Linear matrix inequalities with chordal sparsity patterns and applications to robust quadratic optimization

Martin S. Andersen and Lieven Vandenberghe
Electrical Engineering Department
University of California, Los Angeles
Los Angeles, California 90095
Email: {msa,vandenbe}@ee.ucla.edu

Joachim Dahl
MOSEK ApS
Fruebjergvej 3
2100 København Ø, Denmark
Email: dahl.joachim@gmail.com

***Abstract*— We discuss nonsymmetric interior-point methods for linear cone programs with chordal sparse matrix cone constraints. The algorithms take advantage of fast recursive algorithms for evaluating the function values and derivatives for the logarithmic barrier functions of the cone of positive semidefinite matrices with a given chordal sparsity pattern, and of the corresponding dual cone. We provide numerical results that show that our implementation can be significantly faster than general purpose semidefinite programming solvers. As a specific application, we discuss robust quadratic optimization.**

I. INTRODUCTION

A fundamental difficulty in solving large-scale semidefinite programs (SDPs)

$$\begin{array}{ll} \text{minimize} & \mathbf{tr}(CX) \\ \text{subject to} & \mathbf{tr}(A_iX) = b_i, \quad i = 1, \dots, m \\ & X \succeq 0 \end{array} \tag{1}$$

is that the variable $X \in \mathbf{S}^n$, a symmetric matrix of order n, is generally dense, even when the data matrices $A_i, C \in \mathbf{S}^n$ are sparse and share a common sparsity pattern. Here the generalized inequality $X \succeq 0$ means that X must be in the cone of symmetric positive semidefinite matrices of order n, denoted $\mathbf{S}^n_+$. The dual of (1) is given by

$$\begin{array}{ll} \text{maximize} & b^Ty \\ \text{subject to} & \sum_{i=1}^m y_iA_i + S = C \\ & S \succeq 0, \end{array} \tag{2}$$

where the variables are $y \in \mathbf{R}^m$ and $S \in \mathbf{S}^n$. Unlike the primal variable X, the dual slack variable S has the same (aggregate) sparsity pattern as C and the matrices A_i. This follows immediately from the equality constraint in (2). The inherent sparsity of S therefore makes it more straightforward to exploit sparsity in dual methods than in primal–dual methods when the data matrices are sparse. The inverse of S, however, is generally dense and it is needed to evaluate the gradient and the Hessian of the dual logarithmic barrier function.

To avoid storing the dense primal variable X, Fukuda and Nakata *et al.* [1], [2], Burer [3], and Srijuntongsiri and Vavasis [4] propose to pose the problems (1) and (2) as optimization problems in the subspace of symmetric matrices with a given sparsity pattern. Specifically, suppose $C, A_i \in \mathbf{S}^n_V$, where $\mathbf{S}^n_V$ denotes the set of symmetric matrices of order n with sparsity pattern V. The primal-dual pair (1)-(2) can then be written in the following equivalent form:

$$\begin{array}{ll} \text{minimize} & \mathbf{tr}(CX) \\ \text{subject to} & \mathbf{tr}(A_iX) = b_i, \quad i = 1, \dots, m \\ & X \succeq_{\mathrm{c}} 0, \end{array} \tag{3}$$

and its dual

$$\begin{array}{ll} \text{maximize} & b^Ty \\ \text{subject to} & \sum_{i=1}^m y_iA_i + S = C \\ & S \succeq 0. \end{array} \tag{4}$$

Here the primal variable X and the dual variable S are both matrices in the subspace $\mathbf{S}^n_V$, and $X \succeq_{\mathrm{c}} 0$ means that X has a positive semidefinite completion. Note that the cone $\{X \in \mathbf{S}^n_V \mid X \succeq 0\}$ and its dual cone $\{X \in \mathbf{S}^n_V \mid X \succeq_{\mathrm{c}} 0\}$ are not identical for general sparsity patterns.

In this paper we describe an interior-point solver for problems of the form (3)-(4) with chordal sparsity patterns V. We outline how chordal matrix techniques can be exploited in nonsymmetric (primal or dual) interior-point methods. These chordal matrix techniques include efficient recursive algorithms for evaluating barrier functions and their first and second derivatives. In addition we demonstrate the efficiency of the techniques with some numerical examples arising in robust quadratic optimization.

The rest of the paper is organized as follows. In Section II we review chordal sparsity and define chordal matrix cones and their logarithmic barrier functions. In Section III we describe nonsymmetric interior-point methods for optimization over sparse matrix cones, and in Section IV we give some application examples as well as numerical results. Conclusions are given in Section V.

II. CHORDAL MATRIX CONES

A. Chordal Sparsity and Chordal Matrix Cones

Examples of matrices with chordal sparsity include matrices with band structure, block-diagonal structure, and arrow structure. The sparsity pattern V of a symmetric matrix of order n can be represented by an undirected graph $\mathcal{G}_V$ with nodes $1, 2, \dots, n$ and an edge between nodes i and j ($i \neq j$) if there is a nonzero in positions i, j and j, i. (All diagonal

Research supported in part by NSF grants ECS-0524663 and ECCS-0824003.

entries are assumed to be nonzero[1], but the corresponding self-loops are not included in the graph.) Diagonal blocks in V correspond to connected components of $\mathcal{G}_V$. We say that the sparsity pattern V is chordal if the graph $\mathcal{G}_V$ is chordal, *i.e.*, every cycle of length greater than three has a chord. A *clique* in $\mathcal{G}_V$ is a set of nodes that define a maximal complete subgraph, and the nodes in a clique correspond to a dense principal subblock in the sparsity pattern. The set of cliques can be organized in a so-called clique tree which is used in chordal matrix algorithms. Clique trees of chordal graphs can be efficiently computed (in linear time) by the *maximum cardinality search* algorithm [5], [6], [7], and this also provides a test for chordality.

Now suppose V is a chordal sparsity pattern. The chordal matrix cone $\mathbf{S}^n_{V,+}$ of positive semidefinite matrices with sparsity V is then defined as

$$\mathbf{S}^n_{V,+} = \{X \in \mathbf{S}^n_V \mid X \succeq 0\}. \tag{5}$$

The dual of this cone is the cone of matrices in $\mathbf{S}^n_V$ that have a positive semidefinite completion and it is defined as

$$\mathbf{S}^n_{V,\mathrm{c}+} = \{P_V(X) \mid X \succeq 0\} \tag{6}$$

where $P_V(X)$ is the projection of X on $\mathbf{S}^n_V$. Note that $\mathbf{S}^n_{V,+}$ and $\mathbf{S}^n_{V,\mathrm{c}+}$ are not self-dual in general.

B. Chordal Matrix Algorithms

A number of sparse matrix problems can be solved efficiently by specialized algorithms when the underlying sparsity pattern is chordal. These algorithms consist of one or two recursions over the clique tree, and they are described in more detail in [8]. An implementation is available in the CHOMPACK library [9].

In this section we give an overview of the sparse matrix problems that can be solved efficiently with recursive algorithms, starting with the most well-known example.

1) Cholesky factorization: Positive definite matrices with a chordal sparsity pattern have a Cholesky factorization with zero fill-in [5], [10]. Specifically, suppose $S \succ 0$ has a chordal sparsity pattern V, then there exists a permutation matrix P and a lower triangular matrix L such that $P^TSP = LL^T$ and $P(L+L^T)P^T$ has the sparsity pattern V. This is an important property of chordal graphs, and it is the basis of the chordal matrix algorithms for the problems described henceforth.

2) Value and gradient of dual barrier: The barrier for the cone $\mathbf{S}^n_{V,+}$ is defined as

$$\phi : \mathbf{S}^n_V \to \mathbf{R}, \quad \phi(S) = -\log\det S \tag{7}$$

where $\mathbf{dom}\,\phi = \mathbf{S}^n_{V,++}$ (the interior of $\mathbf{S}^n_{V,+}$). The Cholesky factorization of S provides an efficient method for evaluating $\phi(S)$, *i.e.*,

$$\phi(S) = -2\sum_{i=1}^n \log L_{ii} \tag{8}$$

[1]If the sparsity pattern has a zero on its diagonal, the corresponding row/column of the positive semidefinite dual variable S must be zero, and hence the row/column can be eliminated from the problem.

where L_{ii} is the ith diagonal element of L. The gradient of ϕ is $\nabla\phi(S) = -P_V(S^{-1})$, and given L, it can be computed efficiently using a recursive algorithm without computing S^{-1} which is generally dense.

3) Hessian and inverse Hessian of dual barrier: The Hessian of ϕ at S, evaluated at $Y \in \mathbf{S}^n_V$, is given by

$$\nabla^2\phi(S)[Y] = P_V(S^{-1}YS^{-1}). \tag{9}$$

By exploiting chordal structure, this expression can be evaluated efficiently for large sparse matrices without forming $S^{-1}YS^{-1}$. This requires the Cholesky factor of S and the projected inverse $P_V(S^{-1})$. The algorithm is based on two recursions on the clique tree, and these recursions form a pair of adjoint linear operators that allow us to evaluate the Hessian in the factored form $\phi(S) = \mathcal{L}_{\mathrm{adj}}(\mathcal{L}(S))$. Moreover, the linear operator $\mathcal{L}$ is easily inverted, and this provides a method for evaluating the inverse Hessian $\nabla^2\phi(S)^{-1}[Y]$ at the same cost as the evaluation of the Hessian.

4) Maximum determinant positive definite completion: Given a matrix $X \in \mathbf{S}^n_V$, the maximum determinant positive definite completion problem is defined as follows:

$$\begin{array}{ll} \text{maximize} & \log\det Z \\ \text{subject to} & P_V(Z) = X. \end{array} \tag{10}$$

If V is chordal, the solution (if it exists) can be computed from X via closed-form expressions [11]. Alternatively, the Cholesky factor of $W = Z^{-1}$ can be computed recursively. It follows from convex duality that W has the sparsity pattern V, and satisfies the nonlinear equation

$$P_V(W^{-1}) = X. \tag{11}$$

The algorithm can therefore be interpreted as a method for solving the nonlinear equation (11) with variable $W \in \mathbf{S}^n_V$.

5) Value and gradient of primal barrier: A logarithmic barrier function for the cone $\mathbf{S}^n_{V,\mathrm{c}+}$ can be obtained from the Legendre transform of the barrier ϕ of $\mathbf{S}^n_{V,+}$ [12, p. 48]. Thus, the barrier for the primal cone $\mathbf{S}^n_{V,\mathrm{c}+}$ is defined as

$$\phi_{\mathrm{c}}(X) = \sup_{S \succ 0}(-\mathbf{tr}(XS) - \phi(S)). \tag{12}$$

If the sparsity pattern is chordal, the optimization problem in the definition (12) can be solved analytically. The solution is a positive definite matrix $\hat{S} \in \mathbf{S}^n_V$ that satisfies the equation $P_V(\hat{S}^{-1}) = X$, *i.e.*, $\hat{S}^{-1}$ is the maximum determinant positive definite completion of X. We can now evaluate $\phi_{\mathrm{c}}(X)$ efficiently: first we compute $\hat{S} = LL^T$ and then

$$\phi_{\mathrm{c}}(X) = \log\det\hat{S} - n = 2\sum_{i=1}^n \log L_{ii} - n. \tag{13}$$

The gradient of the primal barrier follows from the properties of the Legendre transform:

$$\nabla\phi_{\mathrm{c}}(X) = -\hat{S}. \tag{14}$$

6) Hessian and inverse Hessian of primal barrier: The Hessian of the primal barrier function is given by

$$\nabla^2\phi_{\mathrm{c}}(X) = \nabla^2\phi(\hat{S})^{-1}, \tag{15}$$

where $\hat{S}$ is the maximizer in the definition of $\phi_{\mathrm{c}}(X)$. This result follows from the standard properties of the Legendre transform. The Hessian of the primal barrier can therefore be evaluated using the method for evaluating the inverse Hessian of the dual barrier.

C. Chordal Embedding

We have seen that it is possible to take advantage of efficient recursive algorithms for evaluating barriers and their first and second derivatives when the data matrices A_i, C have a common chordal sparsity pattern. These chordal matrix algorithms can also be applied to matrices with a nonchordal sparsity pattern by constructing a chordal embedding or triangulation of the nonchordal sparsity pattern. This amounts to adding edges to the sparsity graph to make it chordal. Chordal embedding techniques can also be used to "shape" the clique tree to improve the computational efficiency of the algorithms. The efficiency depends largely on the clique tree, and for example, merging cliques with large relative overlap can often improve performance.

A chordal embedding of a nonchordal sparsity pattern can easily be constructed by computing a symbolic Cholesky factorization of the sparsity pattern. The amount of fill-in (*i.e.* the number of added edges) generally depends heavily on the ordering of the nodes, and fill-in reducing orderings, such as the *approximate minimum degree* (AMD) ordering [13], are often used in practice.

III. INTERIOR-POINT METHODS

The chordal techniques discussed in the previous section provide an efficient way to evaluate barriers and their first and second derivatives when the underlying sparsity pattern is chordal. In this section we outline how these techniques can be exploited in interior-point methods for the pair of cone programs (3)-(4).

A. The Central Path

The central path defines an arc of strictly feasible points, and interior-point methods make use of it to steer the iteration toward the solution. For the pair of cone programs (3)-(4), the central path is defined as the set of points $X \succ_{\mathrm{c}} 0$, y, $S \succ 0$ that satisfy

$$\mathbf{tr}(A_iX) = b_i, \quad i = 1, \ldots, m, \tag{16a}$$

$$\sum_{i=1}^{m} y_iA_i + S = C, \tag{16b}$$

$$S = -\mu\nabla\phi_{\mathrm{c}}(X) \tag{16c}$$

where μ is a positive parameter. Interior-point methods that compute search directions based on linearizing (16a-c) are called *primal scaling* methods. An equivalent formulation can be obtained by replacing (16c) with the equation

$$X = -\mu\nabla\phi(S), \tag{17}$$

and this gives rise to *dual scaling* methods. We will refer to the corresponding linearized equations as the primal or dual Newton equations. We now describe how the Newton equations can be solved when V is chordal.

B. Primal and Dual Methods

By linearizing the central path equations (16a-c) around the current iterate (X, y, S) and eliminating ΔS, we get the primal Newton equations

$$\mathbf{tr}(A_i\Delta X) = r_i, \quad i = 1, \ldots, m, \tag{18a}$$

$$\sum_{i=1}^{m} \Delta y_iA_i - \mu\nabla^2\phi_{\mathrm{c}}(X)[\Delta X] = R, \tag{18b}$$

with variables $\Delta X, \Delta y$ and residuals R and r_i. This is a system of $m + |V|$ equations in $m + |V|$ unknowns where $|V|$ denotes the number of nonzeros in the lower triangle of V. The Newton equations (18a-b) can be further reduced by eliminating ΔX. This yields a system

$$H\Delta y = g \tag{19}$$

where H is positive definite with entries

$$H_{ij} = \mathbf{tr}\left(A_i\nabla^2\phi_{\mathrm{c}}(X)^{-1}[A_j]\right), \quad i, j = 1, \ldots, m, \tag{20}$$

and $g_i = \mu r_i + \mathbf{tr}(A_i\nabla^2\phi_{\mathrm{c}}(X)^{-1}[R])$.

In the following we outline two methods for solving (19). The first method is based on the Cholesky factorization of H (and hence H must be formed explicitly) whereas the second method avoids explicit calculation of H by exploiting the factorization $\nabla^2\phi_{\mathrm{c}}(X)^{-1} = \nabla^2\phi(\hat{S}) = \mathcal{L}_{\mathrm{adj}} \circ \mathcal{L}$. We remark that the dual scaling Newton equations can be derived and solved in a similar way, and the complexity is exactly the same.

1) Cholesky factorization: The first method explicitly computes the lower triangle of H, column by column, and then solves (19) using a dense Cholesky factorization. We distinguish between two techniques for computing column j. The first technique (T1) is a straightforward evaluation of (20), *i.e.*, we first compute $U = \nabla^2\phi(X)^{-1}[A_j]$, and then we compute the inner products of U and $A_j, \ldots, A_m$. Although the algorithm for evaluating $\nabla^2\phi(X)^{-1}[A_j]$ exploits chordal sparsity, this technique can be inefficient when A_j is much more sparse than V (*e.g.* if A_j only has a few nonzeros).

The second technique (T2) is based on the expansion

$$H_{ij} = \sum_{(p,q)\in I_j} (A_j)_{pq}(\mathbf{tr}(A_iP_V(\hat{S}^{-1}e_pe_q^T\hat{S}^{-1}))) \tag{21}$$

where the set I_j indexes the nonzero entries in A_j and e_p is the pth unit vector. Thus, if A_j only has a small number of nonzeros columns, it is advantageous to precompute the vectors $u_k = L^{-T}L^{-1}e_k$ (where $\hat{S} = LL^T$) that occur in (21). This technique is similar to the technique used in SDPA-C [2].

Using the Cholesky factorization method, the columns of H can be computed by selecting one of the two techniques on a column-by-column basis, and a threshold on the number of nonzero columns in A_j can be used as a simple heuristic for selecting which technique to apply when computing the jth column.

2) QR decomposition: The second method for solving the reduced Newton system (19) avoids the explicit calculation of H by exploiting the factorization $\nabla^2\phi_{\rm c}(X)^{-1} = \nabla^2\phi(\hat{S}) = \mathcal{L}_{\rm adj} \circ \mathcal{L}$, *i.e.*,

$$H_{ij} = \mathbf{tr}\left(\mathcal{L}(A_i)\mathcal{L}(A_j)\right). \qquad (22)$$

This factorization allows us to express H as $H = \tilde{A}^T\tilde{A}$ where $\tilde{A}$ is a $|V| \times m$ matrix with columns $\mathbf{vec}(\mathcal{L}(A_i))$. Here $\mathbf{vec}(\cdot)$ is a linear operator that converts matrices in $\mathbf{S}^n_V$ to vectors in $\mathbf{R}^{|V|}$, scaled such that $\mathbf{vec}(U_1)^T\mathbf{vec}(U_2) = \mathbf{tr}(U_1U_2)$. Thus, instead of computing the Cholesky factorization of H, we can compute a QR decomposition of $\tilde{A}$ and use it to solve (19). This is important since the explicit computation of H is a source of numerical instability.

Finally we mention that this method is a variation of the *augmented systems* approach in linear programming interior-point methods. In semidefinite programming the loss of stability in forming H is more severe than in linear programming [14], but the augmented systems approach is generally computationally intractable due to the large size of the Newton equations. However, since we work in the lower dimensional subspace $\mathbf{S}^n_V$ in our present context, the augmented system approach is often feasible.

C. Complexity

The cost of solving the Newton system (18) is dominated by the cost of solving (19). Recall that the first method explicitly forms and factors H whereas the second method forms and factors the matrix $\tilde{A}$. Here we describe the cost of the two methods.

1) Cholesky factorization: The cost of forming H depends on the sparsity of the data matrices as well the technique used to form H. Forming H using technique T1 costs at most $mK + O(m^2|V|)$ where K is the cost of evaluating $\nabla^2\phi_{\rm c}(X)^{-1}[A_j]$. The second term $O(m^2|V|)$ is the worst-case complexity assuming that the data matrices are all "dense" relative to V. This term is negligible if the data matrices only have a small number of nonzeros. With technique T2, the dominating cost of computing the columns of H is solving the systems $LL^Tu_k = e_k$ for each nonzero column in A_i. The cost therefore mainly depends on $|V|$ and the number of nonzero columns in the data matrices. T2 is generally many times faster than T1 when the data matrices have only a few nonzeros. The matrix H is generally dense, and hence the cost of factorizing H is $O(m^3)$.

The cost K depends on the clique tree (*i.e.* the structure of V) in a complicated way, but for special cases such as band and block-arrow matrices we have $|V| = O(n)$ and $K = O(n)$, and hence in these cases the cost of one iteration is *linear* in n.

2) QR decomposition: Solving (19) via a QR decomposition of $\tilde{A}$ costs $O(mK)$ to form $\tilde{A}$ and $O(m^2|V|)$ to compute the QR decomposition. In particular, the cost of one iteration is also linear in n for special cases such as band and block-arrow matrices, when the other dimensions are fixed.

D. Implementation

We have implemented primal and dual scaling variants of an interior-point method based on the techniques described in this section. Our preliminary implementation is a feasible start barrier method and it is available in the form of a Python extension named SMCP. It relies on the Python extensions CVXOPT 1.1.2 [15] and CHOMPACK 1.1 [9] for linear algebra and chordal matrix computations. The algorithm first tests a number of heuristics in order to find a feasible starting point, and if it fails, a phase I problem is solved. Additional implementation details and extensive benchmarks can be found in [16].

IV. Numerical Experiments

Sparsity patterns with block-arrow structure are chordal, and matrix inequalities with block-arrow patterns arise in several applications. Many robust counterparts of quadratically constrained quadratic programs (QCQPs) or second-order cone programs (SOCPs) fall in this category [17], [18], [19]. If the "width" of the arrow is not too large, it is often advantageous to exploit chordal structure. Here we will take robust quadratically constrained quadratic programming (QCQP) and robust least-squares (RLS) as examples, and we demonstrate with some numerical experiments that the chordal matrix algorithms can lead to significant computational savings. Numerical experiments on different sets of benchmarks have been reported in [16].

We conducted all experiments on a PC with an Intel Q6600 quad core CPU, 4 GB of memory, and running Ubuntu 9.10. In the experiments we used both the Cholesky and the QR variants of SMCP. For comparison, we used the SDP solvers DSDP 5.8 [20], SDPA-C 6.2.1 [21], and SDPT3 4.0b [22]. DSDP implements a dual-scaling algorithm that exploits sparsity in the dual slack variable, and the solver makes use of low rank factorizations of the data matrices as well as an iterative method for solving the reduced Newton system. SDPA-C implements a primal–dual path-following method, and like SMCP, it exploits chordal structure. Finally, SDPT3 is a Matlab-based implementation of a primal–dual predictor-corrector path-following method for general conic optimization problems.

A. Robust Quadratic Programming

In our first example we look at quadratic programs with one or more uncertain convex quadratic constrains of the form

$$x^TA^TAx \le 2b^Tx + d. \qquad (23)$$

Here the problem data $A \in \mathbf{R}^{p\times q}$, $b \in \mathbf{R}^q$, and $d \in \mathbf{R}$ are uncertain. If we choose as uncertainty set a bounded ellipsoid

$$\mathcal{U} = \left\{ (\bar{A},\bar{b},\bar{d}) + \sum_{i=1}^r u_i(A_i,b_i,d_i) \;\middle|\; u^Tu \le 1 \right\} \qquad (24)$$

where $\bar{A},\bar{b},\bar{d}$ are nominal values, the robust counterpart of the uncertain quadratic constraint (23) can be formulated as

TABLE I: Average time per iteration (seconds) for randomly generated uncertain QCQPs with $q = 100$ and $r = 5$.

p	SMCP–CHOL.	SMCP–QR	DSDP	SDPA-C	SDPT3
200	0.25	0.13	0.20	0.70	0.54
400	0.51	0.26	0.91	1.4	1.7
800	1.0	0.53	5.1	3.1	5.8
1600	2.1	1.2	33.6	6.5	21.6
3200	4.3	2.7	–	13.8	90.2

a linear matrix inequality (LMI) [18]

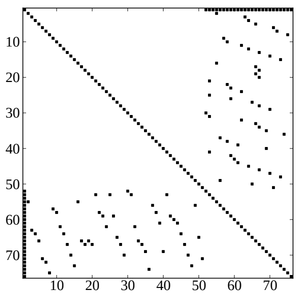
$$\begin{bmatrix} tI & G(x)^T & h(x) \\ G(x) & I & \bar{A}x \\ h(x)^T & (\bar{A}x)^T & f(x) - t \end{bmatrix} \succeq 0 \tag{25}$$

with variables $t \in \mathbf{R}$ and $x \in \mathbf{R}^q$ and where

$$\begin{aligned} G(x) &= [A_1x \ \cdots \ A_rx], \\ h(x) &= (b_1^Tx + d_1/2, \ldots, b_r^Tx + d_r/2), \\ f(x) &= 2\bar{b}^Tx + \bar{d}. \end{aligned}$$

The sparsity pattern associated with the LMI (25) has block-arrow structure, and it can be either chordal or nonchordal, depending on the structure of $G(x)$ (which, in turn, is determined by the choice of uncertainty set). It is typically worthwhile to exploit chordal sparsity in two cases. If $G(x)$ is dense and $p \gg r$ or $r \gg p$, an efficient chordal embedding can easily be constructed by filling the smaller of the two diagonal blocks. This chordal embedding will have cliques of order at most $\min(r + 1, p + 1)$. If on the other hand $G(x)$ is sparse, the sparsity pattern may have an efficient chordal embedding with small cliques even when $p \approx r$. As a special case we mention that if $G(x)$ has at most one nonzero entry in each column for all x (or alternatively, at most one nonzero entry in each row), the sparsity pattern is chordal and the cliques are of order at most three. Here we will consider a numerical experiment with $G(x)$ dense, and in the next section we look at an example where $G(x)$ is sparse and the LMI has a chordal sparsity pattern.

In our first experiment we are interested in the average CPU time per iteration as a function of p for randomly generated uncertain QCQPs with q, r fixed and with r small. Specifically, we minimize a linear objective $f_0(x) = c^Tx$ subject to a single uncertain quadratic constraint of the form (23). For this experiment we choose $q = 100$, $r = 5$, and generate problem instances with $\bar{A}, \bar{b}, A_i$ random, $b_i = 0$, $\bar{d} = 1$, and $d_i = 0$. The results are listed in Table I. It is easily verified that average time per iteration grows roughly linearly for SMCP. Furthermore, notice that SDPA-C, which also exploits chordal structure, is quite fast as well. The other solvers scale quadratically or worse, and hence the benefit of exploiting chordal sparsity becomes evident for p large. We remark that DSDP crashed on the largest problem instance. Finally we note that having multiple uncertain quadratic constraints gives rise to an LMI with block-diagonal structure and with blocks of the form (25).


(a) Before reordering.

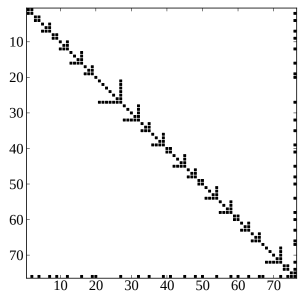

(b) After reordering.

Fig. 1: Sparsity pattern associated with an RLS problem instance with dimensions $p = 25$, $q = 10$, and $r = 50$.

B. Robust Least-Squares

Our next experiment is based on robust least-squares which is a special case of robust QCQP. Suppose we want to minimize $\|Ax - b\|_2$ where $A \in \mathcal{U}$ is uncertain but assumed to lie in the ellipsoidal uncertainty set

$$\mathcal{U} = \{\bar{A} + u_1A_1 + \cdots + u_rA_r \mid \|u\|_2 \le 1\}. \tag{26}$$

Here $\bar{A} \in \mathbf{R}^{p\times q}$ is a known nominal coefficient matrix, $b \in \mathbf{R}^p$ is a known vector, and the matrices $A_i \in \mathbf{R}^{p\times q}$ define the structure of the set $\mathcal{U}$. The RLS problem seeks a solution $x \in \mathbf{R}^q$ that minimizes the worst-case error which is defined as

$$e_{\mathrm{wc}}(x) = \sup_{\|u\|_2 \le 1} \|G(x)u + h(x)\|_2, \tag{27}$$

where $G(x) = [A_1x \ \cdots \ A_rx]$ and $h(x) = \bar{A}x - b$. The RLS problem can be cast as an SDP [17]:

$$\begin{array}{ll} \text{minimize} & t + \lambda \\ \text{subject to} & \begin{bmatrix} t & 0 & h(x)^T \\ 0 & \lambda I & G(x)^T \\ h(x) & G(x) & I \end{bmatrix} \succeq 0 \end{array} \tag{28}$$

with variables $t, \lambda \in \mathbf{R}$ and $x \in \mathbf{R}^q$. Note that the SDP (28) has $m = q + 2$ variables and an LMI of order $n = p + r + 1$.

In the following experiment we consider a family of RLS problems, defined as follows. Suppose $\bar{A}$ has r uncertain entries indexed by $(i_1, j_1), \ldots, (i_r, j_r)$. Furthermore, let the matrices $A_1, \ldots, A_r$ be defined as

$$(A_k)_{ij} = \begin{cases} \gamma & i = i_k, j = j_k \\ 0 & \text{otherwise} \end{cases} \qquad k = 1, \ldots, r, \tag{29}$$

where $\gamma > 0$ is a parameter that controls the size of the uncertainty set $\mathcal{U}$. The resulting SDP has a chordal sparsity pattern with $|V| = 2p + 2r + 1$ nonzeros in the lower triangle of V, and furthermore, the sparsity pattern has $p + r$ cliques of order two. An example of a sparsity pattern from an RLS problem is shown in Fig. 1.

As in the previous experiment, we are interested in the computational cost per interior-point iteration as a function of p. We generate random problem instances as follows. The vector b is computed as $b = \bar{A}\bar{x} + \sigma w$ where $\bar{A}$ is a (dense) random matrix, $\bar{x}$ and w are random vectors, and σ is a positive parameter. The number of uncertain entries of $\bar{A}$ is

TABLE II: Average time per iteration (seconds) for randomly generated RLS problems with $q = 100$ and $r = 5p$.

p	SMCP–CHOL.	SMCP–QR	DSDP	SDPA-C	SDPT3
100	0.19	0.14	0.20	0.18	1.0
200	0.41	0.31	0.81	0.62	4.8
400	0.82	0.64	4.4	2.8	27.6
800	1.7	1.4	25.2	11.1	125.2
1600	3.6	3.2	153.6	45.3	–

parameterized by $d \in (0, 1]$ such that $r = \lceil pqd \rceil$ where $\lceil x \rceil$ denotes the smallest integer larger than or equal to x. The positions of the unknown entries are selected at random.

Table II shows the average CPU time per iteration (in seconds) for problem instances with different values of p and with the remaining problem parameters fixed ($q = 100$ and $d = 0.05$). Notice once again that for SMCP the time per iteration grows roughly linearly with the parameter p. The other solvers scale quadratically or worse, and for the largest instance the general purpose solver SDPT3 ran out of memory. Note also that in this example SDPA-C does not scale as well as in the previous experiment.

V. CONCLUSIONS

We have described how chordal sparsity can be exploited in nonsymmetric interior-point methods for linear optimization over sparse matrix cones. These cones include as special cases the nonnegative orthant, the second-order cone, the cone of positive semidefinite matrices, and direct products of these. More importantly, the sparse matrix cones include a wide range of intermediate cones that fill the complexity gap between the three standard cones. For general sparse nonchordal sparsity patterns the performance of the method depends on the efficiency of the chordal embedding. If no efficient chordal embedding can be found, the chordal techniques are generally not advantageous, although in many cases their efficiency is still comparable to that of the primal-dual interior-point solvers.

The robust counterpart of a convex quadratic optimization problem (least-squares problem, QCQP, or SOCP) with ellipsoidal uncertainty set is typically an SDP with block-arrow structure. Although convex, the robust problem is therefore substantially more expensive to solve using general-purpose solvers than the nominal non-robust problem. Our numerical results show that by exploiting the chordal structure of the arrow pattern, the cost of solving the robust problem can be reduced dramatically, depending on the number of uncertain parameters and the structure of the uncertainty.

REFERENCES

[1] M. Fukuda, M. Kojima, K. Murota, and K. Nakata, "Exploiting sparsity in semidefinite programming via matrix completion I: general framework," *SIAM Journal on Optimization*, vol. 11, pp. 647–674, 2000.

[2] K. Nakata, K. Fujitsawa, M. Fukuda, M. Kojima, and K. Murota, "Exploiting sparsity in semidefinite programming via matrix completion II: implementation and numerical details," *Mathematical Programming Series B*, vol. 95, pp. 303–327, 2003.

[3] S. Burer, "Semidefinite programming in the space of partial positive semidefinite matrices," *SIAM Journal on Optimization*, vol. 14, no. 1, pp. 139–172, 2003.

[4] G. Srijuntongsiri and S. A. Vavasis, "A fully sparse implementation of a primal-dual interior-point potential reduction method for semidefinite programming," 2004. [Online]. Available: http://www.citebase.org/abstract?id=oai:arXiv.org:cs/0412009

[5] D. J. Rose, "Triangulated graphs and the elimination process," *Journal of Mathematical Analysis and Applications*, vol. 32, pp. 597–609, 1970.

[6] D. J. Rose, R. E. Tarjan, and G. S. Lueker, "Algorithmic aspects of vertex elimination on graphs," *SIAM Journal on Computing*, vol. 5, no. 2, pp. 266–283, 1976.

[7] R. E. Tarjan and M. Yannakakis, "Simple linear-time algorithms to test chordality of graphs, test acyclicity of hypergraphs, and selectively reduce acyclic hypergraphs," *SIAM Journal on Computing*, vol. 13, no. 3, pp. 566–579, 1984.

[8] J. Dahl, L. Vandenberghe, and V. Roychowdhury, "Covariance selection for non-chordal graphs via chordal embedding," *Optimization Methods and Software*, vol. 23, no. 4, pp. 501–520, 2008.

[9] J. Dahl and L. Vandenberghe, *CHOMPACK: Chordal Matrix Package*, 2009, abel.ee.ucla.edu/chompack.

[10] E. G. Ng, "Sparse matrix methods," in *Handbook of linear algebra*, L. Hogben, Ed. Chapman & Hall/CRC, Nov. 2006.

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

[12] Y. Nesterov and A. Nemirovskii, *Interior-point polynomial methods in convex programming*, ser. Studies in Applied Mathematics. Philadelphia, PA: SIAM, 1994, vol. 13.

[13] P. Amestoy, T. Davis, and I. Duff, "An approximate minimum degree ordering," *SIAM Journal on Matrix Analysis and Applications*, vol. 17, no. 4, pp. 886–905, 1996.

[14] J. F. Sturm, "Avoiding numerical cancellation in the interior point method for solving semidefinite programs," *Mathematical Programming Series B*, vol. 95, pp. 219–247, 2003.

[15] J. Dahl and L. Vandenberghe, *CVXOPT: A Python Package for Convex Optimization*, abel.ee.ucla.edu/cvxopt, 2008.

[16] M. S. Andersen, J. Dahl, and L. Vandenberghe, *Implementation of nonsymmetric interior-point methods for linear optimization over sparse matrix cones*, 2009, abel.ee.ucla.edu/smcp.

[17] L. El Ghaoui and H. Lebret, "Robust solutions to least-squares problems with uncertain data," *SIAM Journal of Matrix Analysis and Applications*, vol. 18, no. 4, pp. 1035–1064, 1997.

[18] A. Ben-Tal and A. Nemirovski, "Robust convex optimization," *Mathematics of Operations Research*, vol. 23, pp. 769–805, 1998.

[19] M. V. Kothare, V. Balakrishnan, and M. Morari, "Robust constrained model predictive control using linear matrix inequalities," *Automatica*, vol. 32, no. 10, pp. 1361–1379, 1996.

[20] S. J. Benson and Y. Ye, "DSDP5 user guide — software for semidefinite programming," Mathematics and Computer Science Division, Argonne National Laboratory, Argonne, IL, Tech. Rep. ANL/MCS-TM-277, September 2005. [Online]. Available: http://www.mcs.anl.gov/ benson/dsdp

[21] K. Fujisawa, M. Fukuda, M. Kojima, K. Nakata, and M. Yamashita, "SDPA-C (SemiDefinite Programming Algorithm – Completion method) user's manual — version 6.10," Dept. Math. & Comp. Sciences, Tokyo Institute of Technology, Tech. Rep. B-409, 2004.

[22] R. H. Tütüncü, K. C. Toh, and M. J. Todd, "Solving semidefinite-quadratic-linear programs using SDPT3," *Mathematical Programming Series B*, vol. 95, pp. 189–217, 2003.