EXTENDING MEHROTRA AND GONDZIO HIGHER ORDER METHODS TO MIXED SEMIDEFINITE-QUADRATIC-LINEAR PROGRAMMING

JEAN–PIERRE HAEBERLY, MADHU V. NAYAKKANKUPPAM, AND MICHAEL L. OVERTON

Abstract. We discuss extensions of Mehrotra's higher order corrections scheme and Gondzio's multiple centrality corrections scheme to mixed semidefinite-quadratic-linear programming (SQLP). These extensions have been included in a solver for SQLP written in C and based on LAPACK. The code implements a primal–dual path–following algorithm for solving SQLP problems based on the XZ+ZX search direction and Mehrotra's predictor–corrector method. We present benchmarks showing that the use of the higher order schemes yields substantial reductions in both the number of iterations and the running time of the algorithm, and also improves its robustness.

1. Introduction

We introduce extensions of the "higher order corrections" (HOC) and "multiple centrality corrections" (MCC) schemes, introduced for linear programming (LP) by Mehrotra [13] and Gondzio [10] respectively, to mixed semidefinite-quadratic-linear programs (SQLP). More specifically, we describe the use of these methods in the context of a primal–dual path–following algorithm based on Mehrotra's predictor–corrector method. Experience reported for LP has been that, although both methods usually reduce the number of iterations required, Gondzio's MCC method typically yields a reduction in *cpu* time while Mehrotra's HOC method does not. Our experience with SQLP is somewhat different. These methods are attractive for SQLP because the relative cost of performing a single correction, compared to the cost of one basic interior–point iteration, is much less for SQLP than for LP. We find that both methods yield substantial reductions in iteration count and *cpu* time, even on medium–sized problems. While the effects of Gondzio's scheme in SQLP are comparable to those reported for LP, our experience is that the benefits derived from Mehrotra's HOC method are much greater for SQLP than have been reported for LP.

We have implemented these extensions in a collection of routines written in C and based on LAPACK [6], implementing a primal–dual path–following algorithm for solving SQLP. It is an extension of SDPpack, version 0.9 beta, a Matlab package written by the authors in collaboration with F. Alizadeh and S. Schmieta [3], and is based on the XZ+ZX search direction [4] for semidefinite programming (SDP). Several algorithmic improvements have been included.

Date: November 4, 1998. To appear in *Optimization Methods and Software*.

The first author is grateful to Jacek Gondzio for very helpful discussions. This work was supported in part by National Science Foundation Grant CCR-9731777 and in part by U.S. Dept. of Energy Grant DE-FG02-98ER25352.

The paper is organized as follows. In Section 2 we discuss mixed semidefinite–quadratic–linear programs and introduce the necessary notation. We also describe the basic primal–dual predictor–corrector path–following algorithm to solve SQLP's.

In Sections 3 and 4 respectively, we describe the extensions of the "higher order corrections" and "multiple centrality corrections" schemes from LP to SQLP. In Section 5, we discuss miscellaneous implementation details. Finally, in Section 6 we present the results of numerical experiments, demonstrating the relative merits of the higher order schemes.

2. Mixed Semidefinite–Quadratic–Linear Programs

We begin by introducing some notation, based on [3]. Let $N = (N_1, \dots, N_s)$ and $n = (n_1, \dots, n_q)$ denote two vectors of positive integers, and let n_0 be a nonnegative integer. We define the following spaces.

- Let $\mathbb{R}^{n_0}$ denote the space of real n_0–dimensional vectors. It contains the non–negative orthant $\mathbb{R}^{n_0}_+$, which is the cone of vectors $x = (x_1, \dots, x_{n_0})^T$ with $x_i \geq 0$ for $1 \leq i \leq n_0$.
- Let $\mathcal{Q}^n \equiv \mathbb{R}^{n_1+1} \times \cdots \times \mathbb{R}^{n_q+1}$ denote the space of real block vectors with block sizes $n_1 + 1, \dots, n_q + 1$. It contains the quadratic cone $\mathcal{Q}^n_+$ of vectors whose i^{th} block $(x_0, \dots, x_{n_i})^T$ satisfies
$$x_0 \geq \sqrt{\sum_{j=1}^{n_i} x_j^2}\,.$$
- Let $\mathcal{S}^N \equiv \mathcal{S}^{N_1} \times \cdots \times \mathcal{S}^{N_s}$ denote the space of real, symmetric, block diagonal matrices with block sizes $N_1, \dots, N_s$. It contains the semidefinite cone $\mathcal{S}^N_+$ of positive semidefinite matrices with the given block structure.

Each of these cones is closed, convex and self–dual. A mixed semidefinite–quadratic–linear program is an optimization problem of the form

$$\begin{array}{ll} \min & C_S \bullet X_S + c_Q \bullet x_Q + c_L \bullet x_L \\ \text{s.t.} & (A_S)_k \bullet X_S + (a_Q)_k \bullet x_Q + (a_L)_k \bullet x_L = b_k \quad (k = 1, \dots, m) \\ & X_S \succeq 0, \quad x_Q \geq_Q 0, \quad x_L \geq 0, \end{array} \tag{2.1}$$

where

$$C_S, (A_S)_k, X_S \in \mathcal{S}^N, \quad c_Q, (a_Q)_k, x_Q \in \mathcal{Q}^n, \quad c_L, (a_L)_k, x_L \in \mathbb{R}^{n_0},$$

and $b = (b_1, \dots, b_m)^T \in \mathbb{R}^m$. Here, by $X_S \succeq 0$ (respectively $x_Q \geq_Q 0$, $x_L \geq 0$), we mean the cone constraint that X_S (respectively x_Q, x_L) lies in the semidefinite cone (respectively quadratic cone, non–negative orthant). The same symbol $\bullet$ is used to denote the inner product on $\mathcal{S}^N$, $\mathcal{Q}^n$ and $\mathbb{R}^{n_0}$. For $U, V \in \mathcal{S}^N$, $u, v \in \mathcal{Q}^n$, $u_0, v_0 \in \mathbb{R}^{n_0}$, we have

$$U \bullet V \equiv \operatorname{trace}(UV), \quad u \bullet v \equiv u^T v, \quad u_0 \bullet v_0 \equiv u_0^T v_0.$$

Semidefinite programming (SDP), convex quadratically constrained quadratic programming (QCQP), and linear programming (LP) are all special cases of SQLP.

The dual of (2.1) is

$$\max \quad b \bullet y$$

$$
\begin{aligned}
\text{s.t.} \quad & \sum_{k=1}^{m} y_k (A_S)_k + Z_S = C_S \\
& \sum_{k=1}^{m} y_k (a_Q)_k + z_Q = c_Q \\
& \sum_{k=1}^{m} y_k (a_L)_k + z_L = c_L \\
& Z_S \succeq 0, \quad z_Q \geq_Q 0, \quad z_L \geq 0.
\end{aligned}
\tag{2.2}
$$

Let us write $\mathcal{S}^N_{++}$, $\mathcal{Q}^n_{++}$, $\mathbb{R}^{n_0}_{++}$ for the interiors of $\mathcal{S}^N_+, \mathcal{Q}^n_+, \mathbb{R}^{n_0}_+$ respectively. We assume that the following conditions hold:

- *Assumption 1:* (Slater condition) There exist strictly feasible primal and dual points, *i.e.* there exist $X_S, Z_S \in \mathcal{S}^N_{++}$, $x_Q, z_Q \in \mathcal{Q}^n_{++}$, $x_L, z_L \in \mathbb{R}^{n_0}_{++}$, and $y \in \mathbb{R}^m$ satisfying the equality constraints in (2.1) and (2.2), and
- *Assumption 2:* The $((A_S)_k, (a_Q)_k, (a_L)_k)$, $k = 1, \dots, m$, are linearly independent in $\mathcal{S}^N \times \mathcal{Q}^n \times \mathbb{R}^{n_0}$.

Under these assumptions, the optimality conditions [15] for SQLP are that, in addition to the constraints in (2.1) and (2.2), the duality gap is zero, *i.e.*

$$X_S \bullet Z_S + x_Q \bullet z_Q + x_L \bullet z_L = 0. \tag{2.3}$$

In order to describe the Newton system resulting from these optimality conditions, we need to introduce some more notation. We write

$$X = (X_S, x_Q, x_L) \in \mathcal{S}^N \times \mathcal{Q}^n \times \mathbb{R}^{n_0},$$

for the primal variable of SQLP and similarly $Z = (Z_S, z_Q, z_L)$ for the dual slack variable. We write $X \bullet Z \equiv X_S \bullet Z_S + x_Q \bullet z_Q + x_L \bullet z_L$.

For a matrix $T \in \mathcal{S}^N$ (likewise a block vector in $\mathcal{Q}^n$), we refer to the i^{th} block of T as $T(i)$, and write $T = (T(1), \dots, T(s))$. Let

$$d_S = \sum_{i=1}^{s} \frac{N_i(N_i+1)}{2} \quad \text{and} \quad d_Q = \sum_{i=1}^{q} (n_i + 1) \tag{2.4}$$

denote the dimensions of $\mathcal{S}^N$ and $\mathcal{Q}^n$ respectively. Let the operator

$$\mathbf{vec} \; : \; \mathcal{S}^N \longrightarrow \mathbb{R}^{d_S}$$

be an isometry that maps matrices in $\mathcal{S}^N$ onto column vectors.

Following [1], given a block vector $x_Q \in \mathcal{Q}^n$, we define the "arrow matrix" $\mathbf{Arw}(x_Q)$ to be the block diagonal matrix whose i^{th} block is

$$\begin{bmatrix} x_0 & x_1 & x_2 & \cdots & x_{n_i} \\ x_1 & x_0 & 0 & \cdots & 0 \\ x_2 & 0 & x_0 & \cdots & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ x_{n_i} & 0 & 0 & 0 & x_0 \end{bmatrix}$$

where $(x_0, x_1, \dots, x_{n_i})^T$ is the vector $x_Q(i)$. Observe that multiplying a block vector by $\mathbf{Arw}(x_Q)$ requires only a blockwise scalar multiplication and two rank one updates, and is thus an $O(d_Q)$ operation. Similarly, the cost of solving a system of equations whose coefficient matrix is $\mathbf{Arw}(x_Q)$ is $O(d_Q)$, using the Sherman-Morrison-Woodbury formula.

Finally, we need the abbreviations

$$\nu \equiv \sum_{i=1}^{s} N_i + d_Q + n_0, \tag{2.5}$$

$$d \equiv d_S + d_Q + n_0. \tag{2.6}$$

Now for $X_S, Z_S \succeq 0$, $x_Q, z_Q \geq_Q 0$, and $x_L, z_L \geq 0$, the complementarity condition (2.3) reduces to the following system of nonlinear equations:

$$X_S Z_S + Z_S X_S = 0 \tag{2.7}$$

$$\mathbf{Arw}(x_Q)\ \mathbf{Arw}(z_Q)\ e_Q = 0 \tag{2.8}$$

$$\mathbf{Diag}(x_L)\ \mathbf{Diag}(z_L)\ e_L = 0. \tag{2.9}$$

Here, $e_Q = (e_Q(1), \ldots, e_Q(q)) \in \mathcal{Q}^n$ is a block vector with

$$e_Q(i) = (1, 0, \ldots, 0)^T \in \mathbb{R}^{n_i+1} \text{ for } 1 \leq i \leq q,$$

and $e_L = (1, \ldots, 1)^T \in \mathbb{R}^{n_0}$. The complementarity condition for the quadratic cone is discussed in [1]. The complementarity condition $X_S Z_S = 0$ for the semidefinite cone can be found in [2]. The equivalent symmetrized condition $X_S Z_S + Z_S X_S = 0$ in (2.7) results in the XZ+ZX direction [4] (also called the AHO direction in the SDP literature). These issues are also discussed in [5].

It is convenient to assemble the constraints into a matrix as follows:

$$A \equiv \begin{bmatrix} A_S & A_Q & A_L \end{bmatrix} \equiv \begin{bmatrix} (\mathbf{vec}(A_S)_1)^T & (a_Q)_1^T & (a_L)_1^T \\ \vdots & \vdots & \vdots \\ (\mathbf{vec}(A_S)_m)^T & (a_Q)_m^T & (a_L)_m^T \end{bmatrix} \in \mathbb{R}^{m \times d}. \tag{2.10}$$

Similarly we write

$$x \equiv \begin{bmatrix} (\mathbf{vec}(X_S))^T & x_Q^T & x_L^T \end{bmatrix}^T \in \mathbb{R}^d,$$

$$z \equiv \begin{bmatrix} (\mathbf{vec}(Z_S))^T & z_Q^T & z_L^T \end{bmatrix}^T \in \mathbb{R}^d,$$

$$c \equiv \begin{bmatrix} (\mathbf{vec}(C_S))^T & c_Q^T & c_L^T \end{bmatrix}^T \in \mathbb{R}^d.$$

Now, combining the complementarity equations (2.7)—(2.9) with the primal and dual equality constraints, the optimality conditions reduce to a square system of nonlinear equations

$$F(x, y, z) \equiv \begin{bmatrix} A^T y + z - c \\ Ax - b \\ \frac{1}{2}\mathbf{vec}(X_S Z_S + Z_S X_S) \\ \mathbf{Arw}(x_Q)\,\mathbf{Arw}(z_Q) e_Q \\ \mathbf{Diag}(x_L)\,\mathbf{Diag}(z_L) e_L \end{bmatrix} = 0 \tag{2.11}$$

in addition to the cone nonnegativity constraints in (2.1) and (2.2).

For $\mu > 0$ consider the modified system

$$F_\mu(x, y, z) \equiv \begin{bmatrix} A^T y + z - c \\ Ax - b \\ G_\mu(x, z) \end{bmatrix} = 0, \tag{2.12}$$

where

$$G_\mu(x, z) = \begin{bmatrix} \frac{1}{2}\mathbf{vec}(X_S Z_S + Z_S X_S - 2\mu I) \\ \mathbf{Arw}(x_Q)\,\mathbf{Arw}(z_Q) e_Q - \mu e_Q \\ \mathbf{Diag}(x_L)\,\mathbf{Diag}(z_L) e_L - \mu e_L \end{bmatrix}.$$

For any given $\mu > 0$, there exists a unique solution $(x_\mu, y_\mu, z_\mu) \in \mathcal{S}^N_{++} \times \mathcal{Q}^n_{++} \times \mathbb{R}^{n_0}_{++}$ to the system (2.12) [15]. The trajectory $\{(x_\mu, y_\mu, z_\mu) \mid \mu > 0\}$ is called the central path, and any limit point of the central path as $\mu \downarrow 0$ is a solution to the SQLP.

A generic primal–dual path–following algorithm for SQLP approximately solves a sequence of systems $F_{\mu^k}(x, y, z) = 0$ where $\mu^k \downarrow 0$. For each $\mu = \mu^k$, an approximate solution to (2.12) is computed by applying one step of Newton's method. Thus one computes a search direction $(\Delta x, \Delta y, \Delta z)$ as the solution of the linear system

$$\begin{bmatrix} 0 & A^T & I \\ A & 0 & 0 \\ \mathbf{E} & 0 & \mathbf{F} \end{bmatrix} \begin{bmatrix} \Delta x \\ \Delta y \\ \Delta z \end{bmatrix} = -F_\mu(x, y, z) \tag{2.13}$$

where $\mathbf{E} \equiv (I \circledast Z_S, \mathbf{Arw}(z_Q), \mathbf{Diag}(z_L))$ and $\mathbf{F} \equiv (I \circledast X_S, \mathbf{Arw}(x_Q), \mathbf{Diag}(z_L))$ are block diagonal matrices, with $\circledast$ denoting the symmetrized Kronecker product defined in [4]. The matrix on the left–hand side of (2.13) is easily verified to be the Jacobian of F_μ. Thus Δx and Δz are of the form

$$\begin{aligned} \Delta x &= \begin{bmatrix} (\mathbf{vec}(\Delta X_S))^T & \Delta x_Q^T & \Delta x_L^T \end{bmatrix}^T \\ \Delta z &= \begin{bmatrix} (\mathbf{vec}(\Delta Z_S))^T & \Delta z_Q^T & \Delta z_L^T \end{bmatrix}^T. \end{aligned}$$

The current point is then updated as

$$\begin{aligned} x &\leftarrow x + \alpha \Delta x, \\ y &\leftarrow y + \beta \Delta y, \\ z &\leftarrow z + \beta \Delta z, \end{aligned}$$

for suitable values of the stepsizes α and β, chosen to ensure that the new iterates x and z remain inside the cone.

As is well known, the linear system (2.13) may be reduced by block Gauss elimination to a more compact system whose coefficient matrix on the left-hand side is the m by m Schur complement matrix

$$M \equiv A\,\mathbf{E}^{-1}\mathbf{F}\,A^T. \tag{2.14}$$

Observe that $M = M_S + M_Q + M_L$, where

$$M_S = A_S (I \circledast Z_S)^{-1} (I \circledast X_S) A_S^T, \tag{2.15}$$

$$M_Q = A_Q \mathbf{Arw}(z_Q)^{-1} \mathbf{Arw}(x_Q) A_Q^T, \tag{2.16}$$

$$M_L = A_L \mathbf{Diag}(z_L)^{-1} \mathbf{Diag}(x_L) A_L^T. \tag{2.17}$$

Forming and factoring M is the most expensive part of the algorithm. Since M_S and M_Q are generally not symmetric, the code uses the LU factorization provided by LAPACK to factor M. This factorization is then used to first obtain Δy and then Δx and Δz.

The algorithm is described as follows. Let $\Delta X \equiv (\Delta X_S, \Delta x_Q, \Delta x_L)$, and $\Delta Z \equiv (\Delta Z_S, \Delta z_Q, \Delta z_L)$. For an iterate (x, y, z) strictly satisfying the cone constraints, we perform the following steps, which constitute one basic interior–point iteration:

Step 1: Compute a predictor search direction $(\Delta x^{\text{pred}}, \Delta y^{\text{pred}}, \Delta z^{\text{pred}})$ by solving (2.13) with μ set to 0.

Step 2: Line search: compute positive primal and dual step lengths $\widehat{\alpha}^{\text{pred}}$ and $\widehat{\beta}^{\text{pred}}$ such that

$$X + \widehat{\alpha}^{\text{pred}} \Delta X^{\text{pred}} \quad \text{and} \quad Z + \widehat{\beta}^{\text{pred}} \Delta Z^{\text{pred}}$$

lie on the boundary of the cone $\mathcal{S}_+^N \times \mathcal{Q}_+^n \times \mathbb{R}_+^{n_0}$.

Step 3: Let

$$X' \equiv X + \alpha^{\text{pred}} \Delta X^{\text{pred}} \quad \text{and} \quad Z' \equiv Z + \beta^{\text{pred}} \Delta Z^{\text{pred}}$$

where $\alpha^{\text{pred}} = \min(\widehat{\alpha}^{\text{pred}}, 1)$ and $\beta^{\text{pred}} = \min(\widehat{\beta}^{\text{pred}}, 1)$ and set

$$\mu = \left(\frac{X' \bullet Z'}{X \bullet Z} \right)^3 \frac{X \bullet Z}{\nu}, \tag{2.18}$$

where ν is defined in (2.5).

Step 4: Recompute the (relaxed) complementarity conditions in the right hand side of (2.13) as

$$-G_\mu(x,z) - \begin{bmatrix} \frac{1}{2}\mathbf{vec}(\Delta X_S^{\text{pred}} \Delta Z_S^{\text{pred}} + \Delta Z_S^{\text{pred}} \Delta X_S^{\text{pred}}) \\ \mathbf{Arw}(\Delta x_Q^{\text{pred}})\, \mathbf{Arw}(\Delta z_Q^{\text{pred}}) e_Q \\ \mathbf{Diag}(\Delta x_L^{\text{pred}})\, \mathbf{Diag}(\Delta z_L^{\text{pred}}) e_L \end{bmatrix}$$

and solve (2.13) again for a new search direction $(\Delta x, \Delta y, \Delta z)$.

Step 5: Recompute positive primal and dual step lengths $\widehat{\alpha}$ and $\widehat{\beta}$ such that

$$X + \widehat{\alpha} \Delta X \quad \text{and} \quad Z + \widehat{\beta} \Delta Z$$

lie on the boundary of the cone $\mathcal{S}_+^N \times \mathcal{Q}_+^n \times \mathbb{R}_+^{n_0}$.

Step 6: Update the iterates as

$$x \leftarrow x + \alpha \Delta x; \quad y \leftarrow y + \beta \Delta y; \quad z \leftarrow z + \beta \Delta z$$

where $\alpha = \min(\tau \widehat{\alpha}, 1)$ and $\beta = \min(\tau \widehat{\beta}, 1)$, with $\tau < 1$, *e.g.* $\tau = 0.999$.

The performance of the algorithm depends upon the choice of μ in Step 3 of the basic iteration. The rule in (2.18) is Mehrotra's predictor–corrector method [14], originally proposed for LP, which significantly reduces the number of interior–point iterations required to compute a solution of given accuracy. In the next two sections we describe two modifications to the basic interior–point iteration. They are both extensions of well known methods for LP.

3. Mehrotra higher order corrections

Implementations of Mehrotra higher order corrections (HOC) scheme for LP have been described in [13, 7, 17]. The performance of the method has been tested by Mehrotra [13] and Carpenter, Lustig, Mulvey and Shanno [7] on some collections of Netlib test problems. Mehrotra reports that the use of higher order corrections resulted on the average in a reduction of 25% to 35% in the number of iterations, and significant savings in the *cpu* time on several problems. Carpenter, Lustig, Mulvey and Shanno report that the number of outer iterations of the algorithm can often be reduced with the use of higher order corrections but that the total number of backsolves required is usually increased. The consensus seems to be that the method is not practical unless the cost of a backsolve is negligible compared to the cost of factoring the Schur complement matrix.

The situation is more promising for SDP, and hence for SQLP, because the ratio of the cost of forming and factoring the Schur complement matrix defined in (2.14) to that of performing a backsolve to compute a direction $(\Delta x, \Delta y, \Delta z)$ and the associated step lengths α and β is usually very much greater than it is in LP.

Indeed, recalling the definitions of d_S, d_Q and d (see (2.4) and (2.6)), we see that in the case of dense blocks, the cost of forming the Schur complement matrix M is

$$O\left(m\sum_{i=1}^{s} N_i^3 + m^2 d_s + m d_q + m^2 q + m^2 n_0\right), \tag{3.1}$$

and the cost of factoring M is $O(m^3)$. On the other hand, the cost of computing the directions $(\Delta x, \Delta y, \Delta z)$ once the matrix M has been factored is

$$O\left(m^2 + \sum_{i=1}^{s} N_i^3 + md\right) \tag{3.2}$$

and the cost of computing the steps to the cone boundaries is

$$O\left(\sum_{i=1}^{s} N_i^3 + d_q + n_0\right). \tag{3.3}$$

(See Appendix A for the derivation of the operation counts in (3.1), (3.2) and (3.3).)

The higher order corrections scheme extends readily to SQLP. We assume that the directions $(\Delta x^{\text{pred}}, \Delta y^{\text{pred}}, \Delta z^{\text{pred}})$ and the corresponding step lengths $\widehat{\alpha}^{\text{pred}}$, $\widehat{\beta}^{\text{pred}}$, α^{pred} and β^{pred} have been initialized as in Steps 1 and 3 of the basic interior–point iteration, and we replace Steps 4 and 5 in the basic iteration with the following steps:

HOC Step 0: Initialize a loop counter to 1

HOC Step 1: Compute a new direction $(\Delta x,\ \Delta y, \Delta z)$ as in Step 4 and corresponding step lengths $\widehat{\alpha}$ and $\widehat{\beta}$ as in Step 5.

HOC Step 2: If $\widehat{\alpha} \geq \alpha^{\text{pred}}$ or if the loop counter is 1 then accept the primal direction and set

$$\Delta x^{\text{pred}} \leftarrow \Delta x \quad \text{and} \quad \widehat{\alpha}^{\text{pred}} \leftarrow \widehat{\alpha}.$$

If $\widehat{\beta} \geq \beta^{\text{pred}}$ or if the loop counter is 1 then accept the dual direction and set

$$\Delta z^{\text{pred}} \leftarrow \Delta z, \quad \Delta y^{\text{pred}} \leftarrow \Delta y \quad \text{and} \quad \widehat{\beta}^{\text{pred}} \leftarrow \widehat{\beta}.$$

HOC Step 3: If both the primal and the dual direction have been accepted then proceed to HOC Step 4. Otherwise, terminate the HOC iteration: restore

$$\Delta x \leftarrow \Delta x^{\text{pred}} \quad \text{and} \quad \widehat{\alpha} \leftarrow \widehat{\alpha}^{\text{pred}}$$

if the primal direction was rejected; restore

$$\Delta z \leftarrow \Delta z^{\text{pred}}, \quad \Delta y \leftarrow \Delta y^{\text{pred}} \quad \text{and} \quad \widehat{\beta} \leftarrow \widehat{\beta}^{\text{pred}}$$

if the dual direction was rejected; proceed to Step 6 of the basic iteration.

HOC Step 4: Increment the loop counter. If the iteration count exceeds a predefined bound then terminate the HOC iteration and proceed to Step 6 of the basic iteration. Otherwise go to HOC Step 1.

4. Gondzio multiple centrality corrections

The multiple centrality corrections (MCC) scheme for LP was introduced by Gondzio [10]. It is currently implemented in several LP solvers, including HOPDM [11] and PCx [8]. Its extension to SQLP can be summarized as follows. Let $(\Delta x^{\text{pred}}, \Delta y^{\text{pred}}, \Delta z^{\text{pred}})$ denote a predictor direction from the current iterate (x, y, z) and let $\widehat{\alpha}^{\text{pred}}$ and $\widehat{\beta}^{\text{pred}}$ denote the corresponding primal and dual stepsizes to the boundary of the cone $\mathcal{S}_+^N \times \mathcal{Q}_+^n \times \mathbb{R}_+^{n_0}$. Note that these quantities are typically initialized with the direction and steplengths computed in Steps 4 and 5 of the basic iteration. However, they can be initialized with the values obtained at the end of a HOC loop allowing the MCC scheme to be combined with HOC. Let μ be defined as in Step 3 of the basic interior–point iteration. We consider stepsizes of the form $\overline{\alpha} = \min(1, \widehat{\alpha}^{\text{pred}} + \delta)$ and $\overline{\beta} = \min(1, \widehat{\beta}^{\text{pred}} + \delta)$ for some parameter $\delta > 0$, so $\overline{\alpha} > \widehat{\alpha}^{\text{pred}}$ when $\widehat{\alpha}^{\text{pred}} < 1$ and $\overline{\beta} > \widehat{\beta}^{\text{pred}}$ when $\widehat{\beta}^{\text{pred}} < 1$, and we consider the new point

$$(\overline{x}, \overline{y}, \overline{z}) = (x + \overline{\alpha}\Delta x^{\text{pred}}, y + \overline{\beta}\Delta y^{\text{pred}}, z + \overline{\beta}\Delta z^{\text{pred}}).$$

Next we compute a corrector direction $(\Delta x^{\text{corr}}, \Delta y^{\text{corr}}, \Delta z^{\text{corr}})$ aimed at improving the centrality of $(\overline{x}, \overline{y}, \overline{z})$. Let ρ denote the vector of eigenvalues of the matrix $\frac{1}{2}(\overline{X}_S\overline{Z}_S + \overline{Z}_S\overline{X}_S)$ and let Q denote the corresponding orthogonal matrix of eigenvectors, so that

$$\frac{1}{2}(\overline{X}_S\overline{Z}_S + \overline{Z}_S\overline{X}_S) = Q\,\mathbf{Diag}(\rho)Q^T.$$

Let r_S (respectively r_Q, r_L) denote the vector obtained by projecting the entries of ρ (respectively $\mathbf{Arw}(\overline{x}_Q)\,\mathbf{Arw}(\overline{z}_Q)e_Q$, $\mathbf{Diag}(\overline{x}_L)\,\mathbf{Diag}(\overline{z}_L)e_L$) onto a suitable neighborhood of $\mu[1, \ldots, 1]^T$ (respectively μe_Q, μe_L), and let

$$R_S = Q\,\mathbf{Diag}(r_S)Q^T.$$

One may then consider a corrector direction $(\Delta x^{\text{corr}}, \Delta y^{\text{corr}}, \Delta z^{\text{corr}})$ defined by the solution to the system (2.13) with the right–hand side given by

$$\begin{bmatrix} 0 \\ 0 \\ \mathbf{vec}(R_S) - \frac{1}{2}(\overline{X}_S\overline{Z}_S + \overline{Z}_S\overline{X}_S) \\ r_Q - \mathbf{Arw}(\overline{x}_Q)\,\mathbf{Arw}(\overline{z}_Q)e_Q \\ r_L - \mathbf{Diag}(\overline{x}_L)\,\mathbf{Diag}(\overline{z}_L)e_L \end{bmatrix}.$$

We may then combine the predictor and the corrector directions to get

$$(\Delta x, \Delta y, \Delta z) = (\Delta x^{\text{pred}}, \Delta y^{\text{pred}}, \Delta z^{\text{pred}}) + (\Delta x^{\text{corr}}, \Delta y^{\text{corr}}, \Delta z^{\text{corr}}).$$

The motivation for the correction is to obtain an increase in the step lengths when replacing the direction $(\Delta x^{\text{pred}}, \Delta y^{\text{pred}}, \Delta z^{\text{pred}})$ by $(\Delta x, \Delta y, \Delta z)$, and so to obtain a larger reduction in complementarity.

Note that we attempt to adjust the eigenvalues of $\frac{1}{2}(\overline{X}_S\overline{Z}_S + \overline{Z}_S\overline{X}_S)$ rather than those of $\overline{X}_S\overline{Z}_S$. The advantage of using the former matrix is that it is symmetric, but neither matrix is necessarily positive definite. Both have the same trace and therefore the same eigenvalue average. Furthermore, we have the following inequalities:

$$\lambda_{\min}(\frac{1}{2}(\overline{X}_S\overline{Z}_S + \overline{Z}_S\overline{X}_S)) \leq \lambda_{\min}(\overline{X}_S\overline{Z}_S)$$
$$\lambda_{\max}(\frac{1}{2}(\overline{X}_S\overline{Z}_S + \overline{Z}_S\overline{X}_S)) \geq \lambda_{\max}(\overline{X}_S\overline{Z}_S).$$

where $\lambda_{\min}(B)$ and $\lambda_{\max}(B)$ denote the smallest and largest eigenvalues of the matrix B. (See [12], exercise 20, p. 187.)

The correction can be repeated, taking the direction $(\Delta x, \Delta y, \Delta z)$ just computed as the predictor direction for a new correction. This iteration stops when the new direction fails to yield sufficient progress or the iteration count exceeds a predefined bound. Here progress is deemed sufficient if the stepsizes for the new direction are not less than those of the previous direction and either the new primal stepsize or the new dual stepsize exceeds the corresponding stepsize for the previous direction by a preset margin.

We now describe the actual implementation of MCC that we used. Following Gondzio's code for LP (private communication), we define the function $\phi : \mathbb{R}\times\mathbb{R} \to \mathbb{R}$ by

$$\phi(\xi, \mu) = \begin{cases} \mu - \xi & \text{if } \xi < \kappa_{\min}\mu \\ \max(\kappa_{\max}\mu - \xi, -\kappa_{\max}\mu) & \text{if } \xi > \kappa_{\max}\mu \\ 0 & \text{otherwise} \end{cases}$$

where $\kappa_{\min}$, $\kappa_{\max}$ are parameters chosen to satisfy

$$0 < \kappa_{\min} \le 1 \le \kappa_{\max}.$$

We need two more parameters, γ and δ, satisfying $0 < \gamma < 1$ and $\delta > 0$. We assume that the step lengths $\widehat{\alpha}$, $\widehat{\beta}$ and the direction $(\Delta x, \Delta y, \Delta z)$ have been computed as in Steps 4 and 5 of the basic interior–point iteration or by HOC. We insert the following steps in between Steps 5 and 6 of the basic interior–point iteration:

MCC Step 1: Set

$$\Delta x^{\text{pred}} \leftarrow \Delta x,\ \ \Delta y^{\text{pred}} \leftarrow \Delta y,\ \ \Delta z^{\text{pred}} \leftarrow \Delta z,\ \ \widehat{\alpha}^{\text{pred}} \leftarrow \widehat{\alpha},\ \ \widehat{\beta}^{\text{pred}} \leftarrow \widehat{\beta}.$$

MCC Step 2: Compute the modified step lengths as

$$\overline{\alpha} = \min(1, \widehat{\alpha}^{\text{pred}} + \delta) \quad \text{and} \quad \overline{\beta} = \min(1, \widehat{\beta}^{\text{pred}} + \delta)$$

and compute the point $(\overline{x}, \overline{y}, \overline{z})$ as

$$(\overline{x}, \overline{y}, \overline{z}) = (x + \overline{\alpha}\Delta x^{\text{pred}}, y + \overline{\beta}\Delta y^{\text{pred}}, z + \overline{\beta}\Delta z^{\text{pred}}).$$

MCC Step 3: Compute v_S, v_Q and v_L as follows:

1. Let ρ denote the vector of eigenvalues of the matrix $\frac{1}{2}(\overline{X}_S\overline{Z}_S + \overline{Z}_S\overline{X}_S)$, let Q denote the corresponding orthogonal matrix of eigenvectors, and let u_S denote the vector obtained from ρ by setting

$$(u_S)_j = \phi(\rho_j; \mu), \quad 1 \le j \le \sum_{i=1}^{s} N_i.$$

and let

$$v_S = \textbf{vec}(Q\textbf{Diag}(u_S)Q^T).$$

2. Let v_Q denote the block vector obtained from $w_Q = \textbf{Arw}(\overline{x}_Q)\,\textbf{Arw}(\overline{z}_Q)e_Q$ as follows. Let $(w_0, w_1, \ldots, w_{n_i})$ (respectively $(v_0, v_1, \ldots, v_{n_i})$) denote the i^{th} block of w_Q (respectively v_Q) and set

$$v_0 = \phi(w_0, \mu).$$

For $1 \le j \le n_i$, set

$$v_j = \begin{cases} -\epsilon - w_j & \text{if } w_j < -\epsilon \\ \epsilon - w_j & \text{if } w_j > \epsilon \\ 0 & \text{otherwise} \end{cases}$$

where $\epsilon = \min((1 - \kappa_{\min})\mu, v_0/n_i)$.

3. Let v_L denote the vector obtained from $w_L = \mathbf{Diag}(\overline{x}_L)\,\mathbf{Diag}(\overline{z}_L)e_L$ by setting

$$(v_L)_j = \phi((w_L)_j;\mu), \quad 1 \le j \le n_0.$$

MCC Step 4: Compute a corrector direction $(\Delta x^{\mathrm{corr}}, \Delta y^{\mathrm{corr}}, \Delta z^{\mathrm{corr}})$ as the solution of the system of equations (2.13) with right–hand side

$$\begin{bmatrix}0 & 0 & v_S^T & v_Q^T & v_L^T\end{bmatrix}^T.$$

MCC Step 5: Compute the stepsizes $\widetilde{\alpha}$ and $\widetilde{\beta}$ to the boundary of the cones for the direction

$$(\Delta x^{\mathrm{pred}}, \Delta y^{\mathrm{pred}}, \Delta z^{\mathrm{pred}}) + (\Delta x^{\mathrm{corr}}, \Delta y^{\mathrm{corr}}, \Delta z^{\mathrm{corr}}).$$

Let $\alpha^{\mathrm{pred}} = \min(\widehat{\alpha}^{\mathrm{pred}}, 1)$ and $\beta^{\mathrm{pred}} = \min(\widehat{\beta}^{\mathrm{pred}}, 1)$. If $\widetilde{\alpha} \ge \alpha^{\mathrm{pred}}$ then accept the new primal direction and set $\Delta x \leftarrow \Delta x^{\mathrm{pred}} + \Delta x^{\mathrm{corr}}$ and $\widehat{\alpha} \leftarrow \widetilde{\alpha}$. If $\widetilde{\beta} \ge \beta^{\mathrm{pred}}$ then accept the new dual direction and set $\Delta y \leftarrow \Delta y^{\mathrm{pred}} + \Delta y^{\mathrm{corr}}$, $\Delta z \leftarrow \Delta z^{\mathrm{pred}} + \Delta z^{\mathrm{corr}}$ and $\widehat{\beta} \leftarrow \widetilde{\beta}$.

MCC Step 6: If $\widetilde{\alpha} < \alpha^{\mathrm{pred}} + \gamma\delta$ and $\widetilde{\beta} < \beta^{\mathrm{pred}} + \gamma\delta$ then terminate the MCC iteration, because the last correction has failed to achieve sufficient progress. Also terminate the MCC iteration if either the new primal or the new dual direction was rejected or if $\min(\widehat{\alpha}, \widehat{\beta})$ is greater than a predefined threshold value, *e.g.* 0.99. Finally, if the iteration count exceeds a predefined bound then terminate the MCC iteration; otherwise, go back to MCC Step 1.

Several variants of the implementation are possible. For example, one can terminate the MCC iterations only when both the new primal and dual directions have been rejected; one can also modify the criteria that determine whether sufficient progress has been achieved (see MCC Step 6). We have found that the implementation described above performed best on our set of test problems.

5. Miscellaneous Algorithmic Details

In this section, we consider two implementation issues that have considerable impact on the performance of the algorithm. First, we discuss how sparsity in the data is handled. Second, we elaborate on the strategy for setting the value of μ in Step 3 of the basic interior–point iteration.

5.1. **Sparsity.** Many problems arising in applications have very sparse constraint matrices. This is the case, for example, in problems of truss topology design and clock mesh design [16]. In order to obtain the best performance of the algorithm, it is imperative to exploit sparsity in the formation and the factorization of the Schur complement matrix M in (2.14). The code takes advantage of sparsity in the formation of M_S, the semidefinite component of M (see (2.15)).

The code stores sparse blocks in the constraint matrices A_S in the Harwell–Boeing format [9], and implements fast multiplication of a dense matrix with a sparse matrix and fast computation of the inner product of a dense matrix with a sparse matrix. If D is a dense matrix of size p by q and S is sparse of size q by r, then the dense product $B = DS$ is computed as

$$B_j = \sum_{k=1}^{q} S(k,j) D_k,$$

where $B_j, D_k \in \mathbb{R}^r$ denote the j^{th} and k^{th} columns of B and D respectively. These are the only sparse operations that are needed to compute M_S. Indeed, the (k, l) entry of M_S is obtained as

$$M_S(k,l) = \sum_{i=1}^{s} (\mathbf{vec}((A_S)_k(i)))^T (I \circledast Z_S(i))^{-1} (I \circledast X_S(i)) \mathbf{vec}((A_S)_l(i))$$

where the sum is over the blocks and $(A_S)_k(i)$ denotes the i^{th} block of the k^{th} constraint matrix. The terms in this sum are computed as follows. Let $G(i)_l$ denote the solution to the Lyapunov equation

$$G(i)_l Z_S(i) + Z_S(i) G(i)_l = X_S(i)(A_S)_l(i) + (A_S)_l(i) X_S(i). \tag{5.1}$$

Then $M_S(k,l)$ is computed as

$$M_S(k,l) = \sum_{i=1}^{s} (A_S)_k(i) \bullet G(i)_l. \tag{5.2}$$

Thus we need to multiply sparse matrices with dense matrices to compute the right–hand sides in (5.1) and we need to compute inner product of sparse matrices with dense matrices to evaluate the summands in (5.2). Observe that even if the matrices $(A_S)_k$, $1 \le k \le m$, are very sparse, Z_S, whose sparsity pattern is that of a matrix in the span of C_S and $(A_S)_k$, $1 \le k \le m$, is not, in general.

5.2. **Selecting μ.** Let us rewrite equation (2.18) in Step 3 of the basic interior–point iteration as

$$\mu = \sigma \frac{X \bullet Z}{\nu}$$

where σ is chosen by Mehrotra's cubic rule, namely

$$\sigma = \left(\frac{X' \bullet Z'}{X \bullet Z} \right)^3. \tag{5.3}$$

This choice of σ works well in practice but may be too aggressive on some of the more difficult SQLP's, causing a failure of the algorithm due to short steps, *i.e.* the step lengths computed in Step 6 of the basic interior–point iteration converge to zero. On such problems, it is preferable to adopt a more conservative strategy. For example, the algorithm can monitor the size of the step lengths $\widehat{\alpha}^{\text{pred}}$ and $\widehat{\beta}^{\text{pred}}$ computed in Step 2 of the basic interior–point iteration. If one of $\widehat{\alpha}^{\text{pred}}$ and $\widehat{\beta}^{\text{pred}}$ is small then σ is chosen closer to 1, increasing the centering component in $(\Delta x, \Delta y, \Delta z)$.

Here is an example of such a strategy as we implemented it. Modify the choice of σ in Step 5 of the basic interior–point iteration as follows:

$$\begin{array}{rllll}
\text{if} & \widehat{\alpha}^{\text{pred}} < 0.1 & \text{or} & \widehat{\beta}^{\text{pred}} < 0.1 & \text{then} \quad \sigma = 0.5 \\
\text{elseif} & \widehat{\alpha}^{\text{pred}} < 0.2 & \text{or} & \widehat{\beta}^{\text{pred}} < 0.2 & \text{then} \quad \sigma = 0.2 \\
\text{elseif} & \widehat{\alpha}^{\text{pred}} < 0.5 & \text{or} & \widehat{\beta}^{\text{pred}} < 0.5 & \text{then} \quad \sigma = 0.1 \\
\text{else} & \sigma = \left(\dfrac{X' \bullet Z'}{X \bullet Z} \right)^3. & & &
\end{array}$$

If the safeguard is activated, *i.e.* σ is not set according to Mehrotra's cubic formula, then no HOC or MCC type of corrections are attempted for the current iteration. We discuss the effect of this option in the next section.

6. Numerical Experiments

We now present numerical results for the schemes proposed in this paper. We used a variety of test problems described in Table 1. The truss topology design problems were contributed by A. Nemirovski. The clock mesh design problems are circuit design problems dealing with the propagation of a clock signal in integrated circuits; see [16] for details. All the runs were performed on a workstation with an R10000 processor, an R10010 floating point unit and 128 MB of RAM.

All the algorithms use the infeasible initial point

$$y = 0,\ (X_S, x_Q, x_L) = (Z_S, z_Q, z_L) = \eta(I, e_Q, e_L)$$

where η is a scale factor. The choice of η can greatly affect the robustness; generally a larger value of η makes the algorithm more reliable on harder problems but also require more iterations on easier ones. For the runs reported here we used $\eta = 1000$. The choice of steplength parameter τ in Step 6 of the basic iteration affects the results in similar ways: a value close to one gives faster convergence on most problems but may occasionally lead to failure. For the runs reported here, we used $\tau = 0.999$. We also set $\kappa_{\min} = 0.1$, $\kappa_{\max} = 10$, $\delta = 0.1$, and $\gamma = 0.1$ (see Section 4). The threshold value in MCC Step 6 was set to 0.99.

In Figure 1, we show the performance of the higher order corrections (HOC), the multiple centrality corrections (MCC) and the combination of HOC, MCC and SAFE (the safeguard discussed in Section 5.2) on the eight problems in Table 1. The maximum number of corrections per basic interior–point iteration was set to six for both HOC and MCC. These results are shown together with those for the basic algorithm (BASIC). The top two plots in Figure 1 show the final primal infeasibility (the norm of the left-hand side of (2.1)) and the final complementarity (the left-hand side of (2.3)) on log scales (base 10). Looking at these plots, we see that all problems were satisfactorily solved by all schemes, except that MCC did not solve problem 1 (truss5 in Table 1) and BASIC, MCC, and MCC+HOC+SAFE all failed to solve problem 8 (sqlp4 in Table 1)[1]. In all cases (except the failures just noted), the dual infeasibility norm was reduced below 10^{-12}, so this is not shown. The bottom two plots in Figure 1 show the iteration count and the relative *cpu* time, omitting the data points for the failures just noted, so that the other results may be compared more conveniently. The data shown in the latter of these two plots is normalized so that the time for HOC is 1. In other words, it shows the *cpu* time taken by a particular scheme divided by the *cpu* time taken by the HOC scheme.

We conclude that the HOC scheme is the clear winner: it is the only scheme to solve all the problems; it yields the smallest number of iterations on all the problems except problem 7 where MCC wins by one iteration; and it achieves the best *cpu* time on all the problems except problems 4 and 7 where MCC is faster by less than 10%.

In Figure 2, we show the variation in the number of interior–point iterations and the *cpu* time needed to solve truss8 (see Table 1) as a function of the maximum number of corrections allowed per basic interior–point iteration, both for HOC and MCC. We see that, for this particular test problem, the optimal strategy for HOC is to set the maximum number of corrections to six and that even when allowing

[1]The BASIC scheme still does not solve problem 8 when the initial scale factor η is raised to 10000, but it does so when τ is set to 0.95 instead of 0.999.

<table>
<tr><th>#</th><th>Name</th><th>Type</th><th>N</th><th>n</th><th>n_0</th><th>m</th><th>Description</th></tr>
<tr><td>1</td><td>truss5</td><td>S</td><td>$N_1 = 10$
$\vdots$
$N_{33} = 10$
$N_{34} = 1$</td><td>[ ]</td><td>0</td><td>208</td><td rowspan="2">Problems from truss topology design (sparse)</td></tr>
<tr><td>2</td><td>truss8</td><td>S</td><td>$N_1 = 19$
$\vdots$
$N_{33} = 19$
$N_{34} = 1$</td><td>[ ]</td><td>0</td><td>496</td></tr>
<tr><td>3</td><td>clk1</td><td>S, L</td><td>$N_1 = 81$</td><td>[ ]</td><td>288</td><td>144</td><td rowspan="2">Clock mesh design in VLSI circuits (sparse)</td></tr>
<tr><td>4</td><td>clk2</td><td>S, L</td><td>$N_1 = 169$</td><td>[ ]</td><td>624</td><td>312</td></tr>
<tr><td>5</td><td>sqlp1</td><td>S</td><td>$N_1 = 50$
$N_2 = 50$
$N_3 = 50$</td><td>[ ]</td><td>0</td><td>300</td><td rowspan="4">Randomly generated problems (dense)</td></tr>
<tr><td>6</td><td>sqlp2</td><td>Q</td><td>[ ]</td><td>$n_1 = 200$
$n_2 = 200$
$n_3 = 200$</td><td>0</td><td>300</td></tr>
<tr><td>7</td><td>sqlp3</td><td>L</td><td>[ ]</td><td>[ ]</td><td>1000</td><td>300</td></tr>
<tr><td>8</td><td>sqlp4</td><td>S, Q, L</td><td>$N_1 = 50$
$N_2 = 50$
$N_3 = 50$</td><td>$n_1 = 200$
$n_2 = 200$
$n_3 = 200$</td><td>1000</td><td>300</td></tr>
</table>

TABLE 1. The set of test problems.

up to twenty corrections per iteration HOC still performs better than the basic code. On the other hand, MCC performs best when allowing no more than one correction per iteration and the benefits of the scheme rapidly disappear as the maximum number of corrections allowed increases. The best performance of MCC is just slightly better than HOC's best performance.

Finally, in Figure 3, we compare the savings in iteration count and *cpu* time as a function of problem size for HOC and MCC. We consider randomly generated dense SDP's and QCQP's whose sizes are given in Table 2 and Table 3 respectively. For each size, averages were computed over five random instances. The maximum number of corrections per basic interior–point iteration was set to six for both HOC and MCC. We see that, in the case of SDP, all of HOC, MCC, and MCC+HOC+SAFE achieve substantial reductions in the number of iterations in all the problems and substantial reductions in the *cpu* time for problems of size greater than 50. In the case of QCQP, MCC+HOC+SAFE performs poorly, but the performance of HOC and MCC is comparable to the case of SDP. Overall, HOC is the clear winner, but it should be noted that MCC outperforms HOC on small problems.

APPENDIX A.

In this Appendix, we derive the estimates for the costs of forming the Schur complement matrix M in (3.1), solving the linear system once M has been factored (3.2), and computing the step lengths to the boundary of the cone for a given direction (3.3). We assume that all data is dense, and that the basic costs of matrix multiplication and eigenvalue computation are $O(n^3)$ for n by n matrices.

#	N	m
1	$N_1 = 20$	20
2	$N_1 = 50$	50
3	$N_1 = 100$	100
4	$N_1 = 150$	150
5	$N_1 = 200$	200
6	$N_1 = 250$	250

TABLE 2. Randomly generated dense SDP's used to study performance of HOC and MCC as a function of problem size.

#	n	m
1	$n_1 = \ldots = n_{60} = 3$	20
2	$n_1 = \ldots = n_{150} = 3$	50
3	$n_1 = \ldots = n_{300} = 3$	100
4	$n_1 = \ldots = n_{450} = 3$	150
5	$n_1 = \ldots = n_{600} = 3$	200
6	$n_1 = \ldots = n_{750} = 3$	250

TABLE 3. Randomly generated dense QCQP's used to study performance of HOC and MCC as a function of problem size.

We begin by estimating the cost of forming the matrices M_S, M_Q and M_L, defined in (2.15),(2.16) and (2.17), making up the Schur complement matrix (2.14). The cost of forming M_L is clearly $O(m^2 n_0)$. From (2.16) we get that

$$M_Q = A_Q \mathbf{Arw}(z_Q)^{-1} \mathbf{Arw}(x_Q) A_Q^T = \sum_{i=1}^{q} M_Q(i)$$

where

$$M_Q(i) = A_Q(i) \mathbf{Arw}(z_Q(i))^{-1} \mathbf{Arw}(x_Q(i)) A_Q(i)^T,$$

$\mathbf{Arw}(x_Q(i))$ is the i^{th} diagonal block of $\mathbf{Arw}(x_Q)$, and $A_Q(i)$ is the m by $n_i + 1$ matrix whose rows are the transpose of the i^{th} blocks of the vectors $(a_Q)_k$, $1 \leq k \leq m$. Let $(x_0, x_1, \ldots, x_{n_i})^T$ and $(z_0, z_1, \ldots, z_{n_i})^T$ denote the i^{th} blocks of x_Q and z_Q respectively, and let $\widetilde{x} = (0, x_1, \ldots, x_{n_i})^T \in \mathbb{R}^{n_i+1}$, $\widetilde{z} = (0, z_1, \ldots, z_{n_i})^T \in \mathbb{R}^{n_i+1}$, and $e_0 = (1, 0, \ldots, 0)^T \in \mathbb{R}^{n_i+1}$. Finally, write $B = A_Q(i)$ and let $\zeta = z_0^2 - \widetilde{z}^T \widetilde{z}$. Then we have

$$\begin{aligned} M_Q(i) = {} & \frac{x_0}{z_0} BB^T + \frac{x_0}{\zeta z_0}(B\widetilde{z})(B\widetilde{z})^T + \frac{1}{z_0}\left[(Be_0)(B\widetilde{x})^T + (B\widetilde{x})(Be_0)^T\right] \\ & + \frac{1}{\zeta}\left[\left(\frac{x_0}{z_0}\widetilde{z}^T\widetilde{z} - \widetilde{z}^T\widetilde{x}\right)(Be_0)(Be_0)^T - x_0\Big((Be_0)(B\widetilde{z})^T + (B\widetilde{z})(Be_0)^T\Big)\right] \\ & + \frac{1}{\zeta}\left[\frac{\widetilde{z}^T\widetilde{z}}{z_0}(Be_0)(B\widetilde{x})^T - (B\widetilde{z})(B\widetilde{x})^T + \frac{\widetilde{z}^T\widetilde{x}}{z_0}(B\widetilde{z})(Be_0)^T\right]. \end{aligned}$$

Observe that the matrix BB^T needs to be computed only once (for each i) at the start of the algorithm, and we therefore ignore the cost of computing it. Thus

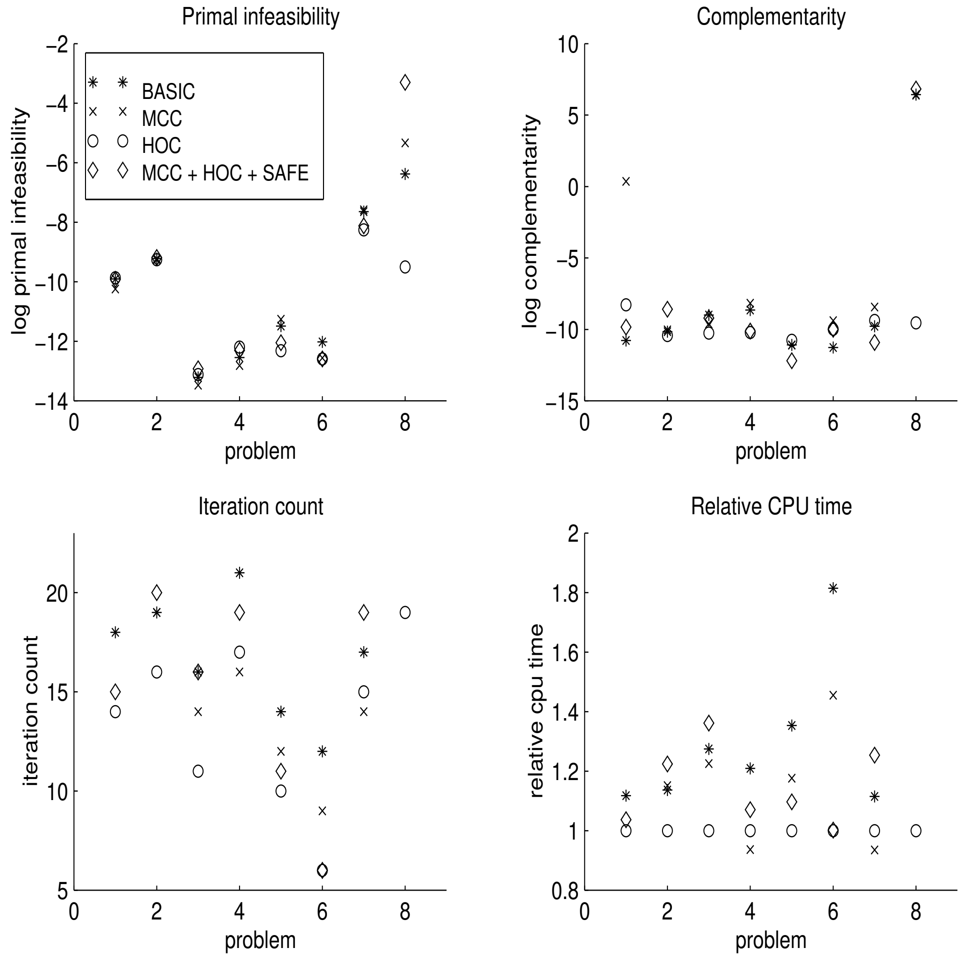

FIGURE 1. Performance of HOC, MCC and MCC+HOC+SAFE in comparison with BASIC on test set in Table 1.

the cost of computing $M_Q(i)$ is $O(mn_i + m^2)$ and the cost of computing M_Q is $O(md_q + m^2q)$.

To estimate the cost of computing M_S, observe that computing the right–hand side of (5.1) and computing the solution to the Lyapunov equation (5.1) require $O(N_i^3)$ operations and there are m such equations to solve. Computing the i^{th} summand in (5.2) requires $O(N_i^2)$ operations so that the cost of computing the (k,l) entry of M_S is $O(d_S)$. Thus the total cost of computing M_S is $O(m\sum_{i=1}^{s} N_i^3 + m^2 d_S)$.

Next we consider the cost of solving the linear system (2.13) once the factorization of the Schur complement matrix M has been obtained. Computing the right–hand side of the Schur complement system (see [4]) requires $O(\sum_{i=1}^{s} N_i^3 + d_Q + n_0)$ operations, while the cost of the backsolve to obtain Δy is $O(m^2)$. The cost of computing Δz is $O(m(d_s + d_Q + n_0)) = O(md)$ and the cost of computing Δx from Δz is $O(\sum_{i=1}^{s} N_i^3 + d_Q + n_0)$. Thus we obtain the estimate (3.2).

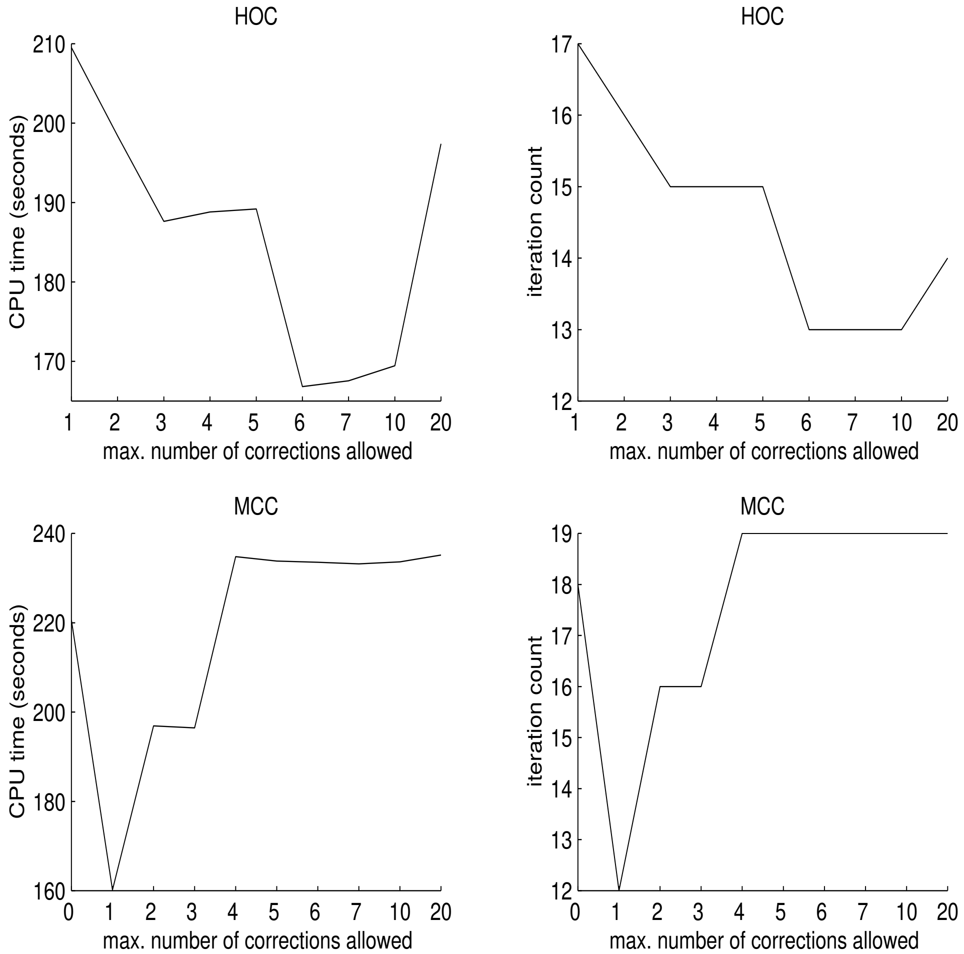

FIGURE 2. Performance of HOC and MCC as a function of the maximum number of corrections allowed per interior–point iteration. The problem selected was truss8 shown in Table 1.

Finally, computing the step lengths to the boundary of the semidefinite cone in the directions ΔX_S and ΔZ_S requires matrix–matrix multiplications and an eigenvalue decomposition and so is an $O(\sum_{i=1}^{s} N_i^3)$ operation. The cost of computing the step lengths to the boundary of the quadratic cone (respectively positive orthant) in the directions Δx_Q, Δz_Q (respectively Δx_L, Δz_L) is $O(d_Q)$ (respectively $O(n_0)$). Summing up the contributions we obtain the estimate (3.3).

REFERENCES

1. I. Adler and F.Alizadeh, *Primal–dual interior point algorithms for convex quadratically constrained and semidefinite optimization problems*, Tech. Report RRR 46–95, RUTCOR. Rutgers University, Piscataway, NJ, December 1995.
2. F. Alizadeh, *Interior–point methods in semidefinite programming with applications to combinatorial optimization*, SIAM Journal on Optimization **5** (1995), 13–51.

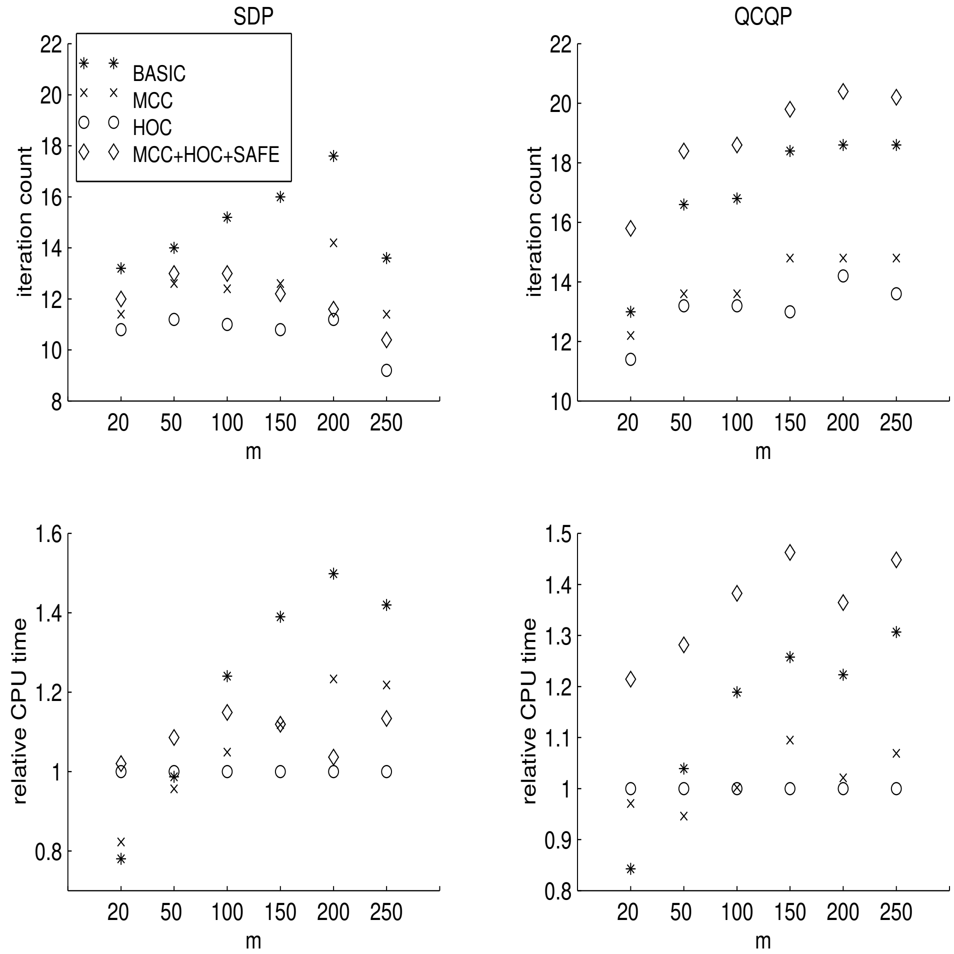

FIGURE 3. Performance of HOC and MCC as function of problem size for randomly generated, dense problems shown in Table 2 and Table 3.

3. F. Alizadeh, J.-P. Haeberly, M.V. Nayakkankuppam, M.L. Overton, and S. Schmieta, *SDPpack User Guide (Version 0.9 Beta)*, Tech. Report 737, Department of Computer Science, New York University, New York, NY, June 1997.
4. F. Alizadeh, J.-P. Haeberly, and M. L. Overton, *Primal–dual interior–point methods for semidefinite programming: convergence rates, stability, and numerical methods*, SIAM Journal on Optimization **8** (1998), 746–768.
5. F. Alizadeh and S. Schmieta, *Optimization with semidefinite, quadratic and linear constraints*, Tech. Report RRR23-97, RUTCOR. Rutgers University, Piscataway, NJ, November 1997.
6. E. Anderson, Z. Bai, C. Bischof, J. Demmel, J. Dongarra, J. Du Croz, A. Greenbaum, S. Hammarling, A. McKenney, S. Oustrouchov, and D. Sorensen, *LAPACK User's Guide*, 2nd ed., SIAM, Philadelphia, PA, 1995.
7. T.J. Carpenter, I.J. Lustig, J.M. Mulvey, and D.F. Shanno, *Higher–order predictor–corrector interior–point methods with application to quadratic objectives*, SIAM Journal on Optimization **3** (1993), 696–725.
8. J. Czyzyk, S. Mehrotra, M. Wagner, and S.J. Wright, *PCx User Guide (Version 1.1)*, Tech. Report OTC 96/01, Optimization Technology Center, Chicago, Il, November 1997.

9. I.S. Duff, R.G. Grimes, and J.G. Lewis, *User's guide for the Harwell–Boeing sparse matrix collection (release 1)*, Tech. Report RAL-92-086, Rutherford Appleton Laboratory, December 1992.
10. J. Gondzio, *Multiple centrality corrections in a primal–dual method for linear programming*, Computational Optimization and Applications **6** (1996), 137–156.
11. J. Gondzio and M. Makowski, *HOPDM modular solver for LP problems, User's guide to version 2.12*, Tech. Report WP-95-50, International Institute for Applied Systems Analysis, Laxenburg, Austria, June 1995.
12. R.A. Horn and C.R. Johnson, *Topics in matrix analysis*, Cambridge University Press, Cambridge, UK, 1994.
13. S. Mehrotra, *Higher order methods and their performance*, Tech. Report 90-16R1, Department of Industrial Engineering and Management Sciences, Northwestern University, Evanston, Ill, July 1991.
14. ———, *On the implementation of a priml-dual interior point method*, SIAM Journal on Optimization **2** (1992), 575–601.
15. Yu. E. Nesterov and A. Nemirovskii, *Interior–point polynomial algorithms in convex programming*, Studies in applied mathematics, vol. 13, SIAM, Philadelphia, PA, 1994.
16. L. Vandenberghe, S. Boyd, and A. El Gamal, *Optimal wire and transistor sizing for circuits with non-tree topology*, Proceedings of the IEEE International Conference on Computer–Aided Design (San Jose, CA), November 1997.
17. S.J. Wright, *Primal–dual interior-point methods*, SIAM, Philadelphia, PA, 1997.

Department of Mathematics, Fordham University, Bronx, NY.
E-mail address: `haeberly@murray.fordham.edu`

Computer Science Department, Courant Institute, NYU, New York.
E-mail address: `madhu@cs.nyu.edu`

Computer Science Department, Courant Institute, NYU, New York.
E-mail address: `overton@cs.nyu.edu`