(P)$ for $P \in \mathbb{R}^{n\times d}$ and dense $D \in \mathbb{S}^d$.

Let F be a sparsity pattern of order n that *contains all of its diagonal elements*, as in $F \supseteq \{(i,i) : i \in \{1, 2, \ldots, n\}\}$. In this case, $\dim(\mathbb{S}^n_F) = |F|$ holds exactly, so we define a *symmetric vectorization* operator $\text{svec}_F : \mathbb{S}^n_F \to \mathbb{R}^{|F|}$ to implement an isometry with the usual Euclidean space, as in

$$\langle \text{svec}_F(X), \text{svec}_F(Y)\rangle = \langle X, Y\rangle \text{ for all } X, Y \in \mathbb{S}^n_F.$$

We will explicitly require $\text{svec}_F(\cdot)$ to be defined according to a column-stacking construction

$$\text{svec}_F(X) = (x_j)_{j=1}^n \quad \text{where } x_j = \Big(X[i,i], \quad \sqrt{2} \cdot (X[i,j] : i > j, \quad (i,j) \in F) \Big).$$

We also define a companion indexing operator $\text{idx}_F(\cdot,\cdot)$ to index elements of the vectorization $x = \text{svec}_F(X)$:

$$\begin{aligned} x[\text{idx}_F(i,i)] &= X[i,i] \quad \text{for all } i \in \{1,2,\ldots,n\}, \\ x[\text{idx}_F(i,j)] &= \sqrt{2} X[i,j] \quad \text{for all } i > j, \quad (i,j) \in F. \end{aligned}$$

As we will see later, the correctness of our proof crucially relies on the fact that $\text{idx}_F(\cdot,\cdot)$ implements a *raster ordering* over the elements of F.

2.2 Sparse Cholesky factorization

To solve $Sx = b$ with $S \succ 0$ via Cholesky factorization, we first compute the lower-triangular Cholesky factor $L = \text{chol}(S)$ according to the following recursive rule

$$\text{chol}\left(\begin{bmatrix} \alpha & b^T \\ b & D \end{bmatrix} \right) = \begin{bmatrix} \sqrt{\alpha} & 0 \\ \frac{1}{\sqrt{\alpha}} b & \text{chol}\left(D - \frac{1}{\alpha} b b^T \right) \end{bmatrix}, \quad \text{chol}(\alpha) = \sqrt{\alpha},$$

and then solve two triangular systems $Ly = b$ and $L^T x = y$ via back-substitution. If S is sparse, then $L = \text{chol}(S)$ may also be sparse. The sparsity pattern of L can be directly computed from the sparsity pattern of S, without needing to examine the numerical values of its nonzeros.

Definition 2.1 (Symbolic Cholesky) The *symbolic Cholesky factor* $\text{chol}(E)$ of a sparsity pattern E of order n is defined as $\text{chol}(E) \equiv E_{n+1}$ where $E_1 = E$ and

$$E_{k+1} = E_k \cup (k,k) \cup \{(i,j) : (i,k) \in E_k, \quad (j,k) \in E_k \text{ for } i > j > k\}.$$

One can verify that $\text{chol}(\text{spar}(S)) = \text{spar}(\text{chol}(S))$. Note that $\text{chol}(E)$ can be computed from E in $O(|\text{chol}(E)|)$ time and memory [18, Theorem 5.4.4]. The efficiency of a sparsity-exploiting algorithm for factorizing $L = \text{chol}(S)$ and solving $Ly = b$ and $L^T x = y$ is determined by the *frontsize* of the sparse matrix S.

Definition 2.2 (Frontsize) The *frontsize* $\omega(E)$ of a sparsity pattern E is defined $\max_j |\text{col}_F(j)|$ where $F = \text{chol}(E)$ and $\text{col}_F(j) \equiv \{j\} \cup \{i > j : (i,j) \in F\}$. The frontsize $\omega(S) \equiv \omega(\text{spar}(S))$ of a symmetric matrix S is the frontsize of its minimum sparsity pattern.

Intuitively, the frontsize $\omega(S)$ is the maximum number of nonzero elements in a single column of the Cholesky factor $L = \text{chol}(S)$. The following is well-known [18].

Proposition 2.3 (Sparse Cholesky factorization) *Given $S \in \mathbb{S}^n$, $S \succ 0$, let $\omega \equiv \omega(S)$. Sparse Cholesky factorization factors $L = \text{chol}(S)$ in T arithmetic operations and M units of memory, where*

$$\frac{1}{6}(\omega-1)^3 + n \le \mathsf{T} \le \omega^2 \cdot n, \qquad \frac{1}{2}(\omega-1)^2 + n \le \mathsf{M} \le \omega \cdot n.$$

Proof Let $\omega_j \equiv |\text{col}_F(j)|$. By inspection, $\mathsf{T} = \sum_{j=1}^n \frac{1}{2}\omega_j(\omega_j+1)$ and $\mathsf{M} = \sum_{j=1}^n \omega_j$. The bounds follow by substituting $\omega \ge \omega_{j+1} \ge \omega_j - 1$. Indeed,

$$\mathsf{M} = \sum_{j=1}^n \omega_j \ge \underbrace{\omega + (\omega-1) + \cdots + 1}_{\omega \text{ terms}} + \underbrace{1 + \cdots + 1}_{n-\omega \text{ terms}} = \frac{1}{2}\omega(\omega+1) + (n-\omega)$$

and similarly $\mathsf{T} = \sum_{j=1}^n \frac{1}{2}\omega_j(\omega_j+1) \ge \frac{1}{6}\omega(\omega+1)(\omega+2) + (n-\omega)$. □

Note that Proposition 2.3 is sharp up to small additive constants: the upper-bound is essentially attained by banded matrices of bandwidth ω, while the lower-bound is essentially attained by a matrix that contains a single dense block of size ω.

2.3 Minimum frontsize and treewidth

The cost of solving $Sx = b$ with sparse $S \succ 0$ can usually be reduced by first permuting the rows and columns of the matrix symmetrically, and then solving $(\Pi S \Pi^T)\Pi x = \Pi y$ for some permutation matrix Π. For $E = \text{spar}(S)$, we write $E_\Pi \equiv \text{spar}(\Pi S \Pi^T)$ to denote its permuted sparsity pattern. It is a fundamental result in graph theory and linear algebra that the problem of minimizing the frontsize $\omega(E_\Pi)$ over the set of $n \times n$ permutation matrices $\Pi \in \text{Perm}(n)$ is the same problem as computing the treewidth of the graph $G = (V, E)$.

Definition 2.4 (Treewidth) A tree decomposition of a graph $G = (V, E)$ is a pair $(\{J_j\}, T)$ in which each *bag* $J_j \subseteq V$ is a subset of vertices and T is a tree such that:

- (Vertex cover) $\bigcup_j J_j = V$;
- (Edge cover) $\bigcup_j (J_j \times J_j) \supseteq E$;
- (Running intersection) $J_i \cap J_j \subseteq J_k$ for every k on the path of i to j on T.

The *width* of the tree decomposition is $\max_j |J_j| - 1$. The *treewidth* of G, denoted $\text{tw}(G)$, is the minimum width over all valid tree decompositions on G.

The connection is an immediate corollary of the following result, which establishes an equivalence between tree decompositions and the sparsity pattern of Cholesky factors.

Proposition 2.5 (Perfect elimination ordering) *Given a sparsity pattern E of order n, let $G = (V, E)$ where $V = \{1, 2, \ldots, n\}$. For every tree decomposition of G with width τ, there exists a perfect elimination ordering $\Pi \in \text{Perm}(n)$ such that $\omega(E_\Pi) = 1 + \tau$.*

We defer a proof to the texts [18, 44], and only note that, given a tree decomposition of width τ, the corresponding perfect elimination ordering Π can be found in $O((1+\tau)\cdot n)$ time.

Corollary 2.6 *We have* $1 + \mathrm{tw}(G) = \min_{\Pi \in \mathrm{Perm}(n)} \omega(E_\Pi)$.

As a purely theoretical result, if we assume that $\mathrm{tw}(G) = O(1)$ with respect to the number of vertices n, then a choice of $\Pi \in \mathrm{Perm}(n)$ that sets $\omega(E_\Pi) = O(1)$ can be found in $O(n)$ time [14] (and so the problem is no longer NP-hard). In practice, it is much faster to use simple greedy heuristics [1], which often find "good enough" choices of Π that yield very small values of $\omega(E_\Pi)$, without a rigorous guarantee of quality.

2.4 General-purpose interior-point methods

The basic approach for solving an SDP using a general-purpose solver is to reformulate the problem into the primal or the dual of the standard-form linear conic program

$$\min_{x \in \mathbb{R}^q} \{\mathbf{c}^T x : \mathbf{A}x = \mathbf{b}, x \in \mathcal{K}\} \geq \max_{y \in \mathbb{R}^p} \{\mathbf{b}^T y : \mathbf{c} - \mathbf{A}^T y \in \mathcal{K}_*\}$$

where the data are the matrix $\mathbf{A} \in \mathbb{R}^{p \times q}$, vectors $\mathbf{b} \in \mathbb{R}^p$ and $\mathbf{c} \in \mathbb{R}^q$, and the problem closed convex cone $\mathcal{K} \subseteq \mathbb{R}^q$, and the notation $\mathcal{K}_*$ means the dual cone of $\mathcal{K}$. We specify the following basic assumptions on this problem to ensure that it can be solved in polynomial time using a self-dual embedding [11]. Below, we denote $\mathbf{1}_\mathcal{K}$ as the identity element on the cone $\mathcal{K}$, and recall that every semidefinite cone is self-dual $\mathcal{K} = \mathcal{K}_*$.

Definition 2.7 (Standard-form SDP) We say that the problem data $\mathbf{A} \in \mathbb{R}^{p \times q}$, $\mathbf{b} \in \mathbb{R}^p$, $\mathbf{c} \in \mathbb{R}^q$, and $\mathcal{K} \subseteq \mathbb{R}^q$ describe an SDP in (n, ω)-*standard form* if:

1. (Dimensions) The cone $\mathcal{K} = \mathrm{svec}(\mathbb{S}_+^{\omega_1}) \times \cdots \times \mathrm{svec}(\mathbb{S}_+^{\omega_\ell})$ is the Cartesian product of semidefinite cones whose orders $\omega_1, \omega_2, \ldots, \omega_\ell$ satisfy

$$\omega = \max_i \omega_i, \qquad n = \sum_{i=1}^{\ell} \omega_i, \qquad q = \frac{1}{2} \sum_{i=1}^{\ell} \omega_i(\omega_i + 1).$$

2. (Linear independence) $\mathbf{A}^T y = 0$ holds if and only if $y = 0$.
3. (Strong duality is attained) There exist a choice of $x^\star$, $y^\star$ that satisfy

$$\mathbf{A}x^\star = \mathbf{b}, \quad x^\star \in \mathcal{K}, \quad \mathbf{c} - \mathbf{A}^T y^\star \in \mathcal{K}, \quad \mathbf{c}^T x^\star = \mathbf{b}^T y^\star.$$

Definition 2.8 (General-purpose solver) We say that ipm implements a *general-purpose solver* if it satisfies the following conditions

1. (Iteration count) Given data $(\mathbf{A}, \mathbf{b}, \mathbf{c}, \mathcal{K})$ in (n, ω)-standard-form, calling $(x, y) = \mathrm{ipm}(\epsilon, \mathbf{A}, \mathbf{b}, \mathbf{c}, \mathcal{K})$ yields iterates $(x, y) \in \mathcal{K} \times \mathbb{R}^p$ that satisfy the following, in at most $O(\sqrt{n} \log(1/\epsilon))$ iterations

$$\|\mathbf{A}x - \mathbf{b}\| \leq \epsilon, \quad \mathbf{c} - \mathbf{A}^T y + \epsilon \cdot \mathbf{1}_\mathcal{K} \in \mathcal{K}, \quad \mathbf{c}^T x - \mathbf{b}^T y \leq \epsilon \cdot n.$$

Algorithm 1 Chordal conversion

Input. Accuracy parameter $\epsilon > 0$, problem data $C, A_1, A_2, \ldots, A_m \in \mathbb{S}^n$, $b \in \mathbb{R}^m$, fill-reducing permutation Π.
Output. Approximate solutions $U \in \mathbb{R}^{n\times\omega}$ and $v \in \mathbb{R}^m$ to the following primal-dual pair:

$$\min_U \left\{ \left\langle C, UU^T \right\rangle : \left\langle A_i, UU^T \right\rangle \le b_i \text{ for all } i \right\} \ge \max_{v\le 0} \left\{ \langle b, v\rangle : \textstyle\sum_i v_i A_i \preceq C \right\}.$$

Algorithm.

1. (Symbolic factorization) Pre-order all data matrices $\tilde{A}_i = \Pi A_i \Pi^T$ and $\tilde{C} = \Pi C \Pi^T$. Compute the permuted aggregate sparsity pattern $E = \text{spar}(\tilde{C}) \cup \bigcup_i \text{spar}(\tilde{A}_i)$, its lower-triangular symbolic Cholesky factor $F = \text{chol}(E)$, and define the following

$$J_j = \text{col}_F(j) \equiv \{j\} \cup \{i > j : (i, j) \in F\}, \quad \omega_j \equiv |J_j|, \quad \omega \equiv \max_j \omega_j.$$

2. (Numerical solution) Call $(x, y) = \text{ipm}(\epsilon, \mathbf{A}, \mathbf{b}, \mathbf{c}, \mathcal{K})$ where ipm is a general-purpose solver (Definition 2.8), and the problem data $\mathbf{A}, \mathbf{b}, \mathbf{c}, \mathcal{K}$ implement the following

$$Y^\star = \arg\max_{Y\in\mathbb{S}^n_F} \left\langle -\tilde{C}, Y \right\rangle \text{ s.t. } \begin{array}{ll} b_i - \left\langle \tilde{A}_i, Y \right\rangle \ge 0 & \text{for all } i \in \{1, 2, \ldots, m\} \\ Y[J_j, J_j] \succeq 0 & \text{for all } j \in \{1, 2, \ldots, n\} \end{array}$$

   as an instance of $y^\star = \arg\max_y \{\mathbf{b}^T y : \mathbf{c} - \mathbf{A}^T y \in \mathcal{K}\}$ with $y^\star = \text{svec}_F(Y^\star)$.
3. (Back substitution) Recover $Y \in \mathbb{S}^n_F$ from $y = \text{svec}_F(Y)$, and compute $\delta = -\min_j \{0, \lambda_{\min}(Y[J_j, J_j])\}$. Solve the positive semidefinite matrix completion

$$\text{find } \tilde{U} \in \mathbb{R}^{n\times\omega} \text{ such that } (\tilde{U}\tilde{U}^T)[J_j, J_j] = Y[J_j, J_j] + \delta I \text{ for all } j$$

   using [39, Algorithm 2]. Output $v = -(x_i)_{i=1}^m$ and $U = \Pi^T \tilde{U}$.

2. (Per-iteration costs) Each iteration costs an overhead of $O(\omega^2 n)$ time and $O(\omega n)$ memory, plus the cost of solving $O(1)$ instances of the *Schur complement equation*

$$\mathbf{A}\nabla^2 f(w)\mathbf{A}^T \Delta y = r, \qquad f(w) = -\log\det(w)$$

   by forming $\mathbf{H} = \mathbf{A}\nabla^2 f(w)\mathbf{A}^T$, factoring $\mathbf{L} = \text{chol}(\Pi\mathbf{H}\Pi^T)$ and then solving $\mathbf{L}z = \Pi r$ and $\mathbf{L}^T(\Pi\Delta y) = z$. Here, the fill-reducing permutation Π is required to be no worse than the natural ordering, as in $\omega(\Pi\mathbf{H}\Pi^T) \le \omega(\mathbf{H})$.

Note that Definition 2.8 is rigorously satisfied by SeDuMi [37, 38], SDPT3 [42, 43], and MOSEK [2, 3, 32]. Given that the correctness of our overall claims crucially depends on the characterization in Definition 2.8, we state a concrete interior-point method in Appendix B that implements these specifications.

3 Main results

Algorithm 1 summarizes the standard implementation of chordal conversion, which is known as the "d-space conversion method using basis representation" in Kim et al. [24]. Our only modification is to recover $X = UU^T$ from $Y = \text{proj}_F(X)$ in Step

3 using the *low-rank* chordal completion [10, Theorem 1.5], instead of the *maximum determinant* chordal completion [20, Theorem 2] as originally proposed by Fukuda et al. [16]. We note that both recovery procedures have the same complexity of $O(\omega^3 n)$ time and $O(\omega^2 n)$ memory, but the former puts X in a more convenient form [39].

The cost of Algorithm 1 is dominated by the cost of solving the *Schur complement equation* at each iteration of the interior-point method in Step 2. At the heart of this paper is a simple but precise upper-bound on the frontsize of its sparsity pattern $E^{(2)}$, given in terms of the extended sparsity pattern $\overline{E}$. Concretely, our result says that if $\omega(\overline{E}) = O(1)$, then $\omega(E^{(2)}) = O(1)$, so the Schur complement matrix can be formed, factored, and backsubstituted in $O(m+n)$ time. Hence, the per-iteration cost of the interior-point method is also $O(m+n)$ time.

To state the Schur complement sparsity $E^{(2)}$ explicitly, note that the problem data $(\mathbf{A}, \mathbf{b}, \mathbf{c}, \mathcal{K})$ in Step 2 of Algorithm 1 are

$$\begin{gathered} \mathbf{A} = [\operatorname{svec}_F(A_1), \ldots, \operatorname{svec}_F(A_m), -\mathbf{P}_1, \ldots, -\mathbf{P}_n], \\ \mathbf{b} = -\operatorname{svec}_F(C), \qquad \mathbf{c} = (b, 0), \\ \mathcal{K} = \mathbb{R}^m_+ \times \operatorname{svec}(\mathbb{S}^{\omega_1}_+) \times \operatorname{svec}(\mathbb{S}^{\omega_2}_+) \times \cdots \times \operatorname{svec}(\mathbb{S}^{\omega_n}_+) \end{gathered} \tag{3a}$$

where $\omega_j \equiv |\operatorname{col}_F(j)|$ and each $\mathbf{P}_j$ is implicitly defined to satisfy

$$\mathbf{P}_j^T \operatorname{svec}_F(Y) = \operatorname{svec}(Y[\operatorname{col}_F(j), \operatorname{col}_F(j)]) \quad \text{for all } Y \in \mathbb{S}^n_F. \tag{3b}$$

The resulting Schur complement matrix reads

$$\begin{gathered} \mathbf{A}\nabla^2 f(w)\mathbf{A}^T = \sum_{i=1}^m d_i \operatorname{svec}_F(A_i) \operatorname{svec}_F(A_i)^T + \sum_{j=1}^n \mathbf{P}_j \mathbf{D}_j \mathbf{P}_j^T \\ \text{where } d_i = w_i^{-2}, \qquad \mathbf{D}_j \operatorname{svec}(X_j) = \operatorname{svec}(W_j^{-1} X_j W_j^{-1}) \end{gathered} \tag{4}$$

and $w = (w_1, \ldots, w_m, \operatorname{svec}(W_1), \ldots, \operatorname{svec}(W_n)) \in \operatorname{Int}(\mathcal{K})$ is a scaling point. The associated sparsity pattern, aggregated over all possible choices of scaling w, is as follows

$$\begin{aligned} E^{(2)} &= \{(i,j) : (\mathbf{A}\nabla^2 f(w)\mathbf{A}^T)[i,j] \neq 0 \text{ for some } w \in \operatorname{Int}(\mathcal{K})\} \\ &= \left(\bigcup_{i=1}^m \operatorname{spar}(a_i a_i^T)\right) \cup \left(\bigcup_{j=1}^n \operatorname{spar}(\mathbf{P}_j \mathbf{D}_j \mathbf{P}_j^T)\right) \\ &= \left(\bigcup_{i=1}^m \operatorname{clique}(\operatorname{supp}(a_i))\right) \cup \left(\bigcup_{j=1}^n \operatorname{clique}(\operatorname{supp}(\mathbf{P}_j))\right) \end{aligned} \tag{5}$$

where we have written $a_i = \operatorname{svec}_F(A_i)$. The result below says that if $\omega(\overline{E}) = O(1)$, then $\omega(\overline{E}^{(2)}) = O(1)$.

Theorem 3.1 (Frontsize of Schur complement sparsity) *Given* $C, A_1, A_2, \ldots, A_m \in \mathbb{S}^n$, *define* $\mathbf{A}, \mathcal{K}$ *as in* (3), *and define* $E^{(2)}$ *as in* (5). *We have*

$$\frac{1}{2}\omega(\omega+1) \leq \omega(E^{(2)}) \leq \frac{1}{2}\overline{\omega}(\overline{\omega}+1)$$

where $\omega \equiv \omega(E)$ *and* $\overline{\omega} = \omega(\overline{E})$ *and* $E, \overline{E}$ *are defined in* (1) *and* (2). *Moreover, if* $E = \overline{E}$, *then we also have* $\text{chol}(E^{(2)}) = E^{(2)}$.

In cases where $E = \overline{E}$, as in the MAX-k-CUT relaxation (Example 1.2) and the Lovász theta problem (Example 1.3), Theorem 3.1 predicts that the Schur complement matrix $\mathbf{H} = \mathbf{A}\nabla^2 f(w)\mathbf{A}^T$ can be factored $\mathbf{L} = \text{chol}(\mathbf{H})$ with *zero fill-in*, meaning that $\text{spar}(\mathbf{L}+\mathbf{L}^T) = \text{spar}(\mathbf{H})$. More generally, if $\omega < \overline{\omega}$ holds with a small gap, then we would also expect $\mathbf{H}$ to factor with very little fill-in.

As previously pointed out by Kobayashi et al. [25], if the Schur complement sparsity $E^{(2)}$ is known to have frontsize $\omega(E^{(2)}) = O(1)$, then the Schur complement matrix $\mathbf{H} = \mathbf{A}\nabla^2 f(w)\mathbf{A}$ can be formed, factored, and backsubstituted in $O(m+n)$ time. Hence, the per-iteration cost of the interior-point method is also $O(m+n)$ time.

Corollary 3.2 (Cost of Schur complement equation) *Given the data matrix* $\mathbf{A}$, *scaling point* $w \in \text{Int}(\mathcal{K})$, *and right-hand side* g, *define* $\mathbf{H} = \mathbf{A}\nabla^2 f(w)\mathbf{A}^T$ *as in* (5). *Suppose that all columns of* $\mathbf{A}$ *are nonzero, and all scaling matrices* $\mathbf{D}_j$ *are fully dense. Then, it takes* T *arithmetic operations and* M *units of storage to form* $\mathbf{H}$ *and solve* $\mathbf{H}\Delta y = g$, *where*

$$\frac{1}{48}(\omega-1)^6 + m + n \leq \mathsf{T} \leq 4\overline{\omega}^4 \cdot (m+\omega n),$$
$$\frac{1}{8}(\omega-1)^4 + m + n \leq \mathsf{M} \leq 2\overline{\omega}^2 \cdot (m+\omega n),$$

in which $\omega \equiv \omega(E)$ *and* $\overline{\omega} \equiv \omega(\overline{E})$ *satisfy* $1 \leq \omega \leq \overline{\omega}$.

Proof Let $a_i \equiv \text{svec}_F(A_i)$ and $\omega_j \equiv |\,\text{col}_F(j)|$. We break the solution of $\mathbf{H}\Delta y = g$ into five steps:

1. (Input) It takes M_{data} memory to state the problem data, where $m+n \leq \mathsf{M}_{\text{data}} \leq 2\overline{\omega}^2 m + 3\omega^2 n$. Indeed, $m+n \leq \text{nnz}(\mathbf{A}) \leq \overline{\omega}^2 m + \omega^2 n$ because $1 \leq \text{nnz}(a_i) \leq \overline{\omega}^2$ (see Step 2 below) and $\text{nnz}(\mathbf{P}_j) = \frac{1}{2}\omega_j(\omega_j+1)$. Also, $m+n \leq \text{nnz}(w) \leq m+\omega^2 n$, and $n \leq \text{nnz}(g) \leq \omega n$.
2. (Build LP part) It takes T_{LP} time to build $\sum_{i=1}^m d_i a_i a_i^T$, where $m \leq \mathsf{T}_{\text{LP}} \leq 2\overline{\omega}^4 m$ time and memory. This follows from $1 \leq \text{nnz}(a_i) \leq \overline{\omega}^2$, where the upper-bound is because $\text{spar}(a_i a_i^T) \subseteq E^{(2)} \subseteq \text{chol}(E^{(2)})$, and that $\omega(E^{(2)})$ is, by definition, the maximum number of nonzero elements in a single column of $\text{chol}(E^{(2)})$.
3. (Build SDP part) It takes T_{SDP} time to build $\sum_{j=1}^n \mathbf{P}_j \mathbf{D}_j \mathbf{P}_j^T$, where $n \leq \mathsf{T}_{\text{SDP}} \leq \omega^4 n$ time and memory. This follows from $\text{nnz}(\mathbf{P}_j) = \frac{1}{2}\omega_j(\omega_j+1)$, which implies $\text{nnz}(\mathbf{P}_j \mathbf{D}_j \mathbf{P}_j^T) = \frac{1}{4}\omega_j^2(\omega_j+1)^2$ for a fully-dense $\mathbf{D}_j \succ 0$.

4. (Factorization) It takes T_{fact} time and M_{fact} memory to factor $\mathbf{L} = \text{chol}(\mathbf{H})$, where $\frac{1}{48}(\omega-1)^6 + n \le \mathsf{T}_{\text{fact}} \le \overline{\omega}^5 n$ and $\frac{1}{8}(\omega-1)^4 + n \le \mathsf{M}_{\text{fact}} \le \overline{\omega}^3 n$. The matrix $\mathbf{H}$ has $|F|$ columns and rows, and frontsize $\omega(E^{(2)})$. The desired figures follow by substituting $\frac{1}{2}\omega^2 \le \omega(E^{(2)}) \le \overline{\omega}^2$ and $n \le |F| \le \omega n$ into Proposition 2.3.
5. (Back-substitution) It takes M_{fact} time and memory to solve each of $\mathbf{L}z = g$ and $\mathbf{L}^T \Delta y = z$ via triangular back-substitution.

The overall runtime is cumulative, so $\mathsf{T} = \mathsf{T}_{\text{LP}} + \mathsf{T}_{\text{SDP}} + \mathsf{T}_{\text{fact}} + 2\mathsf{M}_{\text{fact}}$. The overall memory use is $\mathsf{M} = \mathsf{M}_{\text{data}} + \mathsf{M}_{\text{fact}}$, because the matrix $\mathbf{H}$ can be constructed and then factored in-place. □

Let us now give an end-to-end complexity guarantee for Algorithm 1. We will need the following assumption to ensure that the data $(\mathbf{A}, \mathbf{b}, \mathbf{c}, \mathcal{K})$ previously defined in (3) specifies an SDP in (N, ω)-standard form, where $N = m + \omega n$ and $\omega \equiv \omega(E)$.

Assumption 1 (Strong duality is attained) *There exists a primal-dual pair $X^\star \succeq 0$ and $v^\star \le 0$ that are feasible $\langle A_i, X^\star\rangle \le b_i$ for all i and $\sum_i v_i^\star A_i \preceq C$ and coincide in their objectives $\langle C, X^\star\rangle = \langle b, v^\star\rangle$.*

Theorem 3.3 (Upper complexity) *Let the data $C, A_1, \ldots, A_m \in \mathbb{S}^n$ and $b \in \mathbb{R}^m$ satisfy Assumption 1. Given a tree decomposition of width $\overline{\tau}$ for the extended aggregate sparsity graph $\overline{G} = (V, \overline{E})$, where*

$$V = \{1, 2, \ldots, n\}, \qquad \overline{E} = \text{spar}(C) \cup \text{clique}(A_1) \cup \cdots \cup \text{clique}(A_m),$$

set Π as the associated perfect elimination ordering. Then, Algorithm 1 outputs $U \in \mathbb{R}^{n \times \overline{\omega}}$ and $v \in \mathbb{R}^m$ with $v \le 0$ such that

$$\left\langle A_i, UU^T \right\rangle - b_i \le \epsilon \text{ for all } i, \quad \sum_{i=1}^m v_i A_i - C \preceq \epsilon \cdot I, \quad \left\langle C, UU^T \right\rangle - \langle b, v\rangle \le \epsilon \cdot N,$$

in $O(\sqrt{N}\log(1/\epsilon))$ iterations, with per-iteration costs of $O(\overline{\omega}^4 \cdot N)$ time and $O(\overline{\omega}^2 \cdot N)$ memory, where $N = m + \overline{\omega} \cdot n$ and $\overline{\omega} = 1 + \overline{\tau}$.

Proof One can verify that $(\mathbf{A}, \mathbf{b}, \mathbf{c}, \mathcal{K})$ defined in (3) specifies an SDP in (N, ω)-standard form (see Appendix A for the regularity conditions). Moreover, it follows from the monotonicity of the frontsize (Proposition 4.1) that $\omega \equiv \omega(E_\Pi) \le \omega(\overline{E}_\Pi) \equiv \overline{\omega}$. We will track the cost of Algorithm 1 step-by-step:

1. (Front-reducing permutation) Preordering $A_i \leftarrow \Pi A_i \Pi^T$ and $C \leftarrow \Pi C \Pi^T$ the matrices cost $O(\text{nnz}(C) + \sum_{i=1}^m \text{nnz}(A_i)) = O(\overline{\omega}^2 N)$ time and memory. This follows from $\text{nnz}(C) \le |F| \le N$ and $\text{nnz}(A_i) \le \overline{\omega}^2$.
2. (Conversion) Computing $F = \text{chol}(E)$ costs $O(|F|) = O(N)$ time and space, where we note that $|F| \le \omega n$.

3. (Solution) Let $K = 2 \cdot \max\{1, |\operatorname{tr}(C)|, |\operatorname{tr}(A_1)|, \ldots, |\operatorname{tr}(A_m)|\}$. After $O(\sqrt{N}\log(K/\epsilon))$ iterations, we arrive at a primal $v \leq 0$ and $V_j \succeq 0$ and dual point $Y \in \mathbb{S}^n_F$ satisfying

$$\begin{gathered}\langle A_i, Y\rangle - b_i \leq \epsilon/K, \quad Y[J_j, J_j] \succeq -(\epsilon/K)I, \\ \left\| \textstyle\sum_i v_i A_i + \sum_j P_j V_j P_j^T - C \right\| \leq \epsilon/K, \\ \langle C, Y\rangle - \langle b, v\rangle \leq N \cdot (\epsilon/K).\end{gathered}$$

   Each iteration costs $O(\overline{\omega}^4 N)$ time and $O(\overline{\omega}^2 N)$ memory. This cost is fully determined by the cost of solving $O(1)$ instances of the Schur complement equation, which dominates the overhead of $O(\omega^3 n + \operatorname{nnz}(\mathbf{A})) = O(\overline{\omega}^2 N)$ time and memory.
4. (Recovery) Using the previously recovered Y, we recover U such that $\Pi_F(UU^T) = Y + \delta I$ where $\delta = -\min_j\{0, \lambda_{\min}(Y[J_j, J_j])\} \leq \epsilon/K$. This takes $O(\omega^3 n) = O(\overline{\omega}^2 N)$ time and $O(\omega^2 n) = O(\overline{\omega} N)$ memory.
5. (Output) We output U and v, and check for accuracy. It follows from $K \geq 2|\operatorname{tr}(C)|$ that

$$\left\langle C, UU^T \right\rangle - \langle C, Y\rangle = \delta \cdot \operatorname{tr}(C) \leq \epsilon \cdot \frac{\operatorname{tr}(C)}{K} \leq \epsilon \cdot \frac{|\operatorname{tr}(C)|}{2|\operatorname{tr}(C)|} \leq \frac{1}{2}\epsilon,$$

   and from $K \geq 2$ and $N \geq 1$ that

$$\left\langle C, UU^T \right\rangle - \langle b, v\rangle \leq \frac{N}{K}\epsilon + \frac{1}{2}\epsilon = N\epsilon \cdot \left(\frac{1}{2} + \frac{1}{2N}\right) \leq N \cdot \epsilon.$$

   Similarly, it follows from $K \geq 2|\operatorname{tr}(A_i)|$ and $K \geq 2$ that $\left\langle A_i, UU^T \right\rangle - b_i \leq \epsilon$. Finally, it follows from $v \geq 0$ and $V_j \succeq 0$ that

$$\sum_i v_i A_i + \sum_j P_j V_j P_j^T - C \preceq \epsilon \cdot I \implies \sum_i v_i A_i - C \preceq \epsilon \cdot I.$$

□

Let $\overline{\tau}_\star \equiv \operatorname{tw}(\overline{G})$. In theory, it takes $O(\overline{\tau}_\star^8 \cdot n \log n)$ time to compute a tree decomposition of width $\overline{\tau} = O(\overline{\tau}_\star^2)$ by exhaustively enumerating the algorithm of Fomin et al. [14]. Using this tree decomposition, Theorem 3.3 says that Algorithm 1 arrives at an ϵ-accurate solution in $O((m+n)^{1/2} \cdot \overline{\tau}_\star \cdot \log(1/\epsilon))$ iterations, with per-iteration costs of $O((m+n) \cdot \overline{\tau}_\star^{10})$ time and $O((m+n) \cdot \overline{\tau}_\star^6)$ memory. Combined, the end-to-end complexity of solving (SDP) using Algorithm 1 is $O(\overline{\tau}_\star^{11} \cdot (m+n)^{1.5} \cdot \log(1/\epsilon))$ time.

In practice, chordal conversion works even better. In Sect. 5, we provide detailed numerical experiments to validate that: (i) the minimum degree heuristic usually finds Π that yield $\overline{\omega} = O(1)$; (ii) a primal-dual interior-point method usually converges to ϵ accuracy in dimension-free $O(\log(1/\epsilon))$ iterations. Taking these as formal assumptions improves Theorem 3.3 to $O((m+n)\log(1/\epsilon))$ empirical time.

The following establishes the sharpness of Theorem 3.3.

Corollary 3.4 (Lower complexity) *Given the data* $C, A_1, \ldots, A_m \in \mathbb{S}^n$ *and* $b \in \mathbb{R}^m$, *let* $\tau_\star$ *denote the treewidth of the aggregate sparsity graph* $G = (V, E)$*:*

$$V = \{1, 2, \ldots, n\}, \qquad E = \operatorname{spar}(C) \cup \operatorname{spar}(A_1) \cup \cdots \cup \operatorname{spar}(A_m).$$

There exists no choice of Π *that will allow Algorithm 1 to solve* (SDP) *to arbitrary accuracy* $\epsilon > 0$ *in less than* $\Omega(\tau_\star^6 + m + n)$ *time and* $\Omega(\tau_\star^4 + m + n)$ *memory.*

Proof The cost of Algorithm 1 is at least a single iteration of the interior-point method in Step 2. This is no less than $\Omega((\omega - 1)^6 + m + n)$ time and $\Omega((\omega - 1)^4 + m + n)$ memory according to Corollary 3.2, where $\omega - 1 \geq \tau_\star$ due to Corollary 2.6. □

4 Frontsize of the Schur complement sparsity (Proof of Theorem 3.1)

We now turn to prove the frontsize bound on the Schur complement sparsity $E^{(2)}$ in Theorem 3.1, which we identified as our key technical contribution. Recall that a symmetric sparsity pattern E of order n can be viewed as the edge set of an undirected graph $G = (V, E)$ on vertices $V = \{1, 2, \ldots, n\}$. The underlying principle behind our proof is the fact that the frontsize is monotone under the subgraph relation: if $G' = (V', E')$ is a subgraph of $G = (V, E)$, then $\omega(E) \geq \omega(E')$.

To state this formally, we denote the sparsity pattern induced by a subset of vertices $U = \{u_1, u_2, \ldots, u_p\} \subseteq V$ as follows

$$E[U] \equiv \{(i, j) : (u_i, u_j) \in E\} \text{ where } u_1 < u_2 < \cdots < u_p.$$

Note that we always sort the elements of U. Our definition is made so that if $E = \operatorname{spar}(X)$, then $E[U] = \operatorname{spar}(X[U, U])$, without any reordering of the rows and columns.

Proposition 4.1 (Subgraph monotonicity) *Let* E *be a sparsity pattern of order* n*, and let* $U \subseteq \{1, 2, \ldots, n\}$*. Then, for any sparsity pattern* D *of order* $|U|$ *that satisfies* $E[U] \supseteq D$*, we have* $\operatorname{chol}(E)[U] \supseteq \operatorname{chol}(D)$*, and therefore* $\omega(E) \geq \omega(D)$.

Proof It is known that $(i, j) \in \operatorname{chol}(E)$ for $i > j$ holds if and only if there exists a path $(i, p_1, p_2, \ldots, p_\ell, j)$ whose edges are in E, and whose internal nodes are ordered $p_1, p_2, \ldots, p_\ell < j < i$; see e.g. [44, Theorem 6.1]. It immediately follows this characterization that $\operatorname{chol}(\cdot)$ is monotone with respect to the deletion of edges and isolated vertices: (1) if $D \subseteq E$, then $\operatorname{chol}(D) \subseteq \operatorname{chol}(E)$; (2) we have $\operatorname{chol}(E[U]) = \operatorname{chol}(E)[U]$ for $U = \{1, 2, \ldots, n\} \backslash v$ with isolated vertex v. Therefore, $\operatorname{chol}(\cdot)$ must also be monotone under general vertex and edge deletions, because we can always delete edges to isolate a vertex before deleting it. □

Our lower-bound is a direct corollary of the following result, which gives an exact value for the frontsize of a certain "lifted" sparsity pattern.

Lemma 4.2 (Quadratic lift) *Let* E *be an arbitrary sparsity pattern of order* n*. Define* $F = \operatorname{chol}(E)$ *and the lifted sparsity pattern* $F^{(2)} = \bigcup_{k=1}^{n} \operatorname{clique}(\mathbf{P}_k)$ *in which each*

$\mathbf{P}_k$ *is implicitly defined to satisfy* $\mathbf{P}_k^T \operatorname{svec}_F(Y) = \operatorname{svec}(Y[\operatorname{col}_F(k), \operatorname{col}_F(k)])$ *for all* $Y \in \mathbb{S}_F^n$. *Then, we have* $\operatorname{chol}(F^{(2)}) = F^{(2)}$ *and* $\omega(F^{(2)}) = \frac{1}{2}\omega(E)[\omega(E)+1]$.

Observe that $E^{(2)} \supseteq F^{(2)} = \bigcup_{k=1}^n \operatorname{clique}(\mathbf{P}_k)$ via (5), so it follows immediately from Proposition 4.1

$$\omega(E^{(2)}) \geq \omega(F^{(2)}) = \frac{1}{2}\omega(E)[\omega(E)+1],$$

which is precisely the lower-bound in Theorem 3.1. For the upper-bound, we will use $\overline{F} = \operatorname{chol}(\overline{E})$, the symbolic Cholesky factor of the extended aggregate sparsity pattern $\overline{E}$, to construct a similarly lifted $\overline{F}^{(2)}$. Our key insight is that $E^{(2)}$ can be obtained from $\overline{F}^{(2)}$ via vertex and edge deletions.

Lemma 4.3 (Sparsity overestimate) *Let* $\overline{E}$ *and* $E^{(2)}$ *be the sparsity patterns defined in* (2) *and* (5). *Define* $\overline{F} = \operatorname{chol}(\overline{E})$ *and the lifted sparsity pattern* $\overline{F}^{(2)} = \bigcup_{k=1}^n \operatorname{clique}(\overline{\mathbf{P}}_k)$ *in which each* $\overline{\mathbf{P}}_k$ *is implicitly defined to satisfy* $\overline{\mathbf{P}}_k^T \operatorname{svec}_{\overline{F}}(Y) = \operatorname{svec}(Y[\operatorname{col}_{\overline{F}}(k), \operatorname{col}_{\overline{F}}(k)])$ *for all* $Y \in \mathbb{S}_{\overline{F}}^n$. *Then,* $E^{(2)} \subseteq \overline{F}^{(2)}[V^{(2)}]$ *holds for* $V^{(2)} = \{\operatorname{idx}_{\overline{F}}(i,j) : i, j \in F\}$.

Substituting $E^{(2)} \subseteq \overline{F}^{(2)}[V^{(2)}]$ with the exact frontsize of $\omega(\overline{F}^{(2)})$ from Lemma 4.2 yields

$$\omega(E^{(2)}) \leq \omega(\overline{F}^{(2)}) = \frac{1}{2}\omega(\overline{E})[\omega(\overline{E})+1],$$

which is precisely the upper-bound in Theorem 3.1. Finally, if $E = \overline{E}$, then $F^{(2)} = \overline{F}^{(2)} = E^{(2)}$, so $\operatorname{chol}(E^{(2)}) = E^{(2)}$ via Lemma 4.2.

In the remainder of this section, we will prove Lemmas 4.2 and 4.3.

4.1 Exact frontsize of a lifted sparsity pattern (Proof of Lemma 4.2)

Our proof of Lemma 4.2 is based on a connection between zero-fill sparsity patterns, for which sequential Gaussian elimination results in no additionally fill-in, and a "sorted" extension of the running intersection property.

Definition 4.4 (ZF) The sparsity pattern F is said to be *zero-fill* if $F = \operatorname{chol}(F)$.

Equivalently, if F is zero-fill, then $(i,k) \in F$ and $(j,k) \in F$ implies $(i,j) \in F$ for $i > j > k$ via the definition of the symbolic Cholesky factor (Definition 2.1).

Definition 4.5 (RIP) The sequence of subsets $J_1, J_2, \ldots, J_\ell$ with $J_j \subset \mathbb{N}$ satisfies the *sorted running intersection property* if there exists a parent pointer $\mathrm{p} : \{1, 2, \ldots, \ell - 1\} \to \{2, 3, \ldots, \ell\}$ such that the following holds for all $1 \leq j < \ell$:

$$\mathrm{p}(j) > j, \quad J_{\mathrm{p}(j)} \supseteq J_j \cap (J_{j+1} \cup J_{j+2} \cup \cdots \cup J_\ell), \quad \min\{J_{\mathrm{p}(j)}\} > \max\{J_j \backslash J_{\mathrm{p}(j)}\}.$$

The symbolic Cholesky factor $F = \operatorname{chol}(E)$ for a sparsity pattern E is the canonical example of a zero-fill sparsity pattern. In turn, the corresponding column sets $J_j =$

$\mathrm{col}_F(j)$ are the canonical example of sequence of subsets that satisfy the sorted version of the running intersection property.

Proposition 4.6 (ZF $\Longrightarrow$ RIP) *Let F be a zero-fill sparsity pattern of order n. Then, the sequence of subsets $J_1, J_2, \ldots, J_n$ with $J_j = \mathrm{col}_F(j) \equiv \{j\} \cup \{i > j : (i, j) \in F\}$ satisfies the sorted running intersection property.*

Proof Define $\mathrm{p}(j) = \min\{i > j : i \in J_j\}$ if $|J_j| > 1$ and $\mathrm{p}(j) = n$ if $|J_j| = 1$. Clearly, $\mathrm{p}(j) > j$ holds for all $1 \leq j < n$. To prove $J_{\mathrm{p}(j)} \supseteq J_j \cap \bigcup_{w=j+1}^{\ell} J_w$, let $i \in J_j \cap J_w$ for some $w > j$. We prove that $i \in J_{\mathrm{p}(j)}$ via the following steps:

- We have $i > j$, because $i \in J_w$ implies that $i \geq w > j$.
- If $i = \mathrm{p}(j)$, then $i \in J_{\mathrm{p}(j)}$ by definition.
- If $i > \mathrm{p}(j)$, then $(i, j) \in F$ and $(\mathrm{p}(j), j) \in F$ for $i > \mathrm{p}(j) > j$ implies $(i, \mathrm{p}(j)) \in F$, and hence $i \in J_{\mathrm{p}(j)}$.

Finally, we prove $\min\{J_{\mathrm{p}(j)}\} > \max\{J_j \backslash J_{\mathrm{p}(j)}\}$ by noting that $\max\{J_j \backslash J_{\mathrm{p}(j)}\} = j$ with our construction, and that $i = \min J_{\mathrm{p}(j)}$ must satisfy $i \in J_{\mathrm{p}(j)}$ and therefore $i \geq \mathrm{p}(j) > j$. □

Our proof is based on the fact that the "lifted" sparsity pattern $F^{(2)}$ can be constructed as $F^{(2)} = \bigcup_{k=1}^{n} \mathrm{clique}(J_k^{(2)})$ with respect to the following "lifted" index sets

$$J_k^{(2)} \equiv \mathrm{idx}_F(\mathrm{clique}(J_k)) = \{\mathrm{idx}_F(i, j) : i, j \in J_k, \quad i \geq j\}. \tag{6}$$

We need to show that, if the original index sets $J_1, J_2, \ldots, J_k$ satisfy the running intersection property, then the lifted index sets $J_1^{(2)}, J_2^{(2)}, \ldots, J_k^{(2)}$ will inherit the running intersection property. Our key insight is that the index operator idx_F implements a *raster ordering*.

Lemma 4.7 (Raster ordering) *The ordering $\mathrm{idx}_F : F \to \mathbb{N}$ satisfies the following, for all $(i, j) \in F$ with $i \geq j$ and $(i', j') \in F$ with $i' \geq j'$:*

- *If $j > j'$, then $\mathrm{idx}(i, j) > \mathrm{idx}(i', j')$ holds.*
- *If $j = j'$ and $i > i'$, then $\mathrm{idx}(i, j) > \mathrm{idx}(i', j')$ holds.*

Lemma 4.8 *Let the sequence of subsets $J_1, J_2, \ldots, J_\ell$ with $J_j \subset \mathbb{N}$ satisfy the sorted running intersection property. Then, $J_1^{(2)}, J_2^{(2)}, \ldots, J_\ell^{(2)}$ also satisfy the sorted running intersection property.*

Proof Let $\mathrm{p}(\cdot)$ denote the parent pointer that verifies the sorted running intersection property in $J_1, J_2, \ldots, J_\ell$. We will verify that $\mathrm{p}(\cdot)$ also proves the same property in $J_1^{(2)}, J_2^{(2)}, \ldots, J_\ell^{(2)}$.

First, to prove $J_u^{(2)} \cap \bigcup_{v=u+1}^{\ell} J_v^{(2)} \subseteq J_{\mathrm{p}(u)}^{(2)}$, let $k \in J_u^{(2)} \cap J_v^{(2)}$ for $v > u$. The fact that $k \in J_u^{(2)} = \mathrm{idx}(\mathrm{clique}(J_u))$ implies $k = \mathrm{idx}(i, j)$ for some i, j such that $i, j \in J_u$. Similarly, $k \in J_v^{(2)} = \mathrm{idx}(\mathrm{clique}(J_v))$ and the bijectivity of idx on F imply that the same i, j also satisfy $i, j \in J_v$, where we recall that $v > u$. We conclude $i, j \in J_u \cap J_v \subseteq J_{\mathrm{p}(u)}$ and therefore $k = \mathrm{idx}(i, j) \in J_{\mathrm{p}(u)}^{(2)}$.

Next, we prove $\min\{J_{\mathrm{p}(u)}^{(2)}\} > \max\{J_u^{(2)} \backslash J_{\mathrm{p}(u)}^{(2)}\}$ by establishing two claims:

- $\min\{J^{(2)}_{p(u)}\} = \text{idx}(\alpha, \alpha)$ where $\alpha = \min J_{\text{p}(u)}$. For any $\text{idx}(i, j) \in J^{(2)}_{\text{p}(u)}$ where $i \geq j$, we must have $i, j \in J_{\text{p}(u)}$. It follows from the raster property that $\text{idx}(i, j)$ is minimized with $i = j = \min J_{\text{p}(u)}$.
- $\max\{J^{(2)}_u \backslash J^{(2)}_{\text{p}(u)}\} = \text{idx}(\beta, \gamma)$ where $\beta = \max\{J_u\}$ and $\gamma = \max\{J_u \backslash J_{\text{p}(u)}\}$. We partition J_u into $N_u = J_u \backslash J_{\text{p}(u)}$ and $A_u = J_u \cap J_{\text{p}(u)}$.
  - For any $\text{idx}(i, j) \in J^{(2)}_u \backslash J^{(2)}_{p(u)}$ where $i \geq j$, we can have one of the following three cases: 1) $i \in N_u$ and $j \in A_u$; 2) $i \in A_u$ and $j \in N_u$; or 3) $i \in N_u$ and $j \in N_u$.
  - We observe that the first case $i \in N_u$ and $j \in A_u$ is impossible. Indeed, applying $j \geq \min\{J_{\text{p}(u)}\} > \max\{J_u \backslash J_{\text{p}(u)}\} \geq i$ would yield a contradiction with $i \geq j$.
  - Taking the union of the two remaining cases yields $i \in J_u = A_u \cup N_u$ and $j \in N_u$. It follows from the raster property that $\text{idx}(i, j)$ is maximized with $i = \max J_u$ and $j = \max N_u$.

With the two claims established, the hypothesis that $\min\{J_{\text{p}(u)}\} > \max\{J_u \backslash J_{\text{p}(u)}\}$ implies that $\alpha > \gamma$, and therefore $\text{idx}(\alpha, \alpha) > \text{idx}(\beta, \gamma)$ as desired. □

In reverse, a sequence of subsets $J^{(2)}_1, J^{(2)}_2, \ldots, J^{(2)}_\ell$ that satisfy the sorted running intersection property immediately give rise to a zero-fill sparsity pattern $F^{(2)} = \bigcup_{k=1}^{\ell} \text{clique}(J^{(2)}_k)$ with $\omega(F^{(2)}) = \max_j |J^{(2)}_j|$.

Proposition 4.9 (RIP $\Longrightarrow$ ZF) *Let $J_1, J_2, \ldots, J_\ell$ with $\bigcup_{j=1}^{\ell} J_j = \{1, 2, \ldots, n\}$ satisfy the sorted running intersection property. Then, $F = \bigcup_{j=1}^{\ell} \text{clique}(J_j)$ is zero-fill, and we have $\omega(F) = \max_j |J_j|$.*

Our proof of Proposition 4.9 relies on the following result, which says that every column of F is contain in a subset J_w.

Lemma 4.10 *Let $J_1, J_2, \ldots, J_\ell$ with $\bigcup_{j=1}^{\ell} J_j = \{1, 2, \ldots, n\}$ satisfy the sorted running intersection property. For every j-th column in $F = \bigcup_{j=1}^{\ell} \text{clique}(J_j)$, there exists some J_w such that $\text{col}_F(j) \subseteq J_w$.*

Proof Let $V = \{1, 2, \ldots, n\}$. For arbitrary $j \in V$, denote J_w as the last subset in the sequence $J_1, J_2, \ldots, J_\ell$ for which $j \in J_w$. This choice must exist, because $V = \bigcup_{k=1}^{\ell} J_k$. For every arbitrary $i \in \text{col}_F(j)$, we will prove that $i \in J_w$ also holds:

- There exists $u \leq w$ for which $i, j \in J_u$. Indeed, $(i, j) \in F$ and $F = \bigcup_{k=1}^{\ell} \text{clique}(J_k)$ imply $(i, j) \in \text{clique}(J_u)$ for some J_u, or equivalently $i, j \in J_u$. Given that $j \in J_u$ and $w = \max\{k : j \in J_k\}$ by definition, it follows that $u \leq w$.
- If $u = w$, then $i \in J_w$ holds because $i, j \in J_u$. Otherwise, if $u < w$, we use the sorted running intersection property to assert the following

$$u < w, \qquad i, j \in J_u \quad \Longrightarrow \quad i, j \in J_{\text{p}(u)}. \tag{7}$$

  The fact that $j \in J_{\text{p}(u)}$ follows directly the running intersection property $j \in J_u \cap J_w \subseteq J_{\text{p}(u)}$ for $w > u$. By contradiction, suppose that $i \notin J_{\text{p}(u)}$. Then,

given that $i \in J_u$, it follows from the sorted property that $j \geq \min\{J_{\mathrm{p}(u)}\} > \max\{J_u \backslash J_{\mathrm{p}(u)}\} \geq i$, which contradicts our initial hypothesis that $i > j$.
– Inductively reapplying (7), as in $i, j \in J_{\mathrm{p}(\mathrm{p}(u))}$ and $i, j \in J_{\mathrm{p}(\mathrm{p}(\mathrm{p}(u)))}$, we arrive at some v such that $i, j \in J_{\mathrm{p}(v)}$ and $p(v) = w$. The induction must terminate with $p(v) \geq w$ because each $\mathrm{p}(u) > u$ by construction. It is impossible for $\mathrm{p}(v) > w$ to occur, because $j \in J_{\mathrm{p}(v)}$ and $w = \max\{k : j \in J_k\}$ by definition. We conclude that $i \in J_w$, as desired. □

The equivalence between the sorted running intersection property and zero-fill sparsity pattern then follow as a short corollary of the above.

Proof of Proposition 4.9 To prove that F is zero-fill, we observe, for arbitrary $(i, k) \in F$ and $(j, k) \in F$ with $i > j > k$ that $i, j \in \mathrm{col}_F(k)$. Therefore, it follows from Lemma 4.10 that there exists J_w such that $i, j, k \in J_w$. We conclude that $(i, j) \in F$, because $F = \bigcup_{j=1}^{\ell} \mathrm{clique}(J_j)$ and $i, j \in J_w$ for some $1 \leq w \leq \ell$.

To prove that $\omega(F) = \max_j |J_j|$, we choose $k = \arg\max_j |\mathrm{col}_F(j)|$. It follows from Lemma 4.10 that there exists J_w such that $\mathrm{col}_F(k) \subseteq J_w$, and therefore $\omega(F) = |\mathrm{col}_F(k)| \leq |J_w|$. Finally, given that $\mathrm{clique}(J_w) \subseteq F$ it follows that $(i, j) \in F$ holds for all $i \in J_w$ where $j = \min J_w$. Therefore, we conclude that $J_w \subseteq \mathrm{col}_F(j)$, and therefore $|J_w| \leq |\mathrm{col}_F(j)| \leq |\mathrm{col}_F(k)| = \omega(F)$. □

Finally, we conclude the proof of Lemma 4.2 by verifying that a matrix like $\mathbf{H} = \sum_{k=1}^{n} \mathbf{P}_k \mathbf{D}_k \mathbf{P}_k^T$ does indeed have $F^{(2)} = \bigcup_{k=1}^{n} \mathrm{clique}(J_k^{(2)})$ as its sparsity pattern.

Proof of Lemma 4.2 For an arbitrary sparsity pattern E of order n, let $F = \mathrm{chol}(E)$ and $J_k \equiv \mathrm{col}_F(k)$. First, it follows from Proposition 4.6 that $J_1, J_2, \ldots, J_n$ satisfy the ordered running intersection property. Therefore, it follows from Lemma 4.8, $J_1^{(2)}, J_2^{(2)}, \ldots, J_n^{(2)}$ defined in (6) satisfy the same property. Finally, we verify that

$$\begin{aligned}
\mathrm{supp}(\mathbf{P}_k) \equiv \{\alpha : \mathbf{P}_k[\alpha, \beta] \neq 0\} &= \left\{\alpha : \mathbf{P}_k^T e_\alpha \neq 0\right\} \\
&\overset{(a)}{=} \left\{\mathrm{idx}_F(i, j) : \mathbf{P}_k^T \, \mathrm{svec}_F(e_i e_j^T) \neq 0, \ (i, j) \in F\right\} \\
&\overset{(b)}{=} \left\{\mathrm{idx}_F(i, j) : (e_i e_j^T)[J_k, J_k] \neq 0, \ i \geq j\right\} = J_k^{(2)}.
\end{aligned}$$

Equality (a) is obtained by substituting $e_\alpha = \mathrm{svec}_F(e_i e_j^T)$ for $(i, j) \in F$. Equality (b) follows the identity $\mathbf{P}_k^T \, \mathrm{svec}_F(Y) = \mathrm{svec}(Y[J_k, J_k])$, which was used to define $\mathbf{P}_k$. Therefore, $F^{(2)} = \bigcup_{k=1}^{n} J_k^{(2)}$ and we conclude via Proposition 4.9 that $\omega(F^{(2)}) = \max_k |J_k^{(2)}| = \max_k \frac{1}{2}|J_k|(|J_k| + 1)$, and we recall that $\max_k |J_k| = \omega(F) = \omega(E)$ by definition. □

4.2 Sparsity overestimate (Proof of Lemma 4.3)

We will need some additional notation. For $V = \{1, 2, \ldots, n\}$ and $J \subseteq V$, we denote the subset of J induced by the elements in $U = \{u_1, u_2, \ldots, u_p\} \subseteq V$ as follows

$$J[U] \equiv \{i : u_i \in J\} \text{ where } u_1 < u_2 < \cdots < u_p.$$

Our definition is made so that if $E = \text{clique}(J)$, then $E[U] = \text{clique}(J[U])$. Our desired claim, namely that

$$\begin{aligned} E^{(2)} &= \left(\bigcup_{i=1}^{m} \text{clique}(\text{supp}(a_i)) \right) \cup \left(\bigcup_{j=1}^{n} \text{clique}(\text{supp}(\mathbf{P}_j)) \right) \\ &\subseteq \left(\bigcup_{k=1}^{n} \text{clique}(\text{supp}(\overline{\mathbf{P}}_k)[V^{(2)}]) \right) = \overline{F}^{(2)}[V^{(2)}] \end{aligned}$$

where $V^{(2)} = \{\text{idx}_{\overline{F}}(i, j) : i, j \in F\}$, now follows immediately from the following two lemmas.

Lemma 4.11 *For every $i \in \{1, 2, \ldots, m\}$, there exists $k \in \{1, 2, \ldots, n\}$ such that $\text{supp}(a_i) \subseteq \text{supp}(\overline{\mathbf{P}}_k)[V^{(2)}]$ where $V^{(2)} = \{\text{idx}_{\overline{F}}(i, j) : i, j \in F\}$.*

Proof It follows by repeating the proof of Lemma 4.2 that

$$\begin{aligned} \text{supp}(\overline{\mathbf{P}}_k) &= \{\text{idx}_{\overline{F}}(i, j) : i, j \in \text{col}_{\overline{F}}(k), i \geq j\}, \\ \text{supp}(\overline{\mathbf{P}}_k)[V^{(2)}] &= \{\text{idx}_{F}(i, j) : i, j \in \text{col}_{\overline{F}}(k), i \geq j\}. \end{aligned}$$

Our desired claim follows via the following sequence of inclusions

$$\begin{aligned} \text{supp}(a_i) &= \{v : a_i[v] \neq 0\} = \{\text{idx}_F(u, v) : A_i[u, v] \neq 0, \quad (u, v) \in F\} \\ &\overset{\text{(a)}}{\subseteq} \{\text{idx}_F(u, v) : u, v \in \text{supp}(A_i), \quad (u, v) \in F\} \\ &\overset{\text{(b)}}{\subseteq} \{\text{idx}_F(u, v) : u, v \in \text{col}_{\overline{F}}(k), \quad (u, v) \in F\} = \text{supp}(\overline{\mathbf{P}}_k)[V^{(2)}]. \end{aligned}$$

Inclusion (a) is because $\text{spar}(A_i) \subseteq \text{clique}(\text{supp}(A_i))$. Inclusion (b) is true via the following: If $J = \text{supp}(A_i)$ satisfies $\text{clique}(J) \subseteq \overline{F}$, then $J \subseteq \text{col}_{\overline{F}}(k)$ where $k = \min J$. Indeed, we have $k \in \text{col}_{\overline{F}}(k)$ by definition. For any arbitrary $j \in J$ with $j > k$, we must have an edge $(j, k) \in \text{clique}(J) \subseteq \overline{F}$, and therefore $j \in \text{col}_{\overline{F}}(k)$. □

Lemma 4.12 *For every $k \in \{1, 2, \ldots, n\}$, we have $\text{supp}(\mathbf{P}_k) \subseteq \text{supp}(\overline{\mathbf{P}}_k)[V^{(2)}]$ where $V^{(2)} = \{\text{idx}_{\overline{F}}(i, j) : i, j \in F\}$.*

Proof It again follows by repeating the proof of Lemma 4.2 that

$$\begin{aligned} \text{supp}(\mathbf{P}_k) &= \{\text{idx}_F(i, j) : i, j \in \text{col}_F(k), i \geq j\} \\ &\overset{\text{(a)}}{\subseteq} \{\text{idx}_F(i, j) : i, j \in \text{col}_{\overline{F}}(k), i \geq j\} = \text{supp}(\overline{\mathbf{P}}_k)[V^{(2)}]. \end{aligned}$$

Inclusion (a) is true because $E \subseteq \overline{E}$ implies $F = \text{chol}(E) \subseteq \text{chol}(\overline{E}) = \overline{F}$ via Proposition 4.1, and therefore $\text{col}_F(k) \subseteq \text{col}_{\overline{F}}(k)$. □

5 Large-scale numerical experiments

Our goal in this section is to provide experimental evidence to justify the following four empirical claims made in the paper:

1. Whenever a graph has small treewidth $\tau_\star = O(1)$, a fill-reducing heuristic is also able to find a "good enough" tree decomposition with width $\tau = O(1)$.
2. A primal-dual interior-point method consistently converges to high accuracies of $\epsilon \approx 10^{-6}$ in just tens of iterations, at an essentially dimension-free rate.
3. In theory, clique-tree conversion (CTC) enjoys similar guarantees to chordal conversion (CC). But in practice, (CC) is *much* faster than (CTC), both in solution time and in preprocessing time.
4. Real-world power systems $\mathcal{G}$ yield instances of the AC optimal power flow relaxation (Example 1.6) with small values of $\mathrm{tw}(\overline{G}) = 2 \cdot \mathrm{tw}(\mathcal{G}^2)$.

To this end, we benchmark the following three conversion methods:

- **CC:** Chordal conversion as outlined in this paper in Algorithm 1, implemented in MATLAB, with MOSEK [32] as the general-purpose solver. If $G = \overline{G}$ has small treewidth, then CC is guaranteed to use at most $O(m + n)$ time per-iteration via Theorem 3.3.
- **Dual CTC (heuristic):** The dualized variant of clique-tree conversion of Zhang and Lavaei [46] based on the aggregate sparsity graph G. We take MATLAB / MOSEK implementation directly from the project website[4]. If $G = \overline{G}$ has small treewidth, then this variant is guaranteed to use at most $O(m+n)$ time per-iteration via [46, Theorem 1]. When $G \neq \overline{G}$, this variant reduces to an empirical heuristic.
- **Dual CTC (provable):** The dualized variant of clique-tree conversion of Zhang and Lavaei [46], but forced to use the extended sparsity graph $\overline{G}$ instead of G, as suggested by Gu and Song [21]. It is implemented by padding the elements of the sparse cost matrix C with numerically-zero elements that are structurally nonzero, so that it is recognized as an element of $\mathbb{S}^n_{\overline{E}}$, and then calling Dual CTC (heuristic). If $\overline{G}$ has small treewidth, then this variant is guaranteed to use at most $O(m + n)$ time per-iteration via [46, Theorem 1].

All of our experiments were conducted on a modest workstation, with a Xeon 3.3 GHz quad-core CPU and 32 GB of RAM. Our code was written in MATLAB 9.8.0.1323502 (R2020a), and the general-purpose solver we use is MOSEK v9.1.10 [32]. MOSEK specifies default parameters $\epsilon = 10^{-8}$ and seeks to terminate with $\|\mathbf{A}x - \mathbf{b}\|_\infty \leq \epsilon(1 + \|\mathbf{b}\|_\infty)$ and $\|\mathbf{A}^T y + s - \mathbf{c}\|_\infty \leq \epsilon(1 + \|\mathbf{c}\|_\infty)$ and $\max\{x^T s, \mathbf{c}^T x - \mathbf{b}^T y\} \leq \epsilon \max\{1, |\mathbf{b}^T x|, |\mathbf{c}^T x|\}$ [32, Section 13.3.2]. If MOSEK is unable to achieve this accuracy due to numerical issues, it gives up and accepts the solution as optimal if $\epsilon = 10^{-5}$ [32, Section 13.3.3]. Our calculation for the number of accurate digits is identical to [46, Section 8], which was in turn adapted from the DIMACS challenge [36].

[4] https://github.com/ryz-codes/dual_ctc.

Table 1 Tree decomposition quality, accuracy (in decimal digits) and timing (in seconds) for Lovász theta problems on partial k-trees: $|\mathcal{V}|$—number of vertices; $|\mathcal{E}|$—number of edges; τ—width of tree decomposition used; "prep"—preprocessing time, which includes the conversion process and MOSEK's internal preparation time; "digit"—accurate decimal digits; "iter"—number of interior-point iterations; "per-it"—average time per interior-point iteration

$\lvert\mathcal{V}\rvert$	$\lvert\mathcal{E}\rvert$	Dual CTC (k-tree ordering)					CC (amd ordering)					CC (k-tree ordering)				
		τ	prep	digit	iter	per-it	τ	prep	digit	iter	per-it	τ	prep	digit	iter	per-it
100	126	16	0.05	7.0	8	0.04	9	0.02	6.8	12	0.01	16	0.15	7.9	10	0.01
200	262	23	0.10	8.3	9	0.06	16	0.01	7.0	10	0.01	23	0.03	7.9	10	0.03
500	684	33	0.28	8.8	9	0.12	30	0.03	6.7	9	0.06	33	0.07	7.1	7	0.10
1000	1492	35	0.53	9.1	9	0.17	33	0.05	6.6	8	0.12	35	0.05	8.7	8	0.14
2000	3004	35	0.85	7.3	11	0.14	35	0.09	6.5	8	0.15	35	0.09	6.8	8	0.15
5000	7441	35	2.64	8.6	11	0.63	40	0.23	7.5	9	0.37	35	0.24	6.2	8	0.35
10,000	14,977	35	6.73	7.3	15	1.60	36	0.48	6.7	8	0.81	35	0.47	8.0	9	0.67
20,000	30,032	35	22.21	7.6	13	3.41	42	1.01	7.3	8	1.48	35	1.04	7.9	9	1.31
50,000	75,042	35	114.21	3.8	20	9.13	44	2.91	5.1	8	3.81	35	2.79	6.7	10	3.27
100,000	150,298	35	415.89	4.5	27	18.57	40	5.98	7.8	9	8.20	35	5.93	5.5	10	6.65
200,000	299,407	35	1966.70	3.8	16	36.89	42	11.90	6.1	15	27.65	35	12.06	6.3	11	12.92
500,000	749,285	35	12265.85	3.3	16	102.56	46	31.80	2.8	18	103.85	35	41.41	5.6	16	39.20
1,000,000	1,500,873	35	–	–	–	–	46	65.39	4.4	18	414.80	35	67.05	3.3	17	97.57

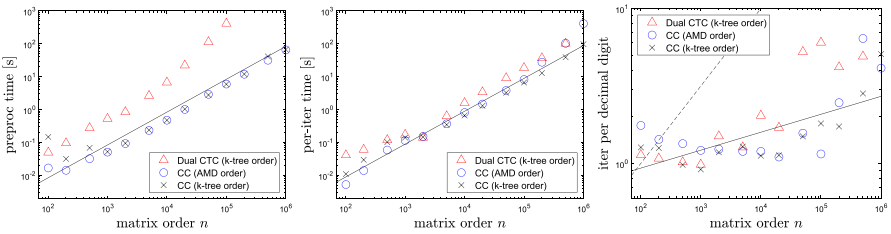

Fig. 1 Lovász theta problems solved via chordal conversion: **a** Preprocessing time, with regression $p_\times(n) = 8.385n \times 10^{-5}$ and $R^2 = 0.87$; **b** Time per iteration, with regression $f_\times(n) = 8.9n \times 10^{-5}$ and $R^2 = 0.98$; **c** Iterations per decimal digit of accuracy, with (solid) regression $g(n) = 0.548n^{0.115}$ and $R^2 = 0.45$ and (dashed) bound $g(n) = 0.1\sqrt{n}$

5.1 Lovász theta problem on synthetic dataset

Our first set of experiments is on the Lovász theta problem (Example 1.3), for which $G = \overline{G}$ always holds with equality. For each trial, we set $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ by randomly generating a k-tree with $k = 35$ (see [44, p. 9] for details) and then deleting edges uniformly at random until $|\mathcal{E}|/|\mathcal{V}| \approx 3/2$. The resulting $\mathcal{G}$ should have treewidth exactly $\tau_\star = 35$ in the limit $|\mathcal{V}| \to \infty$; the optimal ordering Π to yield $\omega(E_\Pi) = 36$ is simply any perfect elimination ordering on the k-tree ("k-tree ordering" in Table 1). We observe that the `amd` heuristic in MATLAB [1] finds high-quality orderings to yield $\omega(E_\Pi) \leq 47$, corresponding to tree decompositions of width $\tau \leq 46$, which is only about 30% worse than the best possible ("amd ordering" in Table 1). Nevertheless, minor differences in τ can still manifest as larger differences in per-iteration time.

We solve the problem on $\mathcal{G}$ using CC and CTC, and observe that in both cases, it takes around 10 iterations to achieve $\epsilon \approx 10^{-6}$ across a wide range of n, until numerical issues at very large scales $n \approx 10^5$ forced more iterations to be taken (see Fig. 1c and Table 1). We find that both CC and CTC achieve comparable $O(m + n)$ runtime per-iteration (see Fig. 1b), but CC is significantly faster in its preprocessing time (see Fig. 1a). As shown in the last few rows of Table 1, CC solved an instance of the Lovász theta problem on a graph with 10^6 vertices and 1.5×10^6 edges in less than 30 min, taking a little over 1 min in the preprocessing. In contrast, CTC for a graph of half this size took 3.5 h just to perform the preprocessing.

To test the zero fill-in prediction in Theorem 3.1, our implementation of CC in this section forces MOSEK to factor its Schur complement matrix $\mathbf{H} = \mathbf{A}\nabla^2 f(w)\mathbf{A}^T$ *without* the use of a fill-reducing ordering, by setting the flag `MSK_IPAR_INTPNT_ORDER_METHOD` to `'MSK_ORDER_METHOD_NONE'`. If Theorem 3.1 is incorrect, then factoring $\mathbf{L} = \text{chol}(\mathbf{H})$ would catastrophic dense fill-in, and the per-iteration runtime would not be $O(m + n)$ as shown in Fig. 1b.

5.2 AC optimal power flow relaxation on real-world dataset

Our second set of experiments is on the AC optimal power flow relaxation (Example 1.6), for which $G \subset \overline{G}$ generally holds with strict inequality. Here, recall that

Table 2 Treewidth bounds for the 72 power system test cases in the MATPOWER dataset: $|V|$—number of vertices; $|E|$—number of edges; lb, ub—lower-bound and upper-bounds on $\mathrm{tw}(\mathcal{G})$; lb_2, ub_2—lower-bound and upper-bounds on $\mathrm{tw}(\mathcal{G}^2)$

#	Name	$\lvert V\rvert$	$\lvert E\rvert$	lb	ub	lb_2	ub_2	#	Name	$\lvert V\rvert$	$\lvert E\rvert$	lb	ub	lb_2	ub_2
1	case4_dist	4	3	1	1	2	2	37	case85	85	84	1	1	4	4
2	case4gs	4	4	2	2	3	3	38	case89pegase	89	206	8	11	17	27
3	case5	5	6	2	2	4	4	39	case94pi	94	93	1	1	4	4
4	case6ww	6	11	3	3	5	5	40	case118zh	118	117	1	1	4	4
5	case9target	9	9	2	2	4	4	41	case118	118	179	4	4	9	12
6	case9Q	9	9	2	2	4	4	42	case136ma	136	135	1	1	8	8
7	case9	9	9	2	2	4	4	43	case141	141	140	1	1	4	4
8	case10ba	10	9	1	1	2	2	44	case145	145	422	7	10	21	33
9	case12da	12	11	1	1	2	2	45	case_ACTIVSg200	200	245	4	8	11	18
10	case14	14	20	2	2	6	6	46	case300	300	409	3	6	11	17
11	case15nbr	15	14	1	1	4	4	47	case_ACTIVSg500	500	584	4	8	14	22
12	case15da	15	14	1	1	4	4	48	case1354pegase	1354	1710	5	12	13	30
13	case16am	15	14	1	1	4	4	49	case1888rte	1888	2308	4	12	14	38
14	case16ci	16	13	1	1	3	3	50	case1951rte	1951	2375	5	12	15	43
15	case17me	17	16	1	1	3	3	51	case_ACTIVSg2000	2000	2667	6	40	16	85
16	case18nbr	18	17	1	1	4	4	52	case2383wp	2383	2886	5	23	9	51
17	case18	18	17	1	1	3	3	53	case2736sp	2736	3263	5	23	9	55

Table 2 continued

#	Name	$\|V\|$	$\|E\|$	lb	ub	lb_2	ub_2	#	Name	$\|V\|$	$\|E\|$	lb	ub	lb_2	ub_2
18	case22	22	21	1	1	3	3	54	case2737sop	2737	3263	5	23	9	57
19	case24_ieee_rts	24	34	3	4	7	8	55	case2746wop	2746	3299	5	23	9	53
20	case28da	28	27	1	1	3	3	56	case2746wp	2746	3273	5	24	9	58
21	case30pwl	30	41	3	3	7	9	57	case2848rte	2848	3442	5	18	14	41
22	case30Q	30	41	3	3	7	9	58	case2868rte	2868	3471	5	17	16	43
23	case30	30	41	3	3	7	9	59	case2869pegase	2869	3968	9	12	17	42
24	case_ieee30	30	41	3	3	7	9	60	case3012wp	3012	3566	5	25	10	55
25	case33bw	33	32	1	1	3	3	61	case3120sp	3120	3684	5	28	9	60
26	case33mg	33	32	1	1	3	3	62	case3375wp	3374	4068	6	27	12	64
27	case34sa	34	33	1	1	3	3	63	case6468rte	6468	8065	5	26	15	65
28	case38si	38	37	1	1	3	3	64	case6470rte	6470	8066	5	26	15	64
29	case39	39	46	3	3	5	7	65	case6495rte	6495	8084	5	26	15	62
30	case51ga	51	50	1	1	3	3	66	case6515rte	6515	8104	5	26	16	62
31	case51he	51	50	1	1	3	3	67	case9241pegase	9241	14,207	21	33	42	78
32	case57	57	78	3	5	6	12	68	case_ACTIVSg10k	10,000	12,217	5	33	17	80
33	case69	69	68	1	1	4	4	69	case13659pegase	13,659	18,625	21	31	42	80
34	case70da	70	68	1	1	3	3	70	case_ACTIVSg25k	25,000	30,110	6	51	17	127
35	case_RTS_GMLC	73	108	4	5	7	11	71	case_ACTIVSg70k	70,000	83,318	6	88	16	232
36	case74ds	74	73	1	1	3	3	72	case_SyntheticUSA	82,000	98,203	6	90	17	242

Table 3 Accuracy (in decimal digits) and timing (in seconds) for 25 largest OPF problems: n—order of matrix variable; "digit"—accurate decimal digits; "prep"—all preprocessing time before interior-point, including the conversion process and MOSEK's internal preparation time; "iter"—number of interior-point iterations; "per-it"—interior-point time per iteration; "post" - post-processing time after interior-point, and to recover U satisfying $\text{proj}_F(UU^T) = Y$ via [39, Algorithm 2]

#	n	m	Dual CTC (provable)				Dual CTC (heuristic)				CC (provable)				post
			prep	digit	iter	per-it	prep	digit	iter	per-it	prep	digit	iter	per-it	
48	1354	4060	2.14	8.4	44	0.84	1.49	8.4	42	0.12	0.32	8.3	47	0.05	0.24
49	1888	5662	3.40	7.8	49	1.98	2.15	7.8	53	0.15	0.41	8.0	58	0.07	0.33
50	1951	5851	3.78	7.9	46	3.17	2.21	7.7	52	0.15	0.43	8.0	59	0.07	0.33
51	2000	5998	34.68	8.1	32	122.93	4.20	8.3	32	2.54	1.89	7.9	28	1.23	0.98
52	2383	7147	7.97	7.6	47	14.46	2.41	7.7	41	0.32	0.81	7.7	44	0.24	0.62
53	2736	8206	9.33	7.1	57	20.59	2.83	7.8	58	0.36	0.86	7.9	57	0.28	0.77
54	2737	8209	12.31	6.7	61	17.16	2.87	7.4	61	0.41	0.85	8.4	56	0.31	0.76
55	2746	8236	9.14	7.2	50	18.34	4.21	7.5	51	0.58	0.92	8.4	52	0.32	0.71
56	2746	8236	10.98	7.2	60	19.83	4.07	7.2	54	0.66	0.93	7.8	59	0.38	1.17
57	2848	8542	7.52	7.5	54	5.25	3.62	7.9	60	0.26	0.69	6.9	58	0.18	0.57
58	2868	8602	5.81	7.6	54	6.63	3.64	7.8	60	0.23	0.70	7.8	55	0.15	0.56
59	2869	8605	9.62	8.0	47	7.36	2.98	8.0	47	0.23	0.81	8.0	46	0.20	0.54
60	3012	9034	9.85	7.7	50	20.65	3.22	7.7	49	0.49	1.07	8.0	52	0.37	0.95
61	3120	9358	11.57	7.1	62	21.13	3.43	7.6	61	0.54	1.12	8.2	66	0.42	1.16
62	3374	10,120	12.11	7.3	57	24.02	3.85	7.7	55	0.80	1.25	8.0	54	0.52	1.12
63	6468	19,402	21.38	7.7	65	43.95	10.75	8.2	61	1.17	1.83	7.9	63	0.73	2.37
64	6470	19,408	21.28	7.9	61	40.90	8.25	8.1	59	0.79	1.92	7.8	60	0.74	2.18
65	6495	19,483	20.48	7.7	68	42.03	7.95	8.1	67	0.88	1.92	8.1	60	0.64	2.27
66	6515	19,543	20.48	7.6	65	44.33	11.07	8.0	61	1.16	1.82	7.9	60	0.62	2.16
67	9241	27,721	48.58	7.8	67	120.46	14.43	7.9	54	3.02	3.69	7.8	63	1.68	3.90

Table 3 continued

#	n	m	Dual CTC (provable)				Dual CTC (heuristic)				CC (provable)				post
			prep	digit	iter	per-it	prep	digit	iter	per-it	prep	digit	iter	per-it	
68	10,000	29,998	55.69	8.0	49	188.19	19.83	8.2	41	3.63	4.58	8.1	42	2.56	4.68
69	13,659	40,975	56.28	7.2	57	126.93	28.78	7.7	49	3.07	4.34	7.8	50	1.84	6.37
70	25,000	74,998	–	–	–	–	69.16	7.4	114	25.05	19.36	7.7	118	14.73	25.06
71	70,000	209,998	–	–	–	–	–	–	–	–	96.85	7.9	65	180.82	113.19
72	82,000	245,994	–	–	–	–	–	–	–	–	99.38	8.0	68	197.34	140.42

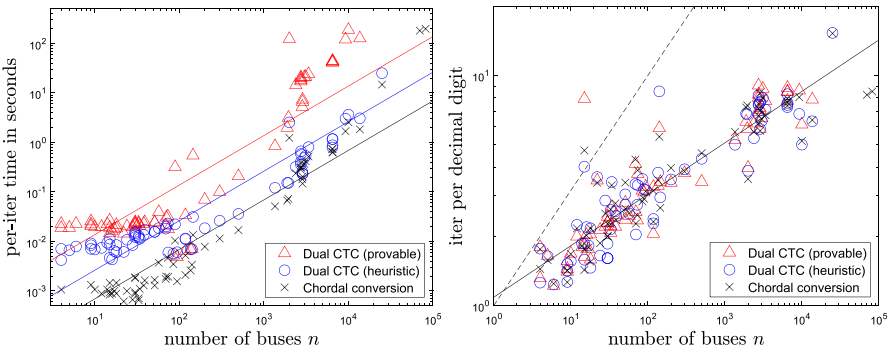

Fig. 2 AC optimal power flow relaxation solved via chordal conversion (×), heuristic Dual CTC (○), and provable Dual CTC (△): **a** Time per iteration, with regression lines of $f_{\times}(n) = 6.908n \times 10^{-5}$ with $R^2 = 0.90$, $f_{\circ}(n) = 2.623n \times 10^{-4}$ with $R^2 = 0.83$, and $f_{\triangle}(n) = 1.375n \times 10^{-3}$ with $R^2 = 0.80$; **b** Iterations per decimal digit of accuracy, with (solid) regression $g(n) = 1.088n^{0.224}$ with $R^2 = 0.84$ and (dashed) bound $g(n) = \sqrt{n}$

$\text{tw}(G) = 2 \cdot \text{tw}(\mathcal{G})$ and $\text{tw}(\overline{G}) = 2 \cdot \text{tw}(\mathcal{G}^2)$, where $\mathcal{G}$ is the graph of the underlying electric grid, and $\mathcal{G}^2$ is its square graph. For the 72 power system graphs taken from the MATPOWER suite [48], we compute upper- and lower-bounds on $\text{tw}(\mathcal{G})$ and $\text{tw}(\mathcal{G}^2)$ using the "FillIn" and the "MMD+" heuristics outlined in [30], and find that $\text{tw}(\mathcal{G}^2)$ is small in all 72 power system graphs (see Table 2).

We solve the problem using CC and the two variants of CTC, and verify that the per-iteration costs are linear (see Fig. 2a). In all three cases, it takes a consistent 50–70 iterations to achieve $\epsilon \approx 10^{-6}$, again until numerical issues at very large scales $n \approx 10^4$ forced more iterations to be taken (see Table 3 and Fig. 2b). For these smaller large-scale problems with $G \neq \overline{G}$, we find that CC and CTC had comparable processing times, but CC is between 2 and 100 times faster than either variant of CTC in its solution time. The largest case is `case_SyntheticUSA` with 82000 buses (due to [5]), which we solve with CC in 4 h. Previously, this was solved using CTC in 8 h on a high-performance computing (HPC) node with two Intel XeonE5-2650v4 processors (a total of 24 cores) and 240 GB memory [13].

6 Conclusions and future directions

Chordal conversion can sometimes allow an interior-point method to solve a large-scale sparse SDP in just $O(m + n)$ time per-iteration. Previously, a well-known necessary condition is that the aggregate sparsity graph $G = (V, E)$ should have an $O(1)$ treewidth (independent of m and n):

$$V = \{1, 2, \ldots, n\}, \quad E = \text{spar}(C) \cup \text{spar}(A_1) \cup \cdots \cup \text{spar}(A_m),$$
$$\text{where } \text{spar}(C) \equiv \{(i, j) : C[i, j] \neq 0 \text{ for } i > j\}.$$

In this paper, provide a companion sufficient condition, namely that the *extended* aggregate sparsity graph $\overline{G} = (V, \overline{E})$ should also have an $O(1)$ treewidth:

$$V = \{1, 2, \ldots, n\}, \quad \overline{E} = \text{spar}(C) \cup \text{clique}(A_1) \cup \cdots \cup \text{clique}(A_m)$$
$$\text{where clique}(A) = \{(i, j) : A[i, k] \neq 0 \text{ or } A[k, j] \neq 0 \text{ for some } k\}.$$

The key to our analysis is to characterize the Schur complement sparsity $E^{(2)}$, the sparsity pattern of the linear equations solved at each iteration, directly in terms of $\overline{E}$.

Our primary focus has been on reducing the per-iteration costs to $O(m+n)$. Naively applying this figure to the $O(\sqrt{m+n}\log(1/\epsilon))$ iterations of a general-purpose interior-point method results in an end-to-end complexity of $O((m+n)^{1.5}\log(1/\epsilon))$ time. By adopting the treewidth-based interior-point method of Dong et al. [12, Theorem 1.3], as was done in the recent preprint of Gu and Song [21], it should be possible to formally reduce the end-to-end complexity down to $O((m+n)\log(1/\epsilon))$ time. However, we mention that interior-point methods in practice often converge to ϵ accuracy in dimension-free $O(\log(1/\epsilon))$ iterations (without the square-root factor), and as such the empirical complexity is already $O((m+n)\log(1/\epsilon))$ time.

In many applications, G and $\overline{G}$ either coincide or are very close, so our analysis becomes either exact or nearly exact; an $O(1)$ treewidth in $\overline{G}$ is both necessary and sufficient for chordal conversion to be fast. In cases where G and $\overline{G}$ are very different, particularly when the treewidth of $\overline{G}$ is $\Omega(n)$ while the treewidth of G is $O(1)$, our preliminary simulations suggest that the per-iteration cost could slow down $\Omega(n^3)$ time, but more work is needed to establish this rigorously. Finally, even where interior-point methods are no longer efficient, it may still be possible to use chordal conversion to improve the efficiency of first-order methods and/or nonconvex approaches.

Acknowledgements I am grateful to Martin S. Andersen for numerous insightful discussions, and for his early numerical experiments that motivated much of the subsequent theoretical analysis in this paper. The paper has also benefited significantly from discussions with Subhonmesh Bose, Salar Fattahi, Alejandro Dominguez–Garcia, Cedric Josz, and Sabrina Zielinski. I thank the associate editor and two reviewers for helpful comments that significantly sharpened the presentation of the paper.

Declarations

Conflict of interest The author has no relevant financial or non-financial interests to disclose.

A Proof of the standard-form assumptions

Given data $C, A_1, \ldots, A_m \in \mathbb{S}^n$ and $b \in \mathbb{R}^m$, define $(\mathbf{A}, \mathbf{b}, \mathbf{c}, \mathcal{K})$ as in (3). In this section, we verify that $(\mathbf{A}, \mathbf{b}, \mathbf{c}, \mathcal{K})$ specifies an SDP that satisfies the regularity assumptions in Definition 2.7.

Lemma A.1 (Linear independence) *We have* $\mathbf{A}^T y = 0$ *if and only* $y = 0$.

Proof We will prove $\sum_{k=1}^{n} \mathbf{P}_k \mathbf{P}_k^T \succeq I$, which implies $\mathbf{A}\mathbf{A}^T \succeq I$ and hence $\|\mathbf{A}^T y\| \geq \|y\|$. For arbitrary $Y \in \mathbb{S}_F^n$ with $y = \operatorname{svec}_F(Y)$, we observe that

$$\|\mathbf{P}_k^T y\|^2 = \|\operatorname{svec}(Y[J_k, J_k])\|^2 = \sum_{i,j \in J_k} (Y[i,j])^2 = \sum_{(i,j) \in \operatorname{clique}(J_k)} \gamma_{ij} (Y[i,j])^2$$

where $\gamma_{ij} = 1$ if $i = j$ and $\gamma_{ij} = 2$ if $i \neq j$. Therefore, we have

$$\sum_{k=1}^{n} \|\mathbf{P}_k^T y\|^2 = \sum_{k=1}^{n} \sum_{(i,j) \in \operatorname{clique}(J_k)} \gamma_{ij} (Y[i,j])^2 \geq \sum_{(i,j) \in F} \gamma_{ij} (Y[i,j])^2 = \|y\|^2.$$

The inequality is because $F = \bigcup_{k=1}^{n} \operatorname{clique}(J_k)$. □

Lemma A.2 (Strong duality is attained) *Under Assumption 1, there exists* $x^\star, s^\star \in \mathcal{K}$ *satisfying* $\mathbf{A}x^\star = \mathbf{b}$, $\mathbf{A}^T y^\star + s^\star = \mathbf{c}$, $(x^\star)^T s^\star = 0$.

Proof Define P_j implicitly to satisfy $P_j^T x = x[\operatorname{col}_F(j)]$ for all $x \in \mathbb{R}^m$. Assumption 1 says that there exists $\hat{X} \succeq 0$ and $\hat{y} \leq 0$ and that satisfy $\mathcal{A}(\hat{X}) \leq b$ and $\mathcal{A}^T(\hat{y}) + \hat{S} = C$ and $\left\langle C, \hat{X} \right\rangle = \langle b, \hat{y} \rangle$. It follows from [44, Theorem 9.2] that $\hat{S} = C - \mathcal{A}^T(\hat{y}) \in \mathbb{S}_F^n \cap \mathbb{S}_+^n$ if and only if there exists $\hat{S}_j \succeq 0$ such that $\hat{S} = \sum_{j=1}^{n} P_j \hat{S}_j P_j^T$. Now, we turn to the primal-dual pair defined by the data $(\mathbf{A}, \mathbf{b}, \mathbf{c}, \mathcal{K})$ in (3), which is written

$$\begin{aligned} &\min_{v \in \mathbb{R}^m, V_j \in \mathbb{S}^{\omega_j}} \left\{ \langle b, v \rangle : \begin{array}{c} \operatorname{proj}_F \left[\mathcal{A}^T(-v) + \sum_{j=1}^{n} P_j V_j P_j^T - C \right] = 0, \\ v \geq 0, \quad V_1, \ldots, V_n \succeq 0 \end{array} \right\} \\ &\quad = \max_{Y \in \mathbb{S}_F^n} \left\{ \langle -C, Y \rangle : \mathcal{A}(Y) \leq b, \ -P_j^T Y P_j \preceq 0 \text{ for all } j \in \{1, 2, \ldots, n\} \right\} \end{aligned}$$

We can mechanically verify that $v^\star = -\hat{y}$ and $V_j^\star = \hat{S}_j$ is feasible for the primal problem, and $Y^\star = \Pi_F(\hat{X})$ is feasible for the dual problem, and that the two objectives coincide $\langle b, v^\star \rangle = \langle -C, Y^\star \rangle$. Therefore, we conclude that $x^\star = (v^\star, \operatorname{svec}(V_1^\star), \ldots, \operatorname{svec}(V_n^\star))$ and $y^\star = \operatorname{svec}_F(Y^\star)$ is a complementary solution satisfying $\mathbf{A}^T x^\star = \mathbf{b}$, $\mathbf{A}^T y^\star + s^\star = \mathbf{c}$, and $\langle x^\star, s^\star \rangle = 0$. □

B Complexity of general-purpose interior-point methods

In this section, we state a concrete interior-point method that implements the specifications outlined in Definition 2.8, roughly following the steps in [11].

Proposition B.1 *Let* $(\mathbf{A}, \mathbf{b}, \mathbf{c}, \mathcal{K})$ *describe an SDP in* (n, ω)*-standard form, and let* opt *denote its optimal value. Let* T *and* M *denote the time and memory needed, given data* $\mathbf{A} \in \mathbb{R}^{M \times N}$, $g \in \mathbb{R}^M$, *and a choice of* $w \in \operatorname{Int}(\mathcal{K})$, *to form and solve* $\mathbf{A}\mathbf{D}\mathbf{A}^T \Delta y = g$ *for* $\Delta y \in \mathbb{R}^M$, *where* $\mathbf{D}^{-1} = \nabla^2 f(w)$ *and* $f = -\log\det(w)$. *Then, a general-purpose*

interior-point method computes (x, y) *that satisfy the following in* $O(\sqrt{n}\log(1/\epsilon))$ *iterations*

$$\mathbf{c}^T x \le \text{opt} + n\cdot\epsilon, \quad \|\mathbf{A}x - \mathbf{b}\| \le \epsilon, \quad x \in \mathcal{K},$$
$$\mathbf{b}^T y \ge \text{opt} - n\cdot\epsilon, \quad \mathbf{c} - \mathbf{A}^T y + \epsilon\mathbf{1}_{\mathcal{K}} \in \mathcal{K},$$

with per-iteration costs of $O(\omega^2 n + \text{nnz}(\mathbf{A}) + \mathsf{T})$ *time and* $O(\omega n + \text{nnz}(\mathbf{A}) + \mathsf{M})$ *memory.*

We prove Proposition B.1 by using the short-step method of Nesterov and Todd [34, Algorithm 6.1] to solve the extended homogeneous self-dual embedding

$$\min_{x,y,s,\kappa,\tau,\theta} \quad (n+1)\theta \tag{8a}$$

$$\text{s.t.} \quad \begin{bmatrix} 0 & +\mathbf{A}^T & -\mathbf{c} & -r_d \\ -\mathbf{A} & 0 & +\mathbf{b} & -r_p \\ +\mathbf{c}^T & -\mathbf{b}^T & 0 & -r_c \\ r_d^T & r_p^T & r_c & 0 \end{bmatrix} \begin{bmatrix} x \\ y \\ \tau \\ \theta \end{bmatrix} + \begin{bmatrix} s \\ 0 \\ \kappa \\ 0 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \\ n+1 \end{bmatrix} \tag{8b}$$

$$x, s \in \mathcal{K}, \quad \kappa, \tau \ge 0, \tag{8c}$$

where $r_d = \mathbf{1}_{\mathcal{K}} - \mathbf{c}$ and $r_p = -\mathbf{A}\mathbf{1}_{\mathcal{K}} + \mathbf{b}$ and $r_c = 1 + \mathbf{c}^T\mathbf{1}_{\mathcal{K}}$. Beginning at the strictly feasible, perfectly centered point in (9) for $\mu = 1$:

$$\theta^{(0)} = \tau^{(0)} = \kappa^{(0)} = 1, \quad y^{(0)} = 0, \quad x^{(0)} = s^{(0)} = \mathbf{1}_{\mathcal{K}}. \tag{9}$$

we take the following steps

$$\mu^+ = \left(1 - \frac{1}{15\sqrt{n+1}}\right)\cdot\frac{x^T s + \tau\kappa}{n+1}, \tag{10a}$$
$$(x^+, y^+, s^+, \tau^+, \theta^+, \kappa^+) = (x, y, s, \tau, \theta, \kappa) + (\Delta x, \Delta y, \Delta s, \Delta\tau, \Delta\theta, \Delta\kappa).$$

along the search direction defined by the linear system [41, Eqn. 9]

$$\begin{bmatrix} 0 & +\mathbf{A}^T & -\mathbf{c} & -r_p \\ -\mathbf{A} & 0 & +\mathbf{b} & -r_d \\ +\mathbf{c}^T & -\mathbf{b}^T & 0 & -r_c \\ +r_p^T & +r_d^T & +r_c & 0 \end{bmatrix} \begin{bmatrix} \Delta x \\ \Delta y \\ \Delta\tau \\ \Delta\theta \end{bmatrix} + \begin{bmatrix} \Delta s \\ 0 \\ \Delta\kappa \\ 0 \end{bmatrix} = 0, \tag{11a}$$

$$s + \Delta s + \nabla^2 f(w)\Delta x + \mu^+\nabla f(x) = 0, \tag{11b}$$

$$\kappa + \Delta\kappa + (\kappa/\tau)\Delta\tau - \mu^+\tau^{-1} = 0. \tag{11c}$$

where $f(w) = -\log\det w$ is the usual self-concordant barrier function, and $w \in \text{Int}(\mathcal{K})$ is the unique point that satisfies $\nabla^2 f(w)x = s$. The following is an immediate consequence of [34, Theorem 6.4].

Lemma B.2 (Short-Step Method) *The sequence in* (10) *arrives at an iterate* $(x, y, s, \tau, \theta, \kappa)$ *satisfying*

$$\frac{x^T s + \tau\kappa}{n+1} \leq \epsilon, \qquad \tau\kappa \geq \gamma\epsilon \tag{12}$$

with $\gamma = \frac{9}{10}$ *in at most* $O(\sqrt{n}\log(1/\epsilon))$ *iterations.*

Finally, the following result (adapted from [11, Lemma 5.7.2]) assures us that a feasible point satisfying the optimality condition (12) will recover a solution to the original problem.

Lemma B.3 (ϵ-accurate and ϵ-feasible) *Let there exist* $(x^\star, y^\star)$ *such that* $\mathbf{A}x^\star = \mathbf{b}$ *with* $x^\star \in \mathcal{K}$ *and* $\mathbf{c} - \mathbf{A}^T y^\star \equiv s^\star \in \mathcal{K}$ *that satisfies strong duality* $\mathbf{b}^T y^\star = \mathbf{c}^T x^\star$. *Then, a point* $(x, y, s, \tau, \theta, \kappa)$ *that is feasible for* (8) *and satisfies the optimality condition* (12) *also satisfies the following, where* $\rho = 1 + \mathbf{1}_{\mathcal{K}}^T(x^\star + s^\star)$*:*

$$\|\mathbf{A}(x/\tau) - \mathbf{b}\| \leq \frac{\rho\|r_p\|}{\gamma} \cdot \epsilon, \quad \|\mathbf{A}^T(y/\tau) + (s/\tau) - \mathbf{c}\| \leq \frac{\rho\|r_d\|}{\gamma} \cdot \epsilon, \quad \frac{(x/\tau)^T(s/\tau)}{n+1} \leq \frac{\rho^2}{\gamma^2} \cdot \epsilon.$$

Proof First, observe that $(\hat{x}, \hat{y}, \hat{s}, \hat{\tau}, \hat{\theta}, \hat{\kappa}) = (\hat{\tau}x^\star, \hat{\tau}y^\star, \hat{\tau}s^\star, \hat{\tau}, 0, 0)$ with $\hat{\tau} = (n+1)/\rho$ is a solution to (8), because $r_d^T x^\star + r_p^T y^\star + r_c = \mathbf{1}_{\mathcal{K}}^T(x^\star + s^\star) + 1 = \rho$. Next, we prove that if $(x, y, s, \tau, \theta, \kappa)$ is feasible for (8) and satisfies (12), then $\tau \geq \frac{\gamma}{\rho}$. Indeed, the skew-symmetry of (8b) yields $\theta = \frac{x^T s + \tau\kappa}{n+1}$ and $(x - \hat{x})^T(s - \hat{s}) + (\tau - \hat{\tau})(\kappa - \hat{\kappa}) = 0$. Rearranging yields $(n+1)\theta = x^T s + \tau\kappa = \hat{x}^T s + x^T\hat{s} + \tau\hat{\kappa} + \hat{\tau}\kappa$ and hence $\kappa \leq (n+1)\theta/\hat{\tau}$. Under (12), we have $\theta = \frac{x^T s + \tau\kappa}{n+1} \leq \epsilon$ and $\tau \geq \frac{\gamma\epsilon}{\kappa} \geq \frac{\gamma\theta}{\kappa} \geq \frac{\gamma\theta}{(n+1)\theta/\hat{\tau}} = \frac{\gamma}{n+1}\hat{\tau} = \frac{\gamma}{\rho}$. Finally, divide (8b) through by τ and substitute $1/\tau \leq \rho/\gamma$. □

Proof of Proposition B.1 Recall that $(\mathbf{A}, \mathbf{b}, \mathbf{c}, \mathcal{K})$ describe an SDP in (n, ω)-standard form, and hence there exist $(x^\star, y^\star)$ such that $\mathbf{A}x^\star = \mathbf{b}$ with $x^\star \in \mathcal{K}$ and $\mathbf{c} - \mathbf{A}^T y^\star \in \mathcal{K}$ that satisfies strong duality $\mathbf{b}^T y^\star = \mathbf{c}^T x^\star$. Combining Lemma B.3 and Lemma B.2 shows that iterations in (10) converges to the desired ϵ-accurate, ϵ-feasible iterate after $O(\sqrt{n}\log(1/\epsilon))$ iterations. The cost of each iteration is essentially equal to the cost of computing the search direction in (11). We account for this cost via the following two steps:

1. (Scaling point) We partition $x = [\text{svec}(X_j)]_{j=1}^\ell$ and $s = [\text{svec}(S_j)]_{j=1}^\ell$. Then, the scaling point $w = [\text{svec}(W_j)]_{j=1}^\ell$ is given in closed-form as $W_j = S_j^{1/2}(S_j^{1/2}X_j S_j^{1/2})^{-1/2}S_j^{1/2}$ [38, Section 5]. Noting that each W_j is at most $\omega \times \omega$, the cost of forming w is at most of order

$$\sum_{j=1}^\ell \omega_j^3 \leq \omega^2 \sum_{j=1}^\ell \omega_j = \omega^2 n \text{ time}, \quad \sum_{j=1}^\ell \omega_j^2 \leq \omega \sum_{j=1}^\ell \omega_j = \omega n \text{ memory}.$$

By this same calculation, it follows that the Hessian matrix–vector products $\nabla^2 f(w)x = [\mathrm{svec}(W_j^{-1} X_j W_j^{-1})]_{j=1}^{\ell}$ and $\nabla^2 f(w)^{-1}x = [\mathrm{svec}(W_j X_j W_j)]_{j=1}^{\ell}$ also cost $O(\omega^2 n)$ time and $O(\omega n)$ memory.

2. (Search direction) Using elementary but tedious linear algebra, we can show that if

$$(\mathbf{A}\mathbf{D}\mathbf{A}^T)\begin{bmatrix} v_1 & v_2 & v_3 \end{bmatrix} = \begin{bmatrix} 0 & -\mathbf{b} & r_p \end{bmatrix} - \mathbf{A}\mathbf{D}\begin{bmatrix} d & \mathbf{c} & r_d \end{bmatrix} \tag{13a}$$

where $\mathbf{D} = [\nabla^2 f(w)]^{-1}$ and $d = -s - \mu^+ \nabla f(x)$, and

$$\begin{bmatrix} u_1 & u_2 & u_3 \end{bmatrix} = \mathbf{D}(\begin{bmatrix} d & c & r_d \end{bmatrix} + \mathbf{A}^T \begin{bmatrix} v_1 & v_2 & v_3 \end{bmatrix}), \tag{13b}$$

then

$$\left(\begin{bmatrix} -\mathbf{D}_0^{-1} & -r_c \\ r_c & 0 \end{bmatrix} - \begin{bmatrix} \mathbf{c} & r_d \\ -\mathbf{b} & r_p \end{bmatrix}^T \begin{bmatrix} u_2 & u_3 \\ v_2 & v_3 \end{bmatrix}\right) \begin{bmatrix} \Delta\tau \\ \Delta\theta \end{bmatrix} = \begin{bmatrix} -d_0 \\ 0 \end{bmatrix} - \begin{bmatrix} \mathbf{c} & r_d \\ -\mathbf{b} & r_p \end{bmatrix}^T \begin{bmatrix} u_1 \\ v_1 \end{bmatrix}, \tag{13c}$$

$$\begin{bmatrix} \Delta x \\ \Delta y \end{bmatrix} = \begin{bmatrix} u_1 \\ v_1 \end{bmatrix} - \begin{bmatrix} u_1 & u_2 \\ v_1 & v_2 \end{bmatrix} \begin{bmatrix} \Delta\tau \\ \Delta\theta \end{bmatrix}, \tag{13d}$$

$$\Delta s = d - \mathbf{D}^{-1}\Delta x, \tag{13e}$$

$$\Delta\kappa = d_0 - \mathbf{D}_0^{-1}\Delta\tau, \tag{13f}$$

where $\mathbf{D}_0 = \tau/\kappa$ and $d_0 = -\kappa + \mu^+\tau^{-1}$. Therefore, we conclude that the cost of computing the search direction in (11) is equal to the cost of solving $O(1)$ instances of the Schur complement equation $\mathbf{A}\mathbf{D}\mathbf{A}^T\Delta y = g$, plus $O(1)$ matrix–vector products with $\mathbf{A}$, $\mathbf{A}^T$, $\mathbf{D}$, $\mathbf{D}^{-1}$, for a total cost of $O(\mathsf{T} + \mathrm{nnz}(\mathbf{A}) + \omega^2 n)$ time and $O(\mathsf{M} + \mathrm{nnz}(\mathbf{A}) + \omega n)$ memory respectively. Note that $\mathbf{A}\mathbf{D}\mathbf{A}^T \succ 0$ because $\mathbf{A}\mathbf{A}^T \succ 0$ by the linear independence assumption, and $\mathbf{D}^{-1} = \nabla^2 f(w) \succ 0$ for all $w \in \mathrm{Int}(\mathcal{K})$.

□

References

1. Amestoy, P.R., Davis, T.A., Duff, I.S.: Algorithm 837: AMD, an approximate minimum degree ordering algorithm. ACM Trans. Math. Softw. **30**(3), 381–388 (2004)
2. Andersen, E.D., Andersen, K.D.: The MOSEK interior point optimizer for linear programming: an implementation of the homogeneous algorithm. In: High performance optimization, pp. 197–232. Springer (2000)
3. Andersen, E.D., Roos, C., Terlaky, T.: On implementing a primal-dual interior-point method for conic quadratic optimization. Math. Program. **95**, 249–277 (2003)
4. Bai, X., Wei, H., Fujisawa, K., Wang, Y.: Semidefinite programming for optimal power flow problems. Int. J. Electr. Power Energy Syst. **30**(6–7), 383–392 (2008)
5. Birchfield, A.B., Xu, T., Gegner, K.M., Shetye, K.S., Overbye, T.J.: Grid structural characteristics as validation criteria for synthetic networks. IEEE Trans. Power Syst. **32**(4), 3258–3265 (2016)

6. Biswas, P., Ye, Y.: Semidefinite programming for ad hoc wireless sensor network localization. In: Proceedings of the 3rd international symposium on Information processing in sensor networks, pp. 46–54 (2004)
7. Borchers, B.: SDPLIB 1.2, a library of semidefinite programming test problems. Optim. Methods Softw. **11**(1–4), 683–690 (1999)
8. Boumal, N., Voroninski, V., Bandeira, A.S.: Deterministic guarantees for Burer–Monteiro factorizations of smooth semidefinite programs. Commun. Pure Appl. Math. **73**(3), 581–608 (2020)
9. Burer, S., Monteiro, R.D.: A nonlinear programming algorithm for solving semidefinite programs via low-rank factorization. Math. Program. **95**(2), 329–357 (2003)
10. Dancis, J.: Positive semidefinite completions of partial Hermitian matrices. Linear Algebra Appl. **175**, 97–114 (1992)
11. de Klerk, E., Terlaky, T., Roos, K.: Self-dual embeddings. In: Handbook of Semidefinite Programming, pp. 111–138. Springer (2000)
12. Dong, S., Lee, Y.T., Ye, G.: A nearly-linear time algorithm for linear programs with small treewidth: a multiscale representation of robust central path. In: Proceedings of the 53rd annual ACM SIGACT symposium on theory of computing, pp. 1784–1797 (2021)
13. Eltved, A., Dahl, J., Andersen, M.S.: On the robustness and scalability of semidefinite relaxation for optimal power flow problems. Optim. Eng. (2019). https://doi.org/10.1007/s11081-019-09427-4
14. Fomin, F.V., Lokshtanov, D., Saurabh, S., Pilipczuk, M., Wrochna, M.: Fully polynomial-time parameterized computations for graphs and matrices of low treewidth. ACM Trans. Algorithms **14**(3), 1–45 (2018)
15. Frieze, A., Jerrum, M.: Improved approximation algorithms for MAX k-CUT and MAX BISECTION. Algorithmica **18**(1), 67–81 (1997)
16. Fukuda, M., Kojima, M., Murota, K., Nakata, K.: Exploiting sparsity in semidefinite programming via matrix completion I: General framework. SIAM J. Optim. **11**(3), 647–674 (2001)
17. Fulkerson, D., Gross, O.: Incidence matrices and interval graphs. Pac. J. Math. **15**(3), 835–855 (1965)
18. George, A., Liu, J.W.: Computer solution of large sparse positive definite. Prentice Hall Professional Technical Reference, Hoboken (1981)
19. Goemans, M.X., Williamson, D.P.: Improved approximation algorithms for maximum cut and satisfiability problems using semidefinite programming. J. ACM **42**(6), 1115–1145 (1995)
20. Grone, R., Johnson, C.R., Sá, E.M., Wolkowicz, H.: Positive definite completions of partial Hermitian matrices. Linear Algebra Its Appl. **58**, 109–124 (1984)
21. Gu, Y., Song, Z.: A faster small treewidth SDP solver. (2022). arXiv:2211.06033
22. Jabr, R.A.: Exploiting sparsity in SDP relaxations of the OPF problem. IEEE Trans. Power Syst. **27**(2), 1138–1139 (2012)
23. Kim, S., Kojima, M., Waki, H.: Exploiting sparsity in SDP relaxation for sensor network localization. SIAM J. Optim. **20**(1), 192–215 (2009)
24. Kim, S., Kojima, M., Mevissen, M., Yamashita, M.: Exploiting sparsity in linear and nonlinear matrix inequalities via positive semidefinite matrix completion. Math. Program. **129**(1), 33–68 (2011)
25. Kobayashi, K., Kim, S., Kojima, M.: Correlative sparsity in primal-dual interior-point methods for LP, SDP, and SOCP. Appl. Math. Optim. **58**(1), 69–88 (2008)
26. Lasserre, J.B.: Global optimization with polynomials and the problem of moments. SIAM J. Optim. **11**(3), 796–817 (2001)
27. Lavaei, J., Low, S.H.: Zero duality gap in optimal power flow problem. IEEE Trans. Power Syst. **27**(1), 92 (2012)
28. Lovász, L.: On the Shannon capacity of a graph. IEEE Trans. Inf. Theory **25**(1), 1–7 (1979)
29. Madani, R., Ashraphijuo, M., Lavaei, J.: Promises of conic relaxation for contingency-constrained optimal power flow problem. IEEE Trans. Power Syst. **31**(2), 1297–1307 (2016)
30. Maniu, S., Senellart, P., Jog, S.: An experimental study of the treewidth of real-world graph data. In: 22nd international conference on database theory (ICDT 2019), Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik, (2019)
31. Molzahn, D.K., Hiskens, I.A.: A survey of relaxations and approximations of the power flow equations. Found. Trends Electric Energy Syst. **4**(1–2), 1–221 (2019)
32. MOSEK ApS. The MOSEK optimization toolbox for MATLAB manual. Version 9.0., (2019)
33. Nakata, K., Fujisawa, K., Fukuda, M., Kojima, M., Murota, K.: Exploiting sparsity in semidefinite programming via matrix completion II: implementation and numerical results. Math. Program. **95**(2), 303–327 (2003)

34. Nesterov, Y.E., Todd, M.J.: Primal-dual interior-point methods for self-scaled cones. SIAM J. Optim. **8**(2), 324–364 (1998)
35. Parrilo, P.A.: Structured semidefinite programs and semialgebraic geometry methods in robustness and optimization. PhD thesis, California Institute of Technology, (2000)
36. Pataki, G., Schmieta, S.: The DIMACS library of semidefinite-quadratic-linear programs, (2002)
37. Sturm, J.F.: Using sedumi 1.02, a matlab toolbox for optimization over symmetric cones. Optim. Method. Softw. **11**(1–4), 625–653 (1999)
38. Sturm, J.F.: Implementation of interior point methods for mixed semidefinite and second order cone optimization problems. Optim. Methods Softw. **17**(6), 1105–1154 (2002)
39. Sun, Y.: Decomposition methods for semidefinite optimization. PhD thesis, UCLA, (2015)
40. Sun, Y., Andersen, M.S., Vandenberghe, L.: Decomposition in conic optimization with partially separable structure. SIAM J. Optim. **24**(2), 873–897 (2014)
41. Todd, M.J., Toh, K.-C., Tütüncü, R.H.: On the Nesterov-Todd direction in semidefinite programming. SIAM J. Optim. **8**(3), 769–796 (1998)
42. Toh, K.-C., Todd, M.J., Tütüncü, R.H.: Sdpt3-a matlab software package for semidefinite programming, version 1.3. Optim. Methods Softw. **11**(1–4), 545–581 (1999)
43. Tütüncü, R.H., Toh, K.-C., Todd, M.J.: Solving semidefinite-quadratic-linear programs using sdpt3. Math. Program. **95**, 189–217 (2003)
44. Vandenberghe, L., Andersen, M.S.: Chordal graphs and semidefinite optimization. Found. Trends Optim. **1**(4), 241–433 (2015)
45. Waki, H., Kim, S., Kojima, M., Muramatsu, M.: Sums of squares and semidefinite program relaxations for polynomial optimization problems with structured sparsity. SIAM J. Optim. **17**(1), 218–242 (2006)
46. Zhang, R.Y., Lavaei, J.: Sparse semidefinite programs with guaranteed near-linear time complexity via dualized clique tree conversion. Math. Program. **188**, 1–43 (2020)
47. Zheng, Y., Fantuzzi, G., Papachristodoulou, A., Goulart, P., Wynn, A.: Chordal decomposition in operator-splitting methods for sparse semidefinite programs. Math. Program. **180**(1), 489–532 (2020)
48. Zimmerman, R.D., Murillo-Sánchez, C.E., Thomas, R.J.: Matpower: steady-state operations, planning, and analysis tools for power systems research and education. IEEE Trans. Power Syst. **26**(1), 12–19 (2011)