# Solving large scale semidefinite programs via an iterative solver on the augmented systems

Kim-Chuan Toh [*][†]

**Abstract**

The search directions in an interior-point method for large scale semidefinite programming (SDP) can be computed by applying a Krylov iterative method to either the Schur complement equation (SCE) or the augmented equation. Both methods suffer from slow convergence as interior-point iterates approach optimality. Numerical experiments have shown that a diagonally preconditioned conjugate residual method on the SCE typically takes a huge number of steps to converge. However, it is difficult to incorporate cheap and effective preconditioners into the SCE. This paper proposes to apply the preconditioned symmetric quasi-minimal residual (PSQMR) method to a reduced augmented equation that is derived from the augmented equation by utilizing the eigenvalue structure of the interior-point iterates. Numerical experiments on SDP problems arising from maximum clique and selected SDPLIB problems show that moderately accurate solutions can be obtained with a modest number of PSQMR steps using the proposed preconditioned reduced augmented equation. An SDP problem with 127600 constraints is solved in about 6.5 hours to an accuracy of $10^{-6}$ in relative duality gap.

**Key words:** Large scale semidefinite programming, interior-point methods, augmented systems, conjugate residual method, symmetric quasi-minimal residual method, preconditioners, maximum-clique problem

**AMS Subject classification:** 90C05.

## 1 Introduction

Let $\mathcal{S}^n$ be the vector space of $n \times n$ real symmetric matrices endowed with the inner product $A \bullet B = \text{Trace}(AB)$. Given a positive integer $n$, we let $\bar{n} = n(n+1)/2$. We

[*]Department of Mathematics, National University of Singapore, 2 Science Drive 2, Singapore 117543, Singapore (`mattohkc@math.nus.edu.sg`). This research is supported in part by National University of Singapore Research Grant R-146-000-032-112.

[†]Singapore-MIT Alliance, 4 Engineering Drive 3, Singapore 117576.

use the notation $X \succeq 0$ ($X \succ 0$) to denote that $X$ is symmetric positive semidefinite (symmetric positive definite). Given $k \times l$ matrices $G, H$, we define the linear map $G \circledast H : \mathcal{S}^l \to \mathcal{S}^k$ by $G \circledast H(M) = (HMG^T + GMH^T)/2$, for $M \in \mathcal{S}^l$.

Consider the standard primal semidefinite program (SDP)

$$\begin{array}{lrcl} \min_X & C \bullet X & & \\ & \mathcal{A}(X) & = & b \\ & X & \succeq & 0, \end{array} \tag{1}$$

where $\mathcal{A} : \mathcal{S}^n \to I\!R^m$ is the linear map defined by

$$\mathcal{A}(X) \;=\; [A_1 \bullet X \;\; \cdots \;\; A_m \bullet X]^T .$$

Here $b \in I\!R^m$ and $A_1, \dots, A_m, C \in \mathcal{S}^n$ are given data. The dual of (1) is

$$\begin{array}{ll} \max_{y,Z} & b^T y \\ & \mathcal{A}^T y + \;\; Z \;=\; C \\ & Z \succeq 0, \end{array} \tag{2}$$

where $\mathcal{A}^T : I\!R^m \to \mathcal{S}^n$ is the adjoint of $\mathcal{A}$ defined by

$$\mathcal{A}^T y \;=\; \sum_{k=1}^{m} y_k A_k.$$

In this paper, we assume that (1) and (2) are strictly feasible, and the set $\{A_1, \dots, A_m\}$ is linearly independent in $\mathcal{S}^n$.

We consider primal-dual path-following methods [32, 35] for SDP using the Nesterov-Todd direction in which the general framework in each iteration is as follows. Given a current iterate $(X, y, Z)$ and a centering parameter $\sigma \in [0, 1)$, where $X, Z \succ 0$, the methods find a search direction $(\Delta X, \Delta y, \Delta Z) \in \mathcal{S}^n \times I\!R^m \times \mathcal{S}^n$ so as to generate the next iterate by solving the following linear system of equations:

$$\mathcal{A}\Delta X \;=\; R_p \;:=\; b - \mathcal{A}X \tag{3a}$$

$$\mathcal{A}^T \Delta y \,+\, \Delta Z \;=\; R_d \;:=\; C - Z - \mathcal{A}^T y \tag{3b}$$

$$\mathcal{E}\Delta X \,+\, \mathcal{F}\Delta Z \;=\; R_c \;:=\; \sigma\mu I - \Sigma^2, \tag{3c}$$

where $\mu = X \bullet Z/n$, $\mathcal{E} = G^{-T} \circledast GZ$, and $\mathcal{F} = G^{-T}X \circledast G$. Here $G$ is the unique matrix such that $\Sigma := GZG^T = G^{-T}XG^{-1}$ is a positive definite diagonal matrix. Note that $W := G^T G$ is the Nesterov-Todd (NT) scaling matrix such that $WZW = X$; see [32]. Instead of solving (3a)–(3c) directly, one can substitute $\Delta Z = R_d - \mathcal{A}^T \Delta y$ from (3b) into (3c) and solve the following augmented system:

$$-\mathcal{U}\Delta X + \mathcal{A}^T \Delta y \;=\; \mathcal{R} \;:=\; R_d - \mathcal{F}^{-1} R_c \;=\; R_d - \sigma\mu X^{-1} + Z, \tag{4a}$$

$$\mathcal{A}\Delta X \;=\; R_p, \tag{4b}$$

where $\mathcal{U} := \mathcal{F}^{-1}\mathcal{E} = W^{-1} \circledast W^{-1}$.

One can further eliminate $\Delta X$ from the augmented system above by substituting $\Delta X = \mathcal{U}^{-1}(\mathcal{A}^T\Delta y - R_d + \mathcal{F}^{-1}R_c)$ from (4a) into (4b) to obtain the following Schur complement equation (SCE) involving only $\Delta y$:

$$\underbrace{\mathcal{A}\mathcal{U}^{-1}\mathcal{A}^T}_{M}\Delta y \;\; = \;\; h \;:=\; R_p + \mathcal{A}\mathcal{U}^{-1}R_d - \mathcal{A}\mathcal{E}^{-1}R_c. \tag{5}$$

The $m \times m$ matrix $M$ is known as the Schur complement matrix and its $(i,j)$ element is given by $M_{ij} = A_i \bullet WA_jW$. Most implementations of interior-point methods for SDP use (5) to compute the search direction. Generally, (5) is solved by a direct method by first computing and storing the matrix $M$, and then computing its Cholesky factorization to find $\Delta y$. Substantial reduction in the cost of computing $M$ is possible when the SDP data is sparse; see [13] for the details. However, $M$ is generally fully dense even when the data is sparse. Thus when $m$ is larger than a few thousands, it is impossible to store $M$ in the memory of most current workstations. Furthermore, the $m^3/3$ flops required to compute the Cholesky factor of $M$ also becomes prohibitively expensive. Consequently when $m$ is large, it is extremely difficult to solve (5) by a direct method, and a Krylov subspace iterative method such as the preconditioned conjugate gradient (PCG) or preconditioned conjugate residual (PCR) method becomes necessary as these methods do not require $M$ to be stored explicitly.

Earlier research works on using the PCG or PCR method to solve the SCE arising from large scale SDPs include [8, 20, 23, 24, 36]. As the coefficient matrix $M$ is dense, traditional preconditioning techniques that are designed for sparse matrices, such as incomplete Cholesky factorizations, cannot be readily applied to $M$ without incurring a significant computational cost and memory usage. Thus in all the above mentioned papers, except [20], only simple preconditioners such as diagonal or block-diagonal preconditioners were used. In [20], attempts had been made to use incomplete Cholesky factors as preconditioners but no substantial improvement over diagonal preconditioners was observed. The preconditioners just mentioned are ineffective when the Schur complement matrix becomes increasingly ill-conditioned as the interior-point iterates approach an optimal solution. As a result, in all these works, only low accuracies in the duality gap can be achieved at reasonable costs.

The difficulties in constructing cheap and effective preconditioners for the SCE lead one to believe that second-order methods like those presented in [1, 18, 22, 32, 35] are too expensive for large scale SDPs. Thus, despite the success of second order methods in solving small and medium size SDPs, attention has been diverted to first order methods for large scale SDPs. Currently, there are 3 main classes of first order methods. In [16], the dual SDP (2) was first formulated as a non-smooth convex optimization problem and was solved by a spectral bundle (SB) method based on standard non-smooth optimization techniques. On the other hand, Burer, Monterio, and Zhang, [4] converted the dual SDP into a nonconvex nonlinear program in $I\!R^n_{++} \times I\!R^m$, and used a log-barrier method to solve the resulting nonlinear program. The third class of first order methods [3] is based on the primal SDP (1). In this class of methods, the primal positive semidefinite constraint $X \succeq 0$ is eliminated by employing the factorization $X = VV^T$ for some matrix $V \in I\!R^{n\times p}$, where $p$ is an estimate on the rank of an optimal primal solution. Such a technique transforms (1) into a nonlinear nonconvex program. In [3], an infeasible first-order augmented Lagrangian method (called BMPR method) is used to solve the resulting nonlinear program.

However, there are recent advances in using second-order methods to solve large SDPs. In [31], Toh and Kojima constructed preconditioners for the SCE based on orthogonal projectors derived from the eigenvalue structure of $W$. It was shown that these preconditioners can improve the convergence rate of the PCR method substantially in solving the SCE. However, each preconditioning step is rather expensive. Furthermore, the construction of these preconditioners require the computation of a dense $\bar{p} \times \bar{p}$ matrix and its Cholesky factorization. Though $\bar{p}$ is generally a few times smaller than $m$, it does grow proportionately with $m$, and when $m$ is very large, computing these preconditioners will require excessive memory space and time. Such a drawback poses a limit on the size of SDPs one can solve using these preconditioners. In [14], Fukuda, Kojima, and Shida used a predictor-corrector approach to numerically trace the central path in the space of Lagrange multipliers. The method uses the BFGS quasi-Newton method in the corrector procedure to locate points on the central path and the PCG method with BFGS preconditioners to solve a Schur complement type equation in the predictor procedure. Preliminary numerical results on small SDPs show that this approach is promising for solving large scale SDPs, but careful numerical implementations have yet to be done to actualize this goal.

We should mention that in a primal-dual interior-point method, memory problem can also occur when $n$ is large, since the primal variable $X$ is typically dense even if the SDP data and the resulting dual variable $Z$ are sparse. However, the root cause of this problem lies in the primal-dual framework used to solve the SDP and it cannot be easily overcome by simply using an iterative method to compute the search direction. For such a problem, it is more appropriate to use methods, such as the dual scaling method in [7], that avoid the need to form $X$ explicitly. Another method that can alleviate the memory demand of the primal variable is the matrix completion method proposed in [15]. However, the implementation of the latter method is more complex than the dual scaling method.

In this paper, we will mainly focus on SDPs where $m$ is large, but $n$ is moderate, say, less than 1000. We propose an efficient preconditioned iterative method to solve the augmented system (4a)–(4b). Like the SCE, the augmented system also suffers from ill-conditioning as the interior-point iterates approach optimality. We overcome the ill-conditioning problem by transforming the original augmented system into a better-conditioned reduced augmented system based on a newly developed block preconditioning technique in [30]. The basic idea is to analyze the eigenvalue structure of the (1,1) block $\mathcal{U}$ of the augmented system and eliminate the small eigenvalues by applying the technique in [30]. For SDP problems that are primal and dual nondegenerate and strict complementarity holds at optimality, the coefficient matrix of the reduced augmented system is shown to have bounded condition number even as the interior-point iterates approach optimality. Like the Schur complement matrix, the reduced augmented matrix is dense even if the SDP data is sparse. Thus, to further improve the conditioning of the reduced augmented matrix without incurring significant computational and storage cost, we are restricted to consider only diagonal preconditioners. Fortunately, the class of diagonal preconditioners that are proposed in [27] for augmented systems arising from soil consolidation problems in civil engineering is also quite effective for our reduced augmented systems. To solve the reduced augmented system, we use the preconditioned symmetric quasi-minimal residual (PSQMR) method [12].

Because the cost of applying the PCR method to the SCE is typically 2 to 3 times cheaper than that of applying the PSQMR method to the reduced augmented system, it is desirable to use the SCE unless the Schur complement matrix is highly ill-conditioned. By using the hybrid approach of applying the PCR method to the SCE when interior-point iterates are not close to optimality and switching to the PSQMR method applied to the reduced augmented system when they are, we are able to solve some large SDPs arising from maximum clique problems of graphs, and selected SDPLIB problems [6] to moderately high accuracies, but at reasonable costs. Numerical experiments indicate that our method is promising in solving large SDPs. But there is a slight limitation in that our method cannot be adapted for the HRVW/KSH/M direction [17, 18, 22], for reasons that we will explain later.

The paper is organized as follows. In Section 2, the derivation of the reduced augmented system is presented. The implementation of the PCR method for solving the SCE is given in Section 3. The implementation of the PSQMR method for solving the reduced augmented system and the class of diagonal preconditioners used are presented in Section 4. This is followed by numerical results in Section 5 showing the effectiveness of the preconditioned reduced augmented systems on two collections of SDPs arising from maximum clique problems of graphs. Section 6 presents further numerical results for SDPs (selected SDPLIB problems and those arising from frequency assignment problems) whose degeneracies make them potentially ill-suited for computation via an iterative solver. In Section 7, we conclude our paper.

We end this section by introducing some notations. We let $\mathbf{svec} : \mathcal{S}^n \to I\!R^{\bar{n}}$ be the isometry defined by

$$\mathbf{svec}(U) \;=\; \Big[U_{11},\; \sqrt{2}U_{12},\; U_{22},\; \sqrt{2}U_{13},\; \sqrt{2}U_{23},\; U_{33},\; \cdots,\sqrt{2}U_{1n},\; \cdots,U_{nn}\Big]^T. \quad (6)$$

Note that for $K, L \in \mathcal{S}^n$, $K \bullet L = \mathbf{svec}(K)^T\mathbf{svec}(L)$. We let $\mathbf{smat} : I\!R^{\bar{n}} \to \mathcal{S}^n$ be the inverse of $\mathbf{svec}$. We define the linear map $\mathbf{vec} : I\!R^{m\times n} \to I\!R^{mn}$ by

$$\mathbf{vec}(U) \;=\; [U_{11},\; \cdots,U_{m1},\; U_{12},\; \cdots,U_{m2},\; \cdots\; ,U_{1n},\; \cdots,U_{mn}]^T, \quad (7)$$

and $\mathbf{tvec} : I\!R^{n\times n} \to I\!R^{\bar{n}}$ by

$$\mathbf{tvec}(U) \;=\; [U_{11},\; U_{12},\; U_{22},\; U_{13},\; U_{23},\; U_{33},\; \cdots,U_{1n},\; \cdots,U_{nn}]^T. \quad (8)$$

For a vector $d$, $\mathrm{diag}(d)$ denotes the diagonal matrix with $d$ as its diagonal. The MATLAB notation $[x\,;\,y]$ is used to denote the column vector formed by appending a column vector $y$ to $x$. We use $\|\cdot\|$ to denote the vector and matrix 2-norms, and $\|\cdot\|_F$ to denote the Frobenius norm. The notation $\mathrm{diag}(P, Q)$ is used to denote the block diagonal matrix with $P$ and $Q$ as its diagonal blocks. The condition number $\kappa(P)$ of a matrix $P$ is defined to be the ratio between the largest and smallest singular values of $P$. For a vector $x \in I\!R^n_{++}$, the notation $x = \Theta(\epsilon)$ means that there exists positive constants $\gamma_1, \gamma_2$ such that $\gamma_1\epsilon \le x_i \le \gamma_2\epsilon$ for all $i = 1, \ldots, n$.

## 2 Reduced augmented system

Given an interior-point iterate $(X, y, Z)$, let $\mu := X \bullet Z/n$ and $W$ be the associated Nesterov-Todd scaling matrix. Let $W^{-1} = QDQ^T$ be the eigenvalue decomposition of $W^{-1}$. Then the eigenvalue decomposition of $\mathcal{U}$ is given by

$$\mathcal{U} \;=\; (Q \circledast Q)(D \circledast D)(Q^T \circledast Q^T). \tag{9}$$

With the above decomposition, the augmented system (4a)–(4b) can be rewritten as follows:

$$-(D \circledast D)\,(Q^T \Delta X Q) \,+\, (Q^T \circledast Q^T)\mathcal{A}^T\,\Delta y \;=\; Q^T \mathcal{R} Q \tag{10a}$$

$$\mathcal{A}(Q \circledast Q)\,(Q^T \Delta X Q) \;=\; R_p. \tag{10b}$$

Suppose $(X, y, Z)$ is close to some optimal solution $(X^*, y^*, Z^*)$ of the primal and dual SDP. If $(X^*, Z^*)$ satisfies the strict complementarity condition defined in [2] (that is, $\mathrm{rank}(X^*) + \mathrm{rank}(Z^*) = n$), then as $(X, Z)$ approaches this optimal solution (i.e., when $\mu$ is sufficiently small), the eigenvalues of $W^{-1}$ will separate into two groups, one with small magnitude of the order $\Theta(\sqrt{\mu})$ and the other with large magnitude of the order $\Theta(1/\sqrt{\mu})$. Now suppose that $W^{-1}$ has a group of $p$ small eigenvalues, and a group of $q := n - p$ large eigenvalues. Let the vector of small and large eigenvalues be $d_1$ and $d_2$, respectively. We can rewrite $W^{-1}$ as

$$W^{-1} \;=\; Q_1 D_1 Q_1^T \,+\, Q_2 D_2 Q_2^T, \tag{11}$$

according to the partition $D = \mathrm{diag}(D_1, D_2)$ and $Q = [Q_1 \;\; Q_2]$, with $D_1 = \mathrm{diag}(d_1) \in I\!R^{p\times p}$, $Q_1 \in I\!R^{n\times p}$ correspond to the small eigenvalues, and $D_2 = \mathrm{diag}(d_2) \in I\!R^{q\times q}$, $Q_2 \in I\!R^{n\times q}$ correspond to the large eigenvalues. When $\mu$ is sufficiently small, the number of eigenvalues of $W^{-1}$ with magnitudes $\Theta(\sqrt{\mu})$ is equal to the rank of $X^*$. Thus $p$ is usually equal to the rank of $X^*$. In actual computation, however, we can set $p$ to be any integer such that $\bar{p} \le m$, and it is not necessary to know the exact rank of $X^*$. Recall that by a theorem of Pataki [26], there exists an optimal solution $X^*$ whose rank $p$ satisfies the inequality $\bar{p} \le m$. Thus it is legitimate to choose $p$ such that $\bar{p} \le m$ in actual computation.

Based on the eigen-structure of $W^{-1}$, we will now propose a method to overcome the ill-conditioning problem in the SCE and augmented equation when $\mu$ is small. We start from the augmented system (10a)–(10b) by diagonalizing $\mathcal{U}$ based on the eigenvalue decomposition of $W^{-1}$.

As a reminder, we have $d_1 = \mathrm{diag}(D_1)$ and $d_2 = \mathrm{diag}(D_2)$.

**Theorem 2.1** *With the partition in (11), the augmented system (10a)–(10b) can be rewritten as*

$$\begin{bmatrix} -\mathcal{D}_{11} & & & \mathcal{B}_{11}^T \\ & -\mathcal{D}_{12} & & \mathcal{B}_{12}^T \\ & & -\mathcal{D}_{22} & \mathcal{B}_{22}^T \\ \mathcal{B}_{11} & \mathcal{B}_{12} & \mathcal{B}_{22} & \end{bmatrix} \begin{bmatrix} \mathbf{svec}(Q_1^T \Delta X Q_1) \\ \sqrt{2}\mathbf{vec}(Q_1^T \Delta X Q_2) \\ \mathbf{svec}(Q_2^T \Delta X Q_2) \\ \Delta y \end{bmatrix} = \begin{bmatrix} \mathbf{svec}(Q_1^T \mathcal{R} Q_1) \\ \sqrt{2}\mathbf{vec}(Q_1^T \mathcal{R} Q_2) \\ \mathbf{svec}(Q_2^T \mathcal{R} Q_2) \\ R_p \end{bmatrix}, \tag{12}$$

*where*

$$\begin{aligned}
\mathcal{B}_{11}^T &= \left[\ \mathbf{svec}(Q_1^T A_1 Q_1) \quad \cdots \quad \mathbf{svec}(Q_1^T A_m Q_1)\ \right] \in \mathbb{R}^{\bar{p}\times m}, \\
\mathcal{B}_{12}^T &= \left[\ \sqrt{2}\mathbf{vec}(Q_1^T A_1 Q_2) \quad \cdots \quad \sqrt{2}\mathbf{vec}(Q_1^T A_m Q_2)\ \right] \in \mathbb{R}^{pq\times m}, \\
\mathcal{B}_{22}^T &= \left[\ \mathbf{svec}(Q_2^T A_1 Q_2) \quad \cdots \quad \mathbf{svec}(Q_2^T A_m Q_2)\ \right] \in \mathbb{R}^{\bar{q}\times m},
\end{aligned}$$

*and*

$$\mathcal{D}_{11} = \mathrm{diag}(\mathbf{tvec}(d_1 d_1^T)), \ \ \mathcal{D}_{12} = \mathrm{diag}(\mathbf{vec}(d_1 d_2^T)), \ \ \mathcal{D}_{22} = \mathrm{diag}(\mathbf{tvec}(d_2 d_2^T)).$$

**Proof.** Using the fact that for any $U \in \mathcal{S}^n$,

$$Q^T U Q = \begin{bmatrix} Q_1^T U Q_1 & Q_1^T U Q_2 \\ Q_2^T U Q_1 & Q_2^T U Q_2 \end{bmatrix},$$

we get from (10a) the following equations:

$$\begin{aligned}
-D_1(Q_1^T \Delta X Q_1) D_1 + \sum_{k=1}^m (Q_1^T A_k Q_1)\Delta y_k &= Q_1^T \mathcal{R} Q_1 \\
-D_1(Q_1^T \Delta X Q_2) D_2 + \sum_{k=1}^m (Q_1^T A_k Q_2)\Delta y_k &= Q_1^T \mathcal{R} Q_2 \\
-D_2(Q_2^T \Delta X Q_2) D_2 + \sum_{k=1}^m (Q_2^T A_k Q_2)\Delta y_k &= Q_2^T \mathcal{R} Q_2.
\end{aligned}$$

It is readily shown that these three equations correspond to the first three block equations in (12). Now, from (10b), we have

$$\mathcal{A}(Q \circledast Q)(Q^T \Delta X Q) = \begin{bmatrix} (Q^T A_1 Q) \bullet (Q^T \Delta X Q) \\ \vdots \\ (Q^T A_m Q) \bullet (Q^T \Delta X Q) \end{bmatrix}$$

$$= \begin{bmatrix} (Q_1^T A_1 Q_1) \bullet (Q_1^T \Delta X Q_1) + 2(Q_1^T A_1 Q_2) \bullet (Q_1^T \Delta X Q_2) + (Q_2^T A_1 Q_2) \bullet (Q_2^T \Delta X Q_2) \\ \vdots \\ (Q_1^T A_m Q_1) \bullet (Q_1^T \Delta X Q_1) + 2(Q_1^T A_m Q_2) \bullet (Q_1^T \Delta X Q_2) + (Q_2^T A_m Q_2) \bullet (Q_2^T \Delta X Q_2) \end{bmatrix}$$

$$= \mathcal{B}_{11}\mathbf{svec}(Q_1^T \Delta X Q_1) + \sqrt{2}\,\mathcal{B}_{12}\mathbf{vec}(Q_1^T \Delta X Q_2) + \mathcal{B}_{22}\mathbf{svec}(Q_2^T \Delta X Q_2).$$

This corresponds to the last block equation in (12). □

Through (10a)–(10b), the system (12) is orthogonally equivalent to the augmented system (4a)–(4b), and thus the condition numbers of the coefficient matrices are the same. To improve the conditioning of (12), we apply the block splitting introduced in [30] to (12) to get a smaller reduced augmented system as shown in the next theorem.

**Theorem 2.2** *Let $\beta \in I\!R^p$ be a given positive vector. Suppose*

$$\begin{aligned} E_{11} &= \operatorname{diag}(\mathbf{tvec}(\beta\beta^T + \beta d_1^T + d_1\beta^T)), && (13)\\ S_{11} &:= \mathcal{D}_{11} + E_{11} = \operatorname{diag}(\mathbf{tvec}((d_1+\beta)(d_1+\beta)^T)). && (14)\end{aligned}$$

*The augmented system (12) can be solved via the following reduced augmented system:*

$$\underbrace{\begin{bmatrix} \mathcal{H} & \mathcal{B}_{11}S_{11}^{-1/2} \\ S_{11}^{-1/2}\mathcal{B}_{11}^T & -\Psi \end{bmatrix}}_{\mathcal{K}} \begin{bmatrix} \Delta y \\ S_{11}^{-1/2}E_{11}\mathbf{svec}(Q_1^T\Delta X Q_1) \end{bmatrix}$$

$$= \begin{bmatrix} R_p + \mathcal{B}\operatorname{diag}(S_{11}^{-1},\, \mathcal{D}_{12}^{-1},\, \mathcal{D}_{22}^{-1})\mathbf{svec}(Q^T\mathcal{R}Q) \\ S_{11}^{-1/2}\mathbf{svec}(Q_1^T\mathcal{R}Q_1) \end{bmatrix}, \qquad (15)$$

*where $\mathcal{B} = [\mathcal{B}_{11}\ \mathcal{B}_{12}\ \mathcal{B}_{22}]$, and*

$$\mathcal{H} = \mathcal{B}\operatorname{diag}(S_{11}^{-1},\, \mathcal{D}_{12}^{-1},\, \mathcal{D}_{22}^{-1})\,\mathcal{B}^T, \quad \Psi = \mathcal{D}_{11}E_{11}^{-1}.$$

*Note that*

$$\mathcal{H} = \mathcal{A}(P_1 \circledast P_1)\mathcal{A}^T + \mathcal{A}((2P_2 + P_3)\circledast P_3)\mathcal{A}^T, \qquad (16)$$

*where*

$$P_1 = Q_1\operatorname{diag}(\beta + d_1)^{-1}Q_1^T, \quad P_2 = Q_1 D_1^{-1}Q_1^T, \quad P_3 = Q_2 D_2^{-1}Q_2^T.$$

*Note that once $\Delta y$ and $Q_1^T\Delta X Q_1$ are computed, $Q^T\Delta X Q$ can be computed as follows:*

$$\begin{aligned} Q_1^T\Delta X Q_2 &= D_1^{-1}\left(Q_1^T(\mathcal{A}^T\Delta y - \mathcal{R})Q_2\right)D_2^{-1} && (17)\\ Q_2^T\Delta X Q_2 &= D_2^{-1}\left(Q_2^T(\mathcal{A}^T\Delta y - \mathcal{R})Q_2\right)D_2^{-1}. && (18)\end{aligned}$$

**Proof.** The derivation of (15) follows readily by applying Theorem 2.1 in [30] to the system in (12). Next we will derive (16). Note that

$$\mathcal{H} = \mathcal{B}_{11}\operatorname{diag}(S_{11}^{-1})\mathcal{B}_{11}^T + \mathcal{B}_{12}\operatorname{diag}(\mathcal{D}_{12}^{-1})\mathcal{B}_{12}^T + \mathcal{B}_{22}\operatorname{diag}(\mathcal{D}_{22}^{-1})\mathcal{B}_{22}^T.$$

We shall just show that $\mathcal{B}_{11}\operatorname{diag}(S_{11}^{-1})\mathcal{B}_{11}^T = \mathcal{A}(P_1 \circledast P_1)\mathcal{A}^T$, and it is easy to simplify the other two terms similarly. For any $v \in I\!R^m$, we have

$$\begin{aligned} \mathcal{B}_{11}^T v &= \mathbf{svec}(G) \\ \operatorname{diag}(S_{11}^{-1})\mathcal{B}_{11}^T v &= \mathbf{svec}(\Lambda G\,\Lambda), \end{aligned}$$

where $G = Q_1^T(\mathcal{A}^T v)Q_1$ and $\Lambda = \mathrm{diag}(\beta + d_1)^{-1}$. Hence

$$\mathcal{B}_{11}\mathrm{diag}(S_{11}^{-1})\mathcal{B}_{11}^T v \;=\; \mathcal{B}_{11}\mathbf{svec}(\Lambda G\,\Lambda) \;=\; \begin{bmatrix} \mathbf{svec}(Q_1^T A_1 Q_1)^T \mathbf{svec}(\Lambda G\,\Lambda) \\ \vdots \\ \mathbf{svec}(Q_1^T A_m Q_1)^T \mathbf{svec}(\Lambda G\,\Lambda) \end{bmatrix}$$

$$= \begin{bmatrix} A_1 \bullet Q_1\Lambda Q_1^T(\mathcal{A}^T v)Q_1\Lambda Q_1^T \\ \vdots \\ A_m \bullet Q_1\Lambda Q_1^T(\mathcal{A}^T v)Q_1\Lambda Q_1^T \end{bmatrix} = \begin{bmatrix} A_1 \bullet P_1(\mathcal{A}^T v)P_1 \\ \vdots \\ A_m \bullet P_1(\mathcal{A}^T v)P_1 \end{bmatrix}$$

$$= \mathcal{A}(P_1 \circledast P_1)\mathcal{A}^T v.$$

Thus, we have derived the first term in (16). $\square$

Notice that the reduced augmented matrix $\mathcal{K} \in \mathcal{S}^{m+\bar{p}}$ in (15) is smaller in size compared to the augmented matrix in (12), whose dimension is $m + \bar{n}$. It is also potentially better conditioned as Theorem 2.4 below shows. Before we present that theorem, it is beneficial for us to recall the concept of primal and dual nondegeneracy introduced in [2].

**Theorem 2.3** *[2] Suppose $(X^*, Z^*)$ satisfies the strict complementarity condition. Then $X^*$ is primal nondegenerate if and only if the matrix $[\mathcal{B}_{11}\ \ \mathcal{B}_{12}]$ has full row rank, and a necessary condition for primal nondegeneracy is $\bar{n} - \bar{q} \geq m$. The solution $Z^*$ is dual nondegenerate if and only if the matrix $\mathcal{B}_{11}$ has full column rank, and a necessary condition for dual nondegeneracy is $\bar{p} \leq m$.*

**Proof.** The proof follows readily from Theorems 6 and 9 in [2]. $\square$

**Theorem 2.4** *Under the assumption that $(X^*, Z^*)$ satisfies the strict complementarity condition, and the primal and dual nondegeneracy conditions defined in [2], the coefficient matrix in (15) has a condition number that is bounded independent of $\mu$ (when $\mu$ is small).*

**Proof.** When $\mu$ is sufficiently small and $(X, Z)$ is close to a strictly complementary optimal solution $(X^*, Z^*)$ with $p = \mathrm{rank}(X^*)$, by Theorem 6 in [2], primal nondegeneracy implies that $[\mathcal{B}_{11}\ \ \mathcal{B}_{12}]$ has full row rank; and by Theorem 9 in [2], dual nondegeneracy implies that $\mathcal{B}_{11}$ has full column rank. By Theorem 3.2 in [30], the theorem follows. $\square$

For an SDP that is primal and dual nondegenerate, and strict complementarity condition holds, Theorem 2.4 implies that one can expect a Krylov subspace method applied to (15) to have a better rate of convergence than one that is applied to (5). There is another advantage in using the reduced augmented system. Because the (1,1) and (1,2) blocks of $\mathcal{K}$ are not ill-conditioned, the task of constructing effective preconditioners for $\mathcal{K}$ is likely to be easier than that for the highly ill-conditioned matrix $M$.

**Remark.** (a) Notice that the derivation of the reduced augmented system (15) depends on our ability to find the eigenvalue decomposition of $W^{-1} \circledast W^{-1}$. For the HRVW/KSH/M direction direction described in [17, 18, 22], $W^{-1} \circledast W^{-1}$ is replaced by $(X \circledast Z^{-1})^{-1}$. Unfortunately, unlike the former, the eigenvalue decomposition of the latter is not readily available even if those of $X$ and $Z$ are known. Because of this reason, the augmented system (12) cannot be reduced to the form in (15) for the HRVW/KSH/M direction. However, for the dual scaling direction in [7], $W^{-1} \circledast W^{-1}$ is replaced by $Z \circledast Z$ and the corresponding reduced augmented system can be found readily once the eigenvalue decomposition of $Z$ is known.

(b) For the numerical experiments in this paper, the vector $\beta$ in Theorem 2.2 is chosen to be:

$$\beta \;=\; \max(1, \max(d_1))\,(1, 1, \ldots, 1)^T.$$

(c) Our reduced augmented system (15) can also be applied to interior-point methods for linear programs (LPs). Such a system can potentially produce a search direction that is numerically more accurate than that computed straightforwardly from the SCE (5). This topic is currently being investigated. Another method that had been proposed to overcome the stability/accuracy problems encountered when solving the SCE arising from LP is the stabilization method of Vujcic and Asic [34]. The stabilization method in [34] is based on a novel pivoting strategy to avoid excessive loss of numerical accuracies due to the mixing of elements corresponding to large and small scaling factors when the Schur complement matrix $M$ is factorized. As far as we are aware of, the reduced augmented system approach and the stabilization method in [34] are not directly related, although both can be used to avoid excessive numerical errors for computing the search directions in interior-point methods for LP.

## 2.1 Nondegeneracy and condition number of the reduced augmented matrix

Now we present some examples to illustrate the validity of Theorem 2.4, as well as examples to demonstrate what may happen to the condition number $\kappa(\mathcal{K})$ when the nondegeneracy conditions in Theorem 2.4 do not hold. In order to know the ranks of $X^*$ and $Z^*$ unambiguously, we need to compute very accurate approximate optimal solutions. But it is well known that the standard approach of computing the search direction from (5) in each interior-point iteration usually does not deliver very accurate approximate optimal solutions because of highly ill-conditioned Schur complement matrices. Thus we have to rely on an alternative approach to compute the search directions.

It turns out that the approach of computing the directions from (15) via the $LDL^T$ factorization of $\mathcal{K}$ can usually deliver more accurate approximate solutions than the standard approach. On a limited set of examples that we have tested, the accuracy gained is usually more than 2 digits in the infeasibilities and duality gap. Better accuracy is plausible because $\mathcal{K}$ is potentially better conditioned than $M$, and so the search direction computed via (15) is potentially more accurate than that computed from (5). When the assumption in Theorem 2.4 holds, the condition number of the coefficient matrix in (15) is bounded independent of $\mu$. This implies that the unknowns $\Delta y$ and $Q_1^T \Delta X Q_1$ can be computed accurately even when $\mu$ is small. From (18), it is easy to see that $Q_2^T \Delta X Q_2$ can be computed accurately since $d_2 = \Theta(1/\sqrt{\mu})$. From (17), we have

$$(Q_1^T \Delta X Q_2)_{ij} \;=\; \frac{\Big(Q_1^T(\mathcal{A}^T \Delta y - \mathcal{R})Q_2\Big)_{ij}}{d_1^{(i)} d_2^{(j)}},$$

thus $Q_1^T \Delta X Q_2$ can also be computed accurately since $d_1^{(i)} d_2^{(j)} = \Theta(\sqrt{\mu})\Theta(1/\sqrt{\mu}) = \Theta(1)$. Therefore $\Delta X$ can be computed accurately from $Q_1^T \Delta X Q_1$, $Q_1^T \Delta X Q_2$, and $Q_2^T \Delta X Q_2$. Finally, $\Delta Z$ can also be computed accurately from $\Delta Z = R_d - \mathcal{A}^T \Delta y$.

As our purpose in this paper is on the application of iterative methods for solving large SDPs, we shall not discuss further the issue of solving an SDP via (15) by using the $LDL^T$ factorization. We leave this issue for a more detailed investigation in the future.

To illustrate the validity of Theorem 2.4, in Table 1, we give the condition number of $\mathcal{K}$ in (15) and $M$ in (5) for some of the interior-point iterates generated by the semidefinite programming software, SDPT3 version 3.0 [33]. The SDP problem is the problem `theta2` (with $m = 498$ and $n = 100$) taken from the SDPLIB [6]. The default parameters in SDPT3 are used. But when $\mu$ is small, the search direction in each interior-point iteration is computed via (15) instead of via the system (5) that is implemented in SDPT3.

The table shows that $\kappa(\mathcal{K})$ is bounded at the level $2.5 \times 10^6$ when $\mu = X \bullet Z/n$ is approaching 0 while $\kappa(M)$ grows like $3 \times 10^4/\mu$. In the table,

$$\phi \;=\; \max\Big(\frac{\|R_p\|}{1+\|b\|}, \frac{\|R_d\|_F}{1+\|C\|_F}\Big). \tag{19}$$

The approximate optimal solution $(X, y, Z)$ of `theta2` is strictly complementary and it satisfies the necessary conditions in Theorem 2.3 for primal and dual nondegeneracy. Suppose the eigenvalues of $X$ and $Z$ are ordered in decreasing and increasing order, respectively. We have $\min_i\{\lambda_i(X) + \lambda_i(Z)\} = 3.2 \times 10^{-3}$, and $p = 16$, $q = 84$. For this problem, the matrix $[\mathcal{B}_{11}\ \mathcal{B}_{12}] \in I\!R^{m\times 1480}$ has singular values in the range $[0.1, 4.1]$, while those of $\mathcal{B}_{11} \in I\!R^{m\times 136}$ are contained in $[5\times 10^{-2}, 4.1]$.

Next we give an example to illustrate what may happen to $\kappa(\mathcal{K})$ when the conditions in Theorem 2.4 are not satisfied. For this purpose, we use the SDPLIB problem `qap6` (with $m = 229$ and $n = 37$). The approximate solution delivered by SDPT3 is strictly complementary with $\min_i\{\lambda_i(X) + \lambda_i(Z)\} = 5.0 \times 10^{-4}$, and we have $p = 12$ and $q = 25$. Although $\bar{n} - \bar{q} = 378 \geq m$ satisfies the necessary condition in Theorem 2.3 for primal nondegeneracy, the problem `qap6` is in fact nearly primal degenerate, because the matrix $[\mathcal{B}_{11}\ \mathcal{B}_{12}] \in I\!R^{m\times 378}$ has 13 small singular values that are in the range $[1\times 10^{-5}, 7\times 10^{-5}]$, while the rest are in the range $[1.2\times 10^{-1}, 4.2\times 10^{1}]$. Note that `qap6` is dual nondegenerate, which can be seen from the fact that the singular values of $\mathcal{B}_{11}$ are contained in the interval $[7.0\times 10^{-2}, 2.3\times 10^{1}]$.

| iteration | $X \bullet Z/n$ | $\phi$ | $\kappa(\mathcal{K})$ | $\kappa(\mathcal{H})$ | $\kappa(\mathcal{B}_{11} S_{11}^{-1/2})$ | $\kappa(M)$ |
|---|---|---|---|---|---|---|
| 13 | 2.8e-08 | 4.8e-15 | 2.0e+06 | 1.9e+06 | 6.8e+01 | 1.4e+12 |
| 14 | 3.0e-09 | 4.0e-16 | 2.4e+06 | 2.3e+06 | 6.8e+01 | 1.0e+13 |
| 15 | 2.5e-10 | 4.0e-16 | 2.5e+06 | 2.3e+06 | 6.9e+01 | 9.9e+13 |
| 16 | 5.9e-12 | 4.2e-16 | 2.5e+06 | 2.4e+06 | 6.9e+01 | 4.2e+14 |
| 17 | 1.7e-13 | 2.1e-16 | 2.4e+06 | 2.3e+06 | 6.9e+01 | 4.5e+14 |

Table 1: *Condition number of reduced augmented and Schur complement matrices corresponding to interior-point iterates generated by SDPT3 for the SDP problem* `theta2`*. The approximate optimal solution has a relative duality gap of 3.6e-14. This SDP is primal and dual nondegenerate.*

Because of near primal degeneracy, we see from Table 2 that for `qap6`, $\kappa(\mathcal{K})$ is no longer bounded independent of $\mu$ due to the fact that the (1,1) block $\mathcal{H}$ of $\mathcal{K}$ is nearly singular. In fact, both $\kappa(\mathcal{K})$ and $\kappa(\mathcal{H})$ have order equal to the reciprocal of the machine precision ($\approx 2 \times 10^{-16}$). This example illustrates that for a nearly primal or dual degenerate problem, the matrix $\mathcal{K}$ can also be very ill-conditioned, just like the matrix $M$.

| iteration | $X \bullet Z/n$ | $\phi$ | $\kappa(\mathcal{K})$ | $\kappa(\mathcal{H})$ | $\kappa(\mathcal{B}_{11} S_{11}^{-1/2})$ | $\kappa(M)$ |
|---|---|---|---|---|---|---|
| 24 | 1.8e-06 | 5.2e-12 | 8.6e+16 | 7.3e+16 | 3.2e+02 | 1.7e+20 |
| 25 | 4.6e-07 | 4.8e-12 | 4.6e+17 | 9.7e+17 | 3.2e+02 | 5.0e+20 |
| 26 | 1.9e-07 | 1.1e-11 | 4.8e+18 | 2.9e+18 | 3.2e+02 | 1.2e+20 |
| 27 | 1.0e-07 | 7.5e-12 | 7.5e+18 | 1.6e+18 | 3.2e+02 | 1.3e+20 |
| 28 | 7.6e-08 | 4.3e-12 | 3.0e+18 | 1.0e+19 | 3.2e+02 | 2.0e+21 |
| 29 | 6.7e-08 | 7.3e-12 | 2.2e+18 | 1.1e+18 | 3.2e+02 | 3.3e+20 |

Table 2: *Same as Table 1, but for the SDP problem* `qap6`*. The approximate optimal solution has a relative duality gap of 6.5e-9. This SDP appears to be primal degenerate but dual nondegenerate.*

Our third example is the problem `mcp250-1` (with $m = 250$, $n = 250$) from SDPLIB. This problem has a strictly complementary approximate optimal solution, with $p = 25$ and $q = 225$. This problem is primal nondegenerate since the singular values of $[\mathcal{B}_{11}\ \mathcal{B}_{12}]$ are contained in the interval $[1.3 \times 10^{-1}, 1]$. But it is clearly dual degenerate since $\bar{p} > m$ violates the necessary condition for dual nondegeneracy in Theorem 2.3. This is also reflected in Table 3 with $\kappa(\mathcal{B}_{11} S_{11}^{-1/2})$ numerically equal to infinity.

A closer inspection of the problem data of `mcp250-1` reveals that it has 20 constraints that fix for a given $i$, $X_{ii} = 1$, and $X_{ij} = 0$ for $j \neq i$. That is, $X$ is actually

| iteration | $X \bullet Z/n$ | $\phi$ | $\kappa(\mathcal{K})$ | $\kappa(\mathcal{H})$ | $\kappa(\mathcal{B}_{11}S_{11}^{-1/2})$ | $\kappa(M)$ |
|---|---|---|---|---|---|---|
| 14 | 2.4e-07 | 1.1e-15 | 1.1e+11 | 9.8e+03 | Inf | 5.9e+08 |
| 15 | 3.8e-08 | 9.7e-16 | 2.9e+11 | 6.1e+03 | Inf | 2.3e+09 |
| 16 | 3.3e-09 | 6.4e-16 | 5.3e+12 | 7.2e+03 | Inf | 2.8e+10 |
| 17 | 1.0e-10 | 6.3e-16 | 1.9e+14 | 7.7e+03 | Inf | 9.0e+11 |
| 18 | 3.3e-12 | 6.7e-16 | 5.5e+15 | 7.3e+03 | Inf | 2.7e+13 |

Table 3: *Same as Table 1, but for the SDP problem* `mcp250-1`*. The approximate optimal solution has a relative duality gap of 1.5e-13. This SDP is primal nondegenerate but it is dual degenerate.*

a block diagonal matrix where one of the block is the $20 \times 20$ identity matrix. The presence of such a fixed block makes the problem dual degenerate. By removing the fixed block, the resulting problem has $m = 230$ and $n = 230$. The new problem becomes dual nondegenerate, and now the condition number of $\mathcal{K}$ is bounded independent of $\mu$, as shown in Table 4. The singular values of $\mathcal{B}_{11}S_{11}^{-1/2}$ is now in the interval $[3.5 \times 10^{-2}, 2.2 \times 10^{-1}]$. This example shows that preprocessing SDP data is an important step to avoid degeneracies, and hence also potential numerical difficulties. Preprocessing to avoid degeneracies is especially important when one chooses to use an iterative solver to compute the search direction since degeneracies can seriously increase the condition number of the coefficient matrix, and hence worsen the convergence rate.

| iteration | $X \bullet Z/n$ | $\phi$ | $\kappa(\mathcal{K})$ | $\kappa(\mathcal{H})$ | $\kappa(\mathcal{B}_{11}S_{11}^{-1/2})$ | $\kappa(M)$ |
|---|---|---|---|---|---|---|
| 14 | 2.5e-07 | 6.4e-16 | 2.7e+06 | 9.2e+03 | 6.2e+00 | 4.5e+08 |
| 15 | 3.5e-08 | 6.8e-16 | 1.2e+06 | 6.1e+03 | 6.2e+00 | 2.2e+09 |
| 16 | 2.7e-09 | 6.1e-16 | 1.7e+06 | 7.5e+03 | 6.2e+00 | 3.7e+10 |
| 17 | 9.0e-11 | 7.0e-16 | 1.8e+06 | 7.8e+03 | 6.2e+00 | 1.0e+12 |
| 18 | 3.0e-12 | 7.7e-16 | 1.7e+06 | 7.3e+03 | 6.2e+00 | 3.0e+13 |
| 19 | 2.3e-13 | 6.7e-16 | 1.7e+06 | 7.5e+03 | 6.2e+00 | 3.9e+14 |

Table 4: *Same as Table 3 for the SDP problem* `mcp250-1`*, but with fixed diagonal block removed. The approximate optimal solution has a relative duality gap of 5.7e-14. This SDP is primal and dual nondegenerate.*

Our last example is on an SDP that is both primal and dual degenerate. This problem, `fap01`, is an SDP relaxation of a frequency assignment problem considered in [5]. This SDP has a semidefinite variable in $\mathcal{S}_+^{52}$ and a linear variable in $I\!R_+^{1160}$. The number of constraints is $m = 1378$. The approximate optimal solution is strictly complementary with $\min_i\{\lambda_i(X) + \lambda_i(Z)\} = 3.9 \times 10^{-4}$. We have $p = 48$, $q = 4$

for the semidefinite block, and $p = 30$, $q = 1130$ for the linear block. The matrix $[\mathcal{B}_{11}\ \mathcal{B}_{12}] \in I\!R^{m\times 1368}$ has 6 singular values that are smaller than $5 \times 10^{-16}$, with the rest contained in the interval $[8.2 \times 10^{-2}, 2]$. It is clear that the problem is primal degenerate since $[\mathcal{B}_{11}\ \mathcal{B}_{12}]$ does not have full row rank. The matrix $\mathcal{B}_{11}$ has 4 singular values that are smaller than $10^{-11}$, with the rest lie in the interval $[1.7 \times 10^{-2}, 2]$. Since $\mathcal{B}_{11}$ has very small singular values, the problem can be considered to be dual degenerate. The condition numbers of $\mathcal{K}$, $\mathcal{H}$, and $\mathcal{B}_{11}S_{11}^{-1/2}$ in Table 5 clearly reflect the fact the problem is primal and dual degenerate.

| iteration | $X \bullet Z/n$ | $\phi$ | $\kappa(\mathcal{K})$ | $\kappa(\mathcal{H})$ | $\kappa(\mathcal{B}_{11}S_{11}^{-1/2})$ | $\kappa(M)$ |
|---|---|---|---|---|---|---|
| 24 | 2.8e-08 | 8.7e-11 | 4.8e+06 | 5.7e+06 | 9.1e+13 | 5.6e+12 |
| 25 | 1.3e-09 | 8.1e-16 | 5.0e+11 | 2.2e+08 | 9.7e+14 | 1.4e+15 |
| 26 | 3.0e-11 | 1.9e-15 | 1.5e+13 | 1.0e+10 | 1.4e+14 | 1.9e+18 |
| 27 | 6.6e-13 | 2.1e-15 | 3.9e+15 | 4.5e+11 | 2.2e+13 | 4.0e+16 |
| 28 | 1.6e-14 | 2.1e-15 | 3.7e+16 | 1.4e+13 | 5.2e+11 | 4.5e+17 |
| 29 | 3.9e-16 | 1.7e-15 | 2.1e+18 | 3.5e+14 | 6.2e+10 | 1.5e+20 |

Table 5: *Same as Table 1, but for the SDP problem* `fap01`. *The approximate optimal solution has a relative duality gap of 1.6e-14. This SDP appears to be both primal and dual degenerate.*

The reader would have noticed that for all the examples, except `qap6`, we are able to compute very accurate approximate solutions (with $\phi$ and relative duality gap both smaller than $2\times 10^{-13}$). It is rather surprising that this is possible for the last example since $\mathcal{K}$ is highly ill-conditioned.

## 3 Solving the SCE via the conjugate residual method

The use of an iterative method to solve the SCE (5) requires less computer memory compared to using a direct method. It also has the added advantage that one can terminate the iterative solver whenever an approximate solution of (5) is deemed sufficiently accurate. This can lead to a significant saving in the CPU time required in each interior-point iteration, especially during the initial phase where accurate computation of the search direction is not necessary. In [19], Kojima, Shida, and Shindoh (KSS) proposed inexact search directions where (3a) and (3b) are satisfied exactly but (3c) is relaxed. If $\Delta y$ is an approximate solution of (5), the KSS inexact search direction requires the computation of the matrix $U := \mathcal{A}^T(\mathcal{A}\mathcal{A}^T)^{-1}(h - M\Delta y)$ for computing $\Delta X$ and determining whether $\|(\mathcal{E}\Delta X + \mathcal{F}\Delta Z) - R_c\|_F = \|\mathcal{E}(U)\|_F$ is sufficiently small. However, such a computation can be expensive when either $\mathcal{A}\mathcal{A}^T$ is not easily invertible or when computing $\mathcal{E}(U)$ is expensive. Due to these drawbacks, we decide to use the heuristic rule described below to compute an inexact search direction.

Suppose $\Delta y$ only satisfies (5) approximately. Let $r = h - M\Delta y$. Given such an $\Delta y$, we compute $\Delta Z$ and $\Delta X$ via the following equations:

$$\Delta Z \;=\; R_d - \mathcal{A}^T\Delta y, \qquad \Delta X \;=\; \mathcal{E}^{-1}R_c - \mathcal{E}^{-1}\mathcal{F}\Delta Z. \tag{20}$$

Then $(\Delta X, \Delta y, \Delta Z)$ satisfies (3a)–(3c) approximately, where the residual vector is $[r\ ;\ 0\ ;\ 0]^T$. As our interest is to solve (3a)–(3c), it is reasonable to insist that the relative residual norm of the approximate solution $(\Delta X, \Delta y, \Delta Z)$ must be smaller than some prescribed threshold, say $\theta$. Let

$$\|(R_p, R_d, R_c)\| \;:=\; \max(\|R_p\|,\, \|R_d\|_F,\, \|R_c\|_F). \tag{21}$$

That is, we want

$$\|r\| \;\le\; \theta\, \|(R_p, R_d, R_c)\|. \tag{22}$$

Note that for the dual variables, once dual feasibility is achieved, it is maintained because (3b) is satisfied exactly. However, for the primal variable, primal infeasibility may deteriorate since (3a) is satisfied only approximately. But we can ensure that the primal infeasibility is reduced proportionately to $\|(R_p, R_d, R_c)\|$ in each iteration. Suppose the new primal iterate is $X^+ = X + \alpha\Delta X$, where $\alpha \in (0, 1]$ is the step-length, then we have

$$\|b - AX^+\| \;\le\; (1-\alpha)\|R_p\| + \alpha\|r\| \;\le\; \Big(1 - \alpha(1-\theta)\Big)\|(R_p, R_d, R_c)\|.$$

The behavior of the preconditioned conjugate residual (PCR) method on the SCE was discussed in detail in [31]. Because the matrix $M$ is dense, it is difficult to adapt existing preconditioning techniques that are mainly designed for sparse matrices to $M$, and the only obvious and easily implementable choices are diagonal preconditioners. In [31], PCR method was applied to following preconditioned version of (5):

$$\underbrace{L^{-1}ML^{-T}}_{\widehat{M}}(L^T\Delta y) \;=\; L^{-1}h, \tag{23}$$

where $L = \mathrm{diag}(\sqrt{M_{11}}, \ldots, \sqrt{M_{mm}})$. It was observed that the PCR method on (23) is highly efficient in computing an approximate solution when the iterate $(X, y, Z)$ is not close to optimality, i.e, when the duality gap $X \bullet Z$ is not too small. However, when the iterate is close to optimality, the PCR method becomes exceedingly slow because the matrix $\widehat{M}$ becomes very ill-conditioned (with condition number of the order $1/\mu$) and also a more accurate solution of the system (23) is needed when the duality gap is small.

As we shall compare with the reduced augmented equation approach later in Section 4, the strength of solving (23) by an iterative method such as the PCR method lies on its simplicity and inexpensive matrix-vector products (where each cost about $3\rho_s n^3 + 2\rho_t mn^2$ flops; $\rho_s$ and $\rho_t$ are defined in Section 4.1). Thus it is desirable to use the PCR method whenever its convergence rate is not too slow.

## 4 Computing the search direction via the reduced augmented system

Assume that $\Delta y$ and $\Delta X$ are computed inexactly from Theorem 2.2 and the residual vector from (15) is denoted by

$$\begin{bmatrix} \xi \\ \eta \end{bmatrix}. \tag{24}$$

Then simple algebraic manipulations show that we have

$$\begin{aligned} -\mathcal{U}\Delta X + \mathcal{A}^T\Delta y &= \mathcal{R} - Q_1\,\mathbf{smat}(S_{11}^{1/2}\eta)\,Q_1^T \\ \mathcal{A}\Delta X &= R_p - \xi + \mathcal{A}\,\mathbf{svec}(Q_1\mathbf{smat}(S_{11}^{-1/2}\eta)Q_1^T). \end{aligned}$$

Now if we compute $\Delta Z$ via the equation

$$\Delta Z = R_d - \mathcal{A}^T\Delta y - Q_1\,\mathbf{smat}(S_{11}^{1/2}\eta)Q_1^T, \tag{25}$$

then we have

$$\begin{aligned} \mathcal{E}\Delta X + \mathcal{F}\Delta Z &= \mathcal{F}\left(\mathcal{U}\Delta X + R_d - \mathcal{A}^T\Delta y - Q_1\,\mathbf{smat}(S_{11}^{1/2}\eta)Q_1^T\right) \\ &= \mathcal{F}\,(R_d - \mathcal{R}) \;=\; R_c, \end{aligned}$$

where $\mathcal{R}$ is defined as in (4a). Thus, for the inexact search direction $(\Delta X, \Delta y, \Delta Z)$ computed from (15) and (25), it satisfies (3a)–(3c) approximately and the residual vector is

$$\begin{bmatrix} \xi - \mathcal{A}\,\mathbf{svec}(Q_1\mathbf{smat}(S_{11}^{-1/2}\eta)Q_1^T) \\ Q_1\,\mathbf{smat}(S_{11}^{1/2}\eta)\,Q_1^T \\ 0 \end{bmatrix}. \tag{26}$$

Again, we want the relative residual norm of our inexact search direction $(\Delta X, \Delta y, \Delta Z)$ to be sufficiently small. That is, we want

$$\max(\|\xi - \mathcal{A}\,\mathbf{svec}(Q_1\mathbf{smat}(S_{11}^{-1/2}\eta)Q_1^T)\|,\ \|S_{11}^{1/2}\eta\|) \;\le\; \theta\,\|(R_p, R_d, R_c)\|. \tag{27}$$

**Remark.** Notice that we computed $\Delta Z$ as in (25) so as to satisfy the linearized complementarity equation (3c) exactly. However, if it is desirable to maintain dual feasibility, then we can compute $\Delta Z$ via $\Delta Z = R_d - \mathcal{A}^T\Delta y$ to make (3b) exact, but (3c) approximately satisfied. In the latter case, if we let $V = \mathbf{smat}(S_{11}^{1/2}\eta)$, then the residual associated with (3c) is given by

$$\mathcal{F}(Q_1VQ_1^T) \;=\; [\Sigma(GQ_1)V(GQ_1)^T + (GQ_1)V(GQ_1)^T\Sigma]/2,$$

which can be computed in $2p^2n + pn^2$ flops (with symmetry taken into account) if $GQ_1$ is pre-computed. Because of the extra cost incurred in the present case, this explains why we prefer to compute $\Delta Z$ via (25).

Observe that (3a) and (3b) are not satisfied exactly, primal and dual feasibilities are not maintained even if the iterate happens to be feasible. However, in each iteration, the infeasibilities are reduced proportionately with $\|(R_p, R_d, R_c)\|$. From (26) and (27), the primal infeasibility for the new iterate satisfies

$$\begin{aligned} \|b - \mathcal{A}X^+\| &\leq (1-\alpha)\|R_p\| + \alpha\|\xi - \mathcal{A}\,\mathbf{svec}(Q_1\mathbf{smat}(S_{11}^{-1/2}\eta)Q_1^T)\| \\ &\leq \Big(1 - \alpha(1-\theta)\Big)\|(R_p, R_d, R_c)\|. \end{aligned}$$

It is easy to see that a similar inequality holds for the new dual iterate.

## 4.1 Preconditioned symmetric quasi-minimal residual method

Recall that the reduced augmented equation (15) is symmetric but indefinite. In this subsection, we will discuss an appropriate Krylov subspace method to solve such a linear system.

The standard Krylov subspace method for solving a symmetric indefinite system are SYMMLQ and MINRES due to Paige and Saunders [25]. When preconditioning is used, both the methods above require the preconditioner to be symmetric positive definite, and this excludes the use of indefinite preconditioners that are perhaps more appropriate since the coefficient matrix itself is indefinite. Here, we choose the preconditioned symmetric quasi-minimal residual (PSQMR) method proposed in [12] that allows the use of symmetric indefinite preconditioners. Note that if no preconditioning is used, the SQMR method and MINRES are mathematically equivalent.

Let $\mathcal{I}$ be the set of indices of nonzero elements of the matrix $\sum_{k=1}^m |A_k|$ (where $|A_k|$ is the matrix whose $(i,j)$ element is the magnitude of the corresponding element of $A_k$), and

$$\begin{aligned} \rho_s &= (\text{number of nonzero elements of the matrix } \textstyle\sum_{k=1}^m |A_k|)/n^2, \\ \rho_t &= (\text{total number of nonzero elements of } A_1, A_2, \ldots, A_m)/(mn^2). \end{aligned}$$

Note that $\rho_s$ and $\rho_t$ are the ratios of the actual number of non-zero elements over the maximum possible number of non-zero elements.

In each PSQMR iteration, we compute the matrix-vector product $\mathcal{K}[u\,;\,v]$ for the reduced augmented system (15) via the procedure described in Table 6, where the cost is also estimated.

The cost of a matrix-vector product for the reduced augmented system is $3p^2n + 3\rho_s\,pn^2 + 7\rho_s\,n^3 + 2\rho_t\,mn^2$, as estimated in Table 6. In contrast, the corresponding cost for the SCE (5) is $3\rho_s n^3 + 2\rho_t mn^2$, as estimated in [31]. In our numerical experiments in Section 5, we have found that the cost of the former range from 2 to 4 times more

| Computing | Number of flops required |
|---|---|
| $T := \mathcal{A}^T u$ | $\rho_t\, mn^2$ |
| $\{U_{ij}^{(1)} \mid (i,j) \in \mathcal{I}\}$, where $U_1 := P_1 \circledast P_1(T)$ | $3\rho_s\, n^3$ |
| $\{U_{ij}^{(2)} \mid (i,j) \in \mathcal{I}\}$, where $U_2 := (2P_2 + P_3) \circledast P_3(T)$ | $4\rho_s\, n^3$ |
| $\{U_{ij}^{(3)} \mid (i,j) \in \mathcal{I}\}$, where $U_3 := Q_1 \circledast Q_1 \mathbf{smat}(S_{11}^{-1/2} v)$ | $2p^2 n + \rho_s\, pn^2$ |
| $\mathcal{A}(U_1 + U_2 + U_3)$ | $\rho_t\, mn^2$ |
| $S_{11}^{-1/2}\mathbf{svec}(Q_1^T \circledast Q_1^T(T)) - \Psi v$ | $p^2 n + 2\rho_s\, pn^2$ |
| $\mathcal{K}[u\,;\,v]$ | $3p^2 n + 3\rho_s\, pn^2 + 7\rho_s\, n^3 + 2\rho_t\, mn^2$ |

Table 6: *computational cost required in the matrix-vector product for (15).*

expensive than the latter. For the projected SCE approach proposed in [31], a matrix-vector product would cost about $6p^2 n + 6\rho_s pn^2 + 4\rho_s n^3 + 6\rho_t mn^2 + 2\bar{p}^2$, and this is usually more expensive than that for the reduced augmented system.

In the current literature, most preconditioning techniques are proposed for a sparse matrix that is stored explicitly, and preconditioners such as incomplete Cholesky factors are generally quite effective for matrices that are not too ill-conditioned [29]. However, as the reader may have recalled, our matrix $\mathcal{K}$ is dense and is not formed explicitly. Thus, most of the current preconditioning techniques [29, Chapter 10] are not applicable to our linear system. The only obvious and easily implementable choices for our system are diagonal preconditioners.

In [27], some effective diagonal preconditioners were proposed for a symmetric indefinite matrix of the form $\mathcal{K}$ that arises from the finite element solution of the Biot's soil consolidation equations. Those diagonal preconditioners were derived from some theoretical forms that are proven to have tight eigenvalue clustering properties. By adapting those preconditioners for our matrix $\mathcal{K}$, we get

$$\begin{bmatrix} \mathrm{diag}(\mathcal{H}) & 0 \\ 0 & \alpha\, \mathrm{diag}\Big(S_{11}^{-1/2}\mathcal{B}_{11}^T \mathrm{diag}(\mathcal{H})^{-1}\mathcal{B}_{11} S_{11}^{-1/2} + \Psi\Big) \end{bmatrix}, \tag{28}$$

where $\alpha$ is a given scalar. In our numerical experiments in Section 5, we take $\alpha = -20$. Notice that the diagonal preconditioner (28) is indefinite.

## 5 Numerical experiments on SDPs arising from maximum clique problems

We will now present numerical experiments to show the convergence behavior of the PCR method on (23) versus the PSQMR method on (15).

All the numerical results presented in this paper are computed using MATLAB on a 700MHz HP workstation c3700 with 1G of RAM. Note that computational intensive parts such as the PCR and PSQMR methods are implemented in C, but with interface to MATLAB. To give an idea on the speed of this machine, we run the MATLAB benchmark command, `bench`. Compared to a 300MHz SGI R1200 IRIX 64 machine, our machine is about 2 times faster on LU factorization and has about the same speed on sparse matrix operations.

The interior-point method we used is the primal-dual path-following method (without corrector) described in [33], except that the direct solver used to solve (5) is replaced by an iterative solver. The following starting iterates (slightly modified from the default in [33]) are used throughout:

$$y^0 = 0, \quad X^0 = \xi I, \quad Z^0 = \eta I,$$

where

$$\begin{aligned} \xi &= n \max\left(\sqrt{n},\ n \ \max_k \left\{ \frac{1+|b_k|}{1+\|A_k\|_F} \right\}\right), \\ \eta &= \sqrt{n}\ \max\left(n,\ \|C\|_F,\ \max_k\{\|A_k\|_F\}\right). \end{aligned}$$

For easy reference, we will call the interior-point method in [33] that uses the PCR method to solve the preconditioned SCE (23) as Algorithm PFsch ("PF" for "path-following"). The parameter $\theta$ in (22) is set to 0.01. In view of the efficiency of the PCR method in computing an inexact search direction via (23) when the duality gap $X \bullet Z$ is not too small, for the experiments, we use a hybrid method that combines the advantage of applying the PCR method to (23) and the PSQMR method to (15) for computing the search direction in each interior-point iteration. The details of the hybrid method are given in Algorithm PFaug in Table 7. The parameter $\theta$ in (27) is set to $\theta = 0.05$.

Our test problems consist of the following 2 collections of SDPs:

1. the first consists of SDPs arising from maximum clique problems on randomly generated graphs;
2. the second consists of SDPs associated with maximum clique problems for graphs from the Second DIMACS Implementation Challenge [9].

We choose these SDP collections because they are likely to be primal and dual nondegenerate. These are problems with $m$ large and $n$ moderate. Thus they are also well suited for solution via a primal-dual interior point method with iterative solver. There are 2 commonly used equivalent SDP relaxations [28, (2.6) and (2.9)] for the maximum clique problems. The relaxation we used for a given simple undirected graph $(G, V)$ follows equation (2.6) in [28]. That is,

$$\min\{-(ee^T) \bullet X \ : \ \mathrm{Trace}(X) = 1,\ X_{ij} = 0 \ \forall\, (i,j) \in E,\ X \succeq 0\},$$

where $e$ is the vector of all ones. We also tested on the second formulation given in [28, (2.9)], but found that the SDPs are typically either primal or dual degenerate.

Let

$$N_k = \begin{cases} \text{the number of PCR/PSQMR steps required at the } k\text{th} \\ \text{interior-point iteration to solve (23)/(15) so that the ad-} \\ \text{missible condition (22)/(27) is satisfied.} \end{cases}$$

The maximum numbers of PCR and PSQMR steps allowed in each interior-point iteration are set to $5m$ and $3m$, respectively.

Table 8 shows the primal and dual objective values obtained by Algorithm PFaug. Table 9 to 12 compare the cumulative CPU time taken by Algorithms PFsch and PFaug at various interior-point iterations so as to achieve the following accuracy:

$$\max(\texttt{relgap}, \phi) \;\leq\; 10^{-4},\; 10^{-5},\; 10^{-6}.$$

Here `relgap` is the relative duality gap defined by

$$\texttt{relgap} \;=\; \frac{X \bullet Z}{1 + (|C \bullet X| + |b^T y|)/2}, \tag{29}$$

and $\phi$ is the infeasibility measure defined in (19). For each problem, 3 rows of data are reported, and they correspond to the CPU time needed to solve the problem to an accuracy of $10^{-4}$, $10^{-5}$, and $10^{-6}$, respectively.

Tables 9–11 show that Algorithm PFaug is much faster than Algorithm PFsch on majority of the problems tested. For example, consider the problem `theta82` with $m = 23872$, Algorithm PFaug is about 7 times faster than Algorithm PFsch to achieve an accuracy of $10^{-6}$ in $\max(\texttt{relgap}, \phi)$. On the set of maximum clique problems on randomly generated graphs considered in Table 9, Algorithm PFaug is 3–14 times faster than Algorithm PFsch. For those SDPs arising from [9] in Table 11, Algorithm PFaug is 2–9 times faster than Algorithm PFsch. The reader may have observed that the speedup in these problems is mainly gained on the last few interior-point iterations. Comparing Tables 9 and 11, we see that the number of PSQMR steps needed to solve (15) is far less than that required by (23) when the iterates are close to optimality. This also confirmed the usefulness of Theorem 2.4. But because computing a matrix-vector product for (15) is more expensive, the saving in the CPU time is not as impressive as the reduction in the number iterative steps.

It is worth noting that in Table 9, we are able to solve an SDP (`theta162`) with 127600 constraints in 6.5 hours to an accuracy of $10^{-6}$. If the required accuracy is $10^{-4}$, then only 3.5 hours is needed.

Observe that in Table 10, the reduced augmented system (15) in Algorithm PFaug is never invoked, indicating that the condition number of the NT scaling matrix $W$ never exceed $5 \times 10^3$ in Algorithm PFaug described in Table 7. It is surprising that the collection of `hamming` problems can be solved so efficiently via the SCE alone. For example, the problem `hamming-9-5-6` is solved to an accuracy of $10^{-6}$ in 3 minutes, whereas the CPU time reported in [5] by using the first order nonlinear programming method in [4] (we will called it the BMZ method for convenience) is more than 10 hours. Although the comparison here between Algorithm PFaug and the BMZ method is not entirely fair because the latter solves a different, though equivalent, SDP relaxation

[28, (2.9)] of the the maximum clique problem, the fact that such a disparity is possible indicates that one should not totally abandon second order methods in favor of first order methods when solving large scale SDPs.

Despite the success of Algorithm PFaug reported in Tables 9–11. For the problems in Table 12, the performance of Algorithm PFaug is worse than Algorithm PFsch. For example, it performs badly compared to Algorithm PFsch on the problem `p-hat300-1`. To understand why the reduced augmented equation approach does not perform well, we need to know whether the problem is degenerate. This can be done by estimating the condition number of $\mathcal{H}$ and $\mathcal{B}_{11}S_{11}^{-1/2}$. Since the matrices are large, we used the Lanczos method to estimate the largest and smallest eigenvalues of the matrices $\mathcal{H}$ and $S_{11}^{-1/2}\mathcal{B}_{11}^T\mathcal{B}_{11}S_{11}^{-1/2}$. The ratio between these eigenvalues would then give a lower bound on $\kappa(\mathcal{H})$ and $\kappa(\mathcal{B}_{11}S_{11}^{-1/2})^2$. A lower bound we get for $\kappa(\mathcal{H})$ is $1.5 \times 10^8$. As for $\kappa(\mathcal{B}_{11}S_{11}^{-1/2})$, we are able to get an accurate estimate of $9.0 \times 10^1$. From these numbers, we may conclude that the problem is dual nondegenerate, but it is possibly primal degenerate due to the large condition number estimate we have for $\mathcal{H}$. The poorer performance of Algorithm PFaug on `G51--G54` compared to Algorithm PFsch is also due to degeneracies.

Methods for solving large SDPs are still in the infancy state. Currently, the most successful methods are the spectral bundle (SB) method in [16], the BMZ method in [4], and the BMPR method in [3]. Detailed comparison between the SB (version 1.1.1) and BMZ methods are given in [5], where the BMZ method appeared to perform generically better than the SB method on the tested set of SDPs arising from maximum clique problems from the Second DIMACS Implementation Challenge. Comparison between the BMPR and SB (version 1.1.1) methods on the same set of SDPs are reported in [3]. Based on the results reported in [3] and [5], it is known that the BMPR is superior to the BMZ method on this set of SDPs. The latter is in turns superior to the SB method. Thus in this section, we shall compare our reduced augmented equation approach mainly with the BMPR method.

For the set of SDPs listed in Tables 10 to 12, except `G43--G47`, `G51--G54`, Algorithm PFaug is comparable to the BMPR method (Tables 4 and 7 in [3]) in terms of computational time (although we must take into account that different machines are used, and our machine is 1-2 times faster based on MATLAB's `bench` command). For example, the problems `brock400-1` and `c-fat-200-1` are solved by Algorithm PFaug in 1016 and 67 seconds, respectively. The corresponding numbers for the BMPR method reported in [3] are 2028 and 742 seconds.

For `G43--G47`, our method is able to solve them in about 1.5 hour to an accuracy of $10^{-6}$. The BMPR method however, is much faster, taking an average of 15 minutes to solve these problems (see Table 7 in [3]), though less accurately than ours. The fact that the latter is far more superior to Algorithm PFaug on these problems can be explained. Firstly, because the matrix variable has a relatively large dimension of $n = 1000$, computing the NT scaling matrix $W$ and eigenvalue decomposition of $W^{-1}$ in Algorithm PFaug takes more than 50% of the total computation time. Secondly, the rank of the optimal primal variable (about 60) is small compared to $n$, the BMPR method can fully exploit such an advantage whereas Algorithm PFaug is not designed to do the same. The reasons above apply also to the problems `G51--G54`. For the problems `G52` and `G53`, our interior-point based algorithms perform much worse than the BMPR method. For example, Algorithm PFsch takes 6.5 hours to achieve an accuracy of $10^{-4}$ whereas the BMPR method takes only 2 hours to achieve a comparable accuracy. If the required accuracy is $10^{-6}$, then Algorithm PFsch would take about 33.5 hours to solve the problem. Comparing the results in Table 11 and 12, obviously `G52` and `G53` are much harder to solve compared to `G43--47`. We suspect that this is because the former are highly degenerate problems. For example, for `G52`, we have $\bar{p} = 48828 \gg m = 5917$, which violates the necessary condition for dual nondegeneracy in Theorem 2.3.

We note that the objective values reported in Table 8 are generally better than those reported in [3]. For example, The primal objective value we obtained for `brock400-1` is $-39.7018863$, with a primal infeasibility of $5.1 \times 10^{-10}$, whereas the corresponding number obtained by the BMPR method is $-39.652$, with a primal infeasibility of $1.4 \times 10^{-4}$.

Other than computational time, we should mention a comparison criterion between interior-point methods (such as Algorithm PFaug) and first order methods (such as the BMPR method) that is perhaps under appreciated. An advantage of the former is that it can produce a duality gap that measures how close the approximate optimal solution is to optimality. The latter, however, can only obtain either an approximate primal or dual optimal solution, and there is no optimality guarantee on the approximate solution delivered.

## 6 Numerical experiments on other SDPs

In this section, we further investigate the performance of Algorithms PFaug and PFsch, but on some SDPs that are not necessarily well suited for the reduced augmented equation or the primal-dual interior-point framework. The problems we considered are as follows.

`mcp`: this collection consists of preprocessed version of the SDPLIB problems, `mcp500-1`–`mcp500-4`. These are SDPs arising from relaxation of maximum cut problems. The original SDPs are dual degenerate, but a simple preprocessing step to remove fixed diagonal blocks render them dual nondegenerate. For these problems, $m \approx 500$ and $n = 500$.

`arch`: this consists of the SDPLIB problems, `arch0, arch2, arch4`, and `arch8`. Each of these problems has a semidefinite variable of dimension 161 and a linear variable of dimension 174, and $m = 174$.

`fap`: these are SDPs arising from semidefinite relaxation of frequency assignment problems [11]. The explicit form of the primal SDP is given in [5, eq. (5)] (note the difference between maximizing and minimizing the objective function in [5, eq. (5)] and (1)). Note that this collection of SDPs are likely to be both primal and dual degenerate (evident from Table 5 for `fap01`). Each of these problems has a semidefinite variable with moderate dimension $n$, and a linear variable with dimension slightly less than $m$, where $m$ is the number of constraints and $m \gg n$.

Table 7: Algorithm PFaug.

**Algorithm PFaug.** Suppose we are given an initial iterate $(X^0, y^0, Z^0)$ with $X^0, Z^0$ positive definite. Set $\gamma^0 = 0.9$ and $\sigma^0 = 0.5$.

**For** $k = 0, 1, \ldots$

Let the current and the next iterate be $(X, y, Z)$ and $(X^+, y^+, Z^+)$ respectively. Also, let the current and the next step-length (centering) parameter be denoted by $\gamma$ and $\gamma^+$ ($\sigma$ and $\sigma^+$) respectively.

1. Set $\mu = X \bullet Z/n$. Stop the iteration if the infeasibility measure $\phi$ defined in (19) and `relgap` defined in (29) are sufficiently small.
2. Compute the Nesterov-Todd scaling matrix $W$ and the eigenvalue decomposition $W^{-1} = QDQ^T$. Let $d = \mathrm{diag}(D)$, where $d$ is sorted in ascending order.
```
If  $\max(d)/\min(d) > 5 \times 10^3$
    choose $p$ to be the integer such that $d_{p+1}/d_p$ is the maximum,
else
    set $p = 0$,
end
```
3. (a) If $p = 0$;
   Compute an inexact direction $(\Delta X, \Delta y, \Delta Z)$ via the PCR method on (23) with diagonal preconditioner $\mathrm{diag}(M)$.

   (b) If $p > 0$;
   Compute an inexact search direction $(\Delta X, \Delta y, \Delta Z)$ via the PSQMR method on (15) with diagonal preconditioner described in (28).
4. Update $(X, y, Z)$ to $(X^+, y^+, Z^+)$ by
$$X^+ = X + \alpha\,\Delta X, \quad y^+ = y + \beta\,\Delta y, \quad Z^+ = Z + \beta\,\Delta Z,$$
where $\alpha = \min(1, -\gamma/\lambda_{\min}(X^{-1}\Delta X))$, $\beta = \min(1, -\gamma/\lambda_{\min}(Z^{-1}\Delta Z))$. (Here $\lambda_{\min}(U)$ denotes the minimum eigenvalue of $U$; if the minimum eigenvalue in either expression is positive, we ignore the corresponding term.)
5. Update the step-length and centering parameters by
$$\gamma^+ = 0.9 + 0.08\min(\alpha, \beta), \quad \sigma^+ = 1 - 0.9\min(\alpha, \beta).$$

Before we discuss the numerical results for the above SDPs, we would like to mention that many of the SDPs in SDPLIB [6] appear to be either primal or dual degenerate; or ill-posed in the sense that the primal and dual problems are not both strictly feasible. The `mcp` problems are dual degenerate if fixed diagonal blocks are not removed. The `qap` problems are nearly primal degenerate; the `control` problems either do not appear to have strictly complementary approximate optimal solutions, or they are primal degenerate. The `gpp` problems do not have strictly primal feasible points.

The `mcp` and `arch` problems are problems with $m \approx n$ and they are not large scale. Thus they are not ideal examples to evaluate the viability of using iterative methods to solve large SDPs in a primal-dual interior-point method. However, they are included here to evaluate the merit of the reduced augmented equation (15) over the SCE (5) when solved via an iterative method. The CPU time given in Table 14 for these problems is not indicative of the time spent in solving these linear systems because a substantial part is spent on computing $W$ and its eigenvalue decomposition. For the `mcp` problems, the iterative solvers use only less than 30% of the total CPU time. Thus the number of iterative steps used to solve the linear systems would be a better indicator of the relative merit between (5) and (15). From Table 14, we observe that the PSQMR method on (15) takes significantly fewer steps to converge than the PCR method on (5) when $\mu$ is small. This confirms again the merit of the reduced augmented equation over the the SCE when $\mu$ is small.

The `fap` problems are SDPs that are both primal and dual degenerate. Because these problems are expected to be hard to solve via an interior-point method using an iterative solver, now the accuracy tolerance is set to $\max(\texttt{relgap}, \phi) \leq 10^{-2}, 10^{-3}, 10^{-4}$ in Table 15. Also, because these problems have convergence difficulty in a purely primal-dual path-following method, thus we use a primal-dual path-following method with Mehrotra's predictor-corrector. Note that in each iteration of the predictor-corrector method, 2 linear systems with the same coefficient matrix have to be solved. But having the same coefficient matrix offers no savings in computation time for an iterative solver, unlike the case of a direct solver where the same factorization can be used for both linear systems. Thus, unless necessary, predictor-corrector approach is not the preferred option when an iterative solver is used.

Because of degeneracies, the reduced augmented equation would offer no advantage over the SCE for the `fap` problems. And since each matrix-vector product in (15) is more expensive, it is only logical to expect that Algorithm PFsch would be more efficient than Algorithm PFaug. This expectation is confirmed by the numerical results presented in Table 15. It is evident that Algorithm PFaug consistently takes longer time than Algorithm PFsch to solve the problems. Furthermore, Algorithm PFaug fails to solve 8 of the problems (entries with bold-face fonts) to the required accuracy of $10^{-4}$, whereas Algorithm PFsch successfully solved all. This set of SDPs illustrates that for problems that are degenerate, it is not advisable to use an iterative method to solve the reduced augmented equation (15). Unless modifications on (15) are done to handle the ill-conditioning of $\mathcal{K}$, it appears that the simplest approach of using the PCR method on (5) should be used.

Table 13 shows the primal and dual objective values for the `mcp`, `arch`, and `fap` problems obtained by Algorithm PFsch.

It has been reported in [5] that the BMZ method is highly successful in solving the `fap` problems compared to the SB method. (The BMPR method is not tested on the `fap` problems in [3].) By comparing the performance of Algorithm PFsch in Table 15 with the results report in [5, Table 6], we observe that our interior-point method fared reasonably well compared to the first order BMZ method. The CPU time taken to solve all the problems, except `fap12`, are comparable for both methods (again, we must take into account that different machines are used, and our machine is 1-2 times faster). For example, the problem `fap11` is solved in 9 hours by Algorithm PFsch, and the CPU time reported in [5] is 10.8 hours.

The objective values we obtained in Table 13 for the `fap` problems are better than those obtained in [5]. Take `fap11` for example, the dual objective value we obtained is 0.0297662, with a dual infeasibility of $1.1 \times 10^{-16}$. This value is better (the larger the absolute value the better) than the absolute value of 0.0296136 reported in [5]. For `fap12`, the BMZ method is superior to Algorithm PFsch, where the former is about 3 times faster if an accuracy requirement of $10^{-4}$ is set for the latter. But if the accuracy requirement is set to $10^{-3}$, then Algorithm PFsch can solve `fap12` in about 19 hours compared to 12.5 hours taken by the BMZ method.

Our comparison here between Algorithm PFsch and the BMZ method indicates interior-point methods are not totally uncompetitive compared to first order methods.

## 7 Conclusion and future research

We introduced the reduced augmented equation for computing the search directions in primal-dual interior-point methods. For SDPs that are primal and dual nondegenerate, and have strictly complementary optimal solutions, the coefficient matrices of the reduced augmented equations have condition numbers that are bounded independent of the barrier parameter $\mu$, even when $\mu$ approaches 0.

We proposed Algorithm PFaug that is based on a hybrid between the PCR method applied the SCE and the PSQMR method applied to the reduced augmented equation. Numerical experiments on SDPs arising from maximum clique problems show that Algorithm PFaug performs much better than Algorithm PFsch that is based solely on applying the PCR method to the SCE.

Our interior-point based methods, Algorithms PFaug and PFsch, are competitive (time-wise) compared to the first order BMPR method on majority of the maximum clique problems considered in [5]. Our interior-point based method, Algorithm PFsch, is also competitive compared to the first order BMZ method on the `fap` problems. The numerical results presented in this paper indicate that interior-point methods like Algorithms PFaug and PFsch are not totally uncompetitive compared first order methods such as the SB, BMZ, and BMPR methods. On many of the problems tested in this paper, we are able to obtain objective values that are better or comparable to those obtained by the first order methods.

Algorithm PFaug is well suited for primal and dual nondegenerate problems with optimal solutions that are strictly complementary. It appears that significant modifications to the reduced augmented equation are needed to effectively solve problems that are degenerate. Besides this important issue, there are a number of other issues that we hope to address in the future.

(a) We would like to investigate the performance of the reduced augmented equation approach in a dual scaling interior-point framework for solving SDPs with $n$ large, especially large SDPs arising from maximum cut and graph partitioning problems.

(b) The construction of more sophisticated preconditioners for the reduced augmented matrix.

(c) The use of a direct method to solve the reduced augmented equation so as to generate accurate approximate optimal solutions. Our numerical results in Section 2.1 indicate that the outcome would be promising.

## Acknowledgments

Part of this paper was written while the author was visiting Tokyo Institute of Technology under the Hitachi Fellowship. He thanks Professor Masakazu Kojima for hosting his visit and for many stimulating discussions. He gratefully acknowledges the financial support from the Hitachi Scholarship Foundation during the visit.

The author thanks Sam Burer for providing him the data to generate the frequency assignment problems. Finally, he thanks the referees for reading the paper very carefully and for their helpful comments.

## References

[1] F. Alizadeh, J.-P.A. Haeberly, and M.L. Overton, *Primal-dual interior-point methods for semidefinite programming: convergence results, stability and numerical results*, SIAM J. Optimization, 8 (1998), pp. 746–768.

[2] F. Alizadeh, J. A. Haeberly, and M. Overton, *Complementarity and nondegeneracy in semidefinite programming*, Math. Prog., 77 (1997), pp. 111–128.

[3] S. Burer, and R. Monterio, *A nonlinear programming algorithm for solving semidefinite programs via low-rank factorization*, Mathematical Programming B, 95 (2003), pp. 329–357.

[4] S. Burer, R. Monterio, and Y. Zhang, *Solving a class of semidefinite programs via nonlinear programming*, Mathematical Programming A, 93 (2002), pp. 97–122.

[5] S. Burer, R. Monterio, and Y. Zhang, *A computational study of a gradient-based log-barrier algorithm for a class of large scale SDPs*, Mathematical Programming B, 95 (2003), pp. 359–379.

[6] B. Borchers, *SDPLIB 1.2, a library of semidefinite programming test problems*, Optimization Methods and Software, 11 & 12 (1999), pp. 683–690. Available at `http://www.nmt.edu/~borchers/sdplib.html`.

[7] S. J. Benson, Y. Ye, and X. Zhang, *Solving large-scale sparse semidefinite programs for combinatorial optimization*, SIAM J. Optimization, 10 (2000), pp. 443-461.

[8] C. Choi, and Y. Ye, *Solving sparse semidefinite programs using the dual scaling algorithm with an iterative solver*, working paper, Computational Optimization Laboratory, University of Iowa, March, 2000.

[9] M. Trick, V. Chvatal, W. Cook, D. Johnson, C. McGeoch, and R. Tarjan *The second DIMACS Implementation Challenge: NP hard problems – maximum clique, graph coloring, and satisfiability*, Rugters University, 1992. See also the website:`http://dimacs.rugters.edu/Challenges/`.

[10] T.A. Driscoll, K.C. Toh and L.N. Trefethen, *From potential theory to matrix iterations in six steps*, SIAM Review, 40 (1998), pp. 547-578.

[11] A. Eisenblätter, M. Grötschel, and A.M.C.A. Koster, *Frequency planning and ramification of coloring*, ZIB-report 00-47, Konrad-Zuse-Zentrum für Informationstechnik Berlin, December 2000.

[12] R.W. Freund, and N.M. Nachtigal, *A new Krylov-subspace method for symmetric indefinite linear systems*, Proceedings of the 14th IMACS World Congress on Computational and Applied Mathematics, Atlanta, USA, W.F. Ames ed., July 1994, pp. 1253–1256.

[13] K. Fujisawa, M. Kojima, and K. Nakata, *Exploiting sparsity in primal-dual interior-point methods for semidefinite programming*, Mathematical Programming, 79 (1997), pp. 235–253.

[14] M. Fukuda, M. Kojima, and and M. Shida, *Lagrangian dual interior-point methods for semidefinite programs*, SIAM J. Optimization, 12 (2002), pp. 1007–1031.

[15] M. Fukuda, M. Kojima, K. Murota, and K. Nakata, *Exploiting sparsity in semidefinite programming via matrix completion I: general framework*, SIAM J. Optimization, 11 (2000), pp. 647–674.

[16] C. Helmberg, and K.C. Kiwiel, *A spectral bundle method with bounds*, Mathematical Programming, 93 (2002), pp. 173–194.

[17] C. Helmberg, F. Rendl, R. Vanderbei, and H. Wolkowicz, *An interior-point method for semidefinite programming*, SIAM J. Optimization, 6 (1996), pp. 342–361.

[18] M. Kojima, S. Shindoh, and S. Hara, *Interior-point methods for the monotone semidefinite linear complementarity problem in symmetric matrices*, SIAM J. Optimization, 7 (1997), pp. 86–125.

[19] M. Kojima, M. Shida, and S. Shindoh, *Search directions in the SDP and the monotone SDLCP: generalization and inexact computation*, Mathematical Programming, 85 (1999), pp. 51–80.

[20] C.-J. Lin, and R. Saigal, *An incomplete Cholesky factorization for dense symmetric positive definite matrices*, BIT, 40 (2000), pp. 536–558.

[21] H.D. Mittelmann, *An independent benchmarking of SDP and SOCP solvers*, Mathematical Programming B, 95 (2003), pp. 407-430.

[22] R. D. C. Monteiro, *Primal-dual path following algorithms for semidefinite programming*, SIAM J. Optimization, 7 (1997), pp. 663–678.

[23] K. Nakata, K. Fujisawa, and M. Kojima, *Using the conjugate gradient method in interior-points methods for semidefinite programs* (in Japanese), Proceedings of the Institute of Statistical Mathematics, 46 (1998), pp. 297–316.

[24] K. Nakata, S.-L. Zhang and M. Kojima, *Preconditioned conjugate gradient methods for large scale and dense linear systems in semidefinite programming*, abstract based on talks delivered at INFORMS Meeting, Philadelphia, November 1999.

[25] C.C. Paige and M.A. Saunders, *Solution of sparse indefinite systems of linear equations*, SIAM J. Numerical Analysis, 12 (1975), pp. 617–629.

[26] G. Pataki, *On the rank of extreme matrices in semidefinite programs and the multiplicity of optimal eigenvalues*, Mathematics of Operations Research, 23 (1998), pp. 339–358.

[27] K.K. Phoon, K.C. Toh, S.H. Chan, and F.H. Lee, *An efficient diagonal preconditioner for finite element solution of Biot's consolidation equations*, Int. J. Numerical Methods in Engineering, 55 (2002), pp. 377–400.

[28] G. Gruber, and F. Rendl, *Computational experience with stable set relaxations*, SIAM J. Optimization, 13 (2003), pp. 1014–1028.

[29] Y. Saad, *Iterative Methods for Sparse Linear Systems*, PWS Publishing Company, Boston.

[30] K. C. Toh, *An analysis of ill-conditioned equilibrium systems*, Technical Report, Department of Mathematics, National University of Singapore, 2002.

[31] K. C. Toh, and M. Kojima, *Solving some large scale semidefinite programs via the conjugate residual method*, SIAM J. Optimization, 12 (2002), pp. 669–691.

[32] M. J. Todd, K. C. Toh, and R. H. Tütüncü, *On the Nesterov-Todd direction in semidefinite programming*, SIAM J. Optimization, 8 (1998), pp. 769–796.

[33] K. C. Toh, M. J. Todd, and R. H. Tütüncü, *SDPT3 — a* MATLAB *software package for semidefinite programming, version 1.3*, Optimization Methods and Software, 11 (1999), pp. 545–581. A copy of the software can be obtained through the worldwide web from `http://www.math.nus.sg/~mattohkc`.

[34] V.V. Kovacevic-Vujcic, and M.D. Asic, *Stabilization of interior-point methods for linear programming*, Computational Optimization and Applications, 14 (1999), pp. 331–346.

[35] Y. Zhang, *On extending some primal-dual interior-point algorithms from linear programming to semidefinite programming*, SIAM J. Optimization, 8 (1998), pp. 365–386.

[36] Q. Zhao, S.E. Karisch, F. Rendl, and H. Wolkowicz, *Semidefinite programming relaxations for the quadratic assignment problem*, J. Comb. Optim., 2 (1998), pp. 71–109.

Table 8: Primal and dual objective values obtained by Algorithm PFaug.

| problem | $n$ | $m$ | primal obj | dual obj |
|---:|---:|---:|---:|---:|
| theta6 | 300 | 4375 | -63.4770649 | -63.4770915 |
| theta62 | 300 | 13390 | -29.6412339 | -29.6412589 |
| theta8 | 400 | 7905 | -73.9535154 | -73.9535717 |
| theta82 | 400 | 23872 | -34.3668848 | -34.3668981 |
| theta83 | 400 | 39862 | -20.3018839 | -20.3018980 |
| theta10 | 500 | 12470 | -83.8059524 | -83.8059706 |
| theta102 | 500 | 37467 | -38.3905171 | -38.3905620 |
| theta103 | 500 | 62516 | -22.5285606 | -22.5285800 |
| theta104 | 500 | 87245 | -13.3361385 | -13.3361438 |
| theta12 | 600 | 17979 | -92.8016040 | -92.8016958 |
| theta123 | 600 | 90020 | -24.6686484 | -24.6686554 |
| theta162 | 800 | 127600 | -37.0097262 | -37.0097436 |
| MANN-a27 | 378 | 703 | -132.7628635 | -132.7628930 |
| johnson8-4-4 | 70 | 561 | -13.9999840 | -14.0000044 |
| johnson16-2-4 | 120 | 1681 | -7.9999998 | -8.0000017 |
| san200-0.7-1 | 200 | 5971 | -29.9999629 | -30.0000002 |
| c-fat200-1 | 200 | 18367 | -11.9999970 | -12.0000002 |
| hamming-6-4 | 64 | 1313 | -5.3333301 | -5.3333351 |
| hamming-8-4 | 256 | 11777 | -15.9999977 | -16.0000010 |
| hamming-9-8 | 512 | 2305 | -223.9996367 | -224.0000138 |
| hamming-10-2 | 1025 | 23040 | -102.3999498 | -102.4000165 |
| hamming-7-5-6 | 128 | 1793 | -42.6666515 | -42.6666678 |
| hamming-8-3-4 | 256 | 16129 | -25.5999744 | -25.6000043 |
| hamming-9-5-6 | 512 | 53761 | -85.3331694 | -85.3333369 |
| brock200-1 | 200 | 5067 | -27.4566346 | -27.4566445 |
| brock200-4 | 200 | 6812 | -21.2934670 | -21.2934817 |
| brock400-1 | 400 | 20078 | -39.7018863 | -39.7019055 |
| keller4 | 171 | 5101 | -14.0122384 | -14.0122440 |
| sanr200-0.7 | 200 | 6033 | -23.8361531 | -23.8361601 |
| G43 | 1000 | 9991 | -280.6245145 | -280.6245830 |
| G44 | 1000 | 9991 | -280.5831314 | -280.5832102 |
| G45 | 1000 | 9991 | -280.1848899 | -280.1851375 |
| G46 | 1000 | 9991 | -279.8365727 | -279.8369756 |
| G47 | 1000 | 9991 | -281.8938988 | -281.8939612 |
| p-hat300-1 | 300 | 33918 | -10.0679626 | -10.0679686 |
| G51 | 1000 | 5910 | -348.9996545 | -349.0001073 |
| G52 | 1000 | 5917 | -348.3860739 | -348.3864065 |
| G53 | 1000 | 5915 | -348.3469748 | -348.3473550 |
| G54 | 1000 | 5917 | -340.9998601 | -341.0000203 |

Table 9: Comparison of Algorithms PFsch and PFaug on a number of SDP problems arising from maximum clique problems on randomly generated graphs.

| | | Algorithm PFsch | | | | Algorithm PFaug | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| $n$ $m$ | iter. no. | `relgap` | $\phi$ | cum. time | $N_k$ | `relgap` | $\phi$ | cum. time | $N_k$ | $p$ |
| theta6 | 22 (22) | 7.8 -5 | 2.2 -5 | 1:11 | 715 | 9.3 -5 | 4.1 -6 | 1:00 | 170 | 47 |
| 300 | 24 (24) | 4.3 -6 | 7.9 -7 | 6:12 | 6782 | 1.8 -6 | 1.6 -7 | 1:49 | 365 | 47 |
| 4375 | 25 (25) | 6.2 -7 | 1.6 -7 | 12:11 | 11641 | 5.0 -7 | 3.1 -8 | 2:28 | 540 | 48 |
| theta62 | 21 (21) | 7.9 -5 | 1.2 -5 | 3:35 | 1327 | 7.9 -5 | 1.2 -5 | 3:17 | 1327 | 0 |
| 300 | 23 (23) | 3.2 -6 | 4.9 -7 | 26:31 | 15796 | 3.2 -6 | 9.2 -8 | 7:43 | 1015 | 106 |
| 13390 | 24 (24) | 8.7 -7 | 1.0 -7 | 1:03:13 | 33831 | 8.4 -7 | 6.9 -9 | 10:38 | 1105 | 106 |
| theta8 | 23 (23) | 2.4 -5 | 5.0 -6 | 7:57 | 3247 | 2.3 -5 | 2.8 -8 | 4:00 | 400 | 66 |
| 400 | 24 (24) | 4.4 -6 | 1.1 -6 | 18:21 | 7896 | 3.0 -6 | 9.5 -9 | 5:18 | 415 | 66 |
| 7905 | 25 (25) | 7.2 -7 | 1.8 -7 | 43:24 | 18436 | 7.6 -7 | 3.9 -8 | 6:37 | 430 | 66 |
| theta82 | 23 (23) | 2.2 -5 | 3.6 -6 | 15:27 | 3036 | 2.2 -5 | 2.3 -7 | 10:35 | 510 | 138 |
| 400 | 24 (24) | 2.7 -6 | 5.1 -7 | 45:44 | 11751 | 2.7 -6 | 2.4 -9 | 14:37 | 620 | 138 |
| 23872 | 25 (25) | 3.9 -7 | 7.6 -8 | 2:19:49 | 36504 | 3.9 -7 | 6.6-10 | 18:16 | 555 | 138 |
| theta83 | 22 (22) | 2.8 -5 | 2.9 -6 | 16:04 | 2959 | 2.8 -5 | 2.9 -6 | 15:49 | 2959 | 0 |
| 400 | 23 (23) | 4.0 -6 | 4.5 -7 | 48:23 | 12031 | 4.0 -6 | 1.4 -7 | 29:18 | 1785 | 204 |
| 39862 | 24 (24) | 6.9 -7 | 8.0 -8 | 2:31:30 | 38507 | 6.9 -7 | 4.4 -9 | 38:21 | 1160 | 201 |
| theta10 | 22 (22) | 9.0 -5 | 2.2 -5 | 10:01 | 1327 | 9.7 -5 | 9.1 -6 | 7:13 | 265 | 81 |
| 500 | 24 (24) | 1.9 -6 | 6.5 -7 | 55:50 | 9120 | 1.8 -6 | 9.5 -8 | 13:36 | 460 | 81 |
| 12470 | 25 (25) | 2.3 -7 | 9.6 -8 | 2:32:49 | 28412 | 2.2 -7 | 2.6 -8 | 17:52 | 560 | 82 |
| theta102 | 23 (23) | 6.3 -5 | 7.9 -6 | 22:25 | 1800 | 6.3 -5 | 7.9 -6 | 22:04 | 1800 | 0 |
| 500 | 24 (24) | 8.8 -6 | 1.3 -6 | 54:53 | 6437 | 8.8 -6 | 2.6 -7 | 32:33 | 820 | 170 |
| 37467 | 26 (26) | 1.7 -7 | 3.3 -8 | 7:51:53 | 62416 | 1.7 -7 | 8.9-10 | 1:02:15 | 1120 | 174 |
| theta103 | 22 (22) | 3.3 -5 | 2.9 -6 | 35:16 | 3719 | 3.3 -5 | 2.9 -6 | 34:50 | 3719 | 0 |
| 500 | 23 (23) | 5.3 -6 | 4.8 -7 | 1:38:14 | 12045 | 5.3 -6 | 3.6 -8 | 54:21 | 1190 | 252 |
| 62516 | 24 (24) | 8.6 -7 | 9.0 -8 | 4:26:17 | 32131 | 8.6 -7 | 1.8 -9 | 1:11:38 | 1005 | 252 |
| theta104 | 22 (22) | 7.3 -5 | 4.0 -6 | 24:28 | 2070 | 7.3 -5 | 4.0 -6 | 24:12 | 2070 | 0 |
| 500 | 24 (24) | 2.2 -6 | 1.0 -7 | 3:31:02 | 26022 | 2.2 -6 | 1.0 -8 | 2:25:53 | 4285 | 332 |
| 87245 | 25 (25) | 4.0 -7 | 2.1 -8 | 9:24:10 | 65693 | 4.0 -7 | 3.4-10 | 3:08:09 | 2160 | 328 |
| theta12 | 24 (24) | 5.3 -5 | 1.5 -5 | 25:21 | 1430 | 5.8 -5 | 3.3 -6 | 18:39 | 255 | 98 |
| 600 | 25 (25) | 9.2 -6 | 2.3 -6 | 1:07:37 | 5626 | 9.2 -6 | 3.7 -8 | 25:13 | 395 | 98 |
| 17979 | 27 (26) | 1.4 -7 | 5.9 -8 | 8:03:22 | 40323 | 9.8 -7 | 3.4 -8 | 33:13 | 485 | 98 |
| theta123 | 23 (23) | 5.2 -5 | 4.7 -6 | 47:35 | 2707 | 5.2 -5 | 4.7 -6 | 47:25 | 2707 | 0 |
| 600 | 24 (24) | 7.9 -6 | 7.4 -7 | 2:10:12 | 9587 | 7.9 -6 | 1.2 -7 | 1:20:28 | 1135 | 301 |
| 90020 | 26 (26) | 2.8 -7 | 2.7 -8 | 16:29:46 | 72630 | 2.8 -7 | 5.0-10 | 2:20:16 | 885 | 301 |
| theta162 | 25 (25) | 1.4 -5 | 1.8 -6 | 3:20:59 | 6126 | 1.4 -5 | 1.8 -6 | 3:21:34 | 6126 | 0 |
| 800 | 26 (26) | 2.3 -6 | 3.0 -7 | 10:00:53 | 20400 | 2.3 -6 | 5.9 -8 | 4:58:07 | 1670 | 335 |
| 127600 | 27 (27) | 4.7 -7 | 6.0 -8 | 27:47:52 | 54234 | 4.7 -7 | 4.2 -9 | 6:34:37 | 1650 | 340 |

Table 10: Comparison of Algorithms PFsch and PFaug on SDPs from the Second DIMACS Challenge on Maximum Clique Problems.

| | | Algorithm PFsch | | | | Algorithm PFaug | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| $n$ $m$ | iter. no. | `relgap` | $\phi$ | cum. time | $N_k$ | `relgap` | $\phi$ | cum. time | $N_k$ | $p$ |
| MANN-a27 | 39 (39) | 7.5 -5 | 1.8 -5 | 1:48 | 28 | 7.5 -5 | 1.8 -5 | 1:43 | 28 | 0 |
| 378 | 41 (41) | 2.0 -6 | 9.5 -7 | 1:56 | 50 | 2.0 -6 | 9.5 -7 | 1:51 | 50 | 0 |
| 703 | 42 (42) | 2.0 -7 | 9.8 -8 | 2:00 | 61 | 2.0 -7 | 9.8 -8 | 1:55 | 61 | 0 |
| johnson8-4-4 | 16 (16) | 1.5 -5 | 1.1-10 | 0:18 | 2 | 1.5 -5 | 1.1-10 | 0:16 | 2 | 0 |
| 70 | 17 (17) | 1.5 -6 | 6.0-10 | 0:19 | 2 | 1.5 -6 | 6.0-10 | 0:17 | 2 | 0 |
| 561 | 18 (18) | 1.5 -7 | 3.6 -9 | 0:20 | 2 | 1.5 -7 | 3.6 -9 | 0:18 | 2 | 0 |
| johnson16-2-4 | 17 (17) | 2.4 -5 | 7.8-13 | 0:20 | 2 | 2.4 -5 | 7.8-13 | 0:18 | 2 | 0 |
| 120 | 18 (18) | 2.4 -6 | 6.4-12 | 0:22 | 2 | 2.4 -6 | 6.4-12 | 0:19 | 2 | 0 |
| 1681 | 19 (19) | 2.4 -7 | 5.3-11 | 0:23 | 2 | 2.4 -7 | 5.3-11 | 0:20 | 2 | 0 |
| san200-0.7-1 | 23 (23) | 1.2 -5 | 8.4 -7 | 0:35 | 22 | 1.2 -5 | 8.4 -7 | 0:32 | 22 | 0 |
| 200 | 24 (24) | 1.2 -6 | 9.6 -8 | 0:37 | 30 | 1.2 -6 | 9.6 -8 | 0:34 | 30 | 0 |
| 5971 | 25 (25) | 1.2 -7 | 1.0 -8 | 0:39 | 49 | 1.2 -7 | 1.0 -8 | 0:36 | 49 | 0 |
| c-fat200-1 | 21 (21) | 2.8 -5 | 1.6 -6 | 0:54 | 175 | 2.8 -5 | 1.6 -6 | 0:51 | 175 | 0 |
| 200 | 22 (22) | 2.8 -6 | 2.1 -7 | 1:01 | 255 | 2.8 -6 | 2.1 -7 | 0:58 | 255 | 0 |
| 18367 | 23 (23) | 2.8 -7 | 1.9 -8 | 1:10 | 313 | 2.8 -7 | 1.9 -8 | 1:07 | 313 | 0 |
| hamming-6-4 | 15 (15) | 9.4 -5 | 5.0 -7 | 0:17 | 3 | 9.4 -5 | 5.0 -7 | 0:15 | 3 | 0 |
| 64 | 16 (16) | 9.4 -6 | 4.9 -8 | 0:18 | 3 | 9.4 -6 | 4.9 -8 | 0:16 | 3 | 0 |
| 1313 | 17 (17) | 9.4 -7 | 4.9 -9 | 0:19 | 3 | 9.4 -7 | 4.9 -9 | 0:17 | 3 | 0 |
| hamming-8-4 | 20 (20) | 2.1 -5 | 1.9 -7 | 0:34 | 4 | 2.1 -5 | 1.9 -7 | 0:31 | 4 | 0 |
| 256 | 21 (21) | 2.1 -6 | 1.3 -8 | 0:36 | 5 | 2.1 -6 | 1.3 -8 | 0:33 | 5 | 0 |
| 11777 | 22 (22) | 2.1 -7 | 1.2 -8 | 0:38 | 4 | 2.1 -7 | 1.2 -8 | 0:35 | 4 | 0 |
| hamming-9-8 | 19 (19) | 1.7 -5 | 1.3 -6 | 2:10 | 4 | 1.7 -5 | 1.3 -6 | 2:08 | 4 | 0 |
| 512 | 20 (20) | 1.7 -6 | 7.6 -7 | 2:17 | 4 | 1.7 -6 | 7.6 -7 | 2:15 | 4 | 0 |
| 2305 | 21 (21) | 1.7 -7 | 2.3 -9 | 2:25 | 5 | 1.7 -7 | 2.3 -9 | 2:23 | 5 | 0 |
| hamming-10-2 | 21 (21) | 6.5 -5 | 3.4-10 | 17:31 | 3 | 6.5 -5 | 3.4-10 | 17:36 | 3 | 0 |
| 1025 | 22 (22) | 6.5 -6 | 1.5 -9 | 18:26 | 3 | 6.5 -6 | 1.5 -9 | 18:31 | 3 | 0 |
| 23040 | 23 (23) | 6.5 -7 | 5.1 -8 | 19:19 | 2 | 6.5 -7 | 5.1 -8 | 19:24 | 2 | 0 |
| hamming-7-5-6 | 17 (17) | 3.8 -5 | 4.0 -6 | 0:21 | 2 | 3.8 -5 | 4.0 -6 | 0:18 | 2 | 0 |
| 128 | 18 (18) | 3.8 -6 | 5.2 -9 | 0:22 | 4 | 3.8 -6 | 5.2 -9 | 0:19 | 4 | 0 |
| 1793 | 19 (19) | 3.8 -7 | 1.9 -8 | 0:23 | 4 | 3.8 -7 | 1.9 -8 | 0:21 | 4 | 0 |
| hamming-8-3-4 | 19 (19) | 1.2 -5 | 2.6 -8 | 0:32 | 3 | 1.2 -5 | 2.6 -8 | 0:30 | 3 | 0 |
| 256 | 20 (20) | 1.2 -6 | 5.5 -8 | 0:34 | 4 | 1.2 -6 | 5.5 -8 | 0:31 | 4 | 0 |
| 16129 | 21 (21) | 1.2 -7 | 1.5 -9 | 0:36 | 6 | 1.2 -7 | 1.5 -9 | 0:33 | 6 | 0 |
| hamming-9-5-6 | 20 (20) | 2.0 -5 | 2.7 -6 | 2:47 | 6 | 2.0 -5 | 2.7 -6 | 2:46 | 6 | 0 |
| 512 | 21 (21) | 2.0 -6 | 8.7 -8 | 2:56 | 5 | 2.0 -6 | 8.7 -8 | 2:56 | 5 | 0 |
| 53761 | 22 (22) | 2.0 -7 | 6.4 -9 | 3:06 | 5 | 2.0 -7 | 6.4 -9 | 3:06 | 5 | 0 |

Table 11: Comparison of Algorithms PFsch and PFaug on SDPs from the Second DIMACS Challenge on Maximum Clique Problems.

| | | Algorithm PFsch | | | | Algorithm PFaug | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| $n$ / $m$ | iter. no. | `relgap` | $\phi$ | cum. time | $N_k$ | `relgap` | $\phi$ | cum. time | $N_k$ | $p$ |
| brock200-1 | 21 (21) | 3.8 -5 | 6.6 -6 | 1:33 | 1646 | 3.6 -5 | 1.1 -7 | 1:14 | 335 | 63 |
| 200 | 22 (22) | 6.2 -6 | 1.1 -6 | 3:35 | 6498 | 5.9 -6 | 8.0 -9 | 1:30 | 330 | 63 |
| 5067 | 24 (24) | 3.2 -7 | 1.0 -7 | 16:50 | 25334 | 3.6 -7 | 3.3-10 | 1:57 | 240 | 63 |
| brock200-4 | 21 (21) | 1.8 -5 | 2.5 -6 | 2:15 | 3185 | 1.8 -5 | 8.7 -9 | 1:29 | 640 | 79 |
| 200 | 22 (22) | 3.5 -6 | 4.8 -7 | 5:52 | 11188 | 3.5 -6 | 1.6 -9 | 2:14 | 900 | 79 |
| 6812 | 23 (23) | 7.2 -7 | 8.2 -8 | 14:17 | 26054 | 6.9 -7 | 1.1 -9 | 2:41 | 545 | 77 |
| brock400-1 | 23 (23) | 3.0 -5 | 5.8 -6 | 11:55 | 2077 | 3.0 -5 | 6.2 -8 | 9:33 | 515 | 123 |
| 400 | 24 (24) | 3.3 -6 | 7.4 -7 | 34:22 | 8883 | 3.3 -6 | 2.9 -9 | 13:04 | 560 | 123 |
| 20078 | 25 (25) | 4.8 -7 | 1.1 -7 | 1:53:34 | 31285 | 4.8 -7 | 5.1-10 | 16:56 | 620 | 123 |
| keller4 | 21 (21) | 4.2 -5 | 4.2 -6 | 0:51 | 1008 | 4.2 -5 | 2.9 -7 | 0:25 | 110 | 67 |
| 171 | 22 (22) | 4.2 -6 | 4.2 -7 | 1:33 | 3180 | 4.2 -6 | 1.3 -7 | 0:34 | 235 | 67 |
| 5101 | 23 (23) | 4.2 -7 | 4.2 -8 | 2:30 | 5063 | 4.2 -7 | 3.4 -8 | 0:58 | 765 | 67 |
| sanr200-0.7 | 21 (21) | 2.6 -5 | 4.4 -6 | 1:45 | 2149 | 2.6 -5 | 7.0 -8 | 1:07 | 410 | 71 |
| 200 | 22 (22) | 4.6 -6 | 7.9 -7 | 4:33 | 8718 | 4.6 -6 | 1.1 -8 | 1:32 | 510 | 71 |
| 6033 | 23 (24) | 9.3 -7 | 1.4 -7 | 11:39 | 22494 | 3.0 -7 | 2.7-10 | 2:15 | 375 | 71 |
| G43 | 27 (27) | 4.0 -5 | 3.1 -5 | 48:39 | 754 | 5.4 -5 | 3.6 -7 | 43:18 | 245 | 56 |
| 1000 | 29 (28) | 4.7 -6 | 8.3 -7 | 3:37:58 | 11230 | 9.7 -6 | 2.2 -7 | 51:22 | 300 | 58 |
| 9991 | 30 (32) | 4.7 -7 | 3.8 -7 | 5:16:54 | 8943 | 2.6 -7 | 3.8 -8 | 1:25:23 | 265 | 58 |
| G44 | 28 (28) | 2.0 -5 | 1.2 -5 | 1:09:00 | 1497 | 1.4 -5 | 1.2 -7 | 54:55 | 220 | 60 |
| 1000 | 29 (29) | 7.4 -6 | 3.7 -6 | 2:38:09 | 8070 | 2.8 -6 | 2.7 -9 | 1:04:46 | 380 | 60 |
| 9991 | 31 (30) | 2.5 -7 | 2.0 -7 | 9:44:55 | 14144 | 2.8 -7 | 2.3-10 | 1:11:52 | 255 | 60 |
| G45 | 28 (29) | 1.8 -5 | 1.4 -5 | 1:48:32 | 854 | 8.9 -5 | 5.9 -7 | 1:09:36 | 310 | 57 |
| 1000 | 29 (30) | 3.6 -6 | 2.4 -6 | 3:09:17 | 7315 | 8.9 -6 | 3.7 -7 | 1:15:55 | 220 | 58 |
| 9991 | 30 (31) | 7.7 -7 | 2.7 -7 | 7:38:25 | 24263 | 8.9 -7 | 4.0 -8 | 1:24:22 | 320 | 58 |
| G46 | 27 (27) | 3.9 -5 | 3.1 -5 | 48:01 | 652 | 5.5 -5 | 1.6 -6 | 44:23 | 235 | 56 |
| 1000 | 29 (28) | 7.6 -6 | 2.9 -6 | 3:35:40 | 10852 | 8.9 -6 | 1.0 -7 | 53:19 | 340 | 60 |
| 9991 | 31 (30) | 2.6 -7 | 2.1 -7 | 7:20:06 | 10383 | 1.8 -7 | 9.8-11 | 1:09:28 | 335 | 60 |
| G47 | 27 (27) | 5.3 -5 | 4.0 -5 | 44:42 | 608 | 6.3 -5 | 6.7 -7 | 44:50 | 190 | 58 |
| 1000 | 30 (28) | 2.0 -6 | 1.4 -6 | 2:36:55 | 2108 | 9.7 -6 | 1.4 -7 | 51:11 | 225 | 58 |
| 9991 | 31 (30) | 4.1 -7 | 1.5 -7 | 4:54:55 | 12555 | 2.2 -7 | 8.3 -9 | 1:07:37 | 375 | 58 |

Table 12: Comparison of Algorithms PFsch and PFaug on SDPs from the Second DIMACS Challenge on Maximum Clique Problems.

| | | Algorithm PFsch | | | | Algorithm PFaug | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| $n$ $m$ | iter. no. | `relgap` | $\phi$ | cum. time | $N_k$ | `relgap` | $\phi$ | cum. time | $N_k$ | $p$ |
| p-hat300-1 | 24 (24) | 9.3 -5 | 1.3 -6 | 13:19 | 4357 | 9.3 -5 | 1.1 -9 | 1:42:24 | 26790 | 209 |
| 300 | 26 (26) | 6.8 -6 | 1.8 -7 | 42:52 | 17512 | 6.9 -6 | 3.8 -8 | 9:49:14 | 101750 | 200 |
| 33918 | 28 (28) | 7.1 -7 | 1.2 -8 | 2:12:47 | 43234 | 5.9 -7 | 1.7-10 | 20:47:54 | 100775 | 200 |
| G51 | 44 (43) | 4.2 -5 | 1.5 -5 | 1:19:28 | 305 | 8.7 -5 | 5.8 -6 | 2:56:49 | 1505 | 5 |
| 1000 | 45 (45) | 4.3 -6 | 3.5 -6 | 1:23:51 | 482 | 1.4 -6 | 1.4 -6 | 3:06:26 | 414 | 0 |
| 5910 | 46 (46) | 4.8 -7 | 4.2 -7 | 1:28:16 | 490 | 1.4 -7 | 1.4 -7 | 3:11:11 | 514 | 0 |
| G52 | 60 (58) | 9.3 -5 | 5.2 -6 | 4:28:59 | 3291 | 8.7 -5 | 1.2 -6 | 13:02:23 | 5395 | 6 |
| 1000 | 68 (65) | 6.8 -6 | 6.7 -7 | 9:04:03 | 4212 | 7.9 -6 | 2.7 -7 | 25:58:35 | 9985 | 8 |
| 5917 | 71 (69) | 6.0 -7 | 2.2 -7 | 11:31:50 | 5907 | 9.5 -7 | 5.9 -8 | 33:19:59 | 14257 | 0 |
| G53 | 56 (58) | 8.4 -5 | 3.0 -6 | 6:25:55 | 6874 | 7.7 -5 | 6.4 -6 | 7:25:49 | 3905 | 6 |
| 1000 | 62 (63) | 8.8 -6 | 1.2 -6 | 14:06:25 | 26162 | 8.7 -6 | 4.6 -7 | 17:32:15 | 12955 | 6 |
| 5915 | 68 (68) | 5.2 -7 | 3.2 -7 | 33:24:45 | 29574 | 8.6 -7 | 4.9 -7 | 37:15:59 | 17740 | 6 |
| G54 | 45 (45) | 6.6 -5 | 3.9 -5 | 1:17:04 | 288 | 3.7 -5 | 6.3 -6 | 3:35:28 | 2290 | 7 |
| 1000 | 46 (46) | 6.6 -6 | 5.8 -6 | 1:21:25 | 477 | 3.7 -6 | 3.5 -6 | 3:40:09 | 494 | 0 |
| 5917 | 48 (47) | 3.4 -7 | 1.1 -7 | 1:45:27 | 2325 | 3.7 -7 | 3.3 -7 | 3:46:16 | 699 | 0 |

Table 13: Primal and dual objective values obtained by Algorithm PFsch.

| problem | $n$ | $m$ | primal obj | dual obj |
|---|---|---|---|---|
| mcp500-1 | 451 | 451 | -598.1479228 | -598.1485310 |
| mcp500-2 | 493 | 493 | -1070.0563326 | -1070.0567704 |
| mcp500-3 | 500 | 500 | -1847.9696030 | -1847.9700289 |
| mcp500-4 | 500 | 500 | -3566.7373504 | -3566.7380666 |
| arch0 | 161 | 174 | -0.5665156 | -0.5665177 |
| arch2 | 161 | 174 | -0.6715133 | -0.6715158 |
| arch4 | 161 | 174 | -0.9726271 | -0.9726275 |
| arch8 | 161 | 174 | -7.0569738 | -7.0569811 |
| fap01 | 52 | 1378 | 0.0329454 | 0.0328773 |
| fap02 | 61 | 1866 | 0.0007310 | 0.0006973 |
| fap03 | 65 | 2145 | 0.0493711 | 0.0493676 |
| fap04 | 81 | 3321 | 0.1749789 | 0.1748222 |
| fap05 | 84 | 3570 | 0.3083974 | 0.3082823 |
| fap06 | 93 | 4371 | 0.4595326 | 0.4593247 |
| fap07 | 98 | 4851 | 2.1180259 | 2.1176137 |
| fap08 | 120 | 7260 | 2.4363666 | 2.4362657 |
| fap09 | 174 | 15225 | 10.7982702 | 10.7976727 |
| fap10 | 183 | 14479 | 0.0096992 | 0.0096708 |
| fap11 | 252 | 24292 | 0.0298764 | 0.0297662 |
| fap12 | 369 | 26462 | 0.2734163 | 0.2732371 |

Table 14: Comparison of Algorithms PFsch and PFaug on `mcp` and `arch` problems from SDPLIB.

| | | Algorithm PFsch | | | | Algorithm PFaug | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| $n$ $m$ | iter. no. | `relgap` | $\phi$ | cum. time | $N_k$ | `relgap` | $\phi$ | cum. time | $N_k$ | $p$ |
| mcp500-1 | 25 (25) | 1.0 -5 | 1.1 -6 | 1:24 | 225 | 1.1 -5 | 3.1 -6 | 1:49 | 95 | 11 |
| 451 | 26 (26) | 1.0 -6 | 1.1 -7 | 1:38 | 411 | 3.3 -6 | 2.9 -7 | 1:58 | 95 | 11 |
| 451 | 27 (27) | 1.0 -7 | 9.9 -9 | 1:55 | 536 | 5.1 -7 | 5.4 -8 | 2:10 | 140 | 11 |
| mcp500-2 | 22 (22) | 8.2 -5 | 6.5 -6 | 1:43 | 336 | 8.2 -5 | 6.5 -6 | 1:47 | 336 | 0 |
| 493 | 24 (24) | 2.5 -6 | 3.3 -7 | 2:21 | 550 | 3.1 -6 | 4.9 -7 | 2:19 | 170 | 8 |
| 493 | 25 (25) | 4.1 -7 | 4.6 -8 | 3:04 | 1161 | 9.2 -7 | 1.8 -7 | 2:44 | 270 | 8 |
| mcp500-3 | 20 (20) | 7.4 -5 | 1.3 -5 | 1:40 | 195 | 7.5 -5 | 2.8 -5 | 1:48 | 85 | 8 |
| 500 | 21 (22) | 9.9 -6 | 2.6 -6 | 1:55 | 322 | 6.2 -6 | 1.6 -6 | 2:17 | 110 | 9 |
| 500 | 23 (24) | 2.3 -7 | 5.2 -8 | 2:55 | 848 | 6.0 -7 | 5.0 -8 | 2:47 | 115 | 9 |
| mcp500-4 | 19 (19) | 5.4 -5 | 2.5 -5 | 1:45 | 183 | 5.4 -5 | 2.5 -5 | 1:54 | 183 | 0 |
| 500 | 21 (21) | 7.7 -6 | 1.7 -6 | 2:43 | 885 | 7.7 -6 | 1.8 -6 | 2:31 | 70 | 11 |
| 500 | 23 (24) | 2.0 -7 | 1.3 -7 | 3:35 | 681 | 7.2 -7 | 9.6 -8 | 3:12 | 60 | 11 |
| arch0 | 52 (52) | 9.1 -5 | 8.3 -7 | 0:57 | 869 | 9.4 -5 | 5.8 -9 | 0:45 | 100 | 8 |
| 161 | 57 (57) | 6.0 -6 | 4.1 -6 | 1:17 | 869 | 6.3 -6 | 2.4-11 | 0:54 | 110 | 8 |
| 174 | 58 (62) | **2.8 -6** | **3.7 -6** | 1:21 | 869 | 5.8 -7 | 7.5-11 | 1:02 | 105 | 8 |
| arch2 | 45 (45) | 5.9 -5 | 1.0 -6 | 0:49 | 869 | 5.9 -5 | 3.0 -9 | 0:30 | 80 | 8 |
| 161 | 48 (48) | 6.0 -6 | 1.0 -6 | 1:03 | 869 | 6.0 -6 | 4.7-10 | 0:35 | 90 | 8 |
| 174 | 50 (52) | **2.2 -6** | **7.2 -6** | 1:11 | 869 | 7.6 -7 | 4.1-12 | 0:41 | 100 | 8 |
| arch4 | 50 (51) | 9.9 -5 | 3.3 -6 | 0:56 | 869 | 5.9 -5 | 8.9-10 | 0:36 | 75 | 5 |
| 161 | 52 (53) | 4.4 -6 | 3.7 -7 | 1:05 | 869 | 2.8 -6 | 1.6-11 | 0:39 | 80 | 5 |
| 174 | 53 (54) | 7.2 -7 | **2.6 -6** | 1:09 | 869 | 2.9 -7 | 2.1-11 | 0:40 | 75 | 5 |
| arch8 | 52 (52) | 2.1 -5 | 1.4 -8 | 0:50 | 587 | 3.5 -5 | 6.8 -9 | 0:33 | 35 | 7 |
| 161 | 53 (53) | 7.0 -6 | 7.2 -7 | 0:54 | 869 | 9.7 -6 | 1.2 -9 | 0:34 | 55 | 7 |
| 174 | 58 (57) | **1.0 -6** | 2.5 -7 | 1:16 | 869 | 1.1 -7 | 6.9-11 | 0:39 | 50 | 8 |

Table 15: Comparison of Algorithms PFsch and PFaug on `fap` problems.

| | | Algorithm PFsch | | | | Algorithm PFaug | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| $n$ $m$ | iter. no. | `relgap` | $\phi$ | cum. time | $N_k$ | `relgap` | $\phi$ | cum. time | $N_k$ | $p$ |
| fap01 | 26 (26) | 2.5 -3 | 4.3 -7 | 0:10 | 4474 | 2.5 -3 | 4.3 -7 | 0:10 | 4474 | 0 |
| 52 | 27 (27) | 3.7 -4 | 9.8 -8 | 0:15 | 10071 | 3.7 -4 | 9.8 -8 | 0:15 | 10071 | 0 |
| 1378 | 28 (28) | 6.7 -5 | 3.4 -7 | 0:21 | 11022 | **3.8 -4** | 4.6 -5 | 0:25 | 6885 | 46 |
| fap02 | 23 (23) | 5.9 -3 | 4.1 -7 | 0:05 | 1116 | 5.9 -3 | 4.1 -7 | 0:05 | 1116 | 0 |
| 61 | 25 (25) | 7.6 -4 | 4.5 -8 | 0:09 | 2660 | 7.6 -4 | 4.5 -8 | 0:09 | 2660 | 0 |
| 1866 | 27 (27) | 3.4 -5 | 4.0 -9 | 0:15 | 6096 | 2.9 -5 | **1.4 -4** | 0:31 | 9325 | 56 |
| fap03 | 30 (30) | 3.3 -3 | 3.3 -7 | 0:17 | 4431 | 3.3 -3 | 3.3 -7 | 0:17 | 4431 | 0 |
| 65 | 31 (31) | 8.2 -4 | 8.6 -8 | 0:25 | 9612 | 3.2 -3 | 1.5 -4 | 0:47 | 10715 | 60 |
| 2145 | 33 (31) | 2.8 -5 | 1.1 -6 | 0:55 | 17158 | **3.2 -3** | **1.5 -4** | 0:47 | 10715 | 60 |
| fap04 | 37 (37) | 4.1 -3 | 3.3 -7 | 1:47 | 23146 | 5.8 -3 | 9.6 -7 | 2:28 | 15250 | 76 |
| 81 | 39 (39) | 4.2 -4 | 8.3 -7 | 3:12 | 26566 | 2.5 -3 | 1.2 -5 | 5:25 | 16600 | 76 |
| 3321 | 40 (39) | 8.7 -5 | 4.7 -6 | 3:55 | 26566 | **2.5 -3** | 1.2 -5 | 5:25 | 16600 | 76 |
| fap05 | 41 (41) | 6.4 -3 | 2.6 -7 | 2:26 | 28558 | 7.5 -3 | 8.6 -7 | 4:12 | 14620 | 79 |
| 84 | 43 (45) | 8.4 -4 | 1.4 -5 | 4:04 | 28558 | 2.7 -3 | 2.6 -6 | 11:38 | 17840 | 79 |
| 3570 | 45 (45) | 3.6 -5 | 2.9 -5 | 5:40 | 28558 | **2.7 -3** | 2.6 -6 | 11:38 | 17840 | 79 |
| fap06 | 43 (43) | 4.2 -3 | 1.7 -7 | 3:36 | 31778 | 5.7 -3 | 2.2 -6 | 10:16 | 21850 | 83 |
| 93 | 45 (45) | 4.6 -4 | 6.9 -6 | 6:30 | 34966 | 9.8 -4 | 5.8 -7 | 16:18 | 20005 | 86 |
| 4371 | 47 (49) | 3.4 -5 | 1.3 -5 | 9:16 | 34966 | **3.9 -4** | 2.7 -6 | 29:09 | 21850 | 86 |
| fap07 | 43 (44) | 7.7 -3 | 6.1 -7 | 4:49 | 34270 | 7.6 -3 | 9.4 -7 | 8:13 | 20145 | 93 |
| 98 | 46 (47) | 4.0 -4 | 4.1 -6 | 10:15 | 38806 | 8.1 -4 | 1.8 -6 | 20:26 | 24250 | 90 |
| 4851 | 48 (50) | 4.3 -5 | 1.1 -5 | 13:49 | 38806 | **2.5 -4** | 8.7 -7 | 33:14 | 24250 | 91 |
| fap08 | 45 (45) | 5.0 -3 | 3.0 -7 | 8:29 | 30394 | 6.2 -3 | 1.1 -6 | 15:51 | 11560 | 110 |
| 120 | 47 (48) | 5.8 -4 | 2.1 -7 | 18:04 | 58078 | 5.6 -4 | 1.2 -7 | 37:13 | 26745 | 110 |
| 7260 | 49 (51) | 4.3 -5 | 3.8 -6 | 27:59 | 58078 | 5.2 -5 | 2.5 -8 | 1:04:43 | 27890 | 110 |
| fap09 | 70 (72) | 5.0 -3 | 8.0 -7 | 28:53 | 23255 | 8.2 -3 | 2.2 -6 | 1:10:58 | 10885 | 157 |
| 174 | 73 (76) | 4.3 -4 | 7.4 -8 | 1:25:14 | 92031 | 7.6 -4 | 2.6 -7 | 2:35:05 | 28440 | 156 |
| 15225 | 75 (77) | 5.3 -5 | 2.2 -6 | 2:38:43 | 121798 | **3.7 -4** | 1.6 -7 | 3:14:07 | 42065 | 156 |
| fap10 | 65 (65) | 7.9 -3 | 1.1 -7 | 25:27 | 25835 | 7.9 -3 | 1.1 -7 | 25:36 | 25835 | 0 |
| 183 | 67 (67) | 5.5 -4 | 1.4 -8 | 1:12:04 | 99556 | 5.5 -4 | 7.9 -6 | 1:47:55 | 72390 | 144 |
| 14479 | 69 (70) | 1.3 -5 | 2.2 -6 | 2:28:29 | 115830 | 7.1 -5 | 5.7 -5 | 5:11:44 | 72390 | 140 |
| fap11 | 71 (71) | 9.3 -3 | 1.2 -7 | 1:22:14 | 30988 | 9.3 -3 | 1.2 -7 | 1:22:58 | 30988 | 0 |
| 252 | 73 (73) | 7.9 -4 | 1.1 -8 | 3:54:12 | 130230 | 7.9 -4 | 1.2 -6 | 10:07:30 | 121455 | 175 |
| 24292 | 75 (75) | 1.5 -5 | 9.2 -7 | 9:01:15 | 194334 | 4.5 -5 | 2.9 -5 | 18:49:02 | 121455 | 171 |
| fap12 | 68 | 9.3 -3 | 1.3 -7 | 5:21:43 | 64851 | | | | | |
| 369 | 70 | 9.5 -4 | 4.0 -8 | 18:43:24 | 211694 | excluded since it will take too long to run | | | | |
| 26462 | 72 | 4.0 -5 | 8.2 -7 | 33:34:41 | 211694 | | | | | |